TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

Giải thuật di truyền và ứng dụng giải thuật di truyền vào thời khóa biểu

Giảng viên hướng dẫn : TS. Phạm Hồng Phong
Chuyên ngành : Khoa học máy tính
Sinh viên thực hiện : Phan Văn Hiếu
MSSV : 0178267
Lớp : 67CNCS

Hà Nội, tháng 12 năm 2025

LỜI CẢM ƠN

Trong suốt quá trình học tập và thực hiện khóa luận tốt nghiệp với đề tài " Nghiên cứu và Ứng dụng Giải thuật Di truyền trong Xếp Thời khóa biểu", em đã nhận được rất nhiều sự quan tâm, giúp đỡ và động viên quý báu.

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc tới Thầy **Phạm Hồng Phong**, người đã tận tình hướng dẫn, định hướng, góp ý chuyên môn và luôn đồng hành cùng em trong suốt quá trình nghiên cứu và hoàn thiện khóa luận này.

Em xin chân thành cảm ơn Ban Giám hiệu Trường **Đại Học Xây Dựng Hà Nội**, **Khoa Công Nghệ Thông Tin**, cùng toàn thể Quý Thầy Cô trong Khoa đã truyền đạt cho em những kiến thức quý báu, tạo điều kiện thuận lợi về môi trường học tập và nghiên cứu trong suốt thời gian học tại trường.

Em cũng xin gửi lời cảm ơn đến các anh chị, bạn bè và gia đình đã luôn động viên, chia sẻ khó khăn và hỗ trợ em cả về tinh thần lẫn vật chất để em có thể hoàn thành tốt khóa luận này.

Mặc dù đã rất cố gắng, đồ án tốt nghiệp khó tránh khỏi những thiếu sót nhất định. Em rất mong nhận được những ý kiến đóng góp của Quý Thầy Cô và các bạn để đề tài được hoàn thiện hơn.

Hà Nội, tháng 12 năm 2025

Sinh viên thực hiện

Phan Văn Hiếu

Mục lục

0.1. Lời Mở Đầu

Trong ngành khoa học máy tính, tìm kiếm lời giải tối ưu cho các bài toán là vấn đề được các nhà khoa học máy tính đặc biệt rất quan tâm. Mục đích chính của các thuật toán tìm kiếm lời giải là tìm ra lời giải tối ưu nhất cho bài toán trong thời gian nhỏ nhất. Các thuật toán như tìm kiếm không có thông tin / vét cạn (tìm kiếm trên danh sách, trên cây hoặc đồ thị) sử dụng phương pháp đơn giản nhất và trực quan nhất hoặc các thuật toán tìm kiếm có thông tin sử dụng heurictics để áp dụng các tri thức về cấu trúc của không gian tìm kiếm nhằm giảm thời gian cần thiết cho việc tìm kiếm được sử dụng nhiều nhưng chỉ với không gian tìm kiếm nhỏ và không hiệu quả khi tìm kiếm trong không gian tìm kiếm lớn. Tuy nhiên, trong thực tiễn có rất nhiều bài toán tối ưu với không gian tìm kiếm rất lớn cần phải giải quyết. Vì vậy, việc đòi hỏi thuật giải chất lượng cao và sử dụng kỹ thuật trí tuệ nhân tạo đặc biệt rất cần thiết khi giải quyết các bài toán có không gian tìm kiếm lớn. Thuật giải di truyền (genetic algorithm) là một trong những kỹ thuật tìm kiếm lời giải tối ưu đã đáp ứng được yêu cầu của nhiều bài toán và ứng dụng.

Thuật giải di truyền đã được phát minh ra để bắt chước quá trình phát triển tự nhiên trong điều kiện quy định sẵn của môi trường. Các đặc điểm của quá trình này đã thu hút sự chú ý của John Holand (ở đại học Michigan) ngay từ những năm 1970. Holand tin rằng sự gắn kết thích hợp trong thuật giải máy tính có thể tạo ra một kỹ thuật giúp giải quyết các vấn đề khó khăn giống như trong tự nhiên đã diễn ra — thông qua quá trình tiến hóa.

Trên thế giới hiện nay, Thuật Giải Di Truyền kết hợp với Công nghệ thông tin được ứng dụng để giải quyết những vấn đề phức tạp trong hệ thống điện một cách rất hiệu quả. Nhưng trong đề tài này, chúng ta nghiên cứu ứng dụng Giải Thuật Di Truyền xếp Thời khoá biểu trong trường Đại học.

1. Lý do chọn đề tài

Trong xu hướng phát triển của xã hội ngày nay, có rất nhiều ngành khoa học mới ra đời. Trong đó có một số ngành khoa học ra đời trên cơ sở phân lập từ các ngành khoa học cổ điển, và một số ngành do sự tích hợp giữa các ngành khoa học khác.

Giải thuật di truyền (GA) là một trong những ngành khoa học ra đời từ sự tích hợp giữa sinh học và máy tính.

Giải thuật di truyền lấy ý tưởng từ quá trình tiến hoá tự nhiên, xuất phát từ một lớp các lời giải tiềm năng ban đầu, giải thuật di truyền tiến hành tìm kiếm trên không gian lời giải bằng cách xây dựng lớp lời giải mới tương đối tốt, cũng có thể là tốt nhất. Quá trình xây dựng lớp lời giải mới được tiến hành dựa trên việc chọn lọc, lai ghép, đột biến từ lớp lời giải ban đầu. Quần thể lời giải trải qua quá trình tiến hoá: ở mỗi thế hệ lại tái sinh các lời giải tương đối tốt hơn các thế hệ lời giải ban đầu, trong khi các lời giải "xấu" thì chết đi.

Bài toán lập lịch có thể được định nghĩa là một bài toán tìm kiếm chuỗi tối ưu để thực hiện một tập các hoạt động chịu tác động của một tập các ràng buộc cần phải được thỏa mãn. Người lập lịch thường cố gắng thử đến mức tối đa sự sử dụng các cá thể, máy móc và tối thiểu thời gian đòi hỏi để hoàn thành toàn bộ quá trình nhằm sắp xếp lịch tối ưu nhất. Vì thế bài toán lập lịch là một vấn đề rất khó để giải quyết.

Hiện nay có nhiều phương pháp tiếp cận để giải quyết bài toán này, như: trí tuệ nhân tạo, hệ chuyên gia, mạng Nơron, lập trình tính toán, lập trình động, tìm kiếm nhánh và đường biên, kỹ thuật mô phỏng, tìm kiếm Tabu và phương pháp nút cổ chai, ... Nhưng trong đề tài này sẽ tìm hiểu và tiếp cận giải thuật di truyền cho lớp bài toán lập lịch và cụ thể là bài toán lập thời khóa biểu học theo hệ tín chỉ cho trường đại học.

2. Mục đích nghiên cứu

Nghiên cứu, tìm hiểu giải thuật di truyền và ứng dụng giải thuật để giải quyết một số bài toán lập lịch, trên cơ sở đó tiếp cận để giải bài toán thời khóa biểu theo hệ tín chỉ và xây dựng ứng dụng hiệu quả và thiết thực.

3. Đối tượng và phạm vi nghiên cứu

- Tìm hiểu giải thuật di truyền.
- Tìm hiểu bài toán lập lịch và các hướng giải quyết truyền thống.
- Ứng dụng giải thuật di truyền vào bài toán lập thời khóa biểu.
- Xây dựng ứng dụng lập thời khóa biểu theo học chế tín chỉ cho trường đại học, cao đẳng.

4. Phương pháp nghiên cứu

Dựa trên các tài liệu thu thập từ nhiều nguồn (sách, báo, Internet, ...) tổng hợp, phân tích và trình bày lại theo sự hiểu biết của bản thân.

Mở rộng cách tiếp cận trước đây trên cơ sở phân tích đặc thù bài toán cần giải quyết để có những cải tiến hợp lý.

Nghiên cứu ứng dụng những kết quả nghiên cứu vào thực tế.

5. Ý nghĩa khoa học và thực tiễn của đề tài

5.1. Ý nghĩa khoa học

Thông qua đề tài sẽ hiểu rõ hơn về bài toán lập lịch và các phương pháp tiếp cận giải bài toán lập lịch, qua đó có sự so sánh và đánh giá các thuật toán.

Tìm hiểu sâu về giải thuật di truyền và ứng dụng vào bài toán thời khóa biểu nhằm có những cải tiến trong các bước của giải thuật di truyền với bài toán cụ thể như việc biểu diễn bài toán, cách chọn cá thể tốt, cách xây dựng hàm đánh giá, ...

5.2. Ý nghĩa thực tiễn

Bài toán lập thời khóa biểu là một bài toán có nhiều ứng dụng trong thực tế, đặc biệt là các trường đại học, cao đẳng đào tạo theo học chế tín chỉ. Ứng dụng giải thuật di truyền để giải bài toán thời khóa biểu là một hướng hy vọng giải quyết được bài toán thời khóa biểu.

Qua đề tài có thể xây dựng ứng dụng thực tế góp phần giảm thiểu thời gian và nguồn lực cho việc lập thời khóa biểu cho một cơ sở.

CHƯƠNG I – GIẢI THUẬT DI TRUYỀN

1.1. Tổng quan về giải thuật di truyền

1.1.1 Giới thiệu

Từ trước đến nay, trong các nghiên cứu và các ứng dụng trong tin học đã xuất hiện nhiều bài toán chưa tìm ra phương pháp giải nhanh và hợp lý, phần lớn đó là các bài toán tối ưu nảy sinh từ các ứng dụng trong thực tế. Để giải các bài toán này người ta thường tìm đến các giải thuật nhanh và hiệu quả mà kết quả chỉ thu được chỉ là xấp xỉ tối ưu. Trong nhiều trường hợp chúng sử dụng giải thuật xác suất, tuy không đảm bảo kết quả tối ưu nhưng có thể chọn các kết quả sao cho sai số có thể chấp nhận được, nhỏ như mong muốn. Trong giải thuật xác suất, việc giải bài toán quy về quá trình tìm kiếm trên không gian tập hợp các lời giải có thể. Tìm lời giải tốt nhất được hiểu là tìm lời giải tối ưu, với các bài toán có miền tìm kiếm nhỏ thì một số thuật toán tìm kiếm cổ điển có thể được sử dụng. Tuy nhiên đối với các bài toán có không gian tìm kiếm lớn như: bài toán lập lịch, thời khóa biểu, bài toán người du lịch ... thì phải sử dụng các kỹ thuật trí tuệ nhân tạo đặc biệt.

Từ các yêu cầu thực tế đặt ra, có nhiều vấn đề cần được giải quyết, cho đến những năm 1950 và 1960 A. S. Fraser là người tiên phong nêu lên sự tương đồng giữa sự tiến hóa của sinh vật và chương trình tin học giả tưởng về Genetic Algorithm. Tuy nhiên, chính John Henry Holland là người triển khai ý tưởng và phương thức giải quyết vấn đề dựa theo quá trình tiến hóa của sinh vật. Từ những bài giảng, bài báo cáo của mình, ông cùng các đồng nghiệp đã đúc kết các ý tưởng vào trong cuốn sách đầu tay "Adaptation in Natural and Altificial Systems", xuất bản năm 1975. Lần đầu tiên Holland nghiên cứu giải thuật này chúng hoàn toàn không có tên, do nguồn gốc của phương pháp này là từ các gen di truyền nên Holland đã đặt tên là giải thuật di truyền.

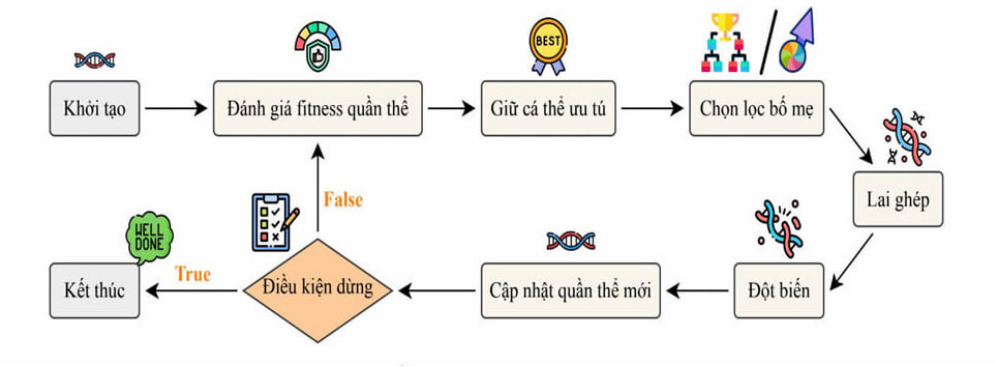
Giải thuật di truyền được hình thành từ đó, giải thuật di truyền hay thuật toán tiến hóa nói chung được hình thành dựa trên quan niệm cho rằng, quá trình tiến hóa tự nhiên là hoàn hảo nhất, hợp lý nhất và tự nó mang tính tối ưu. Quan niệm này được xem như là một tiên đề đúng không chứng minh được, nhưng phù hợp với thực tế khách quan. Quá trình tiến hóa thể hiện tính tối ưu ở chỗ thế hệ sau bao giờ cũng tốt hơn, phát trển hơn, hoàn thiện hơn thế hệ trước. Tiến hóa tự nhiên được duy trì nhờ hai quá trình cơ bản là: sinh sản và chọn lọc tự nhiên. Xuyên suốt quá trình tiến hóa tự nhiên, các thế hệ mới luôn được sinh ra để bổ sung, thay thế thế hệ cũ. Cá thể nào phát triển, thích ứng với môi trường sẽ tồn tại, cá thể nào không thích ứng sẽ bị đào thải. Sự thay đổi môi trường là động lực thúc đẩy quá trình tiến hóa. Ngược lại quá trình tiến hoá cũng tác động lại góp phần thay đổi môi trường.

Trong lĩnh vực nghiên cứu giải thuật di truyền người ta dùng thuật ngữ vay mượn của di truyền học như: cá thể, nhiễm sắc thể, gen, quần thể, độ thích nghi, chọn lọc, lai ghép, đột biến, v.v... Trong đó cá thể (individual, genotypes, structure) biểu diễn một lời giải, giải pháp của bài toán, không giống như trong tự nhiên một cá thể có thể có nhiều nhiễm sắc thể, ở đây chúng ta quy ước mỗi cá thể chỉ có một nhiễm sắc thể (chromosome). Các nhiễm sắc thể có thể là một chuỗi tuyến tính, trong nhiễm sắc thể có thể có các đơn vị nhỏ hơn đó

là gen. Mỗi gen đại diện một thuộc tính, tính chất và có vị trí nhất định trong nhiễm sắc thể.

Quần thể (population) là một tập hợp hữu hạn xác định các cá thể, trong giải thuật di truyền, quần thể là một tập các cá thể biểu diễn một tập các lời giải. Các phép toán chọn lọc (selection), lai ghép (crossover), đột biến (mutation) được thực hiện trên quần thể để tạo ra một quần thể mới. Một bài toán được giải bằng giải thuật di truyền thông thường phải qua các bước sau:

- Biểu diễn lời giải của bài toán (hay nhiễm sắc thể) bằng chuỗi nhị phân, chuỗi ký tự, số thập phân, ...
- Khởi tạo quần thể ban đầu gồm N cá thể một cách ngẫu nhiên.
- Xây dựng hàm thích nghi làm tiêu chuẩn đánh giá các cá thể theo độ thích nghi của chúng.
- Xác định xác suất lai tạo, xác suất đột biến, ...



Hình 1.1: Pipeline Tổng quan các bước tính toán trong GA

1.2. Tìm hiểu chung về Genetic Algorithms

Genetic Algorithms (thuật giải di truyền) là một giải thuật mô phỏng theo quá trình chọn lọc tự nhiên, là kỹ thuật chung giúp giải quyết vấn đề bài toán bằng cách mô phỏng sự tiến hóa của con người hay của sinh vật nói chung (dựa trên thuyết tiến hóa muôn loài của Darwin) trong điều kiện qui định sẵn của môi trường. Lấy ý tưởng từ quá trình tiến hoá tự nhiên, xuất phát từ một lớp các lời giải tiềm năng ban đầu, GA tiến hành tìm kiếm trên không gian lời giải bằng cách xây dựng lớp lời giải mới tốt hơn (tối ưu hơn) lời giải cũ. Quá trình xây dựng lớp lời giải mới được tiến hành dựa trên việc chọn lọc, lai ghép, đột biến từ lớp lời giải ban đầu. Quần thể lời giải trải qua quá trình tiến hoá: ở mỗi thế hệ lại tái sinh các lời giải tương đối tốt, trong khi các lời giải "xấu" thì chết đi.

Trong GA, một tập các biến của bài toán đưa ra được mã hóa sang một chuỗi (hay một cấu trúc mã hóa khác) tương tự như một nhiễm sắc thể trong tự nhiên. Mỗi chuỗi bao gồm một giải pháp có thể của bài toán. Giải thuật di truyền sử dụng các toán tử được sinh ra bởi sự chọc lọc tự nhiên một quần thể các chuỗi nhị phân (hoặc các cấu trúc khác), mã hóa khoảng tham số trên mỗi thế hệ, khảo sát các phạm vi khác nhau của không gian tham số, và định hướng tìm kiếm đối với khoảng mà là xác suất cao để tìm kiếm sự thực hiện tốt hơn. Thuật toán di truyền gồm có bốn quy luật cơ bản là lai ghép, đột biến, sinh sản và chọn lọc tự nhiên.

Quá trình lai ghép (phép lai)

Quá trình này diễn ra bằng cách ghép một hay nhiều đoạn gen từ hai nhiễm sắc thể cha–mẹ để hình thành nhiễm sắc thể mới mang đặc tính của cả cha lẫn mẹ. Phép lai này có thể mô tả như sau:

- Chọn ngẫu nhiên hai hay nhiều cá thể trong quần thể. Giả sử chuỗi nhiễm sắc thể của cha và mẹ đều có chiều dài là m. Tìm điểm lai bằng cách tạo ngẫu nhiên một con số từ 1 đến $m - 1$. Như vậy, điểm lai này sẽ chia hai chuỗi nhiễm sắc thể cha–mẹ thành hai nhóm nhiễm sắc thể con là m_1 và m_2.
- Hai chuỗi nhiễm sắc thể con lúc này sẽ là $m_{11} + m_{22}$ và $m_{21} + m_{12}$.
- Đưa hai chuỗi nhiễm sắc thể con vào quần thể để tiếp tục tham gia quá trình tiến hóa.

Quá trình đột biến (phép đột biến)

Quá trình tiến hóa được gọi là quá trình đột biến khi một hoặc một số tính trạng của con không được thừa hưởng từ hai chuỗi nhiễm sắc thể cha–mẹ. Phép đột biến xảy ra với xác suất thấp hơn rất nhiều lần so với xác suất xảy ra phép lai. Phép đột biến có thể mô tả như sau:

- Chọn ngẫu nhiên một số k từ khoảng $1 \geq k \geq m$.
- Thay đổi giá trị của gen thứ k.
- Đưa nhiễm sắc thể con vào quần thể để tham gia quá trình tiến hóa tiếp theo.

Quá trình sinh sản và chọn lọc (phép tái sinh và phép chọn)

Phép tái sinh: là quá trình các cá thể được sao chép dựa trên độ thích nghi của nó. Độ thích nghi là một hàm được gán các giá trị thực cho các cá thể trong quần thể của nó. Phép tái sinh có thể mô phỏng như sau:

- Tính độ thích nghi của từng cá thể trong quần thể, lập bảng cộng dồn các giá trị thích nghi đó (theo thứ tự gán cho từng cá thể) ta được tổng độ thích nghi. Giả sử quần thể có n cá thể. Gọi độ thích nghi của cá thể thứ i là F_i, tổng dồn thứ i là F_t. Tổng độ thích nghi là F_m.
- Tạo số ngẫu nhiên F có giá trị trong đoạn từ 0 đến F_m.
- Chọn cá thể k đầu tiên thỏa mãn $F \geq F_t$ đưa vào quần thể của thế hệ mới.

Phép chọn: là quá trình loại bỏ các cá thể xấu và để lại những cá thể tốt. Phép chọn được mô tả như sau:

- Sắp xếp quần thể theo thứ tự độ thích nghi giảm dần.
- Loại bỏ các cá thể cuối dãy, chỉ để lại n cá thể tốt nhất.

1.3. Sự khác biệt của giải thuật di truyền và giải thuật khác

Để thấy rõ hơn sự khác biệt của giải thuật di truyền và các giải thuật khác, chúng ta xét bài toán đơn giản sau đây: tối ưu hoá hàm $y = f(x)$ trên khoảng xác định D.

Khi dùng phương pháp truyền thống có một số cách giải sau:

Phương pháp liệt kê: Duyệt tất cả các điểm nằm trong vùng khảo sát D để tìm ra điểm cực trị của nó. Phương pháp này không thích hợp khi dữ liệu đầu quá lớn. Trong trường hợp này miền D có lực lượng lớn hơn đếm được.

Phương pháp giải tích: Tìm điểm cực trị bằng cách giải tập các phương trình khi cho gradient bằng 0. Để xét được gradient phải tính đạo hàm của hàm số. Điều này không giải quyết được trong trường hợp hàm số không liên tục hoặc không có đạo hàm. Ngoài ra đối với hàm nhiều cực trị thì có thể phương pháp này bỏ mất cực trị, cực trị tìm được chỉ mang tính chất địa phương.

Phương pháp tìm kiếm ngẫu nhiên: Là phương pháp kết hợp giữa phương pháp tính toán giải tích và sơ đồ liệt kê. Tuy nhiên những việc làm ngẫu nhiên cùng với giải thuật tìm kiếm ngẫu nhiên cũng phải bị suy yếu bởi thiếu tính hiệu quả.

Đối với giải thuật di truyền, các thông số của bài toán tìm kiếm phải được mã hoá thành một chuỗi hữu hạn các ký tự trên một tập hữu hạn các ký tự. Chuỗi này tương tự như các chuỗi gen của các cơ thể sinh vật. Một cách đơn giản là chúng ta có thể mã hoá thành các chuỗi bit trên tập ký tự $\{0, 1\}$. Mỗi một chuỗi đại diện cho một điểm trong không gian tìm kiếm. Giải thuật di truyền xuất phát với một quần thể các chuỗi được khởi tạo một cách ngẫu nhiên sau đó sẽ sản sinh các quần thể tiếp theo, thông qua việc sử dụng lựa chọn ngẫu nhiên như một hàm chọn. Nhờ đó giải thuật di truyền tìm kiếm trên nhiều điểm song song có khả năng leo lên nhiều cực trị cùng một lúc. Thông qua các toán tử của mình, giải thuật trao đổi thông tin giữa các cực trị với nhau, từ đó làm giảm thiểu khả năng giải thuật kết thúc tại các cực trị địa phương và bỏ qua mất cực trị toàn cục.

Đây là các đặc trưng của giải thuật di truyền so với các phương pháp truyền thống:

- Giải thuật làm việc với sự mã hoá của tập thông số chứ không làm việc với các giá trị của các thông số.
- Giải thuật tìm kiếm từ một quần thể các điểm chứ không phải từ một điểm.
- Giải thuật chỉ sử dụng thông tin về các tiêu chuẩn tối ưu của hàm mục tiêu chứ không dùng các thông tin hỗ trợ nào khác.
- Giải thuật sử dụng các luật chuyển đổi mang tính xác suất chứ không phải là các luật chuyển đổi mang tính xác định.
- Giải thuật thường khó cài đặt, áp dụng. Tuy nhiên không phải lúc nào cũng cho lời giải chính xác. Một số giải thuật di truyền có thể cung cấp lời giải tiềm năng cho một bài toán xác định để người sử dụng lựa chọn.

1.4. Tính chất của giải thuật di truyền

Giải thuật di truyền lập luận có tính chất ngẫu nhiên để tìm giải pháp tối ưu cho những vấn đề phức tạp. Tuy nhiên đây là hình thức ngẫu nhiên có hướng dẫn bởi giá trị hàm thích nghi, chính hàm thích nghi làm điều kiện chỉ đường cho giải thuật di truyền tìm ra lời giải tối ưu trong tập nhiều lời giải có thể.

Vấn đề thích hợp nhất của giải thuật di truyền là tìm điều kiện tối ưu, tối ưu ở đây không phải là tối ưu tuyệt đối mà có thể là tương đối tối ưu trong điều kiện và thời gian cho phép. Trong giải thuật di truyền một trong những vấn đề quan trọng nhất đó là xây dựng hàm thích nghi phải có sự liên hệ trực tiếp với vấn đề cần giải nếu không lời giải sẽ không ý nghĩa. Giải thuật di truyền (GA) thuộc một nhánh trong trí tuệ nhân tạo, giải thuật di truyền lập luận dựa theo sự tiến hóa và xét vấn đề ở mức của gen và nhiễm sắc thể, khác với mạng Nơron dựa trên kinh nghiệm và cách giải quyết vấn đề mà bộ óc con người thường dùng.

1.5. Các thành phần trong giải thuật di truyền

Giải thuật di truyền sử dụng các thuật ngữ vay mượn của di truyền học. Ta có thể nói về các cá thể (hay kiểu gen, cấu trúc), trong một quần thể; những cá thể này cũng còn được gọi là các chuỗi hay các nhiễm sắc thể, các phép toán trong di truyền học như: lai ghép, đột biến, môi trường chọn lọc tự nhiên. Để hiểu rõ các thành phần trong giải thuật di truyền chúng ta tìm hiểu các vấn đề sau.

1.5.1 Biểu diễn nhiễm sắc thể

Một nhiễm sắc thể biểu diễn một giải pháp, lời giải của bài toán. Một nhiễm sắc thể có thể chứa nhiều gen khác nhau để quy định một hay nhiều tính trạng nào đó. Vì thế khi vận dụng giải thuật vào giải bài toán thì phải có biểu diễn phù hợp để nâng cao tính hiệu quả của giải thuật. Có nhiều phương pháp biểu diễn nhiễm sắc thể như: biểu diễn nhị phân, biểu diễn sử dụng hoán vị, biểu diễn bằng giá trị, biểu diễn theo cấu trúc, ...

1.5.1.1 Biểu diễn nhị phân

Biểu diễn nhị phân là cách biểu diễn đơn giản và thông dụng nhất, mỗi nhiễm sắc thể được biểu diễn bằng chuỗi nhị phân, mỗi bit miêu tả đặc tính của nhiễm sắc thể. Biểu diễn nhị phân thường được sử dụng trong các bài toán tối ưu hàm một hay nhiều biến. Khi đó mỗi chuỗi nhị phân sẽ biểu diễn một hay nhiều giá trị của biến, việc mã hóa theo phương pháp này rất thuận lợi trong cách biểu diễn cũng như việc vận dụng các phép toán của giải thuật di truyền như lai tạo, đột biến. Tuy nhiên nếu các bài toán có miền giá trị lớn thì chiều dài của chuỗi lớn sẽ dẫn đến giảm tính hiệu quả của thuật toán.

Ví dụ: hai nhiễm sắc thể được biểu diễn thành chuỗi nhị phân như sau:

- Nhiễm sắc thể 1: `10101000000111101010`
- Nhiễm sắc thể 2: `11100111001101000111`

Để làm rõ phương pháp biểu diễn nhiễm sắc thể bằng chuỗi nhị phân, chúng ta xét ví dụ: giả sử, ta muốn tìm giá trị cực đại của hàm k biến

$$f(x_1, x_2, \ldots, x_k) : \mathbb{R}^k \to \mathbb{R}.$$

Giả sử mỗi biến x_i có thể nhận giá trị trong miền $D_i = [a_i, b_i] \subset \mathbb{R}$ và $f(x_1, x_2, \ldots, x_k) > 0$, $x_i \in D_i$, chúng ta cần tối ưu hóa hàm f với độ chính xác cho trước, giả sử cần 6 số lẻ đối với các giá trị các biến. Rõ ràng để đạt được độ chính xác như vậy mỗi miền D_i được phân cắt thành $(b_i - a_i) \cdot 10^6$ miền con bằng nhau. Gọi m_i là số nguyên nhỏ nhất sao cho

$$(b_i - a_i) \cdot 10^6 \leq 2^{m_i}.$$

Như vậy, mỗi biến x_i được biểu diễn bằng một chuỗi nhị phân có chiều dài m_i. Biểu diễn như trên rõ ràng thỏa mãn điều kiện về độ chính xác yêu cầu. Công thức sau tính giá trị thập phân của mỗi chuỗi nhị phân biểu diễn x_i:

$$x_i = a_i + \frac{b_i - a_i}{2^{m_i} - 1} \text{decimal(chuỗi nhị phân)}.$$

Trong đó, decimal(chuỗi nhị phân) cho biết giá trị thập phân của chuỗi nhị phân đó. Như vậy mỗi nhiễm sắc thể (hay lời giải) được biểu diễn bằng chuỗi nhị phân có chiều dài

$$m = \sum_{i=1}^{k} m_i,$$

m_1 bit đầu tiên biểu diễn các giá trị trong khoảng $[a_1, b_1]$; m_2 bit kế tiếp biểu diễn giá trị trong khoảng $[a_2, b_2]$; nhóm m_k bit cuối cùng biểu diễn giá trị trong khoảng $[a_k, b_k]$. Như vậy tương ứng với thứ tự và giá trị của mỗi bit m_i chúng ta sẽ trình bày được giá trị của mỗi lời giải bài toán.

Một ví dụ kinh điển như bài toán ba lô: có n đồ vật với trọng lượng và giá trị cho trước, và một ba lô có trọng lượng đã biết. Hãy chọn ra các đồ vật cho vào ba lô sao cho trọng lượng không được vượt qua giới hạn, mà tổng giá trị các đồ vật trong túi là lớn nhất. Lưu ý các đồ vật chỉ được chọn một lần (chọn hay không chọn). Để biểu diễn một cách xếp đồ vào túi, ta đánh số các đồ vật từ 1 đến n, và sử dụng một chuỗi nhị phân có độ dài bằng số đồ vật, mỗi bit tương ứng với các đồ vật sẽ có hai giá trị: giá trị 0 nếu đồ vật đó không được cho vào túi và giá trị 1 nếu đồ vật được cho vào túi.

Như vậy, biểu diễn nhị phân truyền thống có một số bất lợi khi áp dụng giải thuật di truyền giải các bài toán cần độ chính xác cao, nhưng đối với các bài toán không gian với số chiều lớn, thì chiều dài của vectơ nhị phân lớn thì giải thuật sẽ làm việc kém hiệu quả.

1.5.1.2 Biểu diễn sử dụng hoán vị

Mã hóa hoán vị phù hợp cho các bài toán liên quan đến trình tự, đối với loại cách biểu diễn này, tương ứng với mỗi cách hoán vị các gen trong nhiễm sắc thể cho ta một lời giải của bài toán. Biểu diễn loại này phù hợp với các bài toán như bài toán người du lịch, bài toán lập lịch, ... với mỗi giải pháp là một chuỗi các số biểu diễn một thứ tự.

Ví dụ:

- Nhiễm sắc thể 1: 1 5 4 3 2 6 7 9 8
- Nhiễm sắc thể 2: 9 1 7 3 8 5 6 4 2

Trong bài toán người du lịch, để biểu diễn một cách đi của người du lịch thì dùng một nhiễm sắc thể mà trình tự các số trong chuỗi cho biết thứ tự các thành phố mà người du lịch đi qua. Ví dụ điển hình của phương pháp biểu diễn sử dụng hoán vị là bài toán người du lịch (Travelling Salesman Problem – TSP).

Bài toán được mô tả như sau: một du khách muốn thăm mọi thành phố anh quan tâm; mỗi thành phố thăm qua đúng một lần; rồi trở về điểm khởi hành. Biết trước chi phí di chuyển giữa hai thành phố bất kỳ. Hãy xây dựng một lộ trình thỏa các điều kiện trên với tổng chi phí nhỏ nhất.

Nếu sử dụng biểu diễn nhị phân cho bài toán người du lịch có n thành phố, mỗi thành phố phải được đánh mã bằng một chuỗi gồm $\lceil \log_2(n) \rceil$ bit; và một nhiễm sắc thể là một chuỗi gồm $n \cdot \lceil \log_2(n) \rceil$ bit. Trong quá trình đột biến hay lai ghép có thể tạo ra một lộ trình không thỏa mãn điều kiện của bài toán và có thể xảy ra trường hợp thăm một thành phố

hai lần. Vấn đề này sẽ gây khó khăn trong quá trình vận dụng giải thuật, thay vì biểu diễn nhị phân ta sử dụng phương pháp biểu diễn bằng hoán vị, chúng ta đánh số các thành phố dùng một vectơ nguyên để biểu diễn nhiễm sắc thể lộ trình. Với cách biểu diễn này, một vectơ các thành phần nguyên

$$v = \langle i_1, i_2, \ldots, i_n \rangle$$

biểu diễn một lộ trình từ $i_1 \to i_2 \to \ldots \to i_{n-1} \to i_n \to i_1$, như vậy mỗi giải pháp là một cách hoán vị của vectơ $\langle 1, 2, 3, \ldots, n \rangle$.

1.5.1.3 Biểu diễn bằng giá trị

Với các bài toán phức tạp nhiều đối tượng thì mã hóa lời giải của bài toán bằng nhị phân hay hoán vị phức tạp hoặc không hiệu quả. Phương pháp mã hoá trực tiếp theo giá trị có thể được dùng trong các bài toán sử dụng giá trị phức tạp, như trong số thực, đối tượng hay các ký tự, ... Trong đó, mỗi nhiễm sắc thể là một chuỗi các giá trị, các giá trị có thể là bất cứ cái gì liên quan đến bài toán, từ số nguyên, số thực, kí tự cho đến các đối tượng phức tạp hơn.

Ví dụ:

- Nhiễm sắc thể 1: 1.23 5.32 0.34 2.98 3.54
- Nhiễm sắc thể 2: `(back), (back), (right), (forward), (left)`

Với cách biểu diễn bằng phương pháp này, khi vận dụng các phép toán của giải thuật di truyền có thể gặp phức tạp trong việc lai ghép, đột biến, vì vậy thường phải có một thuật toán sửa chữa phù hợp với từng yêu cầu từng loại bài toán.

1.5.2 Khởi tạo quần thể ban đầu

Trong phần trên đã trình bày một số phương pháp biểu diễn lời giải của bài toán theo hướng tiếp cận giải thuật di truyền, một trong các yếu tố ban đầu trong giải thuật di truyền đó là khởi tạo một quần thể cá thể (nhiễm sắc thể). Khởi tạo quần thể là khởi tạo ngẫu nhiên một tập các cá thể hay nhiễm sắc thể. Tuy nhiên dựa vào tri thức bài toán hay vận dụng lý thuyết xác suất mà ta có thể lựa chọn cách khởi tạo thích hợp. Nếu cách khởi tạo quần thể ban đầu tốt thì sẽ nâng cao hiệu quả của giải thuật và góp phần làm tăng nhanh tính hội tụ đến giá trị tối ưu của bài toán.

1.5.3 Đánh giá cá thể

Như đã đề cập ở trên, giải thuật di truyền thực hiện tiến trình tìm kiếm lời giải tối ưu theo nhiều hướng, bằng cách duy trì một quần thể các lời giải và thúc đẩy quá trình hình thành và trao đổi thông tin giữa các lời giải. Quần thể trải qua tiến trình tiến hoá: ở mỗi thế hệ lại tái sinh các lời giải tương đối "tốt", trong khi các lời giải tương đối "xấu" thì chết đi. Để phân biệt tính "tốt" của các lời giải khác nhau người ta sử dụng hàm mục tiêu để đóng vai trò môi trường trong quá trình chọn lọc cá thể.

Vì vậy trong giải thuật di truyền hàm đánh giá đóng vai trò làm môi trường chọn lọc cá thể, tuỳ thuộc vào bài toán mà chúng ta xây dựng hàm đánh giá phù hợp.

1.5.4 Phương pháp chọn lọc

Trong một quần thể nếu cá thể nào thích nghi với môi trường sẽ tồn tại và cá thể kém thích nghi sẽ bị đào thải, phù hợp với quá trình chọn lọc tự nhiên. Trong giải thuật di truyền quá

trình chọn lọc tuỳ thuộc vào hàm mục tiêu, với những cá thể nào có hàm mục tiêu cao sẽ đại diện cho những cá thể tốt, thích nghi với môi trường và có xác suất chọn lọc lớn. Toán tử này có thể được xem như là quá trình chọn lọc trong tự nhiên. Chúng ta tìm hiểu một số phương pháp chọn lọc.

1.5.4.1 Chọn lọc tỷ lệ :

Đây là phương pháp chọn lọc đơn giản nhất, ở đây mỗi cá thể trong quần thể chiếm một tỷ lệ trong vòng tròn (Roulette), có độ rộng tỷ lệ với giá trị hàm mục tiêu của cá thể (nhiễm sắc thể). Với mỗi lần quay vòng tròn Roulette chúng ta nhận được một cá thể và coi như đó là cách lựa chọn cá thể cho việc lai tạo.

Các bước thực hiện:

Tính độ thích nghi $\text{eval}(v_i)$ của mỗi nhiễm sắc thể v_i $(i = 1 \dots$

$$\text{eval}(v_i) = \frac{f(v_i)}{\sum_{i=1}^{\text{kích thước quần thể}} f(v_i)}$$

Tìm tổng giá trị thích nghi của quần thể

$$F = \sum_{i=1}^{\text{kích thước quần thể}} \text{eval}(v_i)$$

Tính xác suất chọn P_i cho mỗi nhiễm sắc thể v_i

$$p_i = \frac{\text{eval}(v_i)}{\sum_{i=1}^{\text{kích thước quần thể}} \text{eval}(v_i)}$$

Tính xác suất tích luỹ p_i cho mỗi nhiễm sắc thể P_i

$$q_i = \sum_{j=1}^{i} p_j$$

trong đó:

- *pop_ size*: kích thước của một quần thể đang xét,
- $\text{eval}(v_i)$: hàm đánh giá độ thích nghi của cá thể v_i.

Quá trình chọn lọc được thực hiện bằng cách quay bánh xe Roulette pop_size lần. Mỗi lần chọn một nhiễm sắc thể từ quần thể cũ vào quần thể mới theo cách sau:

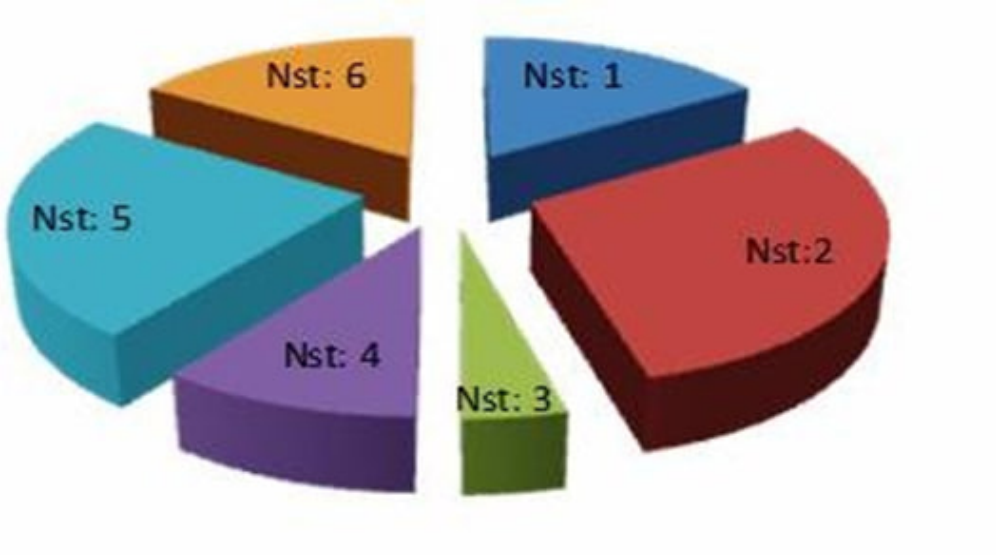
- Sinh ngẫu nhiên một số r trong khoảng $[0, 1]$.
- Nếu $r < q_1$ thì chọn nhiễm sắc thể đầu tiên (v_1), ngược lại chọn nhiễm sắc thể thứ i, v_i $(2 \leq i \leq \text{pop\_size})$ sao cho $q_{i-1} < r \leq q_i$.

Ví dụ: giả sử xét một quần thể ban đầu gồm 6 cá thể (nhiễm sắc thể), tổng giá trị của hàm mục tiêu là 50, và giá trị thích nghi được tính bằng cách chuyển chuỗi mã hoá nhị phân sang thập phân (ví dụ: $01110_2 = 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 14$) như bảng 1.1.

Bảng 1.1: Các nhiễm sắc thể và các giá trị thích nghi

Nhiễm sắc thể	Chuỗi mã hoá	Giá trị thích nghi	Xác suất chọn	Vị trí xác suất
1	01000	8	0.16	0.16
2	01111	15	0.30	0.46
3	00010	2	0.04	0.50
4	00101	5	0.10	0.60
5	01100	12	0.24	0.84
6	01000	8	0.16	1.00

Bánh xe trọng số được thể hiện ở hình 1.2:



Hình 1.2: Ví dụ bánh xe trọng số

Sau đó sẽ khởi tạo các số ngẫu nhiên $x \in [0, 1]$ tương ứng với việc quay vòng tròn bánh xe, đối với mỗi giá trị của x, kỹ thuật chọn lựa trên vòng tròn bánh xe sẽ được áp dụng để chọn một chuỗi nhiễm sắc thể với giá trị hàm thích nghi lớn hơn hay bằng x.

Bảng 1.2 ví dụ cách chọn các cá thể cho một quần thể mới dựa vào giá trị thích nghi ở bảng 1.1 và dãy các số thuộc $[0, 1]$ được sinh ngẫu nhiên.

Bảng 1.2: Ví dụ quần thể mới được chọn

Số ngẫu nhiên	0.55	0.70	0.10	0.80	0.95	0.40
NST chọn	4	5	1	5	6	2

Qua ví dụ ta nhận thấy khả năng của cá thể có giá trị thích nghi lớn thì sẽ được chọn nhiều hơn. Nhưng cũng có thể xảy ra trường hợp một nhiễm sắc thể có thể được chọn nhiều lần. Điều này phù hợp với lý thuyết sơ đồ các nhiễm sắc thể tốt nhất có nhiều bản sao hơn, các nhiễm sắc thể trung bình không thay đổi, các cá thể kém nhất thì chết đi.

1.5.4.2 Chọn lọc xếp hạng

Cơ chế thực hiện chọn lọc xếp hạng được miêu tả như sau:

- Sắp xếp các nhiễm sắc thể trong quần thể theo giá trị thích nghi từ thấp đến cao.

- Đặt lại độ thích nghi cho quần thể đã sắp xếp theo kiểu: nhiễm sắc thể thứ nhất có độ thích nghi là 1, nhiễm sắc thể thứ hai có độ thích nghi là 2, ..., nhiễm sắc thể thứ pop_size có độ thích nghi là pop_size (pop_size: kích thước của quần thể đang xét).

Theo phương pháp này việc một nhiễm sắc thể được chọn nhiều lần như trong lựa chọn theo kiểu bánh xe Roulette đã giảm đi. Nhưng nó có thể dẫn đến sự hội tụ chậm và nhiễm sắc thể có độ thích nghi cao cũng không khác mấy so với các nhiễm sắc thể khác.

1.5.4.3 Chọn lọc cạnh tranh

Mỗi lần chọn lọc ta tiến hành chọn ngẫu nhiên m cá thể từ quần thể hiện tại, cá thể tốt nhất trong m cá thể trên được đưa vào quần thể mới. Tiến hành thực hiện N (kích thước quần thể mới) bước chọn như vậy ta thu được một quần thể mới. Trong đó m được gọi là kích thước cạnh tranh.

1.5.5 Phương pháp lai ghép

Lai ghép trong tự nhiên là sự kết hợp các tính trạng của bố mẹ để tạo ra các thế hệ con. Trong giải thuật di truyền, lai ghép được coi là sự trao đổi thông tin giữa các lời giải, tổ hợp các tính chất trong hai lời giải của cha mẹ để sinh ra một lời giải mới có đặc tính mong muốn là tốt hơn thế hệ bố mẹ. Trong giải thuật di truyền có rất nhiều phương pháp lai ghép được sử dụng khác nhau như: lai ghép một điểm (One Point Crossover), lai ghép đa điểm (Multi Point Crossover), ánh xạ từng phần (Partial Mapped Crossover – PMX), lai ghép có trật tự (Order Crossover – OX), lai ghép dựa trên vị trí (Position Based Crossover – PBX), lai ghép thứ tự tuyến tính (Linear Order Crossover – LOX), ... Tuỳ thuộc vào từng bài toán, từng cách biểu diễn nhiễm sắc thể mà chúng ta sẽ sử dụng phương pháp lai ghép phù hợp. Chúng ta sẽ tìm hiểu một số phương pháp lai ghép phổ biến sau:

1.5.5.1 Lai ghép một điểm

Lai ghép một điểm là một trong những phương pháp lai ghép đơn giản nhất, nó được sử dụng cho hầu hết tất cả các phương pháp biểu diễn. Với cặp bố, mẹ X, Y là các vectơ n chiều, toán tử lai ghép một điểm sẽ chọn ngẫu nhiên một vị trí k ($1 \leq k \leq n$).

Cha:

$$X = (x_1, x_2, x_3, \ldots, x_k, \ldots, x_{n-1}, x_n),$$

Mẹ:

$$Y = (y_1, y_2, y_3, \ldots, y_k, \ldots, y_{n-1}, y_n).$$

Con 1:

$$X' = (y_1, y_2, y_3, \ldots, y_k, \ldots, x_{n-1}, x_n),$$

Con 2:

$$Y' = (x_1, x_2, x_3, \ldots, x_k, \ldots, y_{n-1}, y_n).$$

Với phương pháp lai ghép một điểm thường được sử dụng trong cách biểu diễn nhiễm sắc thể là chuỗi nhị phân. Ví dụ: lai ghép với điểm $k = 10$:

$$\begin{aligned}
\text{Cha: } X &= 1101000101|1101011100,\\
\text{Mẹ: } Y &= 1110010111|0000011101,\\
\text{Con 1: } X' &= 1101000101|11100010111,\\
\text{Con 2: } Y' &= 1110010111|00000011101.
\end{aligned}$$

1.5.5.2 Lai ghép đa điểm

Lai ghép đa điểm là sự mở rộng của lai ghép một điểm, chọn ngẫu nhiên k điểm $j_1, j_2, \ldots, j_k$ từ bố X, mẹ Y, sao cho $1 \leq j_1 < j_2 < \ldots < j_k < n$, để tạo ra thế hệ con X', Y', bằng cách đánh số các đoạn $[j_i, j_{i+1}]$ từ 0. Khi đó: cá thể con X' được tạo ra bằng cách chọn lần lượt các đoạn gen cho cá thể con X' như sau: x'_i lấy bằng x_i tại những đoạn có số hiệu chẵn và lấy y_i tại những đoạn có số hiệu lẻ. Tương tự cho cá thể con Y' được tạo ra bằng cách chọn gen y'_i lấy bằng y_i tại những đoạn có số hiệu lẻ và bằng x_i tại những đoạn có số hiệu chẵn.

Ví dụ: giả sử chọn giá trị $k = 4$, tương ứng với các điểm $5, 9, 15, 18$:

$$\begin{aligned}
\text{Cha: } X &= 11010|0010|111010|111|00,\\
\text{Mẹ: } Y &= 11100|1011|100000|101|01,\\
\text{Con 1: } X' &= 11010|1011|111010|101|00,\\
\text{Con 2: } Y' &= 11100|0010|100000|111|01.
\end{aligned}$$

1.5.5.3 Lai ghép ánh xạ từng phần

Lai ghép ánh xạ từng phần do Golberg và Lingde đề nghị [10], phương pháp tạo ra con mới bằng cách chọn một chuỗi con từ cha, mẹ đồng thời bảo toàn thứ tự và vị trí của tối đa cá thể của cha, mẹ kia. Một chuỗi con được chọn bằng cách chọn hai điểm cắt ngẫu nhiên, được dùng làm hai giới hạn cho các thao tác hoán vị và kết hợp với một thuật toán sửa chữa đặc biệt để giải quyết những vị trí bất hợp lệ. Thuật toán gồm các bước sau:

- Chọn hai điểm cắt nhau cùng với một chuỗi một cách ngẫu nhiên. Chuỗi con được định nghĩa bởi hai điểm cắt được gọi là ánh xạ từng phần.
- Trao đổi hai chuỗi con giữa hai nhiễm sắc thể cha, mẹ để tạo ra nhiễm sắc thể con.
- Xác định ánh xạ giữa các thành phần ánh xạ.
- Hợp thức cá thể con tương ứng với các quan hệ ánh xạ.

Ví dụ minh hoạ cho phương pháp: trong bài toán người du lịch gồm 9 thành phố, bài toán được biểu diễn bằng phương pháp hoán vị các chu trình của các thành phố.

$$\begin{aligned}
&\text{Cá thể cha: } 9\ 3\ 1 \mid 2\ 4\ 7\ 5 \mid 6\ 8,\\
&\text{Cá thể mẹ: } 1\ 7\ 3 \mid 6\ 4\ 8\ 9 \mid 2\ 5.
\end{aligned}$$

Bước đầu tiên là hoán vị giữa hai đoạn gen được chọn trong cá thể bố, mẹ. Trong đó ký hiệu x là những gen chưa được xác định. Thực hiện tạo ra các ánh xạ giữa các thành phần trong hai đoạn gen được chọn: $6 \leftrightarrow 2$, $8 \leftrightarrow 7$, $9 \leftrightarrow 5$.

$$\begin{aligned}
&\text{Cá thể con 1: } x\ x\ x \mid 6\ 4\ 8\ 9 \mid x\ x,\\
&\text{Cá thể con 2: } x\ x\ x \mid 2\ 4\ 7\ 5 \mid x\ x.
\end{aligned}$$

Cuối cùng điều chỉnh các quan hệ ánh xạ và bổ sung các thành phố trong hai cá thể con mà không có xung đột:

$$\begin{aligned}
&\text{Cá thể con 1: } 5\ 3\ 1 \mid 6\ 4\ 8\ 9 \mid 2\ 7,\\
&\text{Cá thể con 2: } 1\ 8\ 3 \mid 2\ 4\ 7\ 5 \mid 6\ 9.
\end{aligned}$$

Lai ánh xạ từng phần khai thác các điểm tương đồng quan trọng trong giá trị và xếp bậc đồng thời khi được sử dụng với một kế hoạch sinh sản phù hợp.

1.5.5.4 Lai ghép có trật tự

Lai ghép có trật tự do Davis đề nghị tạo ra các con bằng cách chọn một chuỗi con từ một cha, mẹ và bảo tồn thứ tự tương đối của cha, mẹ kia. Lai ghép có trật tự có thể được xem là một biến thể của lai ghép ánh xạ từng phần và sử dụng bổ sung một thuật toán sửa chữa khác. Lai ghép có trật tự (OX) có thể thực hiện thông qua các bước sau:

- Chọn ngẫu nhiên một chuỗi con từ cá thể cha, mẹ.
- Tạo ra các cá thể con bằng cách sao chép chuỗi con tương ứng vào những vị trí tương ứng như trong cá thể cha, mẹ. Các vị trí khác xem như chưa biết.
- Tạo ra một trình tự bắt đầu từ điểm cắt của cha (mẹ) được chọn và xoá các gen đã được chọn ở mẹ (cha).
- Cuối cùng bổ sung các gen vào cá thể được chọn bắt đầu điểm cắt.

Ví dụ:

$$\text{Cá thể cha: } 9\ 3 \mid 8\ 5\ 7\ 1 \mid 6\ 4\ 2,$$
$$\text{Cá thể mẹ: } 3\ 5 \mid 2\ 6\ 1\ 4 \mid 8\ 7\ 9.$$

Phân đoạn từ hai điểm cắt để tạo ra cá thể con như sau:

$$\text{Con 1: } x\ x \mid 8\ 5\ 7\ 1 \mid x\ x\ x,$$
$$\text{Con 2: } x\ x \mid 2\ 6\ 1\ 4 \mid x\ x\ x.$$

Tạo ra một thứ tự bắt đầu từ điểm cắt, cá thể được chọn ở đây là cá thể cha: 6 – 4 – 2 – 9 – 3 – 8 – 5 – 7 – 1. Và xoá các gen đã có trong cá thể mẹ: 9 – 3 – 8 – 5 – 7.

Bổ sung các gen vào trong con 2 bắt đầu điểm cắt ta được:

$$\text{Con 2: } 5\ 7\ 2\ 6\ 1\ 4\ 9\ 3\ 8.$$

Tương tự với cá thể con 1 bắt đầu điểm cắt ta được:

$$\text{Con 1: } 6\ 4\ 8\ 5\ 7\ 1\ 9\ 3\ 2.$$

1.5.5.5 Lai ghép dựa trên vị trí

Lai ghép dựa trên vị trí thực chất là một loại lai ghép đồng nhất cho mã hoá theo định nghĩa đột biến kết hợp với một thủ tục sửa chữa. Toán tử lai ghép đồng nhất được đề nghị cho mã hoá chuỗi bit bởi Syswerda. Trước tiên nó sinh ngẫu nhiên một mặt nạ sau đó trao đổi các gen liên quan giữa các cá thể cha, mẹ dựa vào mặt nạ. Một mặt nạ lai ghép là một chuỗi nhị phân đơn giản có kích thước nhiễm sắc thể như nhau, sự tương đương của mỗi bit trong mặt nạ với mỗi bit của cá thể con, xác định cá thể cha, mẹ nào sẽ cung cấp bit đó. Ý tưởng của phương pháp lai ghép dựa trên vị trí là kết hợp sử dụng mặt nạ (nhị phân) làm tiêu chuẩn lựa chọn gen của bố mẹ. Với mỗi giá trị của mặt nạ, nếu mặt nạ có giá trị là 1 thì cá thể con sẽ nhận gen của cha, ngược lại là gen của mẹ.

Các bước thực hiện thuật toán như sau: giả sử nhiễm sắc thể cha, mẹ tương ứng X, Y và mặt nạ M sẽ tạo ra cá thể con X'.

Ví dụ:

$$\text{Cá thể cha: } 9\ 3\ 1\ 2\ 4\ 7\ 5\ 6\ 8,$$
$$\text{Cá thể mẹ: } 1\ 7\ 3\ 6\ 4\ 8\ 9\ 2\ 5,$$
$$\text{Mặt nạ: } M = 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0.$$

Thực hiện lai ghép tạo ra cá thể con bằng cách, với mỗi giá trị tương ứng của mặt nạ M, nếu $m[i] = 1$, thì cá thể con nhận gen của cá thể cha, ngược lại $m[i] = 0$ thì cá thể con nhận gen của cá thể mẹ. Trong quá trình thực hiện kết hợp với thuật toán sửa chữa để tránh các xung đột. Trong ví dụ ta thực hiện từng bước như sau:

- Giá trị $m[1] = 1$ tức gen đầu tiên của cá thể con X' nhận gen của cá thể cha, nếu trong cá thể con chưa nhận gen đó: 9 x x x x x x x.
- Giá trị $m[2] = 0$ gen thứ 2 của cá thể con X' nhận gen của cá thể mẹ, nếu trong cá thể con chưa tồn tại gen đó: 9 7 x x x x x x.
- Tương tự với các giá trị $m[i]$, ta nhận cá thể con X': 9 7 3 6 1 2 4 8 5.

1.5.5.6 Lai ghép thứ tự tuyến tính

Lai ghép thứ tự tuyến tính được phát triển như một sửa đổi của lai ghép dựa trên thứ tự. Lai ghép dựa trên thứ tự có khuynh hướng truyền những vị trí tương đối với các gen thay vì những vị trí tuyệt đối. Trong lai ghép thứ tự, nhiễm sắc thể được xem xét xoay vòng. Ví dụ như trong bài toán người du lịch, bài toán sắp xếp công việc của cửa hàng (job–shop). Vì lý do này, người ta phát triển một biến thể của OX gọi là lai ghép thứ tự tuyến tính (LOX) trong đó nhiễm sắc thể được xem xét tuyến tính thay vì xoay vòng. LOX làm việc như sau:

- Chọn ngẫu nhiên chuỗi con từ hai cá thể cha, mẹ.
- Xoá các gen đã xuất hiện ở vùng chọn ở cá thể cha, mẹ và đánh dấu các vị trí đó bằng ký tự x ở các cá thể.
- Dịch chuyển các ký tự x vào vùng chọn ở các cá thể cha, mẹ cho đến khi chúng gặp miền giao nhau.
- Thay thế chuỗi chọn từ cá thể cha vào cá thể mẹ và ngược lại.

Ví dụ:

$$\text{Cá thể cha: } 2\ 6\ 4\ 7\ 3\ 5\ 8\ 9\ 1,$$
$$\text{Cá thể mẹ: } 4\ 5\ 2\ 1\ 8\ 7\ 6\ 9\ 3.$$

Các bước được thực hiện như sau:

$$\text{Cá thể cha: } 2\ 6 \mid 4\ 7\ 3 \mid 5\ 8\ 9\ 1,$$
$$\text{Con 1: } x\ 6 \mid 4\ 7\ 3 \mid 5\ x\ 9\ x,$$
$$\text{Cá thể mẹ: } 4\ 5 \mid 2\ 1\ 8 \mid 7\ 6\ 9\ 3,$$
$$\text{Con 2: } x\ 5 \mid 2\ 1\ 8 \mid x\ 6\ 9\ x.$$

Sau đó dịch và điền:

$$\text{Con 1: } 6\ 4 \mid x\ x\ x \mid 7\ 3\ 5\ 9 \rightarrow 6\ 4 \mid 2\ 1\ 8 \mid 7\ 3\ 5\ 9,$$
$$\text{Con 2: } 5\ 2 \mid x\ x\ x \mid 1\ 8\ 6\ 9 \rightarrow 5\ 2 \mid 4\ 7\ 3 \mid 1\ 8\ 6\ 9.$$

1.5.5.7 Lai ghép có chu trình

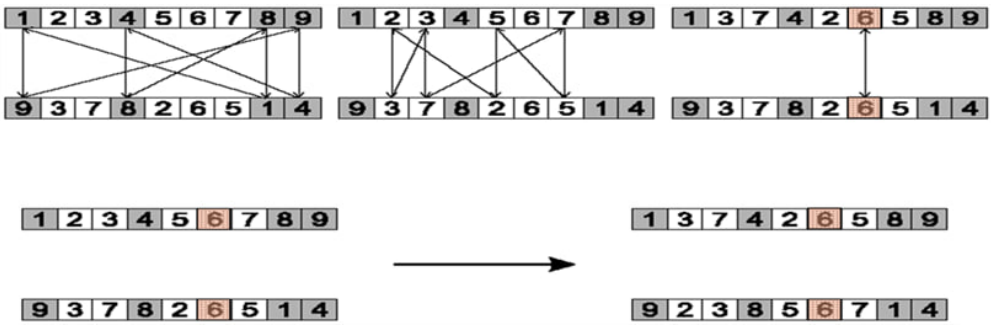
Lai ghép có chu trình do Oliver đề nghị xây dựng cá thể con theo cách mỗi vị trí của nó xuất phát từ một trong các cha, mẹ. Lai ghép có chu trình giống với lai ghép dựa trên vị trí, nó chọn một số gen từ cá thể cha hoặc mẹ và các gen còn lại được chọn từ cá thể cha hoặc mẹ khác. Điểm khác nhau so với lai ghép dựa trên vị trí là các gen không được chọn ngẫu nhiên mà chỉ với các gen được chọn mới xác định một chu trình tương ứng với những vị trí giữa các thể cha, mẹ. Cụ thể chúng ta biểu diễn lai ghép có chu trình làm việc như sau:

- Tìm một chu trình được xác định bởi những vị trí tương ứng của các ký hiệu giữa các cá thể cha, mẹ.
- Sao chép các gen trong chu trình vào cá thể con bởi những vị trí tương ứng trong một cá thể cha hoặc mẹ.
- Xác định các ký hiệu còn lại cho cá thể con bằng cách xoá những ký hiệu này bây giờ là một chu trình của cá thể cha mẹ khác.
- Điền cá thể con với các ký hiệu còn lại.

Ví dụ:

Cá thể cha: 1 2 3 4 5 6 7 8 9,

Cá thể mẹ: 9 3 7 8 2 6 5 1 4.



Hình 1.3: Ví dụ phương pháp lai ghép có chu trình

1.5.6 Toán tử đột biến

Phương pháp lai ghép sẽ tạo ra các cá thể con có sự thừa kế các thuộc tính của bố, mẹ. Nhưng đối với phương pháp đột biến thì quá trình sẽ có thể sinh ra cá thể con có thể không mang tính trạng của bố, mẹ. Đột biến có thể sinh cá thể con có thể tốt hơn hoặc xấu hơn cá thể bố mẹ của nó, xác suất đột biến xảy ra thấp hơn lai ghép và đột biến góp phần làm tăng quá trình hội tụ. Có nhiều phương pháp đột biến, tuỳ thuộc vào quá trình biểu diễn nhiễm sắc thể mà chúng ta vận dụng đột biến phù hợp. Chúng ta sẽ tìm hiểu một số phương pháp đột biến như: đột biến đảo ngược (Inversion Mutation), đột biến chèn (Insertion Mutation), đột biến thay thế (Displacement Mutation), đột biến tương hỗ (Reciprocal Exchange Mutation), đột biến chuyển dịch (Shift Mutation), ...

1.5.6.1 Đột biến đảo ngược:

Tương ứng với nhiễm sắc thể chọn, chọn ngẫu nhiên một đoạn trong nhiễm sắc thể và thực hiện hoán vị đảo ngược đoạn nhiễm sắc thể đó.

Ví dụ: giả sử vị trí chọn đột biến bắt đầu tại 4 có chiều dài 4 như sau:

Nhiễm sắc thể: 2 6 8 1 7 3 5 4 9,

Đảo đoạn: 2 6 8 5 3 7 1 4 9.

1.5.6.2 Đột biến chèn:

Phương pháp chọn ngẫu nhiên một gen ở vị trí bất kỳ trong nhiễm sắc thể và chèn vào vị trí ngẫu nhiên khác trong cùng nhiễm sắc thể.

Ví dụ:

Nhiễm sắc thể: 9 4 2 5 3 8 7 6 1,

Chèn: 9 4 2 8 5 3 7 6 1.

1.5.6.3 Đột biến thay thế:

Là trường hợp mở rộng của đột biến chèn, với đột biến chèn thì chỉ chọn một gen và chèn vào vị trí thích hợp, đột biến thay thế chọn ngẫu nhiên một đoạn gen và chèn vào vị trí tuỳ ý.

Ví dụ:

Nhiễm sắc thể: 9 4 2 5 3 8 7 6 1,

Thay thế đoạn: 9 5 3 8 7 4 2 6 1.

1.5.6.4 Đột biến tương hỗ:

Chọn ngẫu nhiên hai gen bất kỳ trong nhiễm sắc thể và hoán vị chúng trên cùng một nhiễm sắc thể.

Ví dụ:

Nhiễm sắc thể: 9 4 2 5 3 8 7 6 1,

Hoán vị: 9 6 2 5 3 8 7 4 1.

Đột biến chuyển dịch: Chọn ngẫu nhiên một gen và chuyển dịch gen được chọn sang trái hoặc sang phải.

Ví dụ:

Nhiễm sắc thể: 9 4 2 5 3 8 7 6 1,

Chuyển dịch sang trái: 9 4 5 2 3 8 7 6 1,

Chuyển dịch sang phải: 9 4 2 3 5 8 7 6 1.

1.5.7 Điều kiện dừng của giải thuật

Trong một số phiên bản về chương trình tiến hoá không phải mọi cá thể đều tiến hoá lại. Vài cá thể trong đó có khả năng vượt từ thế hệ này sang thế hệ khác mà không thay đổi gì cả, vì vậy phải xác định điều kiện dừng phù hợp. Tuỳ thuộc vào từng yêu cầu, hay thoả mãn điều kiện nào đó của các bài toán mà có các điều kiện dừng khác nhau.

Thông thường có một số phương pháp sau:

- Kết thúc theo kết quả, tức khi giá trị thích nghi của cá thể trong quần thể có giá trị sai số nhỏ hơn một giá trị cho trước thì dừng thuật toán.

- Kết thúc dựa trên số thế hệ, một số vấn đề dựa vào số thế hệ trong quần thể. Khi số thế hệ của quần thể được tạo ra bằng một giới hạn cho trước thì thuật toán dừng, mà không quan tâm đến chất lượng của cá thể trong quần thể như thế nào.
- Tính theo thời gian, thuật toán dừng khi phụ thuộc vào thời gian thực thi chương trình đã được quy định trước và thuật toán dừng.
- Kết hợp nhiều phương pháp khác nhau, giải thuật cũng có thể sử dụng kết hợp nhiều phương pháp khác nhau để giải quyết vấn đề.

1.5.8 Các tham số của giải thuật di truyền

1.5.8.1 Kích thước quần thể

Kích thước quần thể cho biết số lượng cá thể trong một quần thể (trong một thế hệ). Qua các nghiên cứu cũng như các thử nghiệm đã cho thấy nếu kích thước quần thể quá ít thì quá trình lai tạo sẽ ít và không gian tìm kiếm nhỏ, vì vậy có thể bỏ qua các lời giải tốt. Nhưng nếu kích thước quần thể quá lớn, tuy không gian tìm kiếm nhiều có khả năng đạt được kết quả tốt, nhưng tốc độ xử lý sẽ chậm, tốn nhiều tài nguyên và sẽ có ảnh hưởng đến giải thuật.

1.5.8.2 Xác suất lai ghép

Lai ghép được xem là tổ hợp các tính chất của cha mẹ để sinh ra cá thể mới có đặc tính mong muốn là tốt hơn thế hệ cha mẹ của nó, xác suất lai ghép cho biết việc lai ghép tạo ra thế hệ mới được thực hiện mức độ như thế nào. Nếu xác suất lai ghép là p_c, khi đó khả năng để một cá thể được lai ghép là p_c. Nếu không thực hiện lai ghép, con sinh ra sẽ giống hoàn toàn bố mẹ. Nếu được lai ghép, con sinh ra sẽ sở hữu các tính chất tốt của cả cha và mẹ.

1.5.8.3 Xác suất đột biến

Phép đột biến làm cho chất liệu di truyền thêm phong phú, hy vọng góp phần làm tăng nhanh quá trình hội tụ. Nếu xác suất đột biến là p_m, khi đó khả năng để mỗi gen của một nhiễm sắc thể bất kỳ bị đột biến là p_m. Toán tử đột biến có tác dụng ngăn ngừa giải thuật di truyền rơi vào tình trạng cực trị địa phương, tuy nhiên nếu thực hiện đột biến với xác suất quá cao sẽ biến giải thuật di truyền thành giải thuật tìm kiếm ngẫu nhiên.

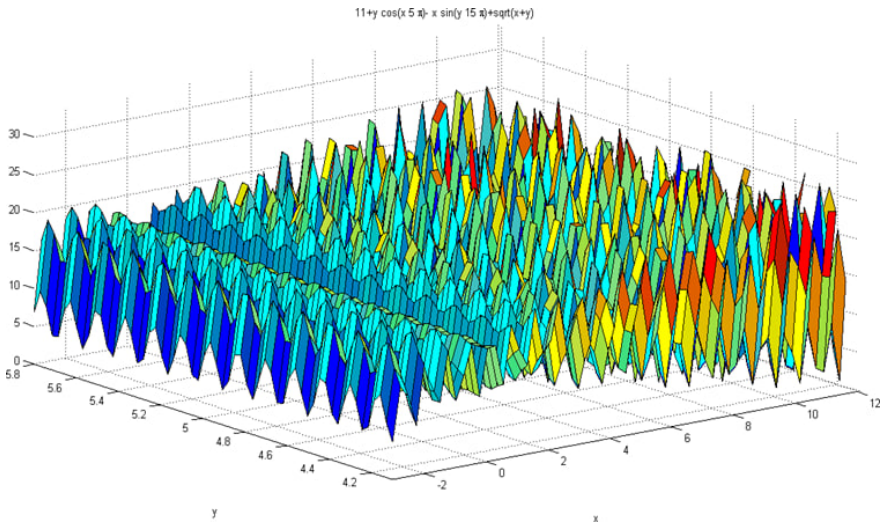
1.6. Ví dụ minh họa

Để hình dung cơ chế hoạt động của giải thuật di truyền chúng ta sẽ thảo luận các tính năng chính của giải thuật qua các ví dụ cụ thể. Trong thí dụ, ta áp dụng giải thuật vào bài toán tìm giá trị lớn nhất của hàm số.

Cho hàm số:

$$f(x,y) = 11 + y\cos(5\pi x) - x\sin(15\pi y) + \sqrt{x+y} \text{ với } -3.0 \le x \le 12.1$$

và $4.1 \le y \le 5.8$. Yêu cầu cần tìm giá trị cực đại của hàm số f(x,y).



Hình 1.4: Đồ thị của hàm số

1.6.1 Biểu diễn nhiễm sắc thể

Với các bài toán thuộc lớp bài toán tìm giá trị tối ưu thường được biểu diễn thành chuỗi nhị phân, một vectơ nhị phân v biểu diễn các giá trị thực của biến x, y. Chiều dài của vectơ v phụ thuộc vào độ chính xác cần có, trong bài toán này, ta tính chính xác đến 4 số lẻ đối với mỗi biến.

Miền của biến x có chiều dài là 15.1; điều kiện chính xác thuộc đoạn $[-3.0, 12.1]$ cần được chia thành các khoảng có kích thước bằng nhau, miền giá trị của x là:

$$12.1 - (-3.0) = 15.1$$

và với độ chính xác là 4 số lẻ, tức:

$$15.1 \times 10^4 = 151000 \text{ khoảng.}$$

Điều này có nghĩa cần 18 bit ($2^{17} < 151000 < 2^{19}$) để biểu diễn nhiễm sắc thể

$$v = (a_1, a_2, a_3, \ldots, a_{18}).$$

Miền giá trị của y có chiều dài 1.7, điều kiện chính xác thuộc đoạn $[4.1, 5.8]$ cần được chia thành các khoảng có kích thước bằng nhau, ít nhất 1.7×10^4 khoảng. Điều này có nghĩa cần 15 bit để biểu diễn nhiễm sắc thể ($2^{14} < 17000 < 2^{15}$).

Hay:

$$v = x' + y' = (a_1, a_2, a_3, \ldots, a_{18}) + (a_{19}, a_{20}, a_{21}, \ldots, a_{33}).$$

Như vậy tương ứng với bài toán này chúng ta cần một nhiễm sắc thể có chiều dài là 33 bit.

1.6.2 Hàm thích nghi

Chuyển đổi giá trị của vectơ nhị phân v thành giá trị thực x, y ta thực hiện như sau:

$$decimal(x') = (a_1, a_2, a_3, \ldots, a_{18})_2 = \left(\sum_{i=1}^{18} a_i \, 2^{18-i} \right)_{10}$$

$$decimal(y') = (a_{19}, a_{20}, a_{21}, \ldots, a_{33})_2 = \left(\sum_{i=19}^{33} a_i \, 2^{33-i} \right)_{10}$$

- Chuyển số nhị phân sang thập phân, tương ứng với x, y.
- Ánh xạ chuyển chuỗi nhị phân x', y' sang số thực tương ứng x, y trong đoạn $[-3.0, 12.1]$ và $[4.1, 5.8]$ như sau:

$$x = -3.0 + \frac{\text{v1}}{2^{18} - 1} \times 15.1, \qquad y = 4.1 + \frac{\text{v2}}{2^{15} - 1} \times 1.7,$$

với $-3.0, 4.1$ là các cận dưới; $15.1, 1.7$ là độ dài miền giá trị.

Ví dụ: một nhiễm sắc thể:

$$101101100001110101101100110111000,$$

với 18 bit đầu tiên (101101100001110101) biểu diễn:

$$x = -3.0 + \text{decimal}(101101100001110101_2) \times \frac{12.1}{2^{18} - 1} \approx 7.741937.$$

Và 15 bit kế tiếp (101100110111000), biểu diễn:

$$y = 4.1 + \text{decimal}(101100110111000_2) \times \frac{1.7}{2^{15} - 1} \approx 5.291624.$$

Như vậy nhiễm sắc thể:

$$101101100001110101101100110111000$$

tương ứng với $\langle x, y \rangle = \langle 7.741937, 5.291624 \rangle$.

Và độ thích nghi của cá thể này là:

$$f(7.741937, 5.291624) = 24.2850.$$

1.6.3 Khởi tạo quần thể

Bước kế tiếp trong giải thuật là khởi tạo quần thể ban đầu, trong ví dụ chúng ta khởi tạo quần thể có 10 cá thể.

Tất cả các bit trong cá thể trong quần thể chúng ta khởi tạo ngẫu nhiên:

$$\begin{aligned}
v_1 &= 010111100001111000101011010101000,\\
v_2 &= 010111100011111010101001010001011,\\
v_3 &= 010010101001101000101111010101001,\\
v_4 &= 010100101101101010101010010101101,\\
v_5 &= 110110101001100110101111010101011,\\
v_6 &= 010111101001001000111011010101100,\\
v_7 &= 101101100001110101101100110111000,\\
v_8 &= 110001101110110111010011010101111,\\
v_9 &= 110011001100000001011111011101010,\\
v_{10} &= 101001100001110101101100110111010.
\end{aligned}$$

Giải mã từng nhiễm sắc thể và tính giá trị thích nghi tương ứng với từng cá thể:

$$\begin{aligned}
\text{eval}(v_1) &= f(2.551464, 5.250938) = 8.27323,\\
\text{eval}(v_2) &= f(2.558953, 5.196307) = 10.07481,\\
\text{eval}(v_3) &= f(1.400343, 5.357244) = 8.25770,\\
\text{eval}(v_4) &= f(1.887081, 5.224634) = 10.88307,\\
\text{eval}(v_5) &= f(9.894010, 5.357347) = 5.44688,\\
\text{eval}(v_6) &= f(2.578192, 5.676158) = 9.64175,\\
\text{eval}(v_7) &= f(7.741936, 5.291613) = 18.51623,\\
\text{eval}(v_8) &= f(8.733730, 4.613782) = 22.26690,\\
\text{eval}(v_9) &= f(9.172946, 5.360615) = 10.89360,\\
\text{eval}(v_{10}) &= f(6.798180, 5.291717) = 26.05350.
\end{aligned}$$

Từ trên ta nhận thấy nhiễm sắc thể v_{10} có giá trị thích nghi lớn nhất và v_5 có giá trị thích nghi nhỏ nhất.

1.6.4 Chọn lọc cá thể

Bước kế tiếp là thực hiện quá trình chọn lọc cá thể, ở đây sẽ sử dụng phương pháp chọn lọc tỷ lệ. Chúng ta xây dựng hệ thống kiến trúc bánh xe roulette cho tiến trình chọn lọc. Ta có tổng độ thích nghi của các cá thể được tính như sau:

$$F = \sum_{i=1}^{10} \text{eval}(v_i) = 129.0281.$$

Xác suất chọn p_i của mỗi nhiễm sắc thể v_i $(i = 1, \ldots, 10)$ là:

$$\begin{aligned}
p_1 &= \text{eval}(v_1)/F = 0.06412, & p_2 &= \text{eval}(v_2)/F = 0.07808,\\
p_3 &= \text{eval}(v_3)/F = 0.05408, & p_4 &= \text{eval}(v_4)/F = 0.08435,\\
p_5 &= \text{eval}(v_5)/F = 0.04221, & p_6 &= \text{eval}(v_6)/F = 0.07473,\\
p_7 &= \text{eval}(v_7)/F = 0.14351, & p_8 &= \text{eval}(v_8)/F = 0.17257,\\
p_9 &= \text{eval}(v_9)/F = 0.08443, & p_{10} &= \text{eval}(v_{10})/F = 0.20192.
\end{aligned}$$

Vị trí xác suất q_i của mỗi nhiễm sắc thể v_i $(i = 1, \ldots, 10)$ được xác định:

$$q_i = \sum_{j=1}^{i} p_j.$$

Cụ thể:

$$\begin{aligned}
q_1 &= 0.06412, & q_2 &= 0.14220, & q_3 &= 0.19628, & q_4 &= 0.28063,\\
q_5 &= 0.32284, & q_6 &= 0.39757, & q_7 &= 0.54108, & q_8 &= 0.71365,\\
q_9 &= 0.79808, & q_{10} &= 1.00000.
\end{aligned}$$

Bây giờ ta quay bánh xe roulette 10 lần; mỗi lần chọn một nhiễm sắc thể cho quần thể mới. Giả sử thứ tự (ngẫu nhiên) của 10 số trong khoảng $[0, 1]$ được sinh ngẫu nhiên ở bảng sau (bảng 1.3):

Bảng 1.3: Chọn lọc nhiễm sắc thể (cá thể)

	1	2	3	4	5	6	7	8	9	10
Số ngẫu nhiên	0.7172	0.7897	0.3412	0.9604	0.0124	0.2806	0.1613	0.675	0.7169	0.1308
NST chọn	v_9	v_9	v_6	v_{10}	v_1	v_4	v_3	v_8	v_8	v_2

Thứ tự chọn nhiễm sắc thể như sau: $q_8 = 0.71365 < 0.7172 < 0.79808 = q_9$, như vậy nhiễm sắc thể được chọn là v_9. Tương tự cho tất cả các cá thể trong quần thể, như bảng trên. Như vậy quần thể mới được chọn như sau:

$$\begin{aligned}
v_1 &= 01011110011000000010111011101010,\\
v_2 &= 01011110011000000010111011101010,\\
v_3 &= 01011110100100100011101010101100,\\
v_4 &= 10100110000111010110110011011010,\\
v_5 &= 01011110000111100010101101010100 0,\\
v_6 &= 01010010110110101010101001010110 1,\\
v_7 &= 01001010100110100010111101010100 1,\\
v_8 &= 11000110010010111010011010101111 0,\\
v_9 &= 11000110010010111010011010101111 0,\\
v_{10} &= 01011110001111101010100101000101 1.
\end{aligned}$$

1.6.5 Phương pháp lai ghép

Sử dụng phương pháp lai ghép một điểm cho các cá thể trong quần thể mới v_i ($i = 1..10$). Với xác suất lai ghép là $p_c = 0.5$ tức 1/2 số cá thể trong quần thể tham gia lai tạo cho ra các cá thể mới (trong ví dụ là 5/10 cá thể) sẽ lai tạo. Đối với mỗi nhiễm sắc thể trong quần thể mới ta sinh một số ngẫu nhiên $r \in [0, 1]$; nếu $r < 0.5$ ta sẽ chọn nhiễm sắc thể đó lai tạo. Nếu trong số các nhiễm sắc thể được chọn lai tạo là số chẵn thì chúng ta chỉ cho chúng thứ tự lai tạo với nhau, trong trường hợp số nhiễm sắc thể được chọn là số lẻ thì chúng ta phải chọn ngẫu nhiên một nhiễm sắc thể bất kỳ để lai tạo, hoặc bỏ bớt một nhiễm sắc thể.

Giả sử chúng ta thực hiện sinh 10 giá trị ngẫu nhiên thuộc $[0, 1]$ như bảng 1.4:

Bảng 1.4: Kết quả chọn các nhiễm sắc thể thực hiện lai ghép

Thứ tự NST	1	2	3	4	5	6	7	8	9	10
Số ngẫu nhiên	0.7232	0.6967	0.3859	0.3147	0.0382	0.4530	0.7928	0.2812	0.6996	0.9793
NST chọn (x)			x	x	x	x		x		

Như vậy các nhiễm sắc thể được chọn để lai tạo sẽ là v_3, v_4, v_5, v_6, v_8 và giả sử ta bỏ chọn nhiễm sắc thể v_3 để thuận lợi trong việc lai tạo.

$$\begin{aligned}
v_4 &= 10100110000111010110110011011010,\\
v_5 &= 01011110000111100010101101010100 0,\\
v_6 &= 01010010110110101010101001010110 1,\\
v_8 &= 11000110010010111010011010101111 0.
\end{aligned}$$

Bước kế tiếp tiến hành cho các cặp (v_4, v_6), (v_5, v_8) lai tạo với nhau, điều này sẽ cho phép lai tạo ngẫu nhiên, để chúng giống quy luật của tự nhiên. Trong ví dụ này chúng ta tiến hành cho lai ghép một điểm, kết quả như sau:

Cặp lai (v_4, v_6):

$$v_4 = 10100110000111010110110011011101 0,$$
$$v_6 = 01010010110110101010100100101011 01,$$
$$v_4' = 01010010110111010110110011011101 0,$$
$$v_6' = 10100110000110101010100100101011 01.$$

Cặp lai (v_5, v_8):

$$v_5 = 01011110000111100010101101010100 0,$$
$$v_8 = 11000110010010111010011010101111 0,$$
$$v_5' = 11000110000111100010101101010100 0,$$
$$v_8' = 01011110010010111010011010101111 0.$$

Như vậy sau khi lai ghép quần thể mới là các cá thể:

$$v_1, v_2, v_3, v_4', v_5', v_6', v_7, v_8', v_9, v_{10}.$$

1.6.6 Phương pháp đột biến

Phép đột biến được thực hiện trên cơ sở từng bit, xác suất đột biến xảy ra thấp hơn lai ghép. Giả sử ta có xác suất xảy ra đột biến là $p_m = 0.02$ (2/100) tức 100 bit ta có thể xảy ra 2 bit đột biến. Với quần thể 10 cá thể thì chúng ta có 10×33 bit trong toàn quần thể. Như vậy ta có số bit đột biến sẽ là 6.6 bit trong mỗi thế hệ. Mỗi bit đều có khả năng đột biến ngang nhau, vì vậy cứ mỗi bit trong quần thể ta sinh ngẫu nhiên số $r \in [0, 1]$; nếu $r < 0.02$ thì bit đó sẽ bị đột biến.

Điều này có nghĩa ta phải sinh 330 số ngẫu nhiên. Giả sử ta có 6 trong số 330 giá trị nhỏ hơn 0.02, giả sử vị trí bit được trình bày như bảng 1.5 và bảng 1.6:

Bảng 1.5: Vị trí các gen bị đột biến

Số ngẫu nhiên	0.0178	0.0135	0.0060	0.0015	0.0114	0.0190
Vị trí bit	53	126	136	172	187	214

Bảng 1.6: Các vị trí gen bị đột biến trong từng nhiễm sắc thể

Vị trí bit trong quần thể	53	126	136	172	214	289
NST	2	4	5	6	7	9
Vị trí bit trong NST	20	27	4	7	16	25

Như vậy nhiễm sắc thể được đột biến $(v_2, v_4, v_5, v_6, v_7, v_9)$, và các vị trí tương ứng đột biến như bảng trên. Với cách biểu diễn nhiễm sắc thể bằng chuỗi nhị phân ta sẽ thực hiện đột biến như sau: nếu bit $0 \to 1$, $1 \to 0$. Các cá thể bị đột biến:

$$v_2' = 01011110011000000011111101110101 0,$$
$$v_4' = 01010010110111010110110111011101 0,$$
$$v_5' = 11010110000111100010101101010100 0,$$
$$v_6' = 10100100000110101010100100101011 01,$$
$$v_7' = 01001010100110101010111101010100 1,$$
$$v_9' = 11000110010010111010011000101111 0.$$

Như vậy quần thể mới (thế hệ mới) nhận được sau một chu kỳ chọn lọc, lai tạo, đột biến:

$$\begin{aligned}
v_1 &= 01011110011000000010111011101010,\\
v_2 &= 01011110011000000011111011101010,\\
v_3 &= 01011110100100100011011010101100,\\
v_4 &= 01010010110111010110110111011010,\\
v_5 &= 11010110000111100010101101010100 0,\\
v_6 &= 10100100000110101010100101011 01,\\
v_7 &= 01001010100110101010111101010100 1,\\
v_8 &= 01011110010010111010011010101111 0,\\
v_9 &= 11000110010010111010011000101111 0,\\
v_{10} &= 01011110001111101010100101000101 1.
\end{aligned}$$

Giải mã từng nhiễm sắc thể và tính giá trị thích nghi tương ứng với từng cá thể:

$$\begin{aligned}
\text{eval}(v_1) &= f(2.55666, 5.36061) = 6.70990,\\
\text{eval}(v_2) &= f(2.56667, 5.78562) = 7.27090,\\
\text{eval}(v_3) &= f(2.57819, 5.67615) = 9.64190,\\
\text{eval}(v_7) &= f(1.40045, 5.35724) = 6.97780,\\
\text{eval}(v_8) &= f(2.56194, 5.12756) = 8.84650,\\
\text{eval}(v_9) &= f(8.69634, 5.11428) = 7.62310,\\
\text{eval}(v_{10}) &= f(2.55895, 5.19630) = 10.07480.
\end{aligned}$$

1.6.7 Các tham số sử dụng trong ví dụ và điều kiện dừng

Ví dụ trên được sử dụng một số tham số như sau:

- Kích thước quần thể: pop_size = 10,
- Xác suất lai ghép: $p_c = 0.5$,
- Xác suất đột biến: $p_m = 0.02$.

Ví dụ sử dụng phương pháp kết thúc dựa trên số thế hệ. Ở trên được trình bày 2 thế hệ quần thể, giá trị thích nghi lớn nhất của các cá thể qua 2 thế hệ trên là

$$\text{eval}(v_5) = 13.0489.$$

Giá trị tối ưu của hàm số trên được tính bằng phương pháp khác là:

$$26.4395.$$

CHƯƠNG II – Tổng Quan Bài Toán Lập Lịch

2.1. Giới thiệu bài toán lập lịch

Việc lập lịch đã được hình thành từ khi con người bắt đầu có sự phân công công việc trong xã hội và hiện nay vẫn tiếp tục tồn tại và phát triển cùng với sự tiến bộ của khoa học kỹ thuật. Một lịch biểu tốt sẽ giúp cho tiến trình thực hiện công việc ngắn lại, tiết kiệm được thời gian, nguồn lực, tài nguyên và nâng cao chất lượng trong sản xuất hay quản lý. Vì thế đã trải qua nhiều thập kỷ người ta không ngừng tìm hiểu, nghiên cứu và phát triển các phương pháp lập lịch để tìm kiếm lời giải tốt nhất cho các bài toán.

Bài toán lập lịch có thể được định nghĩa một cách chung nhất là bài toán cấp phát nguồn lực, tài nguyên để thực hiện tập hợp các công việc trong một chuỗi các tiến trình trên cơ sở thời gian, tài nguyên và các ràng buộc đã được định sẵn. Vì vậy việc lập lịch được xem như tìm kiếm một giải pháp tối ưu trong các điều kiện hạn chế. Người lập lịch cố gắng thử đến mức tối đa sự sử dụng các cá thể, máy móc, nguồn lực, tối thiểu thời gian để hoàn thành toàn bộ quá trình sắp xếp lịch. Như vậy khi giải quyết bài toán lập lịch đưa đến phải trả lời hai câu hỏi:

- Tài nguyên nào sẽ được phân phối cho từng nhiệm vụ.
- Khi nào thì nhiệm vụ sẽ được giải quyết.

Trong bài toán lập lịch bao hàm hai khía cạnh lý thuyết và thực tế, về lý thuyết thể hiện cố gắng giải quyết vấn đề về mô hình toán học. Khía cạnh thực tế là việc xây dựng các ứng dụng phù hợp với tập các ràng buộc trong thực tế. Hiệu quả của việc giải bài toán lập lịch luôn được gắn với việc lựa chọn các quyết định trong không gian quyết định (decision making)

Trong thực tế việc đưa ra quyết định thường được xem xét trên cơ sở:

- Hiệu quả sử dụng tài nguyên (resource).
- Đáp ứng yêu cầu nhanh chóng.
- Giải quyết có trọng tâm.

Tất cả các bài toán lập lịch cụ thể đều xuất phát từ nhu cầu ứng dụng trong thực tiễn để phục vụ mục đích của nhà quản lý. Vì vậy vấn đề tối ưu các chi phí luôn luôn được xem xét như là một trong những ưu tiên hàng đầu của bộ lịch. Việc mô hình hoá và giải về mặt lý thuyết cho đến hiện nay thường dùng quy hoạch động. Tuy nhiên khi thực hiện khi giải bài toán trên máy tính thì không lúc nào cũng khả thi do tính phức tạp của nhiều bài toán dẫn đến một loạt các chi phí khác phát sinh, trong đó chi phí về thời gian là quan trọng nhất. Trong trường hợp nếu máy tính đủ khả năng (về cấu hình) để giải tối ưu thì trong hầu hết các trường hợp, người sử dụng cũng khó chấp nhận về mặt thời gian chờ đợi kết quả, đặc biệt với các bài toán lớn lượng biến sinh ra rất nhiều. Chính vì vậy các ứng dụng thành công trên thực tế đều có sử dụng Heuristic (một phần hay toàn bộ) để kết quả thu được gần tối ưu, có hiệu quả và được người dùng chấp nhận.

Nhưng nhu cầu thực tế về kết quả cho từng bài toán cụ thể lại có thể rất khác nhau, chúng có thể cần "sắp xếp được" cho đến "sắp xếp gần tối ưu" và "sắp xếp tối ưu". Vì vậy, cách giải trong từng bài toán phụ thuộc vào từng mô hình cụ thể của từng bài toán và từng thuật toán.

Một số điểm mấu chốt khi tiếp cận giải bài toán lập lịch:

- Xác định nhu cầu, mục tiêu, giới hạn, độ phức tạp.
- Xác định các loại ràng buộc.
- Xây dựng mô hình.
- Xây dựng giải thuật, các hướng giải quyết.
- Cài đặt chương trình.

Thật dễ thấy rằng, bài toán lập lịch là những bài toán rất quan trọng trong thực tế và chúng xuất hiện hầu hết mọi nơi trong cuộc sống hằng ngày như: lập lịch cho các chuyến bay của hãng hàng không, lập lịch phát sóng cho các tiết mục trong đài truyền hình, lập lịch thi, lập thời khoá biểu học tập, ... Và hiện nay bài toán đã và đang có nhiều nhà nghiên cứu, chuyên gia tìm kiếm các phương pháp tiếp cận để giải quyết bài toán như: trí tuệ nhân tạo, hệ chuyên gia, mạng Nơron, lập trình tính toán, tìm kiếm nhánh và đường biên, kỹ thuật mô phỏng luyện kim, tìm kiếm Tabu, ...

2.2. Các thuộc tính của bài toán lập lịch

- **Tài nguyên:** Là nguồn dữ liệu đầu vào của bài toán, các tài nguyên này có thể phục hồi hoặc không.
- **Tác vụ:** Được đánh giá qua các tiêu chuẩn thực hiện, như thời gian thực hiện, chi phí, mức tiêu tốn nguồn tài nguyên.
- **Ràng buộc:** Là những điều kiện cần được thoả mãn để bài toán có thể đưa ra lời giải tốt nhất.
- **Mục tiêu:** Đánh giá mức độ tối ưu của lời giải của bài toán. Khi các mục tiêu được thoả mãn thì các ràng buộc cũng được thoả mãn.

2.3. Một số loại bài toán lập lịch

Lập lịch tiền định và suy diễn (Deterministic & Stochastic Scheduling): Lập lịch tiền định là tất cả các thông tin dữ liệu liên quan đều đã biết trước hoặc đã chắc chắn. Trong bài toán lập lịch suy diễn, thời gian sẵn sàng cho công việc hay thời gian thực hiện hoạt động là các biến ngẫu nhiên được mô tả bởi một hình thái thống kê đã biết hoặc sự phân bố xác suất.

Lập lịch tĩnh (Static Scheduling): Việc lập lịch được thực hiện dựa trên các hiểu biết hoặc dự báo về các sự kiện, tác vụ thực hiện trong hệ thống (thời điểm xuất hiện, thời gian thực hiện, hạn chót ước tính (deadline)) và được quyết định tại thời điểm thiết kế và được áp dụng cố định trong suốt quá trình hoạt động của hệ thống. Các tác vụ được khởi động ở các thời điểm đã lập trong một bảng trước đó. Một thuật toán lập lịch tĩnh được gọi là tối ưu nếu nó luôn luôn có thể tìm được một lịch điều phối thỏa mãn các ràng buộc đã

cho trong khi một thuật toán tĩnh khác cũng tìm được một lời giải. Như vậy trong lập lịch tĩnh tất cả các công việc được tiến hành lập lịch đã sẵn sàng để sử dụng đồng thời.

Lập lịch động (Dynamic Scheduling): Việc thực hiện lập lịch trong quá trình thực thi dựa trên cơ sở các thông tin hoạt động hiện hành của hệ thống. Sơ đồ lập lịch là không xác định trước và thay đổi động theo quá trình thực hiện. Lập lịch động linh hoạt nhưng tốn thời gian để ra quyết định và cũng không có "nhận thức" tới bối cảnh tổng thể như các yêu cầu tài nguyên, sự phụ thuộc giữa các tác vụ.

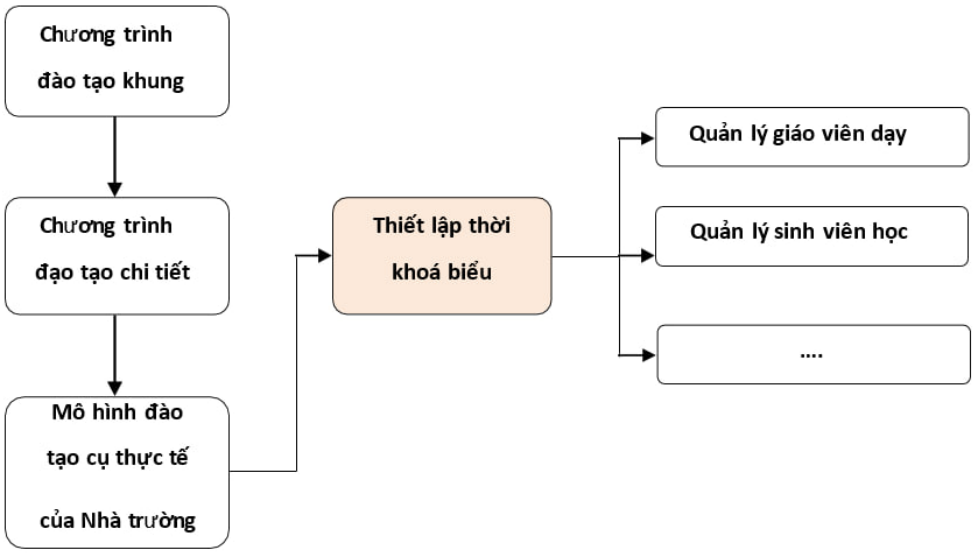
2.4. Bài toán thời khóa biểu

Bài toán lập Thời khóa biểu đã từ lâu trở thành một bài toán nổi tiếng và đã thu hút được sự quan tâm của rất nhiều nhà nghiên cứu, nhiều chuyên gia trong các lĩnh vực liên quan. Sự "nổi tiếng" của bài toán này không chỉ được đo bởi độ phức tạp của vấn đề, mà còn ở tính thực tiễn, khả năng áp dụng nhiều trên thực tế. Bất cứ một nhà trường nào, thời khóa biểu học tập của sinh viên và lịch giảng dạy của giáo viên đã và luôn là bộ xương sống cơ bản nhất, kết nối hầu như toàn bộ các hoạt động của nhà trường. Chính vì lẽ đó bài toán lập Thời khóa biểu trở thành một trong những vấn đề chính và quan trọng vào bậc nhất của mỗi nhà trường.

2.4.1 Giới thiệu bài toán

Bài toán thời khoá biểu là một trong những bài toán thuộc lớp bài toán lập lịch, bài toán yêu cầu tìm giải pháp tối ưu trong tập các tài nguyên, nguồn lực, thời gian hạn chế. Vấn đề xây dựng thời khoá biểu tự động hoặc bán tự động cho các loại công việc khác nhau, bao gồm cả các hoạt động dạy và học trong một cơ sở đào tạo là một vấn đề cấp thiết đã và đang thu hút nhiều sự chú ý trong hai lĩnh vực nghiên cứu và ứng dụng trong thời gian qua.

Như chúng ta đã biết chương trình đào tạo là một bản thiết kế tổng thể cho mọi hoạt động đào tạo, nó được sắp xếp theo một quy trình cụ thể. Và trong đó thời khoá biểu là một khâu quan trọng trong mô hình quản lý đào tạo của nhà trường. Hình 2.1 biểu diễn mối quan hệ không tách rời giữa chương trình đào tạo và thời khoá biểu được minh hoạ như sau:



Hình 2.1: Quy trình quản lý đào tạo của trường Đại học và Cao đẳng

Hình 2.1 cho thấy, Thời khoá biểu như là bộ xương sống trong quá trình đào tạo và quản

lý của nhà trường, đặc biệt hiện nay một số trường đại học, cao đẳng nước ta đã và đang chuyển sang đào tạo theo học chế tín chỉ thì một thời khoá biểu ổn định, phù hợp với sinh viên, thuận lợi cho giáo viên và có tính khoa học sẽ tạo điều kiện cho sinh viên, giáo viên lên kế hoạch học tập, nghiên cứu.

Tuy nhiên, các kết quả đạt được trong lĩnh vực này hiện nay vẫn chưa có được một ứng dụng rộng rãi do các nguyên nhân sau đây.

Về mặt học thuật, bài toán sắp thời khoá biểu tự động là một bài toán có độ phức tạp giải thuật rất cao (NP–đầy đủ). Hiện nay vẫn chưa tồn tại giải thuật sắp thời khoá biểu tổng quát có thời gian chạy chấp nhận được trong môi trường thực tế.

Khi áp dụng vào các đơn vị đào tạo, do mỗi đơn vị đào tạo có một mô hình đào tạo, đặc thù, dẫn đến các mối quan hệ ràng buộc khác nhau. Việc đưa ra một mô hình biểu diễn đặc trưng tổng quát cho tất cả các mối quan hệ ràng buộc này là rất khó khăn.

Khi triển khai, do mỗi đơn vị đào tạo có một hệ thống thông tin quản lý riêng sử dụng nhiều công nghệ khác nhau, việc đưa ra một giải pháp công nghệ để tích hợp module xử lý thời khoá biểu vào hệ thống thông tin sẵn có của mỗi đơn vị đào tạo sao cho ít gây nên xáo trộn về quy trình và phương thức làm việc nhất cũng là một thách thức không nhỏ.

Trong luận văn này chúng ta sẽ cố gắng phân tích mô hình bài toán lập Thời khóa biểu tổng quát và các đặc thù chính của các trường Cao đẳng, Đại học theo học chế tín chỉ của nước ta. Từ các phân tích đó sẽ đưa ra định nghĩa bài toán chung và định hướng xây dựng một giải thuật để tiếp cận giải bài toán. Trên cơ sở đó làm tiền đề xây dựng ứng dụng góp phần hỗ trợ công việc lập Thời khóa biểu cho cơ sở, đồng thời thể hiện nó là một bộ phận không thể tách rời trong các quan hệ tổng thể của công việc quản lý và đào tạo của nhà trường.

Để tìm hiểu những thông tin, đối tượng cấu thành nên bài toán thời khoá biểu, chúng ta sẽ tìm hiểu một số thành phần liên quan đến dữ liệu của bài toán sẽ được trình bày ở phần sau.

2.4.2 Dữ liệu bài toán

Phần này chúng ta sẽ tìm hiểu, khảo sát các thành phần, đối tượng thông tin có tác động trực tiếp hoặc gián tiếp đến bài toán thời khoá biểu:

Giáo viên: Là người trực tiếp giảng dạy theo các học phần của môn học được quy định chặt chẽ về thời lượng, kiến thức và hình thức học. Trong quy trình đào tạo theo học chế tín chỉ thì mỗi giáo viên có thể dạy được nhiều môn và mỗi giáo viên sẽ có mã quản lý riêng do nhà trường quy định.

Học phần (môn học): Là khối lượng kiến thức tương đối trọn vẹn, thuận lợi cho sinh viên tích luỹ kiến thức trong quá trình học tập. Phần lớn học phần có khối lượng từ 2 đến 4 tín chỉ, nội dung được bố trí giảng dạy trọn vẹn và phân bố đều trong một học kỳ. Kiến thức trong mỗi học phần phải gắn với một mức trình độ theo năm học thiết kế và được kết cấu riêng như một phần của môn học hoặc được kết cấu dưới dạng tổ hợp từ nhiều môn học. Đối với từng học phần phải được ký hiệu bằng một mã số riêng do trường quy định.

Tín chỉ (Credit): Là đơn vị quy chuẩn dùng để lượng hoá khối lượng kiến thức và khối lượng học tập giảng dạy trong quy trình đào tạo. Tín chỉ cũng là đơn vị để đo lường tiến độ học tập của sinh viên – đánh giá dựa trên số lượng tín chỉ sinh viên đã tích lũy được. Một tín chỉ được tính bằng: 15 tiết học lý thuyết, hoặc 30–45 tiết thảo luận, 45–90 giờ thực tập tại cơ sở, 45–60 giờ làm tiểu luận, bài tập lớn, khóa luận tốt nghiệp. Một tiết học được tính bằng 50 phút. Để tiếp thu khối lượng kiến thức của 1 tín chỉ sinh viên phải dành ít nhất 30 giờ chuẩn bị cá nhân.

Lớp học phần: Là lớp của các sinh viên cùng đăng ký một học phần, có cùng thời khoá biểu của học phần trong cùng một học kỳ. Mỗi lớp học phần được gán một mã số riêng.

Số lượng sinh viên của một lớp học phần được giới hạn bởi sức chứa của phòng học/phòng thực hành/phòng thí nghiệm, ... hoặc được sắp xếp theo các yêu cầu riêng đặc thù của học phần đó.

Phòng học: Là nơi học tập của sinh viên, phòng học bao gồm các thông tin như loại phòng chuyên môn, và khả năng tổ chức, sức chứa của phòng học. Mỗi phòng học được quản lý bằng các thông tin như: địa chỉ, cơ sở, dãy, tầng, đặc trưng, ... mỗi phòng học sẽ có một mã số quy định, được phân biệt rõ so với các phòng học khác.

Tiết học (giờ học): Đơn vị thời gian tổ chức một tiết học. Một tiết học thường bắt đầu từ 8 giờ–20 giờ hằng ngày. Mỗi tiết thường có 50 phút giảng dạy.

Ngoài các dữ liệu được trình bày như trên, một số yếu tố cũng góp phần không nhỏ đến việc sắp xếp thời khoá biểu như: bảng phân công giảng dạy từ các khoa, lịch yêu cầu giảng dạy từ các giáo viên, kế hoạch các ngày nghỉ như lễ, tết.

Các cơ sở đào tạo của nước ta hiện nay tất cả các tài nguyên đều bị hạn chế do một số nguyên nhân chủ quan và khách quan. Vì vậy, để sắp xếp thời khoá biểu tốt thoả mãn tất cả các yêu cầu là hết sức khó khăn. Tuy nhiên không chỉ khó khăn về sự thiếu thốn các tài nguyên trên mà còn có sự ảnh hưởng của một số ràng buộc, các dữ liệu liên quan đến thực tế đời sống. Các ràng buộc này sẽ được trình bày ở phần sau.

2.4.3 Ràng buộc của bài toán

Một trong những yếu tố tạo nên sự phức tạp của bài toán thời khoá biểu là các yêu cầu và các ràng buộc. Các ràng buộc là các yêu cầu cần phải được thoả mãn, nếu một trong những yêu cầu này không thoả mãn thì thời khoá biểu sẽ không thể đưa vào sử dụng. Một số yêu cầu về phòng học như: hai lớp học khác nhau không thể học cùng một phòng học tại một thời điểm, và các phòng học phải đảm bảo chỗ ngồi cho sinh viên để sinh viên có chỗ ngồi học tập. Đối với yêu cầu về giáo viên là một giáo viên không thể dạy được hai lớp trong cùng một thời gian. Về chương trình, các môn học trong cùng một chương trình phải được sắp xếp khác thời điểm để sinh viên được lựa chọn học. Và với mỗi môn học có số tiết được quy định trước và thời khoá biểu phải đảm bảo số tiết học của môn học đó.

Như phần trên chúng ta đã tìm hiểu định nghĩa bài toán lập lịch và cụ thể là bài toán thời khoá biểu của một trường đại học, cao đẳng. Như đã biết, trên thực tế bài toán lập lịch, bài toán thời khoá biểu đã có nhiều chuyên gia nghiên cứu, tìm hiểu các giải thuật để giải bài toán. Trong phần sau chúng tôi sẽ trình bày một số thuật toán đã được sử dụng giải các bài toán trên.

CHƯƠNG III – ỨNG DỤNG GIẢI THUẬT DI TRUYỀN VÀO BÀI TOÁN LẬP THỜI KHÓA BIỂU

3.1. Bài toán thời khóa biểu theo học chế tín chỉ

Bài toán thời khoá biểu có vai trò rất quan trọng trong bất cứ một nhà trường nào, thời khóa biểu học tập của sinh viên và lịch giảng dạy của giáo viên luôn là bộ xương sống cơ bản nhất, kết nối hầu như toàn bộ các hoạt động của nhà trường. Chính vì lẽ đó bài toán lập Thời khóa biểu trở thành một trong những vấn đề chính và quan trọng vào bậc nhất của mỗi trường.

Thời khoá biểu như là bộ xương sống trong quá trình đào tạo và quản lý của nhà trường, đặc biệt hiện nay ngành giáo dục đại học nước ta đã và đang chuyển sang đào tạo theo học chế tín chỉ thì một thời khoá biểu ổn định, phù hợp với sinh viên, thuận lợi cho giáo viên và có tính khoa học sẽ tạo điều kiện cho sinh viên, giáo viên lên kế hoạch học tập và nghiên cứu.

Bài toán thời khoá biểu có tầm quan trọng trong thực tế nhưng lại được xếp vào bài toán thuộc lớp NP–khó và có sự kết hợp các ràng buộc không tầm thường thuộc nhiều loại, vì thế nó đã và đang được rất nhiều nhà nghiên cứu quan tâm và cần nghiên cứu về độ phức tạp của bài toán.

Đối với các bài toán không gian lời giải nhỏ thì có thể sử dụng phương pháp cổ điển như vét cạn là đủ để tìm được giải pháp tối ưu. Nhưng với bài toán có không gian lời giải lớn và kết hợp nhiều ràng buộc thì đòi hỏi phải có những phương pháp trí tuệ nhân tạo đặc biệt, giải thuật di truyền là một trong những phương pháp đó.

Trong chương trước chúng ta đã tìm hiểu các khái niệm, các đối tượng liên quan đến bài toán thời khoá biểu và giải thuật di truyền, trong phần sau sẽ đưa ra một cách tổng quát bài toán để ứng dụng giải thuật di truyền.

3.1.1 Định nghĩa bài toán

Từ các định nghĩa các yếu tố, thành phần liên quan đến bài toán thời khoá biểu ở chương trước, phần này chúng ta đưa ra định nghĩa một cách tổng quát cho bài toán như sau:

- Một tập các chương trình đào tạo:

  $$CT = \{CT_1, CT_2, \ldots, CT_l\}.$$

  Mỗi chương trình gồm những môn học theo kế hoạch của một ngành học, cho một khóa học.

- Một tập các môn học:

  $$M = \{M_1, M_2, \ldots, M_t\}.$$

  Mỗi môn học gồm số tín chỉ, danh sách các chương trình học môn học đó.

- Một tập các nhóm sinh viên (lớp học phần):

$$SV = \{SV_1, SV_2, \ldots, SV_n\}.$$

  Mỗi lớp học phần gồm môn học, giảng viên dạy, số sinh viên học (dự kiến hoặc chính thức).

- Một tập các phòng học:

$$P = \{P_1, P_2, \ldots, P_m\}.$$

  Mỗi phòng học có số chỗ ngồi.

- Một tập các giảng viên:

$$G = \{G_1, G_2, \ldots, G_k\}.$$

- Một tập các tiết học trong tuần:

$$T = \{T_1, T_2, \ldots, T_h\}.$$

- Tập phân công giáo viên dạy:

$$E = \{(SV_i, M_i, G_i) \mid SV_i \in SV,\ M_i \in M,\ G_i \in G\}.$$

Bài toán được định nghĩa như trên, việc lập thời khoá biểu phải thể hiện các thông tin của bài toán và qua đó phải thoả mãn tất cả các yêu cầu ràng buộc được trình bày ở phần sau.

3.1.2 Các ràng buộc của bài toán

Xếp lịch học cho các lớp vào các phòng học tại các thời điểm sao cho thỏa mãn các điều kiện sau:

(C1) Không có hai lớp học cùng một phòng tại một thời điểm.

(C2) Một giáo viên không dạy hai lớp tại cùng một thời điểm.

(C3) Xếp các lớp học vào các phòng học đảm bảo đủ chỗ ngồi cho sinh viên.

(C4) Xếp các tiết học đảm bảo đủ số tiết cho mỗi môn học.

(C5) Không xếp các môn học của cùng một chương trình đào tạo vào cùng một tiết học.

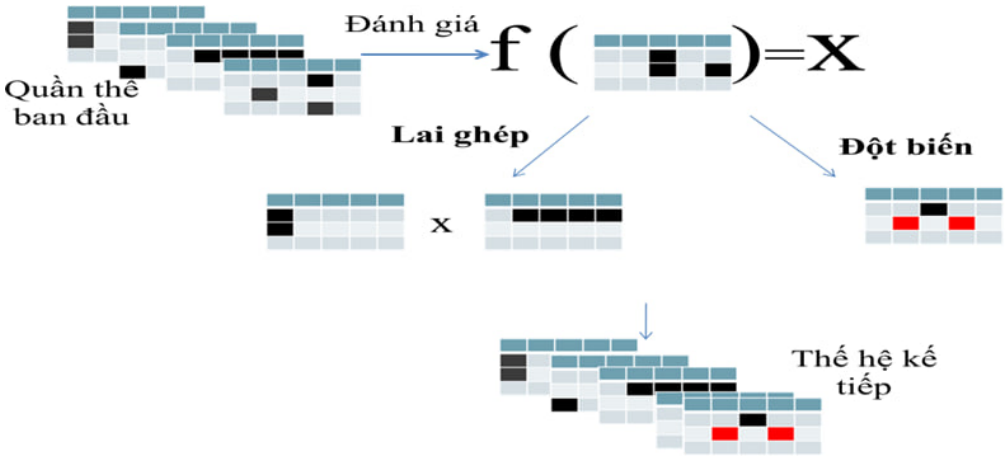
Từ các dữ liệu, các ràng buộc của bài toán, cần xây dựng một giải thuật hiệu quả và đáp ứng các yêu cầu. Ở đây yêu cầu của bài toán tìm lời giải của bài toán sao cho thoả mãn tất cả các ràng buộc $\{C\}$.

3.2. Phát biểu bài toán theo hướng tiếp cận giải thuật di truyền

Từ ý tưởng của giải thuật di truyền ta định nghĩa cho bài toán thời khoá biểu như sau: Khởi tạo N lời giải ban đầu (tập các thời khoá biểu) gọi là quần thể (ở đây chúng ta sử dụng thuật toán sinh ngẫu nhiên được trình bày ở phần sau), mỗi cá thể trong quần thể được xác định giá trị thích nghi bằng hàm đánh giá. Hàm đánh giá có vai trò rất quan trọng trong

việc xác định sự tồn tại của cá thể trong quần thể hay không, hàm đánh giá giống với quá trình chọn lọc tự nhiên trong thuyết tiến hoá: cá thể nào thích nghi với môi trường sẽ tồn tại, cá thể nào không thích nghi sẽ bị đào thải. Và qua mỗi thế hệ, trải qua các quá trình chọn lọc tự nhiên, lai ghép, đột biến sẽ tạo ra các cá thể mong muốn là tốt hơn bố mẹ của chúng. Dựa vào tiêu chí hàm đánh giá để chọn ra cá thể phù hợp với yêu cầu của bài toán, và đối với bài toán thời khoá biểu thì chọn ra cá thể (lời giải) thoả mãn tất cả các ràng buộc.

Việc áp dụng giải thuật di truyền vào bài toán có thể được biểu diễn bằng hình 3.1.



Hình 3.1: Biểu diễn một vòng lặp của giải thuật di truyền trong bài toán thời khoá biểu

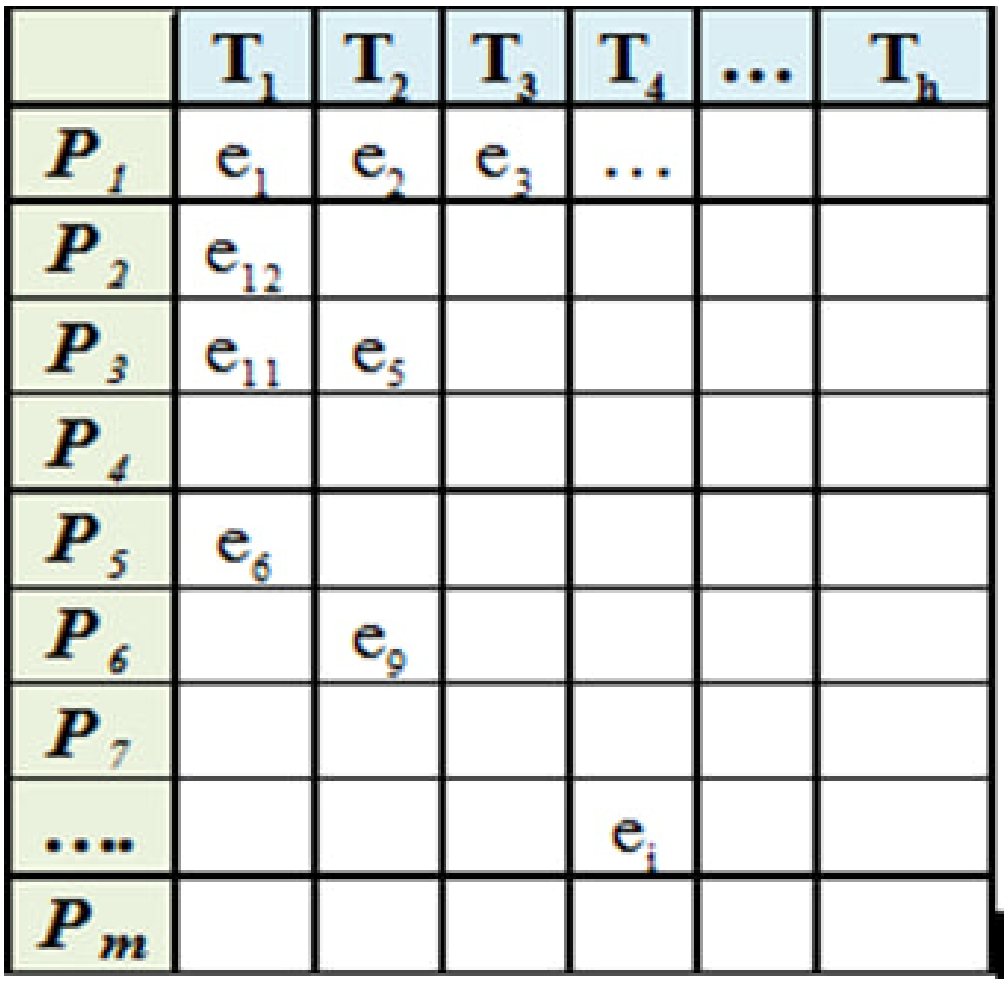
3.3. Áp dụng giải thuật di truyền vào bài toán thời khóa biểu

3.3.1 Biểu diễn nhiễm sắc thể

Nhiễm sắc thể biểu diễn một lời giải của bài toán. Nhiễm sắc thể chứa nhiều gen khác nhau, mỗi gen thể hiện một sự kiện bao gồm các thuộc tính: lớp học phần, giáo viên, phòng học để quy định một hay nhiều tính trạng nào đó.

Thời khóa biểu của cơ sở khả thi khi tất cả các sự kiện (tập phân công giảng dạy) được sắp xếp vào các giờ học trong tuần và thỏa mãn tất cả các điều kiện bắt buộc $\{C1, C2, C3, C4, C5\}$. Như vậy một thời khóa biểu được biểu diễn là ma trận $X_{m \times h}$, trong đó h, m là số các tiết học trong tuần và số phòng học trong một cơ sở. Với mỗi giá trị của ma trận là một đối tượng sự kiện, mỗi sự kiện gồm có giảng viên, lớp học phần và môn học và đây cũng là một giá trị trong tập phân công giảng dạy đã được định nghĩa như trên. Như vậy một nhiễm sắc thể được biểu diễn là một thời khóa biểu của cơ sở trong đó mỗi gen của nhiễm sắc thể là đối tượng sự kiện. Với mỗi cách sắp xếp các gen vào nhiễm sắc thể cho ta một nhiễm sắc thể (cá thể) mới.

Như vậy thời khóa biểu X của một cơ sở sẽ có cấu trúc được trình bày ở hình 3.2.



Hình 3.2: Biểu diễn nhiễm sắc thể (cá thể) của bài toán

Trong đó:

$$e_i = (SV_i, M_i, G_i) \mid SV_i \in SV,\ M_i \in M,\ G_i \in G$$

(lớp học phần của nhóm sinh viên SV_i học môn M_i do giảng viên G_i dạy), hay được gọi là tập phân công giảng dạy. Dựa vào số tiết của môn học trên tuần chúng ta chia nhỏ thành số các sự kiện trong tuần, với mỗi sự kiện sẽ được gán một số nguyên để thuận lợi cho việc biểu diễn sau này.

Một phần của thời khoá biểu tường minh như sau:

	T_1	T_{12}	T_{19}	T_{30}	T_{45}
P_1	G_1, M_1, L_1	G_3, M_2, L_2		G_3, M_2, L_2	G_1, M_1, L_1
P_2	G_3, M_2, L_2	G_2, M_3, L_3		G_4, M_4, L_4	G_3, M_4, L_4
P_3	G_2, M_3, L_3	G_4, M_4, L_4	G_2, M_3, L_3		
P_4				G_3, M_4, L_4	

3.3.2 Khởi tạo quần thể

Khởi tạo quần thể là bước đầu trong giải thuật di truyền, thuật toán có hội tụ nhanh hay chậm đến giá trị tối ưu cũng phụ thuộc vào quần thể khởi tạo ban đầu. Khi khởi tạo quần thể phải khởi tạo tập dữ liệu ban đầu, bao gồm tập các yêu cầu bài toán, khởi tạo tập phân công giảng dạy.

Thuật toán khởi tạo quần thể

```
Procedure Population()
Input:  N        // Số lượng cá thể yêu cầu trong quần thể
Output: Po       // Quần thể các cá thể
Begin
  While (i < N) do
    P = Individual()   // tạo một cá thể mới
    Po = Po  P        // đặt cá thể mới vào quần thể
    i = i + 1
  Endwhile
End.
```

Trong đó, `Individual()` là hàm tạo ra cá thể mới, nó được thực hiện trên ý tưởng: với mỗi T_j trong tập T và với mỗi P_i trong tập P. Chọn ngẫu nhiên một sự kiện e thuộc tập sự kiện (tập phân công giảng dạy) đặt vào vị trí trống (T_j, P_i) và loại bỏ sự kiện e ra khỏi tập sự kiện. Thực hiện cho đến khi hết số sự kiện trong tập phân công hoặc các vị trí (T_j, P_i) đã xét hết.

Thuật toán sinh cá thể cho quần thể

```
Function Individual()
Input:  tập các phân công giảng dạy
        E = {e1, e2, e3, ..., en}, {T}, {P}   // n: số sự kiện
Output: TKB   // Thời khoá biểu (cá thể)
Begin
  For each Tj in {T} do
    For each Pi in {P} do
      // lấy ngẫu nhiên một sự kiện e
      TKB[Tj][Pi] = e    // Đặt vào thời khoá biểu
      E = E \ {e}        // Loại bỏ sự kiện e ra khỏi tập sự kiện
    Endfor
  Endfor
  Return TKB
End.
```

3.3.3 Lai ghép

Bài toán sử dụng phương pháp lai ghép dựa trên vị trí và kết hợp sử dụng ma trận mặt nạ nhị phân (mặt nạ có giá trị $\{0, 1\}$ và có kích thước bằng với kích thước của nhiễm sắc thể) làm tiêu chuẩn lai ghép gen của bố mẹ và có xác suất lai ghép là p_c thể hiện số lượng các cá thể được lai ghép.

Ý tưởng của phương pháp lai ghép: với mỗi giá trị của mặt nạ, nếu mặt nạ có giá trị là 1 thì cá thể con sẽ nhận gen của cha (mẹ), ngược lại là gen của mẹ (cha). Các bước thực hiện như sau:

Bước 1: Xét tuần tự mỗi giá trị $g[i, j] \in M$ (với M là ma trận nhị phân làm mặt nạ, $i = 1..m$, $j = 1..h$). Với mỗi giá trị $g[i, j]$ kiểm tra:

- Nếu $g[i, j] = 1$:
  - Tìm gen x thuộc cá thể cha chưa được xét và không có trong cá thể con. Đặt x vào cá thể con.
  - Đánh dấu đã xét gen x trong cá thể cha.

– Ngược lại, nếu $g[i,j] = 0$:

  * Tìm gen x thuộc cá thể mẹ chưa được xét và không có trong cá thể con. Đặt x vào cá thể con.
  * Đánh dấu đã xét gen x trong cá thể mẹ.

Bước 2: Lặp lại bước 1 cho đến khi các phần tử của mặt nạ M đã được xét.

Bước 3: Kết thúc thuật toán và trả về kết quả.

Giả sử ta có đầu vào của thuật toán là cá thể cha `TKBCha`, cá thể mẹ `TKBMe` và sử dụng một ma trận nhị phân M làm mặt nạ, M có cùng kích thước với cá thể cha, mẹ. Đầu ra của thuật toán là một cá thể con `TKBCon` sau khi thực hiện lai tạo.

Thuật toán được giải mã như sau:

```
Input:  Cá thể cha TKBCha, cá thể mẹ TKBMe, M (ma trận nhị phân)
Output: Cá thể con TKBCon
Begin
  For each g[i,j] in M do        // g[i,j]: giá trị thuộc mặt nạ M
    If (g[i,j] == 1) then
      While (x in TKBCha) and (x in TKBCon) do
        // tìm gen x, chưa xét thuộc TKBCha và x chưa thuộc vào TKBCon
      Endwhile
      TKBCon[i,j] = x
    Else
      While (x in TKBMe) and (x in TKBCon) do
        // tìm gen x, chưa xét thuộc TKBMe và x chưa thuộc vào TKBCon
      Endwhile
      TKBCon[i,j] = x
    Endif
  Endfor
End.
```

Ví dụ: Giả sử có hai nhiễm sắc thể cha, mẹ `NSTCha`, `NSTMe` (các sự kiện được gán bằng các số nguyên để thuận lợi trong việc biểu diễn) và ma trận mặt nạ M:

NSTCha

	T_1	T_2	T_3	T_4
P_1	12	13	1	9
P_2	4	2	5	16
P_3	5	10	11	14
P_4	8	1	7	6

NSTMe

	T_1	T_2	T_3	T_4
P_1	13	4	14	7
P_2	5	3	15	8
P_3	6	10	9	2
P_4	12	11	16	1

Mặt nạ (M)

	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	1	0	0	0
4	0	1	1	0

Cách lai ghép dựa vào giá trị của mặt nạ: nếu giá trị đang xét của mặt nạ bằng 1, gen được nhận là của cha `NSTCha`, ngược lại nhận của mẹ `NSTMe`. Từ ví dụ trên ta có:

- **Giá trị thứ nhất,** $m[1,1] = 0$**.** Cá thể con sẽ nhận gen của mẹ, `NSTMe`.

$$\text{NSTMe}[1,1] = 13, \text{ cá thể con chưa có gen này nên:}$$

$$\text{NSTCon}[1,1] = 13,$$

và đánh dấu đã xét trong cá thể cha và mẹ gen số 13:

NSTCha	T_1	T_2	T_3	T_4
P_1	12	13	1	9
P_2	4	2	5	16
P_3	5	10	11	14
P_4	8	1	7	6

NSTMe	T_1	T_2	T_3	T_4
P_1	13	4	14	7
P_2	5	3	15	8
P_3	6	10	9	2
P_4	12	11	16	1

NSTCon	T_1	T_2	T_3	T_4
P_1	13			
P_2				
P_3				
P_4				

- **Giá trị thứ hai,** $m[1,2] = 1$**.** Cá thể con sẽ nhận gen của cha, `NSTCha`.

$$\text{NSTCha}[1,1] = 12, \text{ cá thể con chưa có gen này nên:}$$

$$\text{NSTCon}[1,2] = 12,$$

và đánh dấu đã xét trong cá thể cha và mẹ gen số 12:

NSTCha	T_1	T_2	T_3	T_4
P_1	12	13	1	9
P_2	4	2	5	16
P_3	5	10	11	14
P_4	8	1	7	6

NSTMe	T_1	T_2	T_3	T_4
P_1	13	4	14	7
P_2	5	3	15	8
P_3	6	10	9	2
P_4	12	11	16	1

NSTCon	T_1	T_2	T_3	T_4
P_1	13	12		
P_2				
P_3				
P_4				

Chuyển sang giá trị kế tiếp của mặt nạ M.

- **Giá trị thứ ba,** $m[1,3] = 1$**.** Cá thể con sẽ nhận gen của cha, `NSTCha`. Các giá trị `NSTCha[1,1]`, `NSTCha[1,2]` đã xét nên lấy giá trị kế tiếp trong `NSTCha[1,3]` $= 1$, cá thể con chưa có gen này nên:

$$\text{NSTCon}[1,3] = 1,$$

và đánh dấu trong cá thể cha và mẹ gen số 1:

NSTCha	T_1	T_2	T_3	T_4
P_1	12	13	1	9
P_2	4	2	5	16
P_3	5	10	11	14
P_4	8	1	7	6

NSTMe	T_1	T_2	T_3	T_4
P_1	13	4	14	7
P_2	5	3	15	8
P_3	6	10	9	2
P_4	12	11	16	1

NSTCon	T_1	T_2	T_3	T_4
P_1	13	12	1	
P_2				
P_3				
P_4				

Tương tự các giá trị kế tiếp, kết quả cá thể con (`NSTCon`) sau khi lai tạo giữa cha `NSTCha`, mẹ `NSTMe`, dựa vào mặt nạ M như hình sau:

	T_1	T_2	T_3	T_4
P_1	13	12	1	4
P_2	14	7	9	5
P_3	2	3	15	8
P_4	6	16	10	11

Hình 3.3: Kết quả ví dụ sau khi thực hiện lai ghép

3.3.4 Đột biến

Trong bài toán, nhiễm sắc thể đại diện cho lời giải của bài toán và mỗi gen trong nhiễm sắc thể có một xác suất đột biến là p, ví dụ: $p = 0{,}03$ tức với 100 cá thể trong quần thể thì có $0{,}03 \times 100 = 3$ cá thể sẽ bị đột biến trong mỗi thế hệ, và quá trình đột biến được thực hiện bằng phương pháp đột biến tương hỗ bằng cách hoán vị 2 gen bất kỳ trong một nhiễm sắc thể.

Các bước thực hiện đột biến (gọi N là số cá thể trong quần thể, p là xác suất đột biến):

- **Bước 1:** Tính số cá thể sẽ bị đột biến.
  Số cá thể đột biến:
  $$K = N \times p.$$
- **Bước 2:** Với mỗi giá trị k ($k \in [1..K]$) thực hiện:
  - Xác định vị trí cá thể bị đột biến: sinh ngẫu nhiên số nguyên
    $$x \in [1..N] \quad (x : \text{vị trí cá thể trong quần thể}).$$
  - Với cá thể x, xác định vị trí gen đột biến bằng cách sinh ngẫu nhiên hai cặp số nguyên
    $$vt_1, vt_2 \in [1..m], \quad vt_3, vt_4 \in [1..h],$$
    với h là số tiết học/tuần, m là số phòng học.
  - Hoán vị hai cặp gen của cá thể x tại hai vị trí (vt_1, vt_2) và (vt_3, vt_4).
- **Bước 3:** Lặp lại bước 2 cho đến khi hết số cá thể bị đột biến.

Thuật toán được giải mã như sau:

```
Procedure Mutation()
Input : Quần thể cá thể (TKB), p
        {Quần thể các cá thể, p: xác suất đột biến}
        N : số cá thể trong quần thể
Output: Quần thể các cá thể sau khi bị đột biến (TKB')
Begin
  {Gọi vt1, vt2, vt3, vt4: là vị trí sẽ bị đột biến trong cá thể}
  K = N * p       // K: giá trị nguyên

  For each k in [1..K] do        // k: số nguyên (1 <= k <= K)
    x = randomInt(1, N)          // x: vị trí cá thể trong quần thể

    vt1 = randomInt(1, h)
    vt2 = randomInt(1, m)
    vt3 = randomInt(1, h)
    vt4 = randomInt(1, m)

    {Hoán vị hai vị trí gen trong một cá thể x}
    temp           = TKB_x[vt1, vt2]   // cá thể thứ x bị đột biến
    TKB_x[vt1,vt2] = TKB_x[vt3, vt4]
    TKB_x[vt3,vt4] = temp
  Endfor
End.
```

Thuật toán trên sử dụng hàm `randomInt(min, max)` trả về một giá trị nguyên thuộc đoạn $[min, max]$.

Ví dụ: Nhiễm sắc thể hình 3.4 có 2 gen bị đột biến tại vị trí số (1) và (8), được thực hiện bằng cách hoán vị giá trị của hai vị trí đó:

	T_1	T_{12}	T_{19}	T_{30}	T_{45}
P_1	$G_1, M_1, L_1^{(1)}$	$G_3, M_2, L_2^{(2)}$		$G_3, M_2, L_2^{(4)}$	$G_1, M_1, L_1^{(5)}$
P_2	$G_3, M_2, L_2^{(6)}$	$G_2, M_3, L_3^{(7)}$		$G_4, M_4, L_4^{(8)}$	$G_3, M_4, L_4^{(9)}$
P_3	$G_2, M_3, L_3^{(10)}$	$G_4, M_4, L_4^{(11)}$	$G_2, M_3, L_3^{(12)}$		
P_4			$G_3, M_4, L_4^{(17)}$		

Hai vị trí được hoán vị là:

$$(G_1, M_1, L_1)^{(1)} \quad \text{và} \quad (G_4, M_4, L_4)^{(8)}.$$

Sau khi hoán vị, ta thu được nhiễm sắc thể mới:

	T_1	T_{12}	T_{19}	T_{30}	T_{45}
P_1	$G_4, M_4, L_4^{(8)}$	$G_3, M_2, L_2^{(2)}$		$G_3, M_2, L_2^{(4)}$	$G_1, M_1, L_1^{(5)}$
P_2	$G_3, M_2, L_2^{(6)}$	$G_2, M_3, L_3^{(7)}$		$G_1, M_1, L_1^{(1)}$	$G_3, M_4, L_4^{(9)}$
P_3	$G_2, M_3, L_3^{(10)}$	$G_4, M_4, L_4^{(11)}$	$G_2, M_3, L_3^{(12)}$		
P_4			$G_3, M_4, L_4^{(17)}$		

Hình 3.4: Ví dụ cá thể bị đột biến

3.3.5 Hàm đánh giá

Trong bài toán, hàm đánh giá được đo mức độ lời giải thỏa mãn các yêu cầu của bài toán, tương ứng với mức độ thoả mãn các ràng buộc của bài toán. Mỗi ràng buộc sẽ được gán cho một giá trị thích nghi; phụ thuộc vào giá trị thích nghi sẽ quyết định có hay không nhiễm sắc thể (cá thể) được giữ lại cho các thế hệ sau.

Trong luận văn, hàm thích nghi sẽ được thực hiện đánh giá thông qua ràng buộc: ràng buộc phải thoả mãn $\{C\}$.

Một thời khoá biểu chấp nhận được thì phải thoả mãn tất cả các ràng buộc trong bài toán. Chúng ta định nghĩa tập các ràng buộc

$$C = \{C_1, C_2, C_3, C_4, C_5\}.$$

Tương ứng, xây dựng thuật toán đánh giá mức độ thoả mãn với các ràng buộc:

Ràng buộc C_1. Yêu cầu thoả mãn về phòng học, không xếp hai lớp cùng học một phòng vào cùng một thời điểm. Với cách tổ chức dữ liệu lời giải bài toán (nhiễm sắc thể) như đã trình bày trên, chúng ta đã thoả mãn được ràng buộc C_1, vì tương ứng với mỗi giá trị của ma trận chỉ có một và chỉ một sự kiện. Như vậy giá trị đánh giá cho ràng buộc loại này được xác định bằng:

$$C_1 = 0.$$

Ràng buộc C_2. Yêu cầu về giáo viên: một giáo viên không đồng thời dạy hai lớp cùng một thời điểm. Nếu với mỗi giáo viên bị vi phạm thì giá trị phạt được tăng lên một đơn vị.

Với ràng buộc C_2 chúng ta trình bày giải thuật kiểm tra sự thoả mãn ràng buộc như sau:

Bước 1: Với mỗi tiết học $T_i \in \{T\}$, $(i = 1..h)$:

- Đánh dấu tất cả các giáo viên là "chưa xét".
- Với mỗi phòng học $P_j \in \{P\}$, $(j = 1..m)$:
  - Lấy thông tin giáo viên tại phòng P_j:

$$gv \leftarrow TKB[i, j].$$

  - Nếu gv đã được xét thì tăng giá trị phạt.
    Ngược lại, đánh dấu gv là đã xét.
- Lặp lại cho đến khi xét hết các phòng.

Bước 2: Lặp lại Bước 1, cho đến khi các tiết học đều xét.

Bước 3: Trả về kết quả, và kết thúc thuật toán.

Thuật toán được giả mã như sau.

Thuật toán sử dụng mảng một chiều `giaovien[k]` kiểu `Boolean`, có kích thước bằng số giáo viên, mỗi giá trị k đại diện cho một giáo viên. Ví dụ: giá trị `giaovien[1]` đại diện cho G_1, `giaovien[2]` đại diện cho G_2 (trong đó $G_1, G_2 \in \{G\}$). Mảng dùng để kiểm tra trong một tiết học xem một giáo viên xuất hiện nhiều hơn một lần.

```
Function TinhGiaTriPhat_C2(TKB)
Input : Cá thể (TKB)
Output: Giá trị phạt (số nguyên)
Begin
  C2 = 0
  For i = 1 to h do         // h: số tiết học
    For each gvl in {G} do // l = 1..k
      giaovien[l] = false
    Endfor
    For j = 1 to m do       // m: số phòng học
      gvl = TKB[i,j]
      if giaovien[l] then  // nếu giáo viên bị trùng
        C2 = C2 + 1         // tăng giá trị phạt
      else
        giaovien[l] = true // đánh dấu giáo viên đã có
      endif
    Endfor
  Endfor
  Return C2                 // Trả về giá trị phạt của ràng buộc
End.
```

Ví dụ: Thời khoá biểu sau thể hiện vi phạm vào yêu cầu C_2, một giáo viên dạy hai lớp cùng một thời điểm. Ở đây tồn tại hai cặp vi phạm (T_1, P_2) và (T_1, P_4) của giáo viên G_3; và (T_{19}, P_1) và (T_{19}, P_4) của giáo viên G_3:

	T_1	T_{12}	T_{19}	T_{30}	T_{45}
P_1	G_1, M_1, L_1	G_3, M_2, L_2	G_3, M_2, L_2		G_1, M_1, L_1
P_2	G_3, M_2, L_2	G_2, M_3, L_3		G_4, M_4, L_4	
P_3	G_2, M_3, L_3	G_4, M_4, L_4	G_2, M_3, L_3		
P_4	G_3, M_4, L_4		G_3, M_4, L_4		

Hình 3.5: Ví dụ vi phạm ràng buộc C_2

Ví dụ ở hình 3.5, có hai cặp vi phạm ràng buộc C_2 là (T_1, P_2) và (T_1, P_4), và (T_{19}, P_1) và (T_{19}, P_4) của giáo viên G_3. Như vậy giá trị phạt của thời khoá biểu trên là 2.

Ràng buộc C_3. Mỗi phòng học có sức chứa và đặc điểm riêng của phòng, vì vậy sắp xếp lớp học vào các phòng sao cho đảm bảo chỗ ngồi cho sinh viên. Đối với yêu cầu này, mỗi thời khoá biểu phải thoả mãn về sức chứa, do đó phải kiểm tra sự thoả mãn của ràng buộc. Các bước thực hiện kiểm tra như sau:

Bước 1: Với mỗi giá trị $TKB[i, j]$, $i = 1..m$, $j = 1..h$:

- Xác định lớp `lop` tại $TKB[i, j]$, trong đó `lop` là nhóm sinh viên.
- Lấy khả năng chứa của phòng học thứ i.
- So sánh sĩ số của lớp học phần (nhóm sinh viên) và khả năng chứa phòng học thứ i.
  Nếu sĩ số lớp học phần `lop` > sức chứa của phòng học thứ i thì tăng giá trị phạt.

Bước 2: Lặp Bước 1, cho đến khi tất cả các giá trị đều được xét.

Bước 3: Trả về kết quả, dừng thuật toán.

Thuật toán được giả mã như sau.
Gọi tập

$$S = \{s_1, s_2, s_3, \ldots, s_m\}$$

là tập biểu diễn khả năng chứa của các phòng $(1..m)$ trong tập phòng $\{P\}$, mỗi giá trị s_i đại diện sức chứa cho phòng thứ i. Và gọi hàm `siso(lop)` cho biết sĩ số sinh viên của lớp `lop`.

```
Function TinhGiaTriPhat_C3(TKB)
Input : Cá thể (TKB)
Output: Giá trị phạt (số nguyên)
Begin
  C3 = 0
  For each TKB[i,j] do        // i = 1..m, j = 1..h
    lop = TKB[i,j]
    if (s_i < siso(lop)) then
      C3 = C3 + 1             // Tăng giá trị phạt
    endif
  Endfor
  Return C3                   // Trả về số lượng ràng buộc bị vi phạm
End.
```

Ràng buộc C_4. Mỗi chương trình tương ứng với các môn học và mỗi môn được quy định một số tín chỉ, mỗi tín chỉ được phân bố số tiết học trong tuần, vì vậy thời khoá biểu phải xếp đầy đủ các tiết học của môn học theo tuần. Yêu cầu: xếp các tiết học đảm bảo đủ số tiết cho mỗi môn học. Các bước thực hiện kiểm tra số lượng các tiết học trong tuần của môn được thực hiện như sau:

Gọi mảng số nguyên `dem_tiet[]` chứa số tiết học đã được xếp lịch tương ứng với từng môn, mỗi giá trị của mảng đại diện cho một môn học, ví dụ `dem_tiet[1]` đại diện cho môn học M_1, `dem_tiet[2]` cho môn M_2, với $M_1, M_2 \in \{M\}$.

Bước 1: Với mỗi giá trị $TKB[i, j]$, $i = 1..m$, $j = 1..h$:

- Xác định môn học M_k tại $TKB[i, j]$.
- Đếm số lượng tiết học tương ứng của môn M_k, và lưu trong mảng:

$$dem\_tiet[k] = dem\_tiet[k] + 1.$$

Lặp lại Bước 1, cho đến khi các giá trị đều được xét.

Bước 2: Với mỗi môn $M_i \in \{M\}$, $i = 1..t$:

- So sánh: nếu số tiết quy định học của môn $M_i >$ `dem_tiet[i]` (số tiết được xếp lịch) thì tăng giá trị phạt.

Lặp Bước 2, cho đến khi các môn đều được xét.

Bước 3: Trả về kết quả, dừng thuật toán.

Thuật toán được giả mã như sau.
Gọi

$$F = \{f_1, f_2, f_3, \ldots, f_t\}$$

là tập chứa số tiết học quy định của các môn học, mỗi giá trị f_i đại diện cho môn học M_i.

```
Function TinhGiaTriPhat_C4(TKB)
Input : Cá thể (TKB)
Output: Giá trị phạt (số nguyên)
Begin
  C4 = 0
  // Khởi tạo mảng đếm số tiết cho mỗi môn
  For i = 1 to t do           // mi in {M}
    dem_tiet[i] = 0
  Endfor

  For each TKB[i,j] do        // i = 1..m, j = 1..h
    mk = mon tai TKB[i,j]
    dem_tiet[k] = dem_tiet[k] + 1
  Endfor

  For i = 1 to t do           // t: số môn học
    if (dem_tiet[i] < f_i) then
      C4 = C4 + 1             // tăng giá trị phạt
    endif
  Endfor

  Return C4                   // trả về số lượng ràng buộc vi phạm
End.
```

Ràng buộc C_5. Trong học chế tín chỉ, mỗi chương trình được thiết kế bởi một chương trình khung; mỗi chương trình đều quy định một số môn học, một số chương trình có những môn học cơ bản có thể giống hoặc khác nhau. Thời khoá biểu được phân yêu cầu: các môn trong một chương trình đào tạo phải có thời gian học khác nhau (C_5). Các bước thực hiện kiểm tra vi phạm của ràng buộc C_5 được thực hiện như sau:

Gọi mảng `CT[]` có giá trị `boolean`, có kích thước bằng số lượng chương trình, mỗi giá trị của mảng đại diện cho một chương trình, ví dụ `CT[1]` đại diện cho CT_1, `CT[2]` đại diện CT_2, ... ($CT_1, CT_2 \in \{CT\}$).

Bước 1: Với mỗi tiết học $T_i \in \{T\}$, ($i = 1..h$):

- Đánh dấu tất cả các chương trình là "chưa xét" (`CT[k] = false`, $k = 1..l$).
- Với mỗi phòng học $P_j \in \{P\}$, ($j = 1..m$):
  - Lấy thông tin môn học `mon` tại phòng P_j:

$$mon \leftarrow TKB[i, j].$$

  - Xác định chương trình của môn `mon` (gọi là chương trình thứ k).
  - Nếu `CT[k]` đã được xét thì tăng giá trị phạt.
    Ngược lại, đánh dấu `CT[k]` là đã xét (`CT[k] = true`).
- Lặp lại cho đến khi xét hết các phòng.

Bước 2: Lặp lại Bước 1, cho đến khi các tiết học đều xét.

Bước 3: Trả về kết quả, và dừng thuật toán.

Thuật toán được giả mã như sau.
Gọi hàm `layCT(mi)` cho biết môn M_i thuộc chương trình nào và hàm trả về số nguyên.

```
Function TinhGiaTriPhat_C5(TKB)
Input : Cá thể (TKB)
Output: Giá trị phạt (số nguyên)
Begin
  C5 = 0
  For i = 1 to h do               // h: số tiết học
    For k = 1 to l do             // l: số chương trình
      CT[k] = false               // khởi tạo mảng CT
    Endfor
    For j = 1 to m do             // m: số phòng học
      mon = TKB[i,j]
      if (CT[layCT(mon)]) then    // kiểm tra trùng chương trình
        C5 = C5 + 1               // tăng giá trị phạt
      else
        CT[layCT(mon)] = true     // đánh dấu chương trình đã có
      endif
    Endfor
  Endfor
  Return C5
End.
```

Ví dụ: Giả sử ta có tập môn học của hai chương trình là

$$CT_1 = \{M_1, M_2, M_4, M_6\}, \qquad CT_2 = \{M_3, M_5\}.$$

Thời khoá biểu sau thể hiện vi phạm vào ràng buộc C_5 của CT_1. Các cặp môn vi phạm là (M_2, M_6) và (M_1, M_4):

	T_1	T_2	T_3	T_4
P_1	(L_1, M_2, G_6)	(L_3, M_{21}, G_2)		
P_2		(L_{11}, M_6, G_9)		(L_{11}, M_1, G_9)
P_3	(L_7, M_3, G_7)		(L_1, M_4, G_8)	
P_4	(L_{10}, M_6, G_5)			(L_5, M_4, G_7)
P_5		(L_9, M_8, G_7)		
P_6		(L_6, M_1, G_8)		
P_7			(L_{10}, M_6, G_3)	

Hình 3.6: Ví dụ vi phạm ràng buộc C_5

Ví dụ ở hình 3.6, có hai cặp môn vi phạm ràng buộc C_5 là (M_2, M_6) và (M_1, M_4) cùng thuộc một chương trình. Như vậy giá trị phạt của thời khoá biểu trên là 2.

Hàm đánh giá tổng quát. Một thời khoá biểu được gọi là khả thi khi các ràng buộc được thoả mãn. Trong giải thuật di truyền, để đánh giá một cá thể (nhiễm sắc thể) chúng ta sẽ đánh giá các ràng buộc thông qua hàm đánh giá, vì vậy tương ứng với mỗi ràng buộc sẽ được gán một giá trị thích nghi, tuỳ thuộc vào loại ràng buộc mà người sử dụng cho điểm đánh giá. Trong đồ án, xin trình bày một phương pháp đánh giá: trong thời khoá biểu, tương ứng với một yêu cầu vi phạm thì được gán một điểm.

Giai đoạn quyết định giải thuật là đánh giá một lịch học của một cơ sở có thoả mãn các yêu cầu của bài toán hay không. Cứ mỗi ràng buộc bị vi phạm thì chúng ta cho giá trị phạt là 1 điểm. Như vậy, tổng hợp giá trị thích nghi của các ràng buộc được trình bày như trên, chúng ta khái quát thành công thức như sau:

$$F(x) = \sum_{i=1}^{5} w_i \cdot C_i, \tag{3.1}$$

trong đó:

- w_i: trọng số đánh giá mức độ quan trọng của ràng buộc thứ i, $w_i \in [0, 1]$,

$$\sum_{i=1}^{5} w_i = 1;$$

- C_i: tương ứng với giá trị phạt của ràng buộc thứ i;
- x: thời khoá biểu cần đánh giá.

Như vậy, mục tiêu của hàm đánh giá $F(x)$ là đạt được giá trị nhỏ nhất. Yêu cầu của bài toán là phải thoả mãn được tất cả các ràng buộc, tức là:

$$F(x) = 0.$$

CHƯƠNG IV – Xây dựng ứng dụng có tích hợp chức năng lập Thời khóa biểu

Trong chương này, chúng ta trình bày việc xây dựng một ứng dụng web hỗ trợ quản lý và lập Thời khóa biểu tự động cho trường Đại học, dựa trên mô hình **client–server** với kiến trúc **frontend–backend–database** tách biệt. Ứng dụng được triển khai với:

- **Backend**: sử dụng **FastAPI** (Python) để xây dựng RESTful API, hiện thực các chức năng nghiệp vụ (quản lý giáo viên, lớp học phần, phòng học, ràng buộc lịch, thuật toán di truyền lập Thời khóa biểu, ...).
- **Frontend**: sử dụng **React** để xây dựng giao diện người dùng, cho phép cán bộ đào tạo và quản trị viên thao tác trực quan với dữ liệu (nhập thông tin, thiết lập tham số GA, xem và chỉnh sửa lịch, ...).
- **Cơ sở dữ liệu**: sử dụng **SQLite** làm hệ quản trị CSDL quan hệ nhúng, lưu trữ toàn bộ dữ liệu của hệ thống trong một tệp duy nhất, đơn giản cho việc triển khai và phù hợp với quy mô đồ án.

Mô hình tổng quát: phía **React** gửi request HTTP (REST API) tới **FastAPI**; backend xử lý nghiệp vụ, truy vấn/ghi dữ liệu vào **SQLite** thông qua ORM (ví dụ SQLAlchemy), thực hiện thuật toán di truyền để sinh lịch học, sau đó trả kết quả JSON về cho frontend hiển thị.

4.1. Công nghệ sử dụng

4.1.1 FastAPI

FastAPI là một web framework hiện đại cho **Python**, được thiết kế để xây dựng **API web hiệu năng cao** một cách nhanh chóng và dễ dàng. FastAPI tận dụng **ASGI, type hints và Pydantic** để cung cấp cơ chế kiểm tra dữ liệu tự động, tài liệu hoá API theo chuẩn OpenAPI và hỗ trợ lập trình bất đồng bộ với `async/await`.

- **Hiệu năng cao và bất đồng bộ:** Dựa trên ASGI (Starlette) và sử dụng cơ chế `async/await` của Python, FastAPI cho phép xử lý lượng lớn request đồng thời, phù hợp với các hệ thống tải cao, microservices và ứng dụng real-time.

- **Type hints và validation tự động:** Sử dụng type hints của Python kết hợp với Pydantic để định nghĩa schema dữ liệu. Dữ liệu từ request (body, query, path, header) được **kiểm tra và ràng buộc tự động**, giảm lỗi runtime và giúp code rõ ràng, dễ bảo trì.

- **Tự sinh tài liệu API:** Tự động tạo tài liệu API tương tác theo chuẩn OpenAPI và JSON Schema. FastAPI cung cấp sẵn giao diện **Swagger UI** tại `/docs` và **ReDoc** tại `/redoc`, hỗ trợ test và mô tả API một cách trực quan.

- **Routing rõ ràng và RESTful:** Định nghĩa endpoint bằng các decorators như `@app.get`, `@app.post`, `@app.put`, `@app.delete` gắn với từng đường dẫn cụ thể, hỗ trợ path parameters, query parameters và request body, giúp xây dựng **RESTful API** rõ ràng và dễ bảo trì.

- **Kiến trúc module và khả năng mở rộng:** Cho phép tổ chức ứng dụng thành nhiều router, module, lớp service và schema riêng biệt. Kiến trúc này giúp tách biệt các chức năng (người dùng, xác thực, phân công giảng dạy, lập lịch, ...), thuận tiện cho việc mở rộng, bảo trì và làm việc nhóm.

- **Tích hợp hệ sinh thái Python:** Dễ dàng kết nối với các thư viện như **SQLAlchemy, Alembic, Celery, Redis** và các thư viện AI/ML (NumPy, Pandas, TensorFlow, PyTorch), phù hợp cho việc xây dựng backend API, hệ thống xử lý dữ liệu và triển khai mô hình học máy.

- **Bảo mật và xác thực:** Hỗ trợ tích hợp **OAuth2, JWT, API key, cookies** thông qua cơ chế dependency injection, cho phép xây dựng các cơ chế đăng nhập, phân quyền và bảo vệ tài nguyên API một cách linh hoạt và an toàn.

Trong đề tài, FastAPI được sử dụng để:

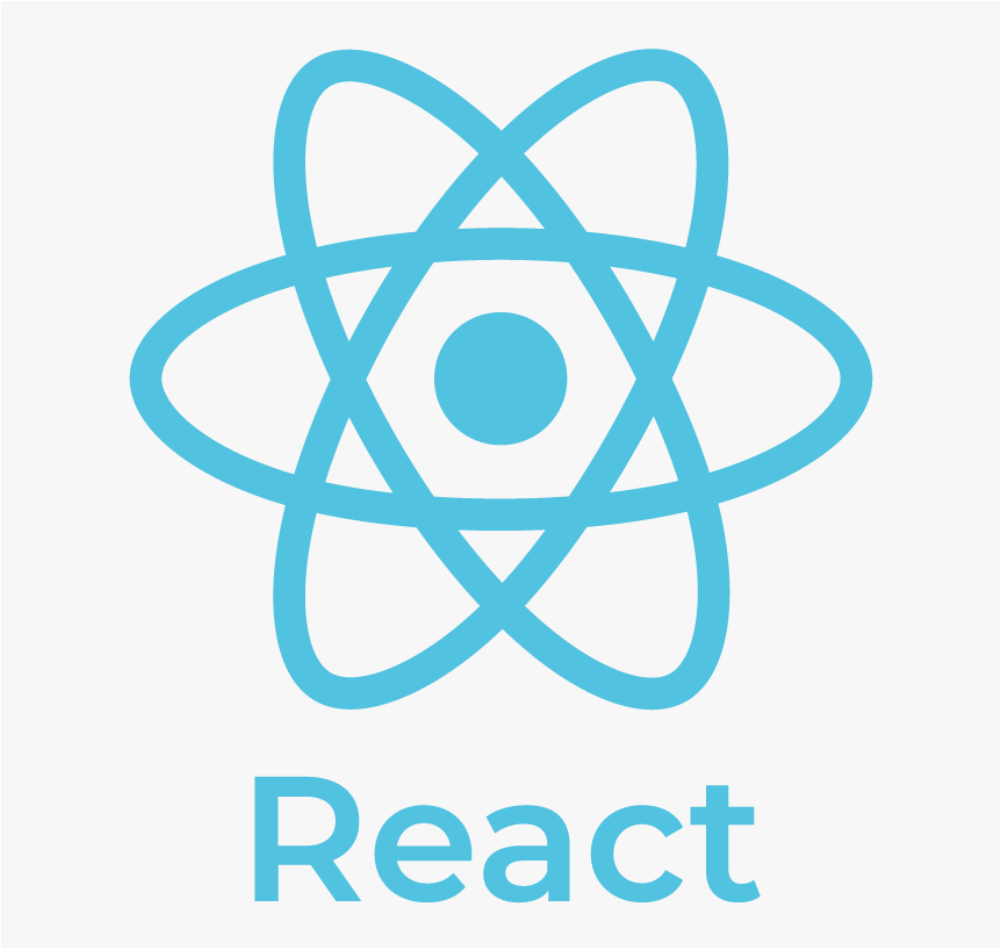
- Cung cấp các API quản lý dữ liệu cơ bản: giáo viên, sinh viên, lớp học phần, phòng học, học phần, chương trình đào tạo, ...

- Cài đặt các API phục vụ thuật toán di truyền: khởi tạo quần thể, chạy GA, trả về thời khóa biểu đề xuất, lưu lời giải tối ưu xuống cơ sở dữ liệu.

- Xử lý xác thực người dùng (quản trị, cán bộ đào tạo) và phân quyền truy cập đối với các API quan trọng.

4.1.2 React

React là một thư viện **JavaScript** dùng để xây dựng **giao diện người dùng (UI)**, được phát triển bởi Facebook và được cộng đồng sử dụng rộng rãi. React tập trung vào việc xây dựng **component** giao diện có thể tái sử dụng, quản lý trạng thái hiệu quả và cập nhật UI tối ưu cho các ứng dụng web hiện đại.



- **Kiến trúc component-based:** Giao diện được chia thành các **component** nhỏ, độc lập như nút bấm, form, layout, danh sách, ... Mỗi component có thể tái sử dụng và kết hợp lại để tạo ra các màn hình phức tạp, giúp tổ chức mã nguồn rõ ràng và dễ bảo trì.
- **Virtual DOM và tối ưu hiệu năng:** React sử dụng **Virtual DOM** để so sánh sự thay đổi trạng thái và chỉ cập nhật những phần tử thực sự cần thiết trên DOM thật, giúp **giảm số lần thao tác DOM** và cải thiện hiệu năng, mang lại trải nghiệm mượt mà cho người dùng.
- **Quản lý state linh hoạt:** Cho phép quản lý **state** bên trong từng component và truyền dữ liệu giữa các component thông qua props. Đối với ứng dụng lớn, có thể sử dụng thêm **Context API, Redux, Zustand, MobX** để quản lý trạng thái tập trung, đảm bảo luồng dữ liệu nhất quán.
- **JSX và khả năng biểu đạt cao:** Sử dụng **JSX** để viết mã giao diện dạng kết hợp giữa JavaScript và HTML, giúp mô tả cấu trúc UI và logic xử lý trong cùng một file, tăng tính trực quan và dễ bảo trì cho mã nguồn.
- **Single Page Application (SPA):** Kết hợp với các thư viện như **React Router** để xây dựng **SPA**, cho phép chuyển trang mượt mà trên client mà không cần tải lại toàn bộ trang, nâng cao trải nghiệm người dùng và giảm tải cho server.
- **Tích hợp API backend:** Dễ dàng kết nối với các backend như **FastAPI, NestJS, Node.js, Django** thông qua **REST API** hoặc **GraphQL**. React có thể gửi request tới backend, nhận dữ liệu và cập nhật state để hiển thị lên giao diện.

- **Hệ sinh thái phong phú:** Sở hữu cộng đồng lớn và nhiều thư viện hỗ trợ (UI kits, charts, form validation, internationalization, ...), giúp rút ngắn thời gian phát triển giao diện và đáp ứng tốt các yêu cầu thực tế của dự án.

Trong đề tài, React được dùng để:

- Xây dựng các màn hình nhập liệu (giáo viên, lớp học phần, phòng học, ràng buộc, tham số GA).
- Hiển thị kết quả lập Thời khóa biểu dưới dạng bảng, lịch tuần, cho phép xem chi tiết, lọc và chỉnh sửa thủ công nếu cần.
- Giao tiếp với backend FastAPI qua HTTP (fetch/axios), nhận dữ liệu JSON và cập nhật state để đồng bộ giao diện với dữ liệu hệ thống.

4.1.3 SQLite

SQLite là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) dạng **nhúng**, được thiết kế gọn nhẹ và dễ tích hợp trực tiếp vào trong ứng dụng. Khác với các hệ quản trị cơ sở dữ liệu kiểu client–server như MySQL hay PostgreSQL, SQLite không cần một máy chủ CSDL riêng mà lưu trữ toàn bộ dữ liệu trong một tệp duy nhất trên hệ thống file.

- **Cơ sở dữ liệu nhúng, không cần server:** SQLite hoạt động như một thư viện nhúng trong ứng dụng, toàn bộ dữ liệu được lưu trong một file `.db` hoặc `.sqlite`. Điều này giúp đơn giản hoá quá trình cài đặt, cấu hình và triển khai, đặc biệt phù hợp cho các ứng dụng nhỏ, ứng dụng di động, desktop và các bài tập, đồ án.
- **Mô hình quan hệ và SQL chuẩn:** Mặc dù nhẹ, SQLite vẫn tuân theo mô hình cơ sở dữ liệu quan hệ, hỗ trợ bảng (table), cột (column), dòng (row) và các kiểu dữ liệu cơ bản. SQLite sử dụng ngôn ngữ truy vấn **SQL chuẩn** để tạo bảng, chèn, sửa, xóa và truy vấn dữ liệu với các mệnh đề như `WHERE`, `JOIN`, `GROUP BY`, `ORDER BY`.
- **Hỗ trợ ràng buộc và toàn vẹn dữ liệu:** Cho phép định nghĩa **khóa chính, khóa ngoại, UNIQUE, NOT NULL** và các ràng buộc khác, giúp đảm bảo tính toàn vẹn dữ liệu ở mức ứng dụng nhỏ và vừa, đáp ứng tốt yêu cầu lưu trữ của đa số đồ án tốt nghiệp.
- **Tích hợp dễ dàng với Python và FastAPI:** SQLite được hỗ trợ sẵn trong Python thông qua module `sqlite3` và tương thích tốt với các ORM như **SQLAlchemy**. Khi kết hợp với FastAPI, ứng dụng backend có thể kết nối tới tệp SQLite, định nghĩa các model tương ứng với bảng và thao tác dữ liệu một cách thuận tiện.
- **Phù hợp cho bài tập và đồ án:** Với ưu điểm **cài đặt đơn giản, cấu hình nhẹ**, SQLite rất phù hợp cho môi trường học thuật và các hệ thống quy mô nhỏ. Sinh viên có thể tập trung vào thiết kế kiến trúc hệ thống (frontend–backend–database) mà không mất nhiều thời gian cho việc quản trị server cơ sở dữ liệu.

Trong ứng dụng lập Thời khóa biểu, SQLite được sử dụng để lưu trữ:

- Thông tin giáo viên, sinh viên, lớp học phần, phòng học, học phần, chương trình đào tạo.
- Các tham số, cấu hình ràng buộc, kết quả quần thể và lịch tối ưu thu được từ thuật toán di truyền.
- Lịch sử các lần chạy thuật toán, giúp so sánh và lựa chọn phương án lịch tốt nhất.

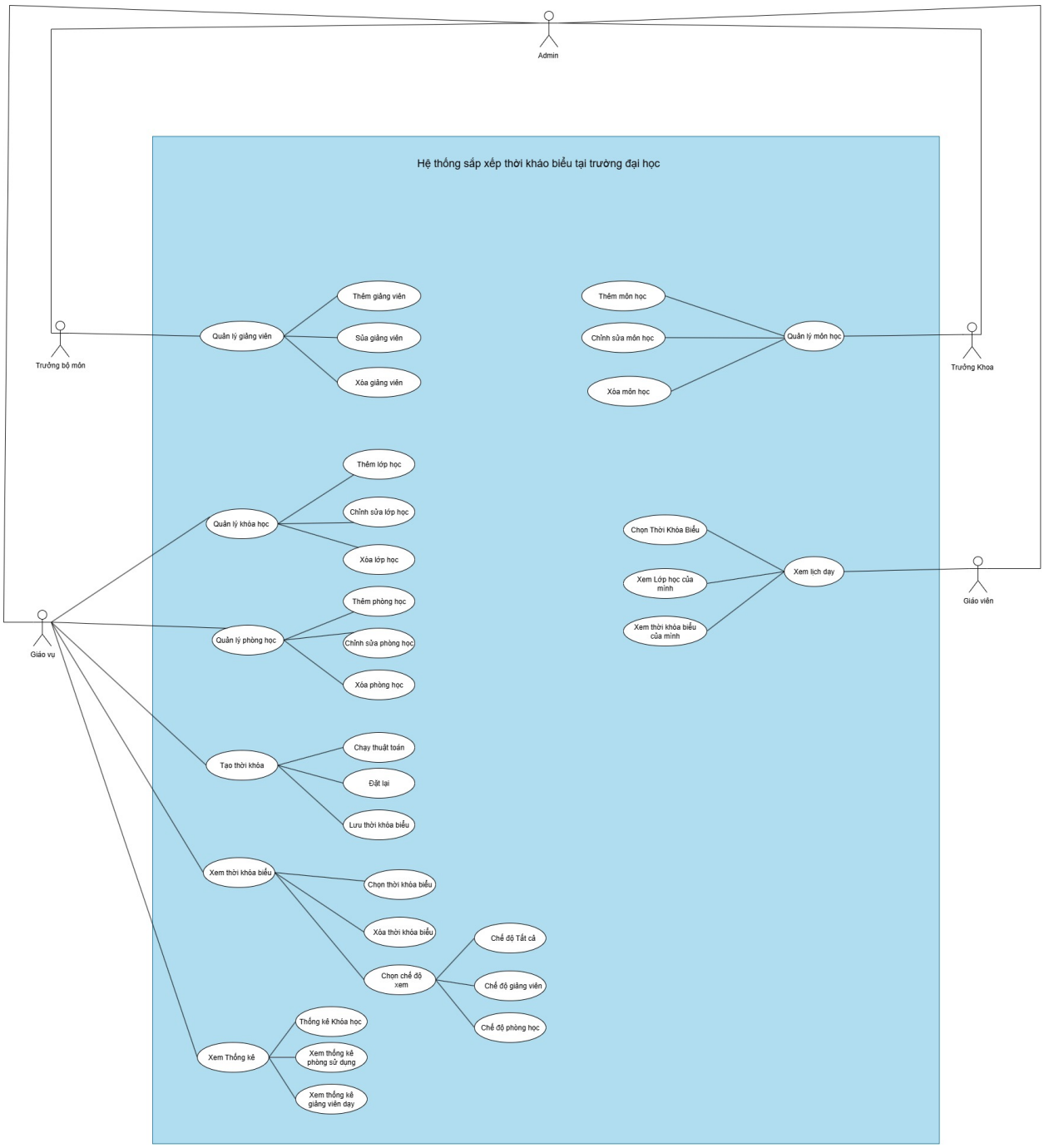
4.2. Kiến trúc hệ thống

Ứng dụng lập Thời khóa biểu được thiết kế theo mô hình ba lớp **Frontend** – **Backend** – **Database** với cơ chế trao đổi dữ liệu thông qua **REST API** (định dạng JSON) qua giao thức HTTP/HTTPS. Kiến trúc tổng quát được minh họa trên Hình 4.1.

Hình 4.1: Kiến trúc hệ thống ứng dụng lập Thời khóa biểu

4.3. Use case tổng quát của hệ thống

Các chức năng của hệ thống được thiết kế xoay quanh ba nhóm người dùng chính: **Admin**, **Trưởng bộ môn/Trưởng khoa** và **Giáo vụ/Giảng viên**. Biểu đồ use case tổng quát được minh họa trên Hình 4.2.



Hình 4.2: Biểu đồ use case tổng quát của hệ thống sắp xếp Thời khóa biểu

Dựa trên Hình 4.2, có thể mô tả các nhóm chức năng chính như sau:

Nhóm chức năng dành cho Admin

- Quản lý người dùng hệ thống (tạo tài khoản, phân quyền cho Trưởng bộ môn, Trưởng khoa, Giáo vụ, Giảng viên).
- Cấu hình các tham số hệ thống chung (năm học, học kỳ, khoảng thời gian sử dụng phòng, . . .).

Nhóm chức năng dành cho Trưởng bộ môn / Trưởng khoa

- **Quản lý giảng viên:** thêm giảng viên, sửa thông tin giảng viên, xóa giảng viên.
- **Quản lý môn học:** thêm môn học, chỉnh sửa môn học, xóa môn học.
- Kiểm tra và phê duyệt thông tin đầu vào (danh sách lớp, số lượng sinh viên, phân công giảng dạy) trước khi chạy thuật toán lập lịch.

Nhóm chức năng dành cho Giáo vụ

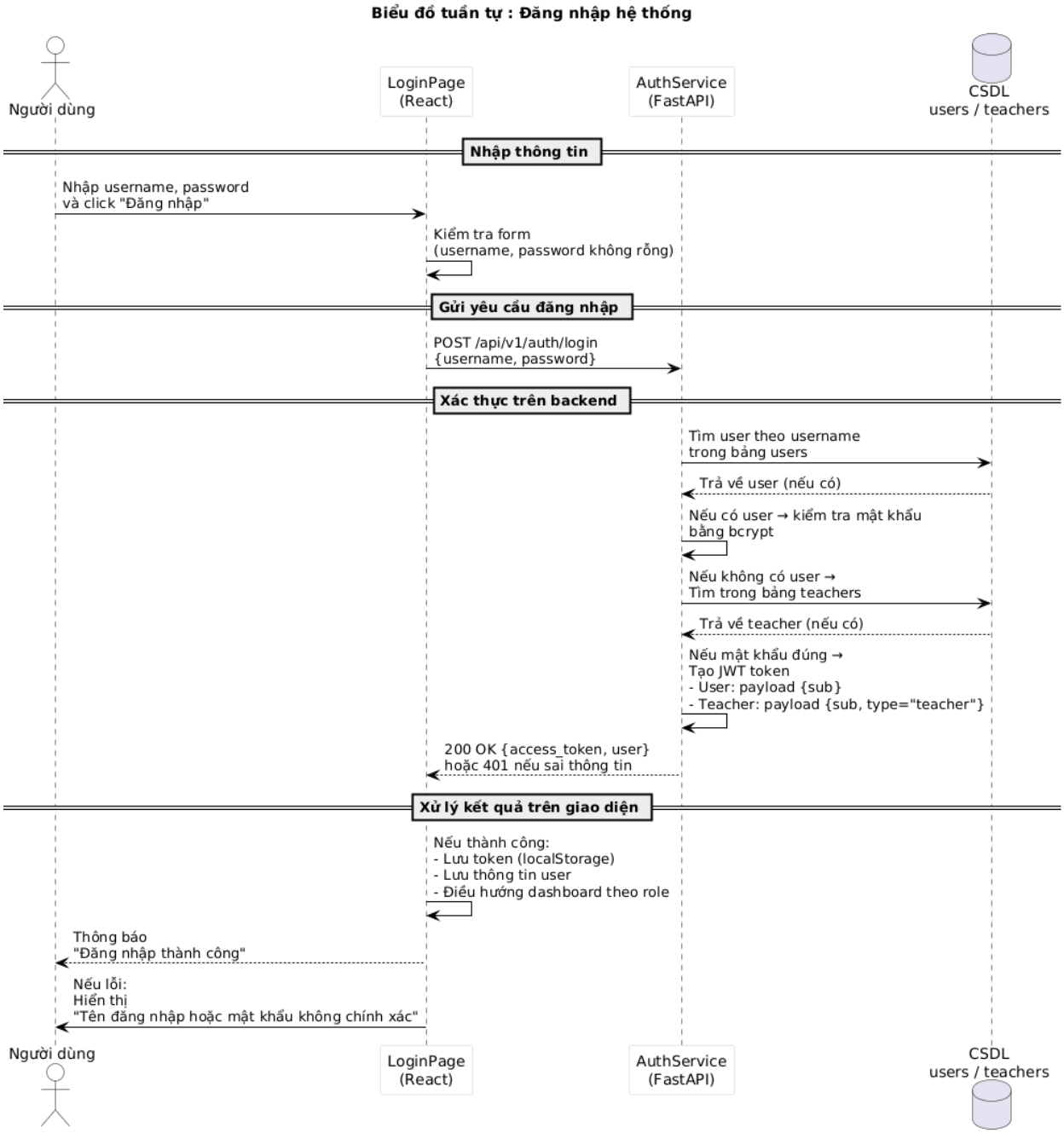
- **Quản lý lớp học:** thêm lớp học, chỉnh sửa lớp học, xóa lớp học.
- **Quản lý phòng học:** thêm phòng học, chỉnh sửa phòng học, xóa phòng học.
- **Tạo Thời khóa biểu:**
  - Nhập/kiểm tra dữ liệu phân công giảng dạy.
  - Cấu hình tham số thuật toán di truyền (kích thước quần thể, số thế hệ, xác suất lai ghép, xác suất đột biến, trọng số ràng buộc, ...).
  - Khởi chạy thuật toán, theo dõi tiến trình tối ưu.
  - Lưu Thời khóa biểu sau khi đạt tiêu chí chấp nhận.
- **Xem và chỉnh sửa Thời khóa biểu:**
  - Chọn Thời khóa biểu cần xem/xem lại.
  - Xem TKB theo nhiều chế độ: theo tổ/nhóm lớp, theo giảng viên, theo phòng học.
  - Chỉnh sửa thủ công (di chuyển tiết, đổi phòng, đổi ca) trong phạm vi cho phép, kèm theo kiểm tra lại các ràng buộc.
  - Xem thống kê khóa học, thống kê sử dụng phòng, thống kê khối lượng giảng dạy.

Nhóm chức năng dành cho Giảng viên

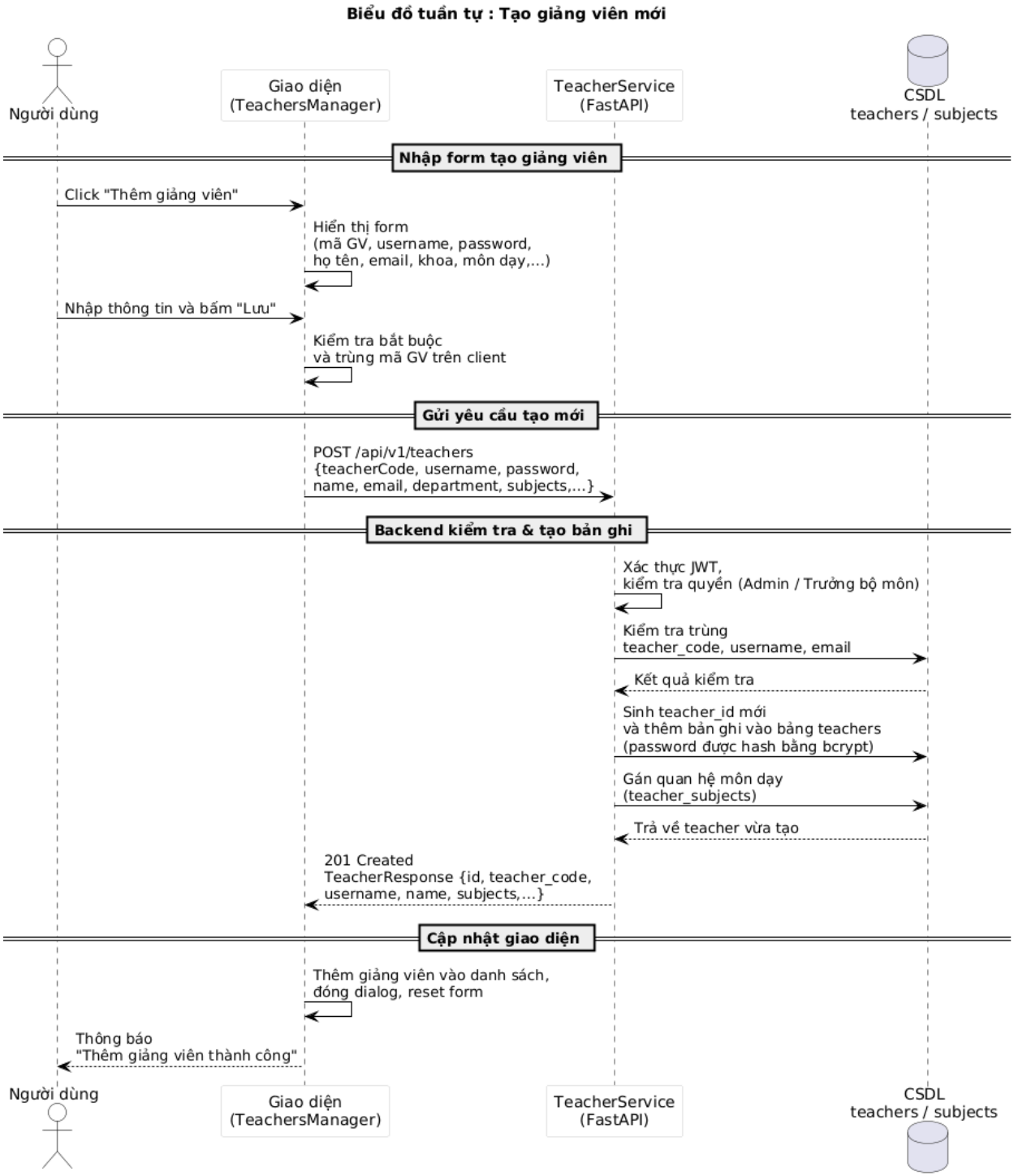
- Xem Thời khóa biểu cá nhân (theo tuần, theo học kỳ).
- Xem danh sách lớp học được phân công giảng dạy.
- Tra cứu thông tin chi tiết từng lớp (sĩ số, phòng học, thời gian, ...).

Các use case trên sẽ được hiện thực thông qua các màn hình React tương ứng và các API của FastAPI, dựa trên mô hình kiến trúc đã trình bày ở mục trước.“‘

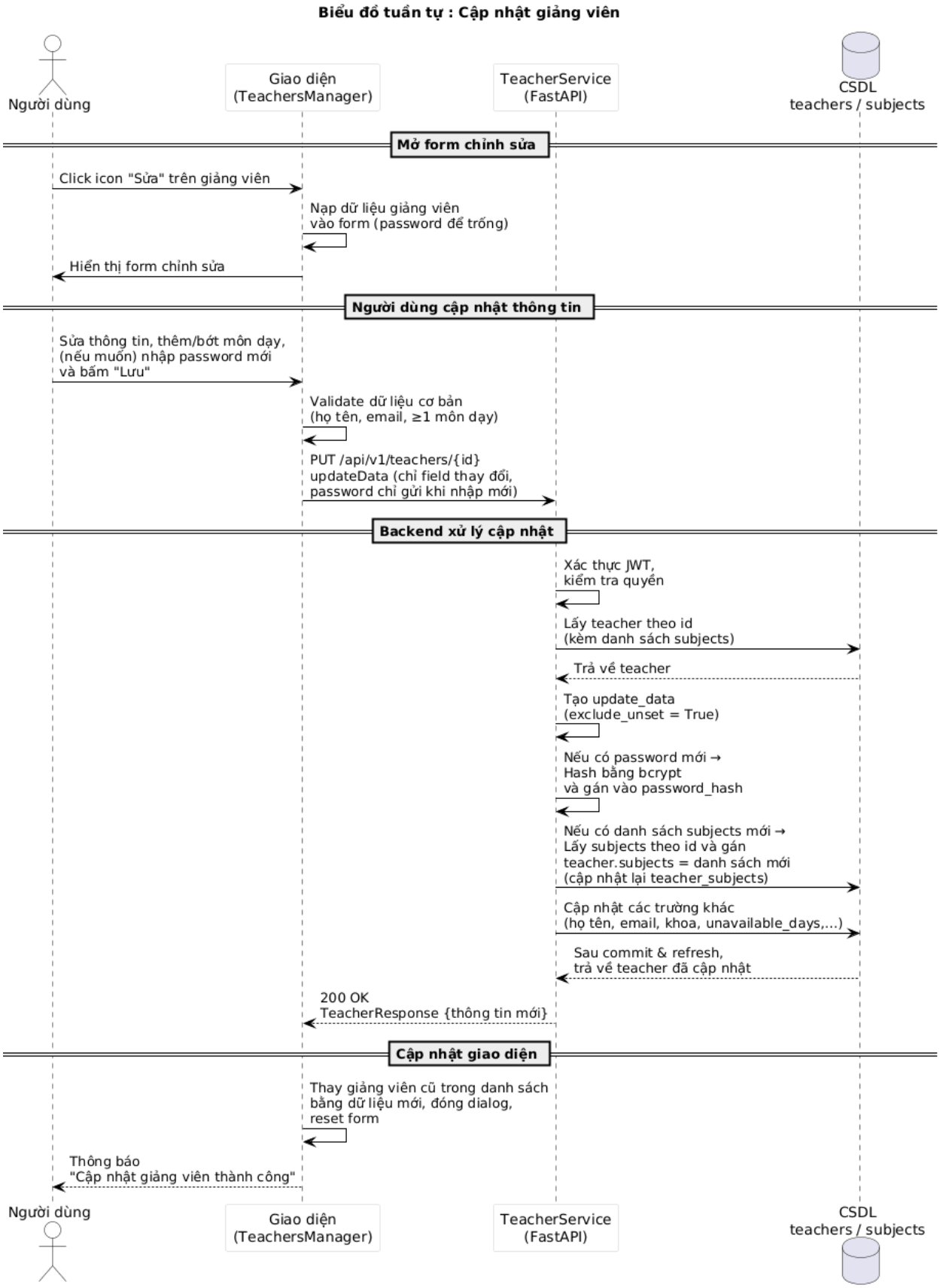
4.4. Sequence Diagrams cho từng UseCase



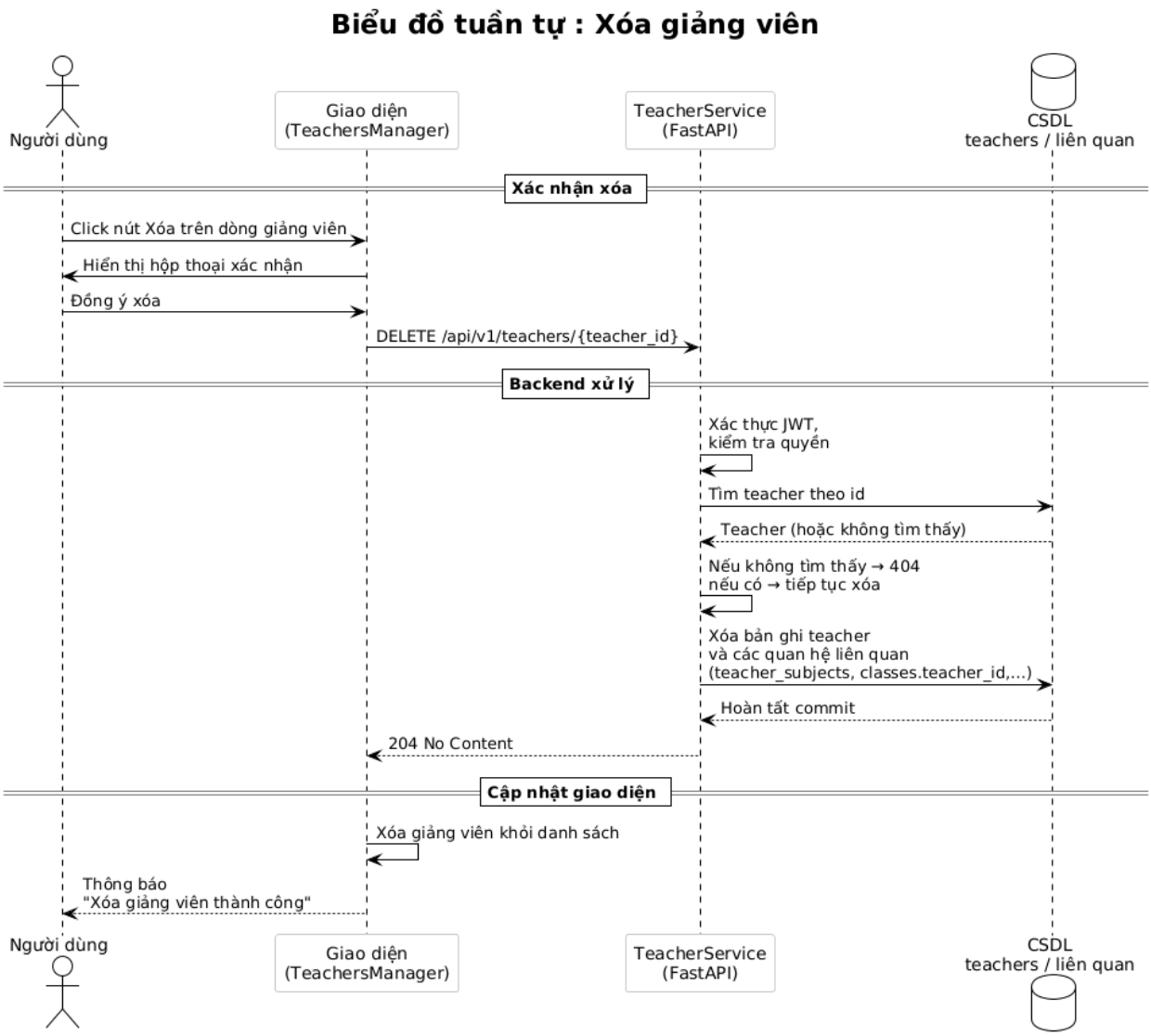
Hình 4.3: Biểu đồ tuần tự đăng nhập



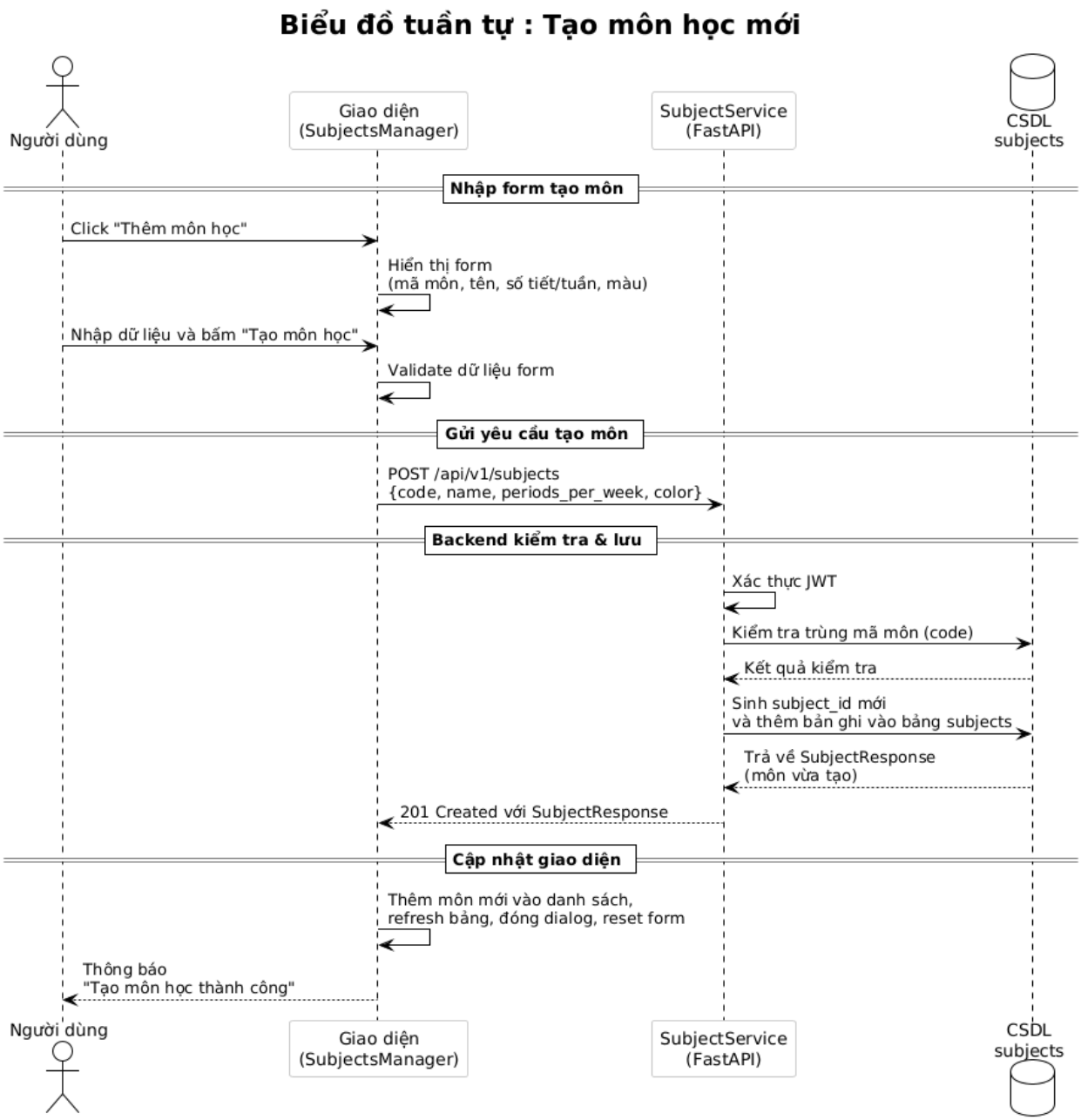
Hình 4.4: Biểu đồ tuần tự tạo giảng viên



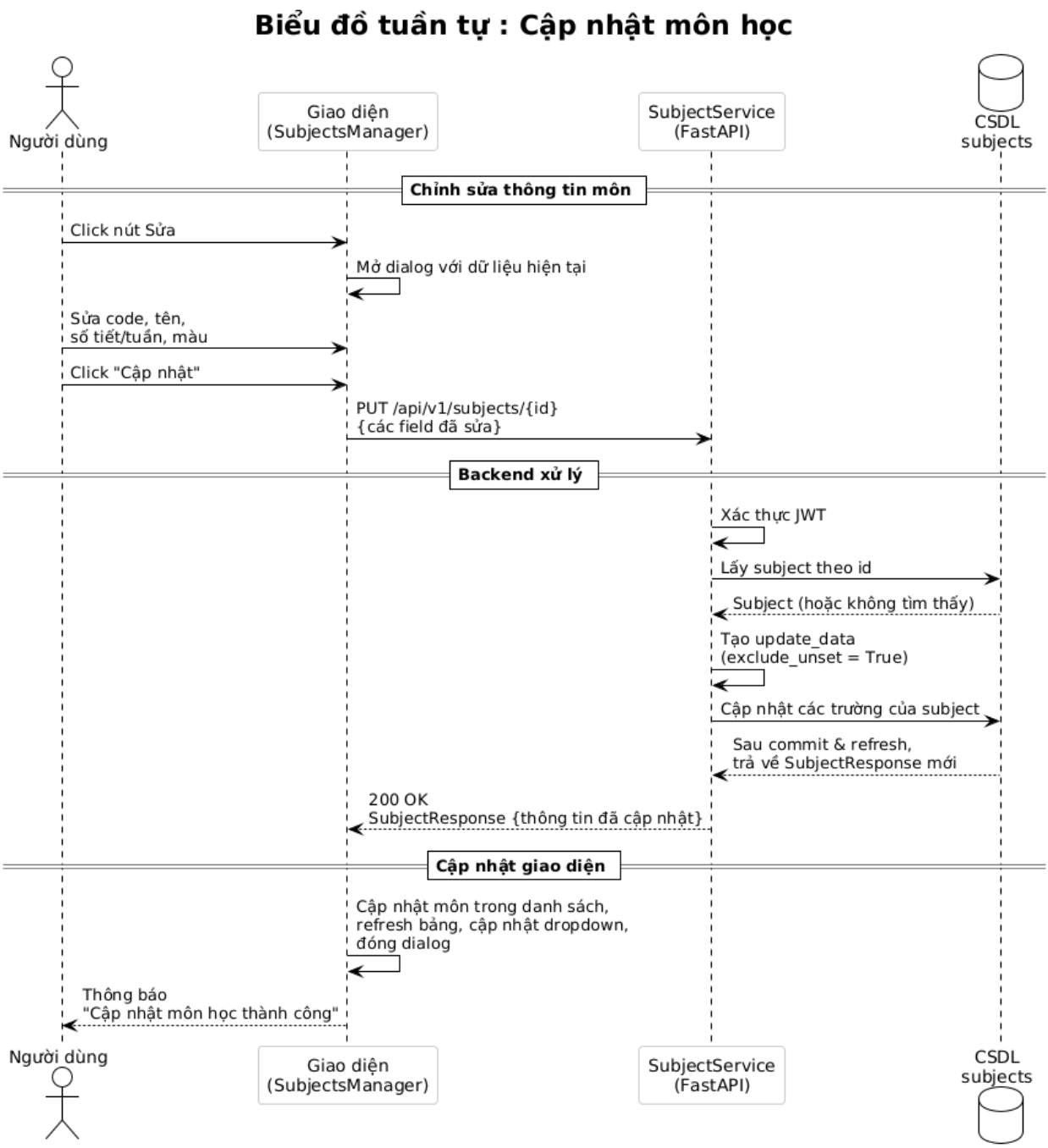
Hình 4.5: Biểu đồ tuần tự cập nhật giảng viên



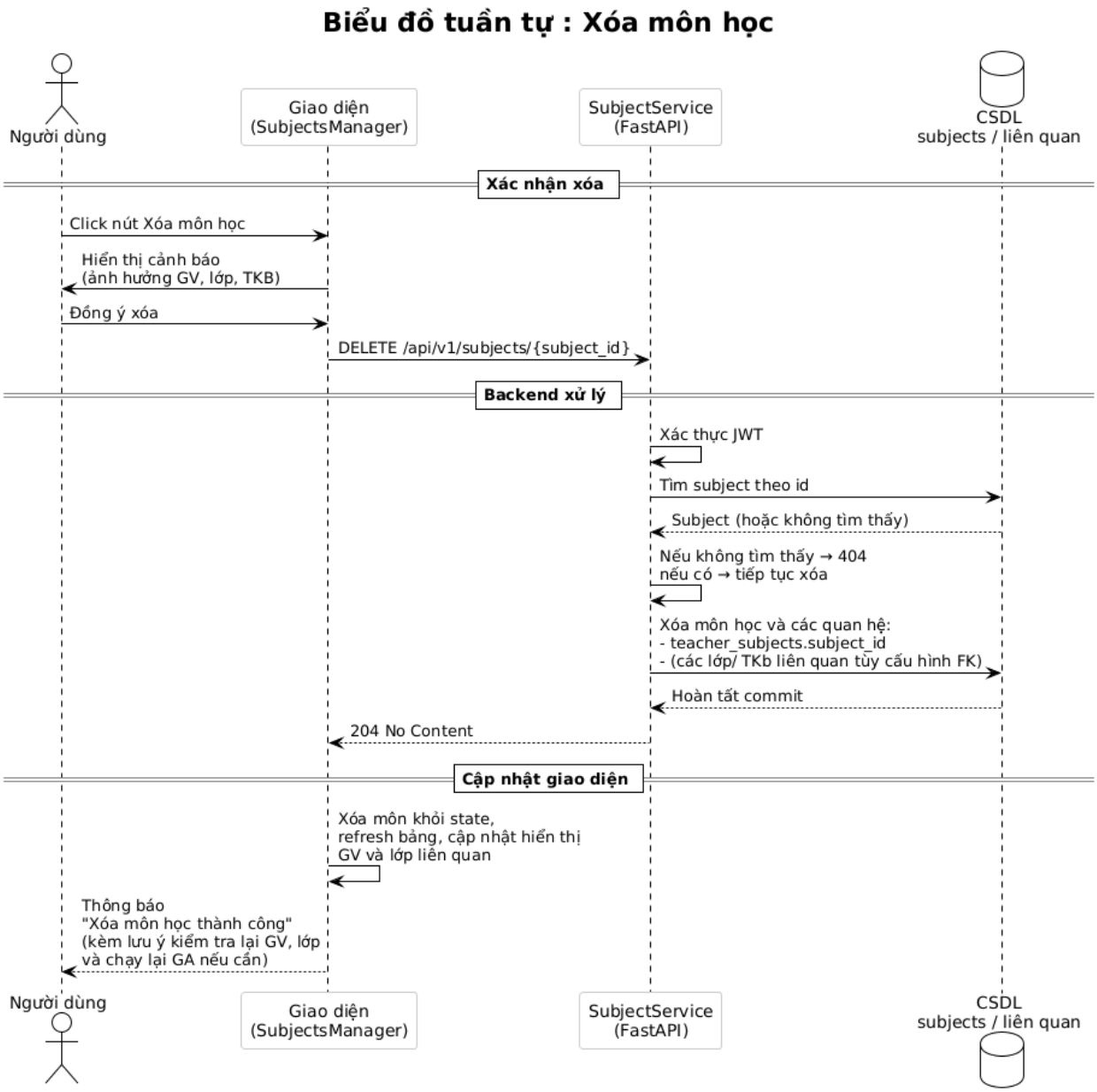
Hình 4.6: Biểu đồ tuần tự Xóa giảng viên



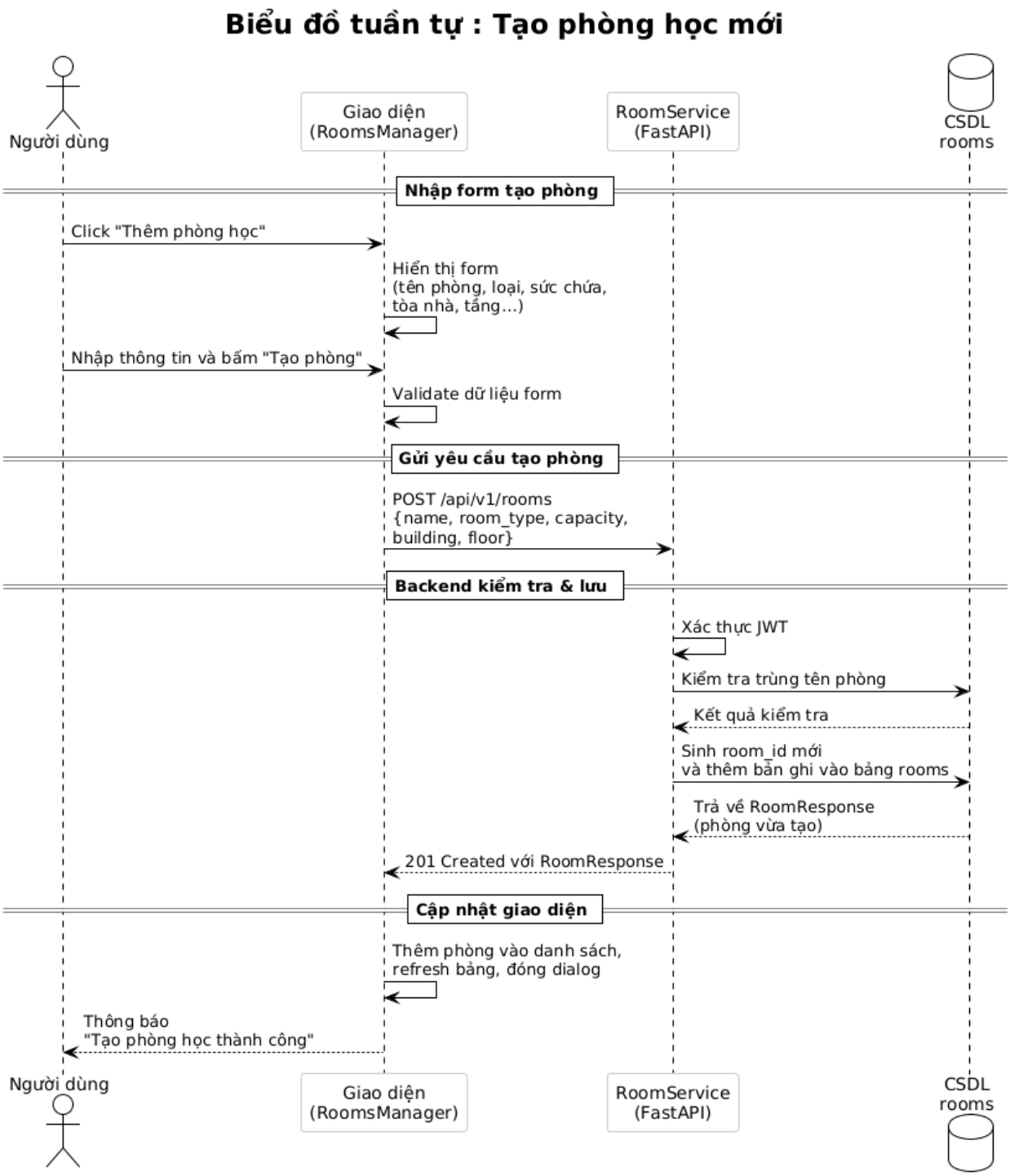
Hình 4.7: Biểu đồ tuần tự tạo môn học



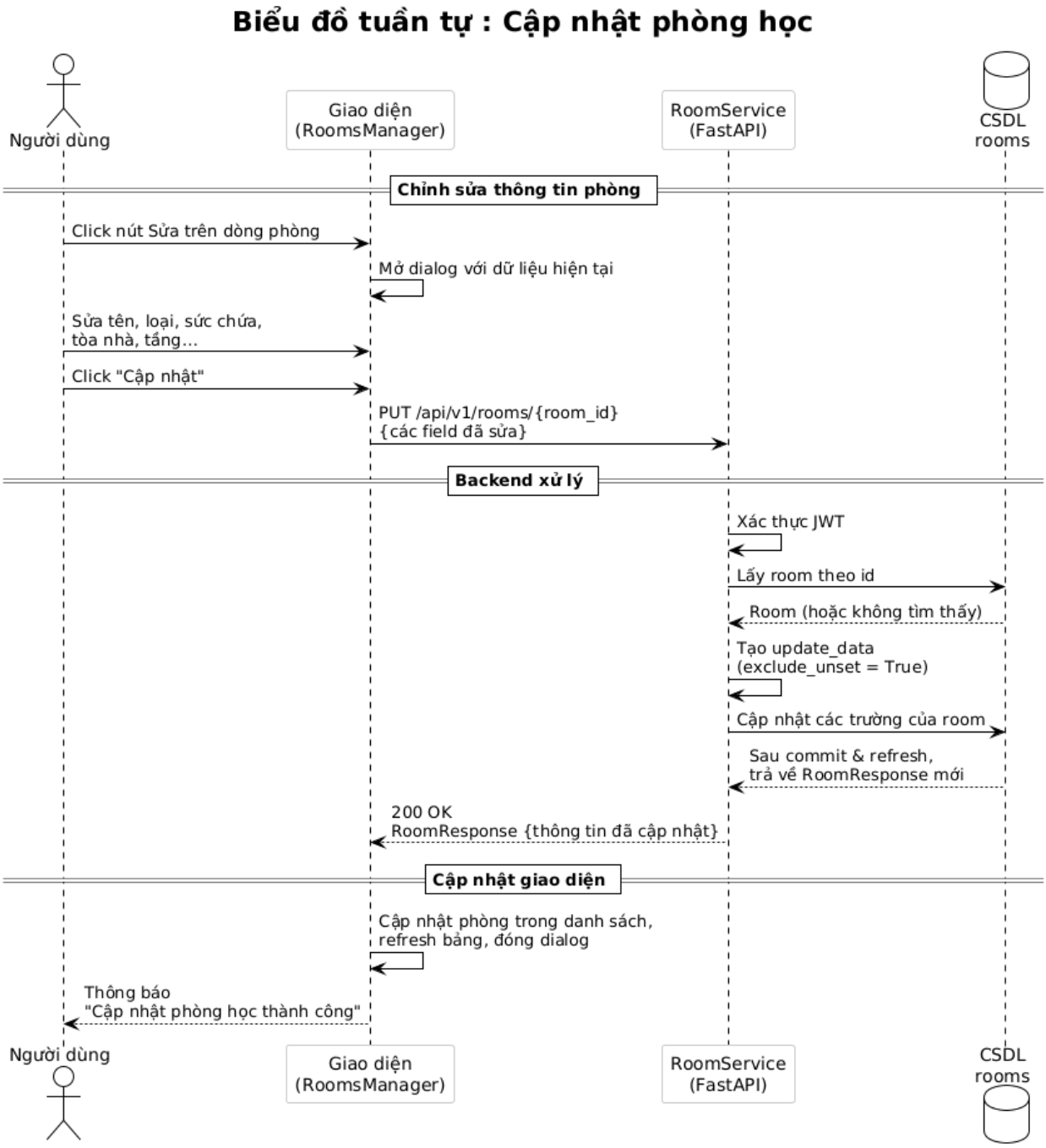
Hình 4.8: Biểu đồ tuần tự cập nhật môn học



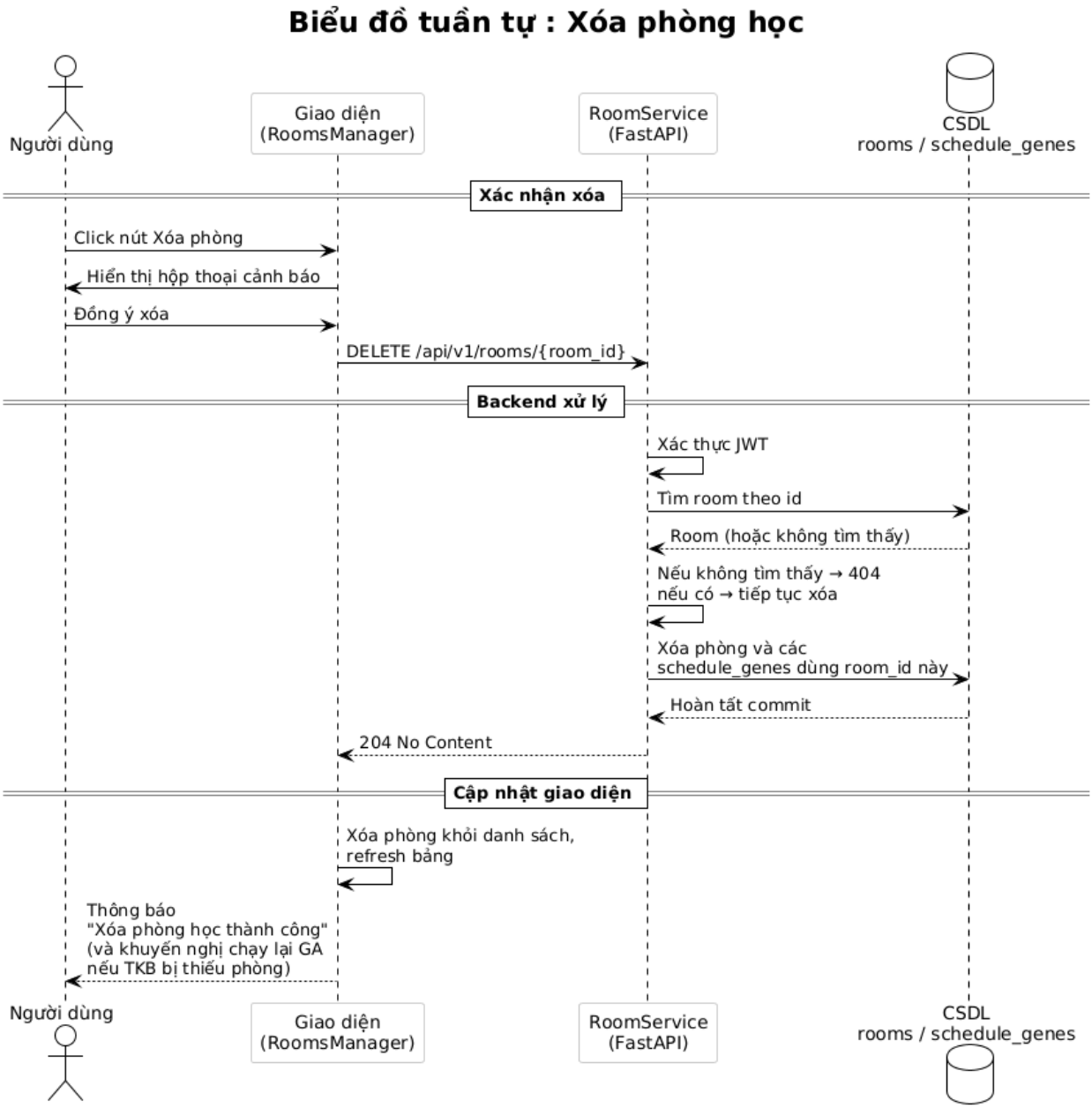
Hình 4.9: Biểu đồ tuần tự xóa môn học



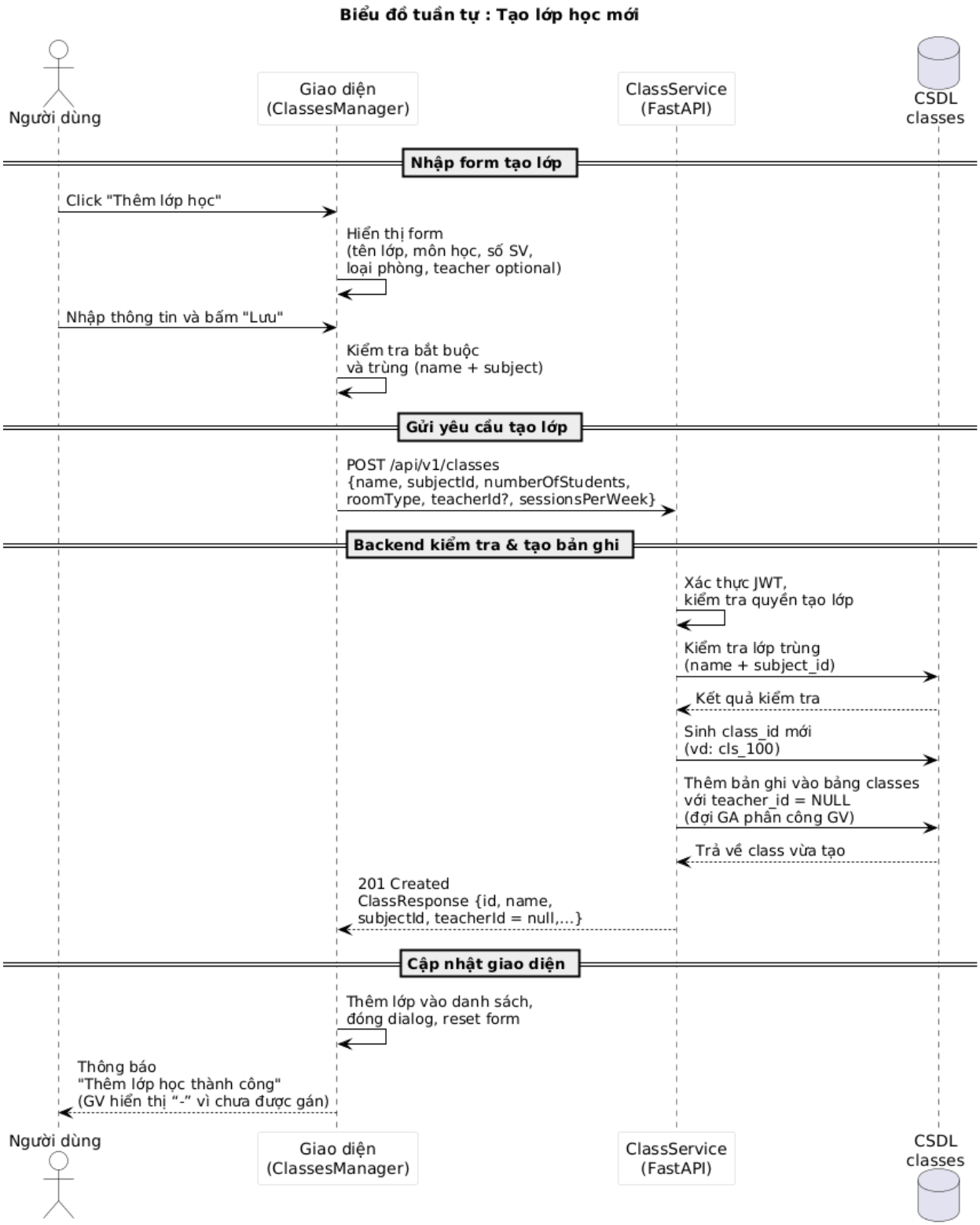
Hình 4.10: Biểu đồ tuần tự tạo phòng học



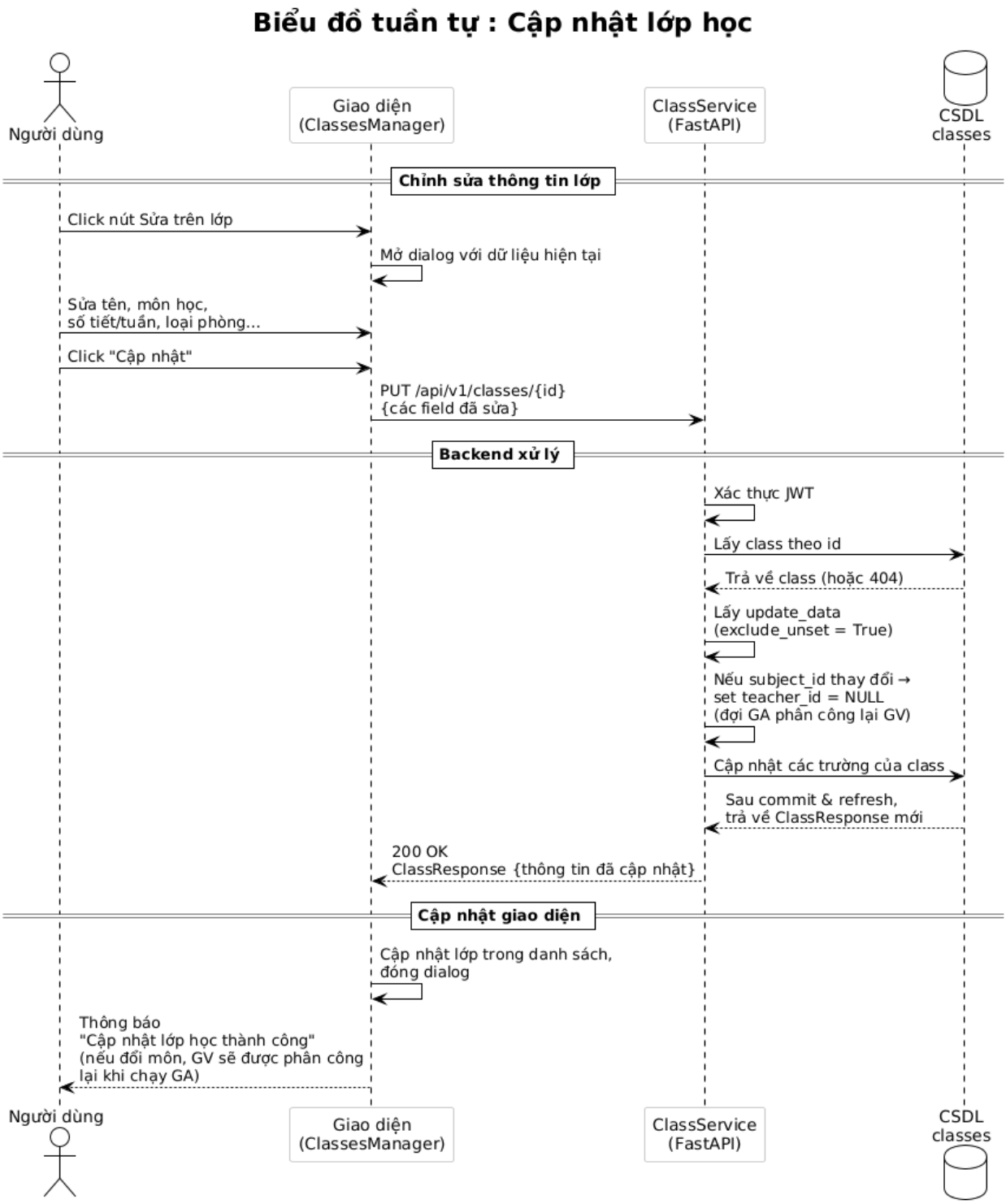
Hình 4.11: Biểu đồ tuần tự cập nhật phòng học



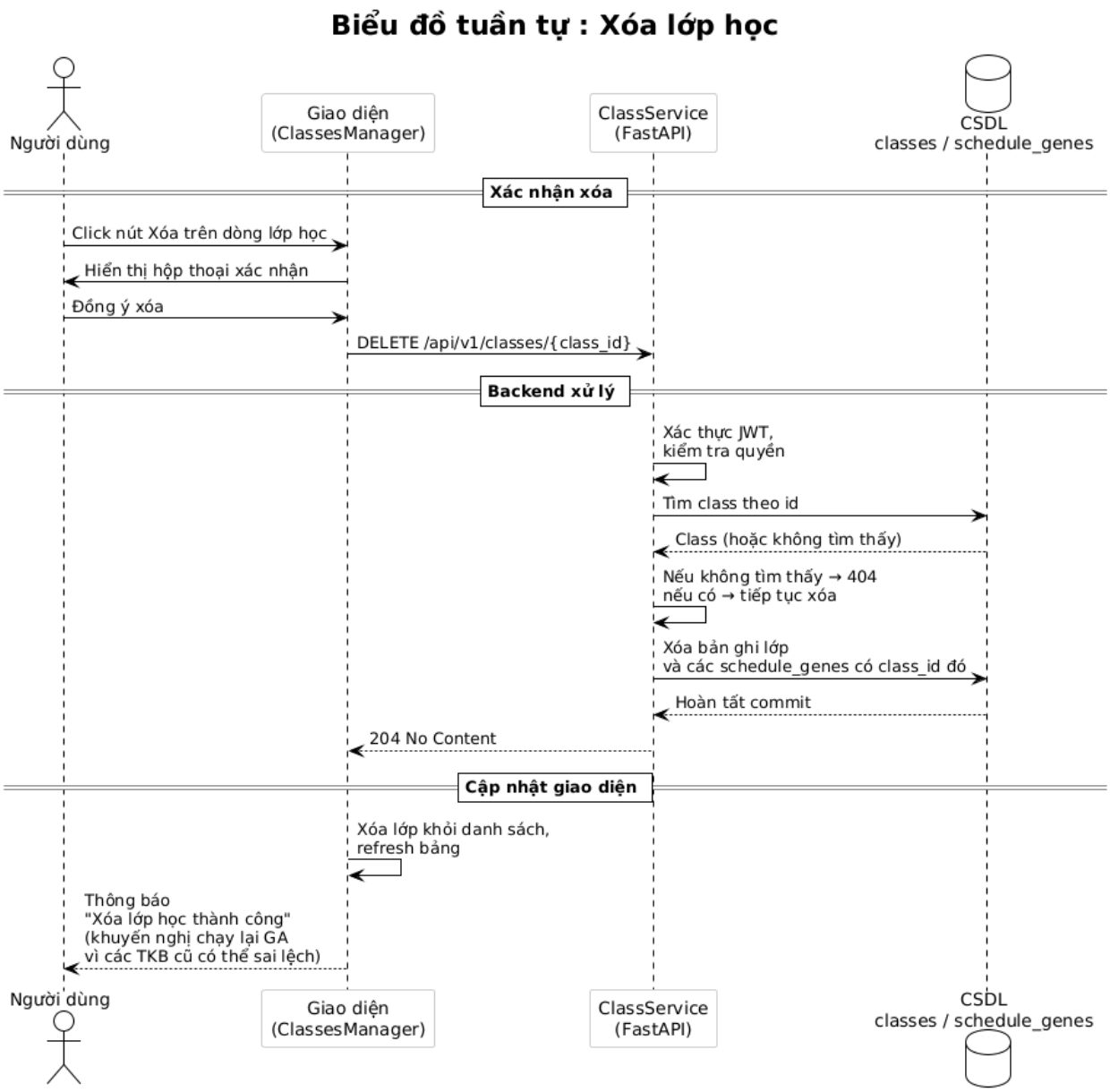
Hình 4.12: Biểu đồ tuần tự xóa phòng học



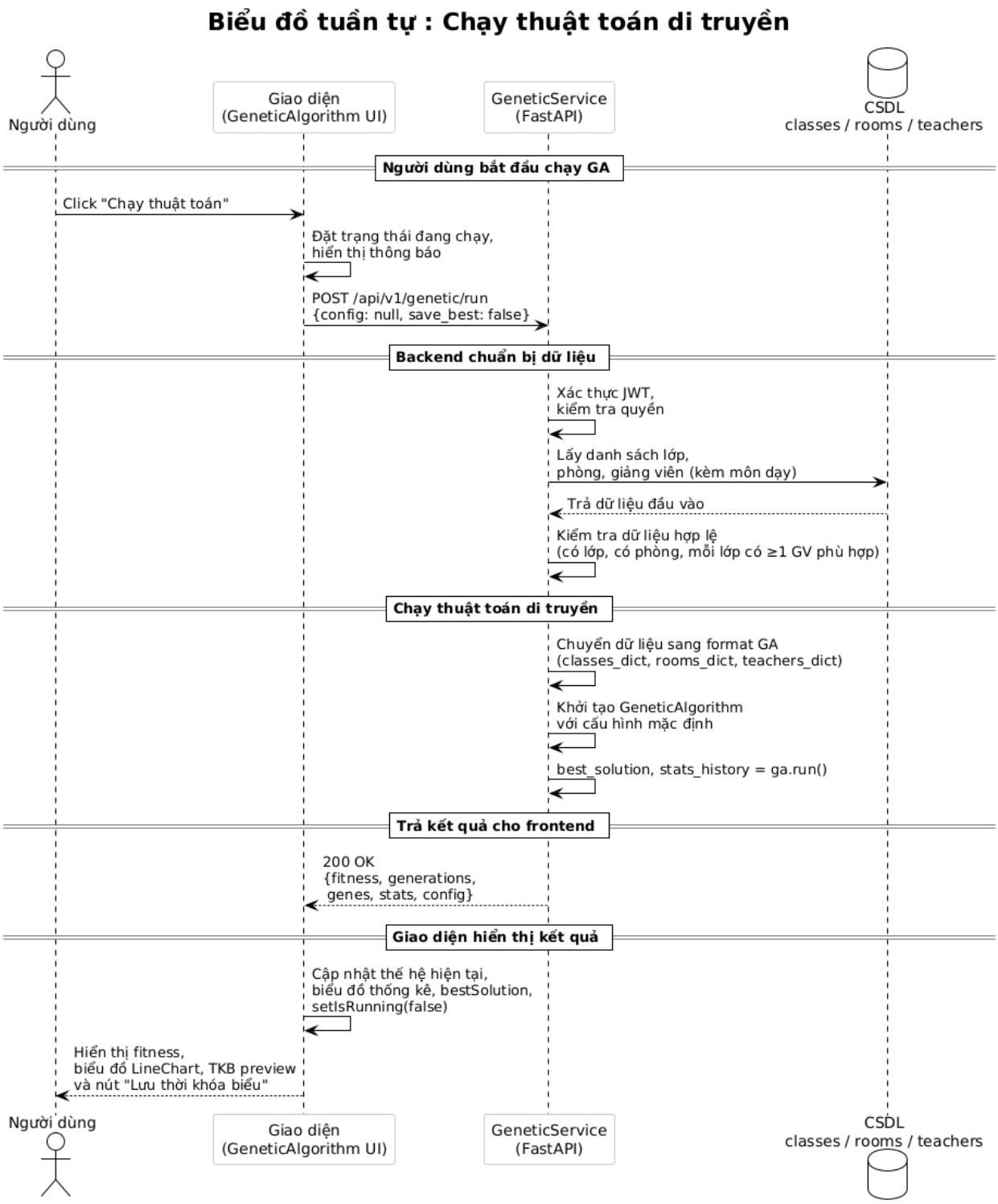
Hình 4.13: Biểu đồ tuần tự tạo lớp học



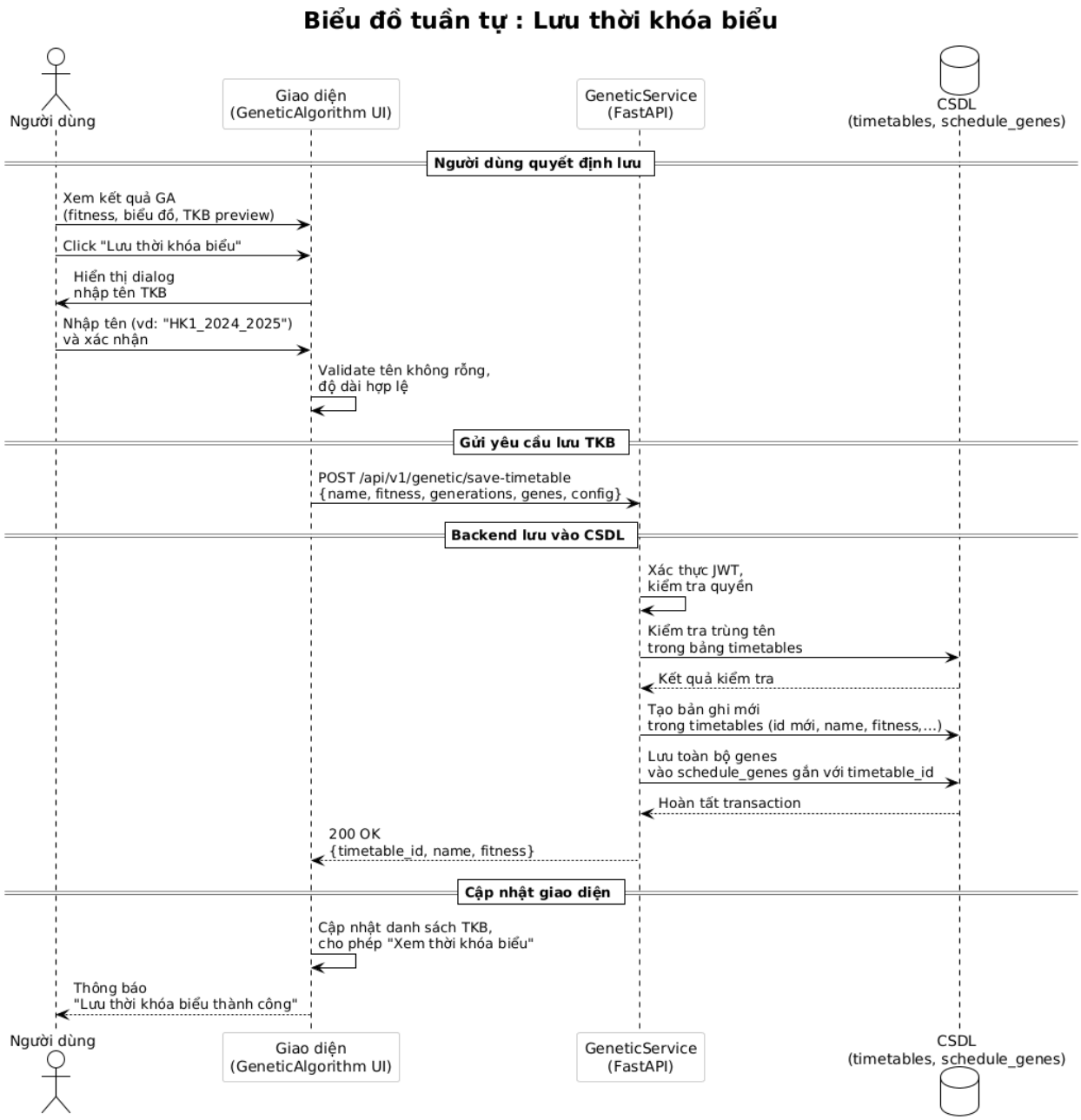
Hình 4.14: Biểu đồ tuần tự cập nhật lớp học



Hình 4.15: Biểu đồ tuần tự xóa lớp học

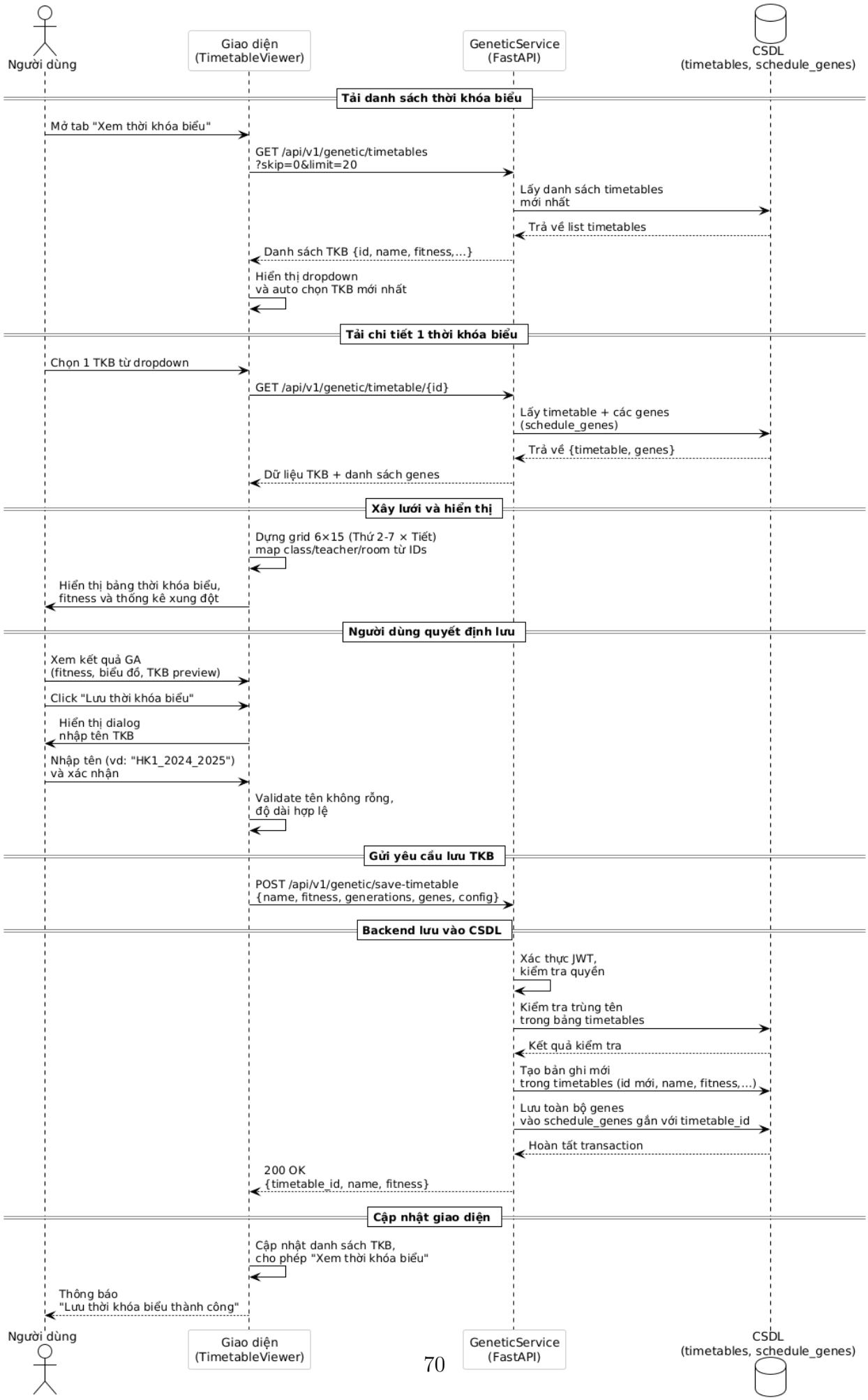


Hình 4.16: Biểu đồ tuần tự chạy thuật toán

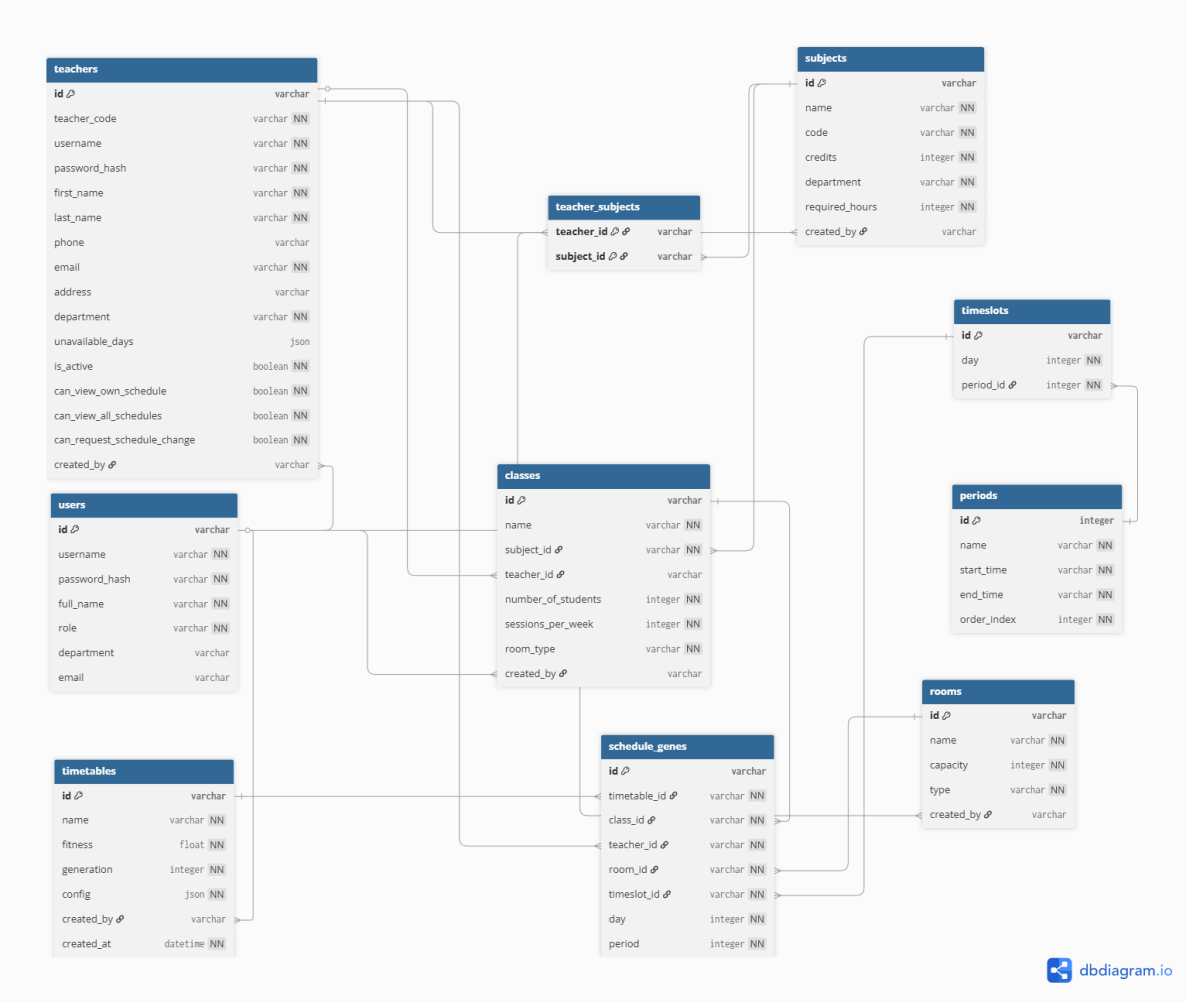


Hình 4.17: Biểu đồ tuần tự lưu thời khóa biểu

Biểu đồ tuần tự : Xem thời khóa biểu



4.5. Mô hình ERD của hệ thống



Hình 4.18: Biểu đồ ERD hệ thống quản lý giảng viên và thời khóa biểu

Mô hình ERD của hệ thống mô tả các thực thể chính và mối quan hệ giữa chúng trong cơ sở dữ liệu phục vụ bài toán quản lý giảng viên và xếp thời khóa biểu. Các bảng dữ liệu (entity) được thiết kế theo mô hình quan hệ, đảm bảo chuẩn hoá và hỗ trợ tốt cho việc truy vấn, cập nhật trong quá trình vận hành hệ thống.

- **Bảng *teachers*:** Lưu trữ thông tin giảng viên, bao gồm các thuộc tính như `teacher_code`, `username`, `password_hash`, `first_name`, `last_name`, `phone`, `email`, `address`, `department`, `unavailable_days`, cùng các cờ phân quyền như `can_view_own_schedule`, `can_view_all_schedu` `can_request_schedule_change`, `is_active`. Trường `created_by` cho biết tài khoản đã tạo giảng viên. Bảng *teachers* có quan hệ một–nhiều với bảng *classes* (một giảng viên phụ trách nhiều lớp) và *schedule_ genes*, đồng thời tham gia quan hệ nhiều–nhiều với bảng *subjects* thông qua bảng trung gian *teacher_ subjects*.

- **Bảng *subjects*:** Lưu trữ thông tin môn học với các thuộc tính `name`, `code`, `credits`, `department`, `required_hours`, `created_by`. Mỗi môn học có thể được giảng dạy bởi nhiều giảng viên và được mở thành nhiều lớp khác nhau. Quan hệ nhiều–nhiều giữa môn học và giảng viên được hiện thực thông qua bảng *teacher_ subjects*, trong khi quan hệ một–nhiều giữa môn học và lớp học được thể hiện qua khoá ngoại `subject_id` trong bảng *classes*.

- **Bảng *teacher_ subjects*:** Là bảng liên kết thể hiện quan hệ nhiều–nhiều giữa giảng viên và môn học. Bảng này gồm hai khoá ngoại `teacher_id` và `subject_id`, đồng thời đóng vai trò là khoá chính tổng hợp. Thông qua bảng *teacher_ subjects*, hệ thống xác định được mỗi giảng viên có thể giảng dạy những môn học nào và ngược lại.

- **Bảng *classes*:** Biểu diễn các lớp học hoặc nhóm học phần, với các thuộc tính như `name`, `subject_id`, `teacher_id`, `number_of_students`, `sessions_per_week`, `room_type`, `created_by`. Mỗi lớp gắn với một môn học cụ thể và một giảng viên phụ trách chính. Bảng *classes* có quan hệ một–nhiều với bảng *schedule_ genes*, bởi vì một lớp có thể xuất hiện ở nhiều khung giờ khác nhau trong thời khóa biểu.

- **Bảng *users*:** Lưu trữ tài khoản người dùng hệ thống (quản trị viên, cán bộ phòng đào tạo, . . .) với các trường `username`, `password_hash`, `full_name`, `role`, `department`, `email`. Các tài khoản này được sử dụng cho mục đích xác thực, phân quyền và thường được tham chiếu trong trường `created_by` của các bảng khác để ghi nhận người tạo dữ liệu.

- **Bảng *rooms*:** Chứa thông tin về phòng học, bao gồm `name`, `capacity`, `type`, `created_by`. Mỗi phòng có thể được sử dụng cho nhiều tiết học khác nhau trong thời khóa biểu. Bảng *rooms* có quan hệ một–nhiều với bảng *schedule_ genes* thông qua khoá ngoại `room_id`.

- **Bảng *periods*:** Định nghĩa các ca/tiết học trong ngày với các thuộc tính `name`, `start_time`, `end_time`, `order_index`. Bảng này giúp hệ thống chuẩn hóa các khung giờ dạy học (ví dụ: Tiết 1, Tiết 2, . . .) và được tham chiếu bởi bảng *timeslots*.

- **Bảng *timeslots*:** Đại diện cho các khung giờ cụ thể trong tuần, kết hợp giữa thứ trong tuần (`day`) và ca học (`period_id`). Mỗi bản ghi trong *timeslots* tương ứng với một thời điểm dạy học xác định (ví dụ: Thứ 2, Tiết 3). Các timeslot này được sử dụng bởi bảng *schedule_ genes* để gắn lớp, giảng viên và phòng vào một khung giờ cụ thể.

- **Bảng *timetables*:** Lưu trữ thông tin về các phương án thời khóa biểu, thường là kết quả của thuật toán tối ưu. Các trường chính bao gồm `name`, `fitness`, `generation`, `config`, `created_at`. Một thời khóa biểu có thể bao gồm nhiều gene lịch (các bản ghi trong *schedule_ genes*), thể hiện toàn bộ sắp xếp lớp – giảng viên – phòng – thời gian.

- **Bảng *schedule_ genes*:** Là bảng trung tâm mô tả từng "gene" của thời khóa biểu. Mỗi bản ghi liên kết một thời khóa biểu (`timetable_id`) với một lớp học (`class_id`), một giảng viên (`teacher_id`), một phòng học (`room_id`) và một khung giờ cụ thể (`timeslot_id`, `day`, `period`). Thông qua bảng này, hệ thống có thể truy vấn nhanh thời khóa biểu của từng giảng viên, từng lớp hoặc từng phòng học, đồng thời phục vụ cho việc đánh giá và tối ưu lịch học.

hiệu quả.

4.6. Xây dựng ứng dụng

4.6.1 Input

Bài toán xếp thời khóa biểu nhận đầu vào:

- **Classes** $C = \{c_1, c_2, \ldots, c_n\}$: Tập n lớp học cần xếp lịch.

  - Mỗi lớp c_i có các thông tin: tên lớp, môn học tương ứng, số lượng sinh viên, số buổi học mỗi tuần ($sessions\_per\_week$), loại phòng yêu cầu (lý thuyết, thực hành hoặc bất kỳ).

- **Teachers** $T = \{t_1, t_2, \ldots, t_m\}$: Tập m giảng viên.
  - Mỗi giảng viên t_j có: danh sách các môn có thể giảng dạy, danh sách các ngày/tiết **không rảnh** (unavailable), giới hạn số tiết dạy tối đa trong một ngày hoặc trong tuần.

- **Rooms** $R = \{r_1, r_2, \ldots, r_k\}$: Tập k phòng học.
  - Mỗi phòng r_l có: sức chứa tối đa (capacity), loại phòng (lý thuyết / thực hành / cả hai), và các thuộc tính khác (toà nhà, thiết bị, ... nếu cần).

- **TimeSlots** TS: Tập các khung giờ học khả dụng trong tuần.
  - 6 ngày/tuần (Thứ 2 – Thứ 7).
  - 15 tiết/ngày, từ 06:30 đến 20:55.
  - Tổng cộng: $6 \times 15 = 90$ timeslot, mỗi timeslot được biểu diễn bởi cặp $(day, period)$.

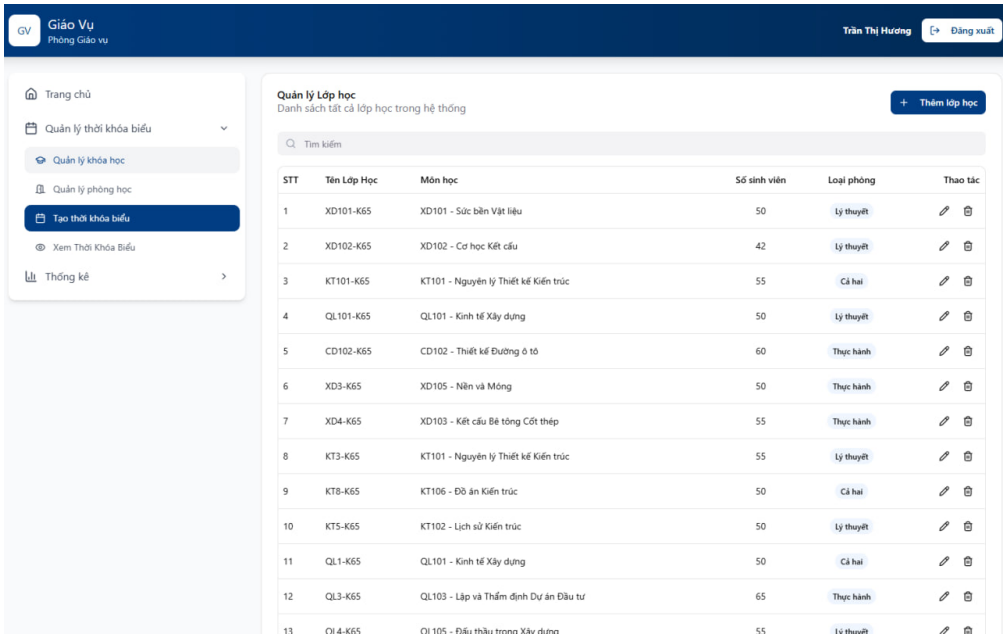
Ngoài ra, dữ liệu đầu vào của bài toán được lấy trực tiếp từ cơ sở dữ liệu của hệ thống, bao gồm các bảng: *classes* (danh sách lớp học), *teachers* (danh sách giảng viên), *rooms* (danh sách phòng học), *subjects* (môn học) và các bảng cấu hình liên quan.

4.6.2 Output

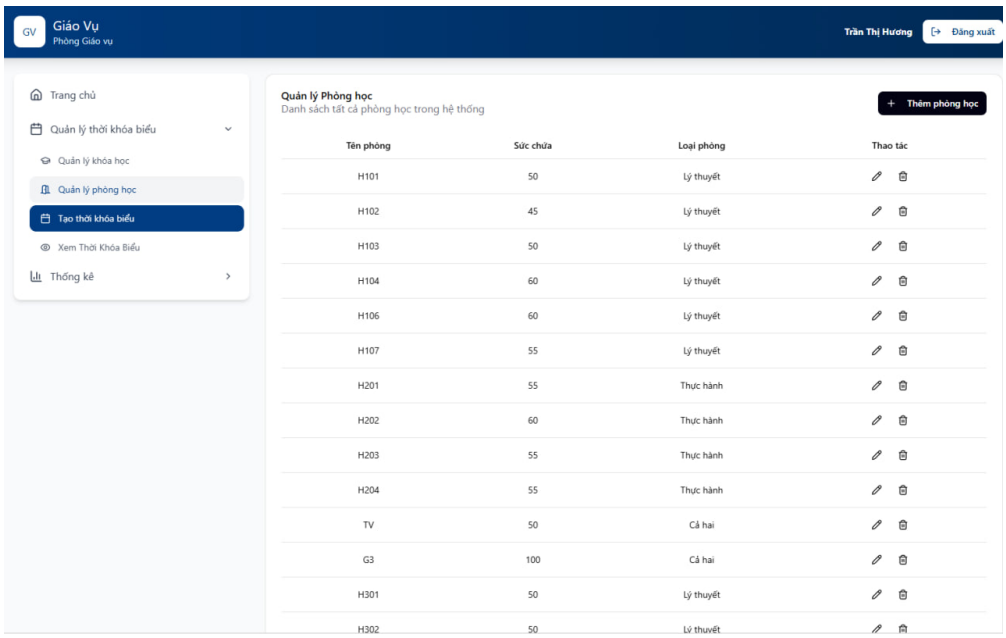
Đầu ra của bài toán là **một thời khóa biểu tối ưu** (hoặc gần tối ưu) thỏa mãn các tiêu chí sau:

- Mọi lớp được xếp **đủ số buổi/tuần** theo yêu cầu.
- Không xảy ra **xung đột** về phòng học, giảng viên và lớp học (không có hai lớp sử dụng cùng một phòng, cùng một giảng viên hoặc cùng một lớp tại cùng một timeslot).
- **Tối ưu hóa** việc phân bố lịch học: các buổi học được phân bố đều trong tuần, khối lượng giảng dạy (workload) giữa các giảng viên được cân bằng, hạn chế tối đa các khung giờ không mong muốn (giờ quá sớm, giữa trưa, tối muộn).
- Thỏa mãn các **ràng buộc** và tiêu chí đánh giá (fitness) đã được định nghĩa trong thuật toán xếp thời khóa biểu.

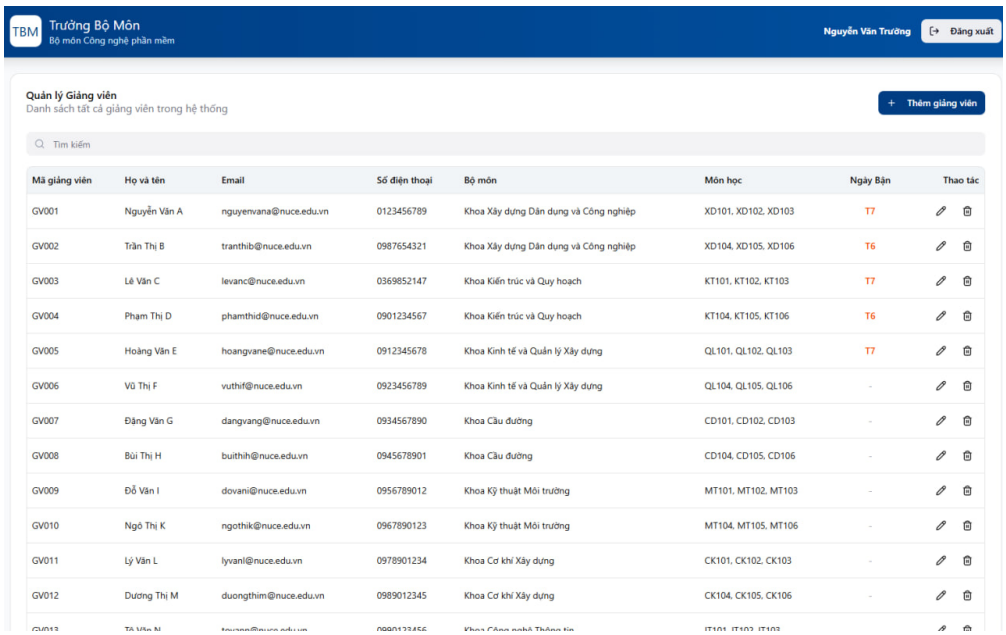
4.6.3 Dữ liệu từ ứng dụng web:



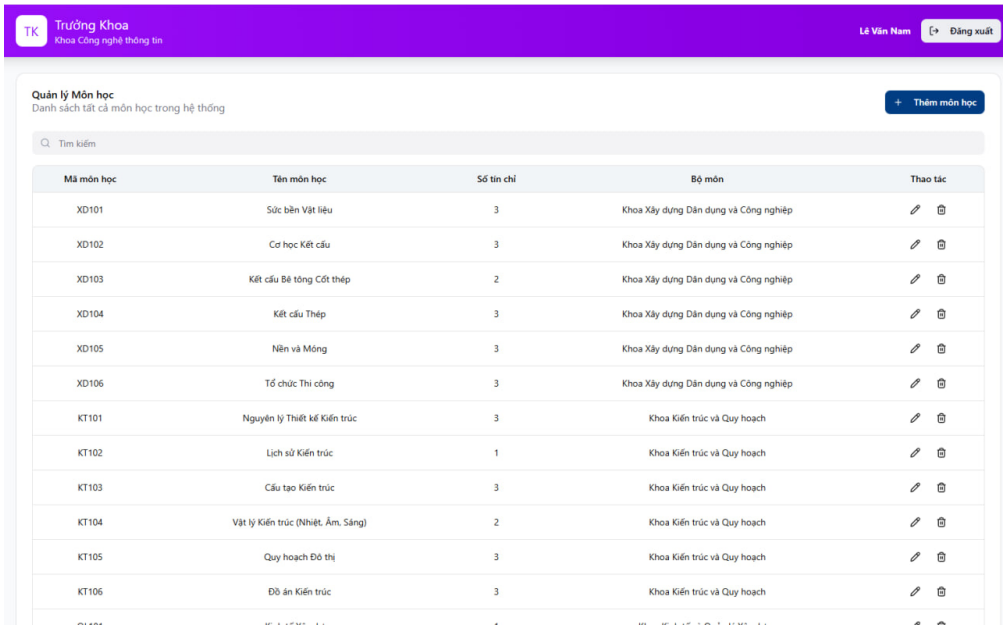
Hình 4.19: Hình ảnh Quản lý lớp học



Hình 4.20: Hình ảnh Quản lý Phòng Học



Hình 4.21: Hình ảnh Quản lý Giảng viên



Hình 4.22: Hình ảnh quản lý Môn học

4.6.4 Biểu diễn nhiễm sắc thể (Chromosome Encoding)

4.6.4.1 Cấu trúc Gen

Mỗi **gen** (gene) biểu diễn một buổi học, bao gồm:

$$Gene = (class\_id, teacher\_id, room\_id, day, period)$$

Trong đó:

- $class\_id \in C$: Lớp học nào
- $teacher\_id \in T$: Giảng viên nào dạy

- $room\_id \in R$: Phòng học nào
- $day \in \{0, 1, 2, 3, 4, 5\}$: Ngày nào (T2–T7)
- $period \in \{0, 1, \ldots, 14\}$: Tiết nào trong ngày

Ví dụ thực tế từ dự án:
Gen $(CS101\text{-}K65, GV001, H101, 0, 2)$ nghĩa là:

> Lớp **CS101-01** (Cơ học kết cấu, 50 sinh viên) học với giảng viên **Nguyễn Văn A** tại phòng **H101** (50 chỗ, loại THEORY) vào **Thứ 2, tiết 3** (07:55–08:45).

Bảng phân bố tiết học trong ngày:

Tiết	Giờ	Ghi chú
0	06:30 – 07:20	Sáng sớm (nên tránh)
1	07:25 – 08:15	
2	08:20 – 09:10	
3	09:15 – 10:05	
4	10:15 – 11:05	
5	11:10 – 12:00	
6	12:30 – 13:20	Nghỉ trưa (nên tránh)
7	13:25 – 14:15	
8	14:20 – 15:10	
9	15:15 – 16:05	
10	16:10 – 17:00	
11	17:05 – 17:55	
12	18:00 – 18:50	
13	19:10 – 20:00	Buổi tối (nên tránh)
14	20:05 – 20:55	Buổi tối (nên tránh)

Bảng 4.1: Khung giờ học trong ngày

4.6.4.2 Cấu trúc nhiễm sắc thể (Chromosome Structure)

Một **nhiễm sắc thể** (chromosome), hay một **cá thể** trong quần thể, biểu diễn **một thời khóa biểu hoàn chỉnh** cho toàn bộ các lớp. Về mặt cấu trúc, mỗi nhiễm sắc thể là một tập hợp các gen:

$$Individual = \{Gene_1, Gene_2, \ldots, Gene_N\}$$

```python

@dataclass
class TimeSlot:
    """Time slot representation"""
    day: int  # 0-5 (Monday-Saturday)
    period: int  # 0-14 (15 periods per day)


@dataclass
class ScheduleGene:
    """A gene in the timetable chromosome

    Now includes teacher_id to allow GA to optimize teacher assignment
    """
    class_id: str
    teacher_id: str  # GA can change this to balance workload
    room_id: str
    time_slot: TimeSlot


@dataclass
class Individual:
    """An individual in the population (a complete timetable)"""
    genes: List[ScheduleGene]
    fitness: float = 0.0

```

Hình 4.23: Cấu trúc nhiễm sắc thể

Trong đó N là **tổng số buổi học cần được xếp lịch**, được tính bằng:

$$N = \sum_{i=1}^{n} sessions\_per\_week(c_i)$$

với n là số lớp học, $sessions\_per\_week(c_i)$ là số buổi/tuần của lớp c_i.
Ví dụ: Với 50 lớp, mỗi lớp cần 3 buổi/tuần, khi đó:

$$N = 50 \times 3 = 150 \text{ gen.}$$

Nhiễm sắc thể tương ứng sẽ chứa 150 gen, mỗi gen mô tả một buổi học cụ thể. Các toán tử di truyền (lai ghép, đột biến) khi áp dụng lên nhiễm sắc thể tương đương với việc điều chỉnh, hoán đổi các buổi học trong toàn bộ thời khóa biểu.

4.6.5 Tham số của thuật toán di truyền

Để giải bài toán xếp thời khóa biểu, hệ thống sử dụng **thuật toán di truyền (Genetic Algorithm – GA)** với các tham số cấu hình như sau:

```python
def __init__(
    self,
    classes: List[Dict],
    rooms: List[Dict],
    teachers: List[Dict],
    population_size: int = 250,
    mutation_rate: float = 0.1,
    crossover_rate: float = 0.8,
    elitism_rate: float = 0.12,
    max_generations: int = 250,
):
```

Hình 4.24: Tham số của thuật toán di truyền

Tham số	Giá trị	Ý nghĩa
`population_size`	250	Số lượng cá thể trong mỗi quần thể
`mutation_rate`	0.1	Xác suất đột biến (10% số gen)
`crossover_rate`	0.8	Xác suất lai ghép (80% số cặp cha mẹ)
`elitism_rate`	0.12	Tỷ lệ cá thể ưu tú được giữ lại (12%)
`max_generations`	250	Số thế hệ tối đa cho phép
`tournament_size`	5	Kích thước nhóm trong chọn lọc tournament

Bảng 4.2: Các tham số thuật toán di truyền (GA) sử dụng trong dự án

- `population_size`: Quy định số lượng thời khóa biểu (cá thể) trong mỗi thế hệ. Quần thể lớn giúp đa dạng nghiệm nhưng tăng thời gian tính toán.
- `mutation_rate`: Xác suất đột biến trên từng gen, dùng để duy trì đa dạng di truyền bằng cách thay đổi ngẫu nhiên một số buổi học (ví dụ: đổi phòng, đổi tiết, đổi ngày).
- `crossover_rate`: Xác suất lai ghép giữa hai cá thể cha mẹ để tạo cá thể con, giúp kết hợp các phần thời khóa biểu tốt từ hai cha mẹ.
- `elitism_rate`: Tỷ lệ cá thể tốt nhất được sao chép trực tiếp sang thế hệ kế tiếp, đảm bảo nghiệm tốt không bị mất đi.
- `max_generations`: Giới hạn số thế hệ tiến hóa; thuật toán dừng khi đạt ngưỡng này hoặc khi nghiệm đã hội tụ.
- `tournament_size`: Số cá thể tham gia mỗi lần chọn lọc kiểu tournament; cá thể có độ thích nghi cao nhất trong nhóm sẽ được chọn làm cha/mẹ.

Nhờ việc mã hóa thời khóa biểu dưới dạng nhiễm sắc thể và lựa chọn các tham số GA phù hợp, hệ thống có khả năng tìm kiếm trong không gian nghiệm rất lớn và dần tiến tới các phương án thời khóa biểu vừa thoả mãn ràng buộc cứng, vừa tối ưu các tiêu chí mềm đã đặt ra.

4.6.6 Hàm đánh giá thích nghi (Fitness Function)

4.6.6.1 Công thức tổng quát

Hàm fitness đánh giá chất lượng của một thời khóa biểu. Sử dụng **weighted cost function**:

$$Cost = \sum_{i=1}^{N_h}(H_i \times W_{hard,i}) + \sum_{j=1}^{N_s}(S_j \times W_{soft,j})$$

$$Fitness = \frac{1}{1 + Cost}$$

Trong đó:

- H_i: Số vi phạm ràng buộc cứng thứ i
- S_j: Số vi phạm ràng buộc mềm thứ j
- $W_{hard,i}$: Trọng số ràng buộc cứng $(15 - 20)$
- $W_{soft,j}$: Trọng số ràng buộc mềm $(2 - 8)$

Giải thích:

- $Cost = 0 \Rightarrow Fitness = 1.0$ (lời giải hoàn hảo)
- $Cost$ càng lớn $\Rightarrow$ $Fitness$ càng nhỏ $\Rightarrow$ TKB càng tệ

Ví dụ tính fitness cụ thể:
Xét cá thể có các vi phạm:

- **H1 - Xung đột phòng**: A101 bị trùng 2 lớp cùng lúc
  - $(CS101\text{-}01, GV001, A101, 0, 2)$ - Thứ 2 tiết 3
  - $(CS201\text{-}01, GV002, A101, 0, 2)$ - Thứ 2 tiết 3 $\Rightarrow$ **1 vi phạm**
- **H2 - Xung đột GV**: GV001 dạy 2 lớp cùng lúc
  - $(CS101\text{-}01, GV001, A101, 1, 3)$ - Thứ 3 tiết 4
  - $(CS102\text{-}01, GV001, C301, 1, 3)$ - Thứ 3 tiết 4 $\Rightarrow$ **1 vi phạm**
- **S3 - Buổi tối**: CS301-01 học tiết 13 (19:10-20:00) $\Rightarrow$ **1 vi phạm**
- **S6 - Thứ 7**: CS401-01 học Thứ 7 $\Rightarrow$ **1 vi phạm**
- **S8 - Workload không cân bằng**:
  - GV001: 8 buổi/tuần, GV002: 2 buổi/tuần
  - Chênh lệch $= 6 > 4 \Rightarrow$ **1 vi phạm**

Tính toán Cost:

$$\begin{aligned}\text{Cost} &= \text{H1} \times 20 + \text{H2} \times 20 + \text{S3} \times 8 + \text{S6} \times 5 + \text{S8} \times 3 \\ &= 1 \times 20 + 1 \times 20 + 1 \times 8 + 1 \times 5 + 1 \times 3 \\ &= 20 + 20 + 8 + 5 + 2 = \mathbf{55}\end{aligned}$$

Tính toán Fitness:

$$\text{Fitness} = \frac{1}{1 + 55} = \frac{1}{56} \approx \mathbf{0.0179}$$

Ý nghĩa: Cá thể này rất kém (fitness = 0.0179) do vi phạm **2 ràng buộc cứng** (xung đột phòng + xung đột GV). Cần sửa gene để tránh xung đột.

4.6.7 Ràng buộc của Thời Khóa biểu

4.6.7.1 Ràng buộc cứng (Hard Constraints)

Các ràng buộc **BẮT BUỘC** phải thỏa mãn (trọng số cao $15 - 20$):

Ràng buộc	**Mô tả**	**Trọng số**
H1	Xung đột phòng học *Không được 2 lớp dùng 1 phòng cùng lúc*	20
H2	Xung đột giảng viên *GV không dạy 2 lớp cùng lúc*	20
H3	Xung đột lớp học *Lớp không học 2 môn cùng lúc*	20
H4	Vượt sức chứa phòng *Số SV* $\leq$ *capacity của phòng*	20
H5	GV dạy quá 8 tiết/ngày $workload\_per\_day \leq 8$	15
H6	Lớp học quá 6 tiết/ngày $sessions\_per\_day \leq 6$	12
H7	Phòng không phù hợp *Lab class phải dùng Lab room*	20
H8	GV dạy ngày không rảnh *Kiểm tra unavailable_days*	20

Bảng 4.3: Ràng buộc cứng và trọng số

4.6.7.2 Ràng buộc mềm (Soft Constraints)

Các ràng buộc **MONG MUỐN** để tối ưu hóa (trọng số thấp $2 - 8$):

Ràng buộc	Mô tả	Trọng số
S1	Phân bố không đều tuần *Buổi học nên rải đều các ngày*	3
S2	GV dạy liên tiếp $>$ 3 tiết *Tránh dạy nhiều tiết liên tục*	5
S3	Lớp buổi tối (tiết 13-14) *Tránh 19:10-20:55*	8
S4	Lớp sáng sớm (tiết 0) *Tránh 6:30-7:20*	8
S5	Lớp giờ nghỉ trưa (tiết 6) *Tránh 12:30-13:20*	8
S6	Lớp thứ 7 *Ưu tiên T2-T6*	5
S7	GV không nghỉ trưa *GV nên có ít nhất 1 tiết nghỉ 11:30-13:30*	6
S8	Buổi học quá gần *Khoảng cách $\geq$ 2 tiết*	2
S9	Mất cân bằng workload *GV cùng môn nên dạy đều*	2

Bảng 4.4: Ràng buộc mềm và trọng số

4.6.8 Các toán tử di truyền trong thuật toán

4.6.8.1 Khởi tạo quần thể (Population Initialization)

Tạo N cá thể ngẫu nhiên với **heuristic strategies**:

Algorithm 2 Khởi tạo quần thể thông minh

```
Population ← ∅
for i = 1 đến population_size do
    genes ← ∅
    for mỗi lớp c trong Classes do
        suitable_rooms ← Lọc phòng phù hợp (đủ sức chứa + đúng loại)
        qualified_teachers ← GV có thể dạy môn của lớp c
        for j = 1 đến c.sessions_per_week do
            day ← Random (ưu tiên T2-T6, tránh T7)
            period ← Random (ưu tiên tiết 1-5, 7-12, tránh 0, 6, 13-14)
            teacher ← Chọn GV tối ưu (cân bằng workload)
            room ← Chọn phòng luân phiên (balance usage)
            Thêm (c.id, teacher, room, day, period) vào genes
        end for
    end for
    fitness ← Tính fitness cho genes
    Thêm Individual(genes, fitness) vào Population
end for
return Population
```

- **Chọn ngày**: 85% T2-T6, 15% T7

- **Chọn tiết**: 95% tiết đẹp (1-5, 7-12), 5% khác
- **Chọn GV**: Thuật toán tối ưu (không random!)
- **Chọn phòng**: Luân phiên để phân bố đều

4.6.8.2 Chọn lọc (Selection) - Tournament Selection

```python
def selection(self, population: List[Individual]) -> Individual:
    """
    Chọn lọc bằng Tournament Selection
    """
    tournament_size = 5
    tournament = random.sample(population, min(tournament_size, len(population)))
    return max(tournament, key=lambda ind: ind.fitness)
```

Chọn cá thể cha mẹ bằng **Tournament Selection**:
Ưu điểm:

- Tốc độ nhanh (không cần sắp xếp toàn bộ quần thể)
- Áp lực chọn lọc cao với $tournament\_size = 5$
- Tránh hội tụ sớm (premature convergence)

Ví dụ cụ thể Tournament Selection:
Chọn ngẫu nhiên 7 cá thể từ quần thể 200:

STT	Mô tả vi phạm	Fitness
1	GV001 dạy CS101-01 T2,T4 - Không vi phạm	0.0667
2	GV002 xung đột T3 tiết 4 (H2)	0.0476
3	A101 trùng 2 lớp T2 tiết 3 (H1)	0.0323
4	CS301-01 học T7 (S6) + tiết 13 (S3)	0.0714
5	Cân bằng tốt, phân bố đều	**0.0909**
6	CS102-01 vượt sức chứa A102 (H4)	0.0270
7	GV003 dạy 4 tiết liên tiếp (S2)	0.0588
Winner: Cá thể số 5		**0.0909**

Bảng 4.5: Ví dụ Tournament Selection

Cá thể số 5 được chọn làm cha/mẹ vì có fitness cao nhất.

4.6.8.3 Lai ghép (Crossover) - Uniform Crossover

Kết hợp 2 cha mẹ để tạo con cái:

```python
838
839     def crossover(self, parent1: Individual, parent2: Individual) -> List[ScheduleGene]:
840         """
841         Lai ghép (Crossover) hai cá thể cha mẹ
842
843         Sử dụng uniform crossover: mỗi gen có 50% cơ hội được chọn từ parent1 hoặc parent2.
844         Phương pháp này giữ được các đặc điểm tốt của cả hai cha mẹ.
845         """
846         if random.random() > self.crossover_rate:
847             return parent1.genes.copy()
848
849         # Uniform crossover - mix genes from both parents
850         child_genes = []
851         for gene1, gene2 in zip(parent1.genes, parent2.genes):
852             if random.random() < 0.5:
853                 child_genes.append(gene1)
854             else:
855                 child_genes.append(gene2)
856         return child_genes
857
```

Uniform Crossover: Mỗi gen có 50% cơ hội được chọn từ cha hoặc mẹ.
Ví dụ cụ thể Uniform Crossover:
Parent 1 (TKB CS101-01 - Cơ học kết cấu, 2 buổi/tuần):

```
Gene 1: (CS101-01, GV001, A101, 0, 2)  # Thứ 2 tiết 3
Gene 2: (CS101-01, GV001, A101, 2, 4)  # Thứ 4 tiết 5
```

Parent 2 (TKB CS102-01 - Vật liệu xây dựng, 2 buổi/tuần):

```
Gene 1: (CS102-01, GV001, C301, 1, 3)  # Thứ 3 tiết 4
Gene 2: (CS102-01, GV001, C301, 3, 7)  # Thứ 5 tiết 8
```

Quá trình lai ghép:

- **Vị trí 0** (CS101-01 buổi 1):
  $random() = 0.3 < 0.5$
  $\Rightarrow$ Chọn từ Parent 1: $(CS101\text{-}01, GV001, A101, 0, 2)$
- **Vị trí 1** (CS101-01 buổi 2):
  $random() = 0.7 > 0.5$
  $\Rightarrow$ Chọn từ Parent 2: $(CS101\text{-}01, GV001, C301, 3, 7)$
- **Vị trí 2** (CS102-01 buổi 1):
  $random() = 0.6 > 0.5$
  $\Rightarrow$ Chọn từ Parent 2: $(CS102\text{-}01, GV001, C301, 1, 3)$
- **Vị trí 3** (CS102-01 buổi 2):
  $random() = 0.2 < 0.5$
  $\Rightarrow$ Chọn từ Parent 1: $(CS102\text{-}01, GV001, A101, 2, 4)$

Child (kết quả lai ghép):

```
Gene 1: (CS101-01, GV001, A101, 0, 2)  # Từ P1: Thứ 2 tiết 3
Gene 2: (CS101-01, GV001, C301, 3, 7)  # Từ P2: Thứ 5 tiết 8
Gene 3: (CS102-01, GV001, C301, 1, 3)  # Từ P2: Thứ 3 tiết 4
Gene 4: (CS102-01, GV001, A101, 2, 4)  # Từ P1: Thứ 4 tiết 5
```

Con cái kế thừa 50% gene từ mỗi cha mẹ, tạo đa dạng di truyền.

4.6.8.4 Đột biến (Mutation) - Smart Mutation

Thay đổi gen với các ràng buộc:

```python
    def mutate(self, genes: List[ScheduleGene]) -> List[ScheduleGene]:
        """
        Đột biến thông minh (Smart Mutation)

        Thay đổi gen với các ràng buộc:
        - Chọn phòng có sức chứa phù hợp
        - Tránh thứ 7 và buổi tối
        - Ưu tiên phòng phù hợp với loại lớp
        - Kiểm tra teacher availability khi chọn ngày
        - Có thể thay đổi giáo viên để cân bằng workload
        """
        mutated_genes = []
        for gene in genes:
            if random.random() < self.mutation_rate:
                # Find suitable rooms for this class
                class_item = next((c for c in self.classes if c['id'] == gene.class_id), None)
                if class_item:
                    # Get suitable rooms based on class type
                    suitable_rooms = self._get_suitable_rooms(class_item)

                    if not suitable_rooms:
                        suitable_rooms = self.rooms

                    # Get qualified teachers for this subject
                    subject_id = class_item['subject_id']
                    qualified_teachers = self.subject_to_teachers.get(subject_id, [])

                    # SAFETY: Should not be empty after validation
                    if not qualified_teachers:
                        # Fallback to keep original gene
                        mutated_genes.append(gene)
                        continue

                    # Step 1: Choose day - prefer weekdays
                    day = None
                    attempts = 0
                    max_attempts = 20

                    while day is None and attempts < max_attempts:
                        # 85% chance weekday (0-4), 15% chance Saturday (5)
                        candidate_day = random.randint(0, 4) if random.random() < 0.85 else 5
                        day = candidate_day
                        attempts += 1
```

```python

                    if day is None:
                        day = random.randint(0, self.DAYS - 1)

                    # Step 2: Chọn period - ưu tiên giờ đẹp, tránh sáng sớm, nghỉ trưa và buổi tối
                    # Giờ đẹp: 1-5 (sáng: 7:25-12:00), 7-12 (chiều: 13:25-18:00)
                    # Tránh: 0 (6:30 quá sớm), 6 (12:30 nghỉ trưa), 13-14 (19:10-20:55 tối)
                    if random.random() < 0.95:  # 95% chọn giờ đẹp
                        good_periods = list(range(1, 6)) + list(range(7, 13))  # 1-5, 7-12
                        period = random.choice(good_periods)
                    else:
                        period = random.randint(0, self.PERIODS - 1)

                    # Step 3: Select teacher - 30% chance to try changing if multiple qualified
                    if len(qualified_teachers) > 1 and random.random() < 0.3:
                        # Use optimal teacher selection based on constraints
                        teacher_id = self._select_optimal_teacher(qualified_teachers, class_item, mutated_genes, day, period)
                    else:
                        teacher_id = gene.teacher_id

                    mutated_genes.append(ScheduleGene(
                        class_id=gene.class_id,
                        teacher_id=teacher_id,  # May have changed teacher
                        room_id=self._select_room_balanced(suitable_rooms, mutated_genes, class_item),
                        time_slot=TimeSlot(day=day, period=period)
                    ))
                else:
                    mutated_genes.append(gene)
            else:
                mutated_genes.append(gene)
        return mutated_genes
```

Smart Mutation: Không random hoàn toàn mà áp dụng heuristic để tăng chất lượng.
Ví dụ cụ thể Smart Mutation:
Trước đột biến (Gene của CS201-01 - Thiết kế kiến trúc, 50 SV):

```
Gene: (CS201-01, GV002, A101, 0, 0)
# Thứ 2, tiết 1 (06:30-07:20) - GIỜ XẤU!
```

```
# GV002 Trần Thị B dạy KT101
# Phòng A101 (60 chỗ THEORY)
```

Vi phạm:

- **S4**: Tiết 0 (sáng sớm 06:30) ⇒ Cost +8
- Chưa tối ưu: có thể chọn phòng C301 lớn hơn

Quá trình đột biến:

1. **Class**: CS201-01 (50 SV, KT101) - giữ nguyên
2. **Qualified teachers**: GV002 (KT101+KT102), GV005 (XD101+KT101)
3. **Chọn GV mới**: $random() = 0.25 < 0.3 \Rightarrow$ Đổi GV
   - GV002: workload = 6 buổi
   - GV005: workload = 4 buổi ⇒ **Chọn GV005** (cân bằng hơn)
4. **Suitable rooms**: A101 (60), A102 (60), C301 (80), C302 (80)
5. **Chọn phòng**: C301 (luân phiên, phòng lớn hơn)
6. **Chọn ngày**: $random() = 0.6 \Rightarrow day = 2$ (Thứ 4)
7. **Chọn tiết**: $random() = 0.4 \Rightarrow period = 3$ (9:15-10:05, GIỜ ĐẸP)

Sau đột biến:

```
Gene: (CS201-01, GV005, C301, 2, 3)
# Thứ 4, tiết 4 (09:15-10:05) - GIỜ ĐẸP!
# GV005 Hoàng Văn E dạy KT101 (cân bằng workload)
# Phòng C301 (80 chỗ BOTH, dư giả hơn)
```

Kết quả:

- **Loại bỏ S4**: Không còn tiết 0 ⇒ Cost -8
- **Cân bằng workload**: GV005 tăng từ 4 lên 5, GV002 giảm từ 6 xuống 5
- **Fitness tăng**: Cá thể tốt hơn sau đột biến

4.6.8.5 Ưu tú (Elitism)

Giữ lại top 12% cá thể tốt nhất qua mỗi thế hệ:

$$elite\_count = \lfloor population\_size \times 0.12 \rfloor$$

Với $population\_size = 200 \Rightarrow elite\_count = 24$ cá thể tốt nhất được giữ lại.

4.6.9 CÁC CẢI TIẾN TRONG TRIỂN KHAI

4.6.9.1 Adaptive Restart Strategy

Để tránh **hội tụ sớm** (stuck in local optimum), áp dụng **adaptive restart**:

```
986
987         if generations_without_improvement >= 30 and generation < self.max_generations - 10:
988
989             elite_count = int(self.population_size * 0.2)
990             new_random = []
991             for _ in range(self.population_size - elite_count):
992                 genes = self.generate_random_schedule()
993                 new_random.append(Individual(
994                     genes=genes,
995                     fitness=self.calculate_fitness(genes)
996                 ))
997             population = population[:elite_count] + new_random
998             generations_without_improvement = 0
999             continue
1000
```

Ý nghĩa: Nếu không cải thiện sau 30 thế hệ ⇒ Inject diversity để thoát local optimum.

4.6.9.2 Pre-validation Strategy

Trước khi chạy GA, **kiểm tra tính khả thi**:

1: *subject_to_teachers* ← Build mapping từ teacher_subjects
2: **for** mỗi lớp c trong *Classes* **do**
3: $\quad qualified \leftarrow subject\_to\_teachers[c.subject\_id]$
4: $\quad$ **if** *qualified* rỗng **then**
5: $\quad\quad$ **Raise Error**: "Lớp *c.name* không có GV nào có thể dạy!"
6: $\quad$ **end if**
7: **end for**
8: **OK** - Có thể chạy GA

Lợi ích: Tránh lãng phí 200 thế hệ rồi mới phát hiện lỗi.

4.6.9.3 Balanced Room Selection

Chọn phòng theo chiến lược **luân phiên công bằng**:

$$score_{room} = -(waste \times 10) - (usage \times 100) + type\_bonus$$

Trong đó:

- $waste = capacity - num\_students$: Sức chứa dư thừa
- $usage$: Số lần phòng đã được sử dụng
- $type\_bonus = 100$ nếu đúng loại, 0 nếu sai

Ví dụ với lớp THEORY 50 SV:

- A101 (THEORY, 60 chỗ, 0 lần): $score = -100 - 0 + 100 = 0$ (dư 10 chỗ)
- A102 (THEORY, 60 chỗ, 0 lần): $score = -100 - 0 + 100 = 0$ **Chọn luân phiên**
- C301 (BOTH, 80 chỗ, 0 lần): $score = -300 - 0 + 100 = -200$ (dư 30 chỗ, lãng phí)

- A101 (THEORY, 60 chỗ, 1 lần): $score = -100 - 100 + 100 = -100$ Tránh (đã dùng)

Ví dụ thực tế:
Lớp CS201-01 (Thiết kế kiến trúc KT101, 50 SV):

1. **Lọc phòng phù hợp**: Loại bỏ B201, B202 (LAB, chỉ 40 chỗ)
2. **Ứng viên**: A101 (60 THEORY), A102 (60 THEORY), C301 (80 BOTH), C302 (80 BOTH)
3. **Tính điểm**:
   - A101: $waste = 10, usage = 0 \Rightarrow score = -100 + 100 = 0$
   - A102: $waste = 10, usage = 0 \Rightarrow score = -100 + 100 = 0$
   - C301: $waste = 30, usage = 0 \Rightarrow score = -300 + 100 = -200$
4. **Chọn**: A101 hoặc A102 (có điểm cao nhất = 0)

4.6.9.4 Optimal Teacher Selection

Chọn GV **không phải random** mà dựa trên:

$$score_{teacher} = conflict + unavailable + workload\_deviation$$

- **Conflict**: +100 nếu GV đang dạy cùng lúc
- **Unavailable**: +50 nếu GV không rảnh ngày đó
- **Workload deviation**: Phạt nếu dạy nhiều hơn trung bình

Chọn GV có **score thấp nhất** (tối ưu nhất).
Ví dụ thực tế:
Lớp CS101-01 (Cơ học kết cấu XD101) cần chọn GV cho buổi học **Thứ 3, tiết 4**:
Bước 1: Lọc GV qualified (có thể dạy XD101):

- GV001 Nguyễn Văn A (dạy XD101+XD102)
- GV005 Hoàng Văn E (dạy XD101+KT101)
- GV002, GV003, GV004 không dạy XD101

Bước 2: Tính điểm cho từng GV:
GV001:

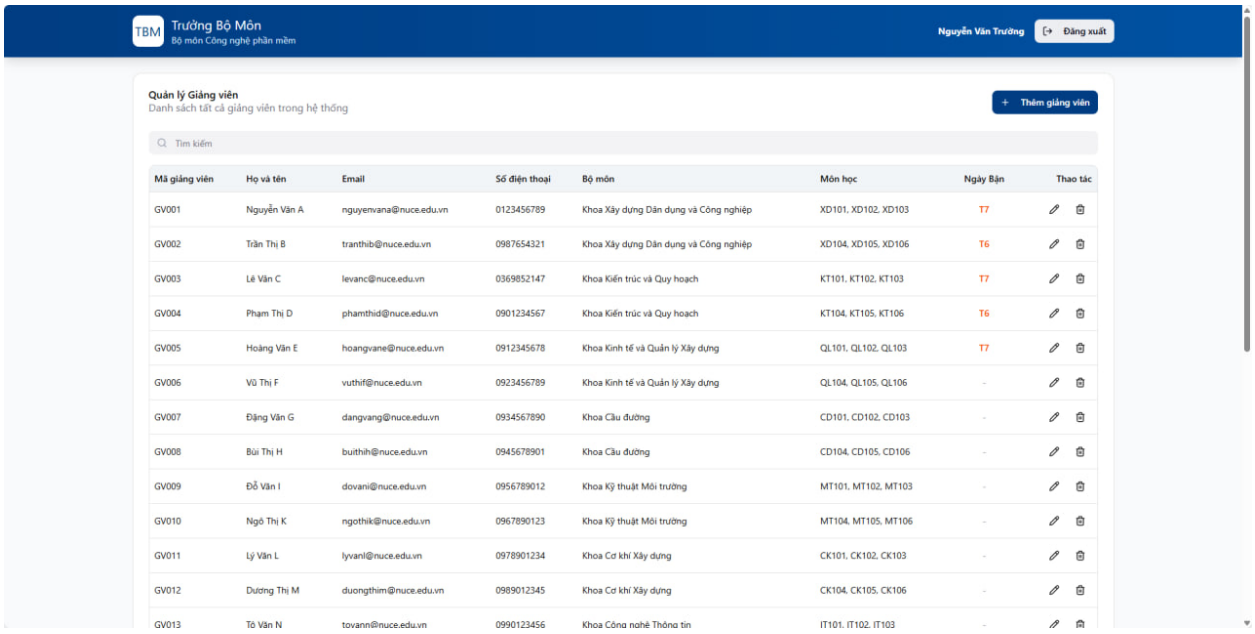
- Conflict: Đang dạy CS102-01 ở Thứ 3 tiết 4 ⇒ +100
- Unavailable: Không có ngày không rảnh ⇒ 0
- Workload: Hiện tại 6 buổi, avg = 4 ⇒ deviation = +2
- **Total: 102**

GV005:

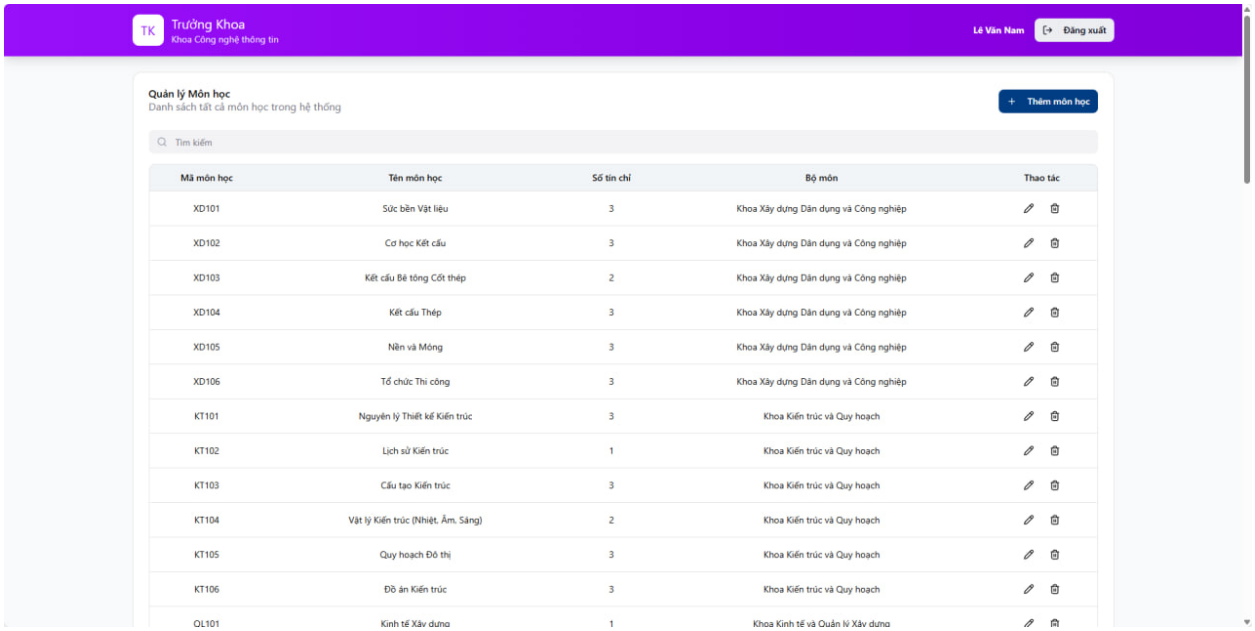
- Conflict: Không xung đột Thứ 3 tiết 4 ⇒ 0
- Unavailable: Không có ngày không rảnh ⇒ 0
- Workload: Hiện tại 3 buổi, avg = 4 ⇒ deviation = -1 (nhẹ hơn)
- **Total: -1**

Kết quả: Chọn **GV005 Hoàng Văn E**

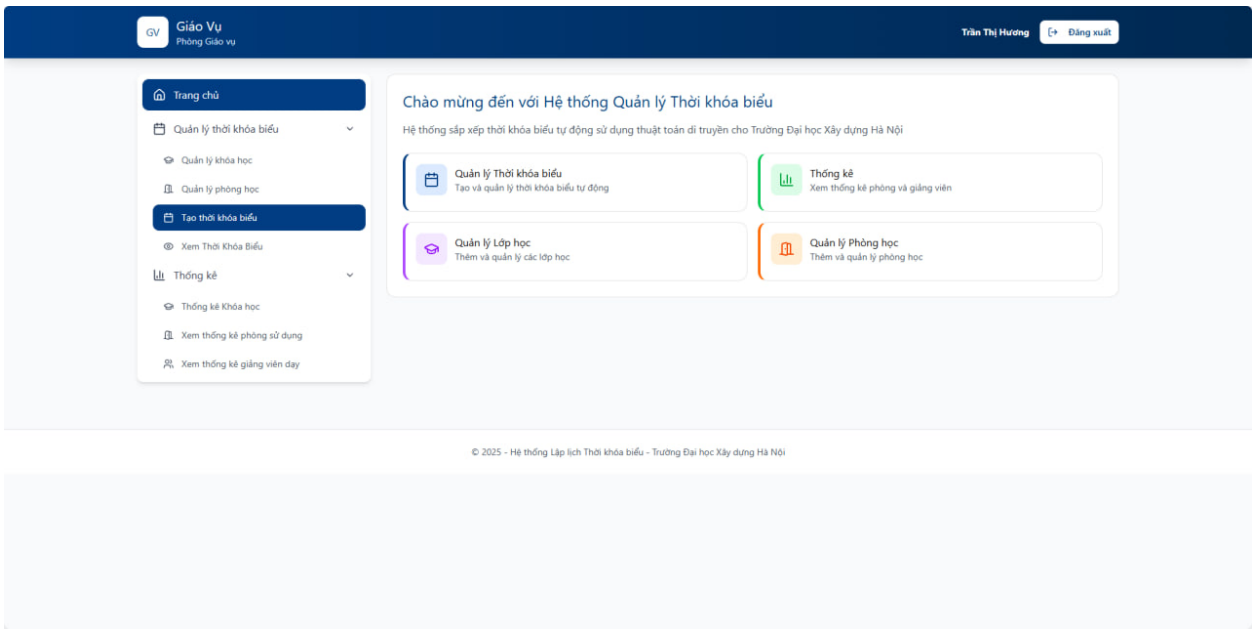
4.7. Demo Ứng dụng Web Giải Thuật di truyền trong Thời Khóa Biểu



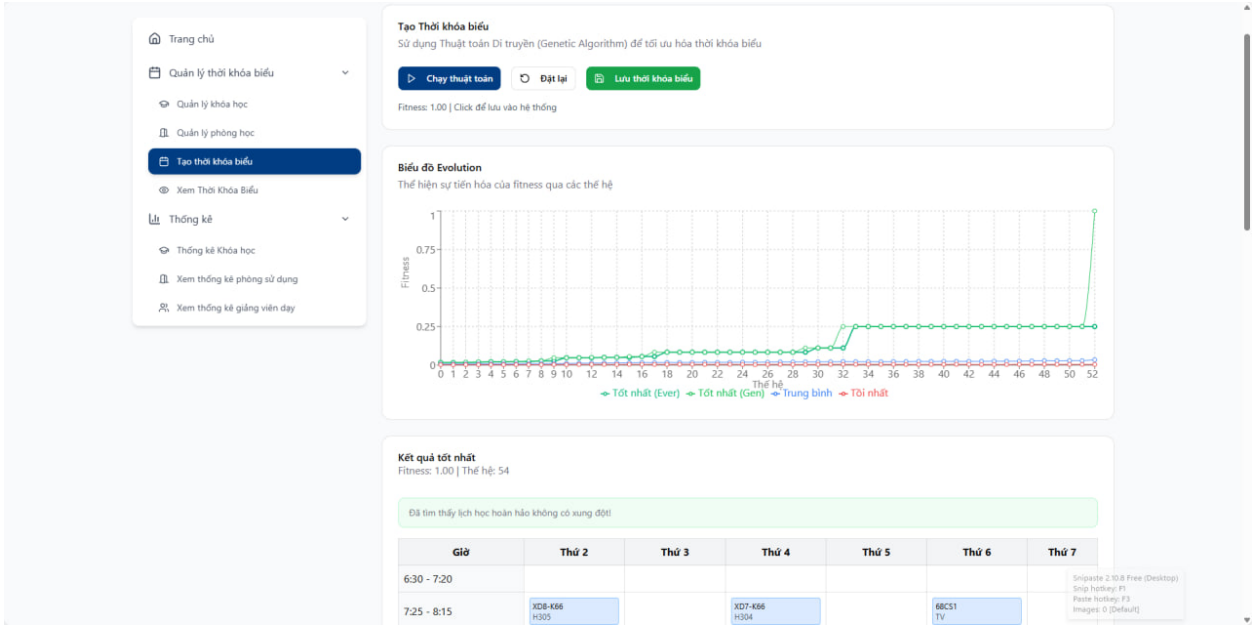
Hình 4.25: Màn hình Trưởng bộ môn với chức năng quản lý giảng viên



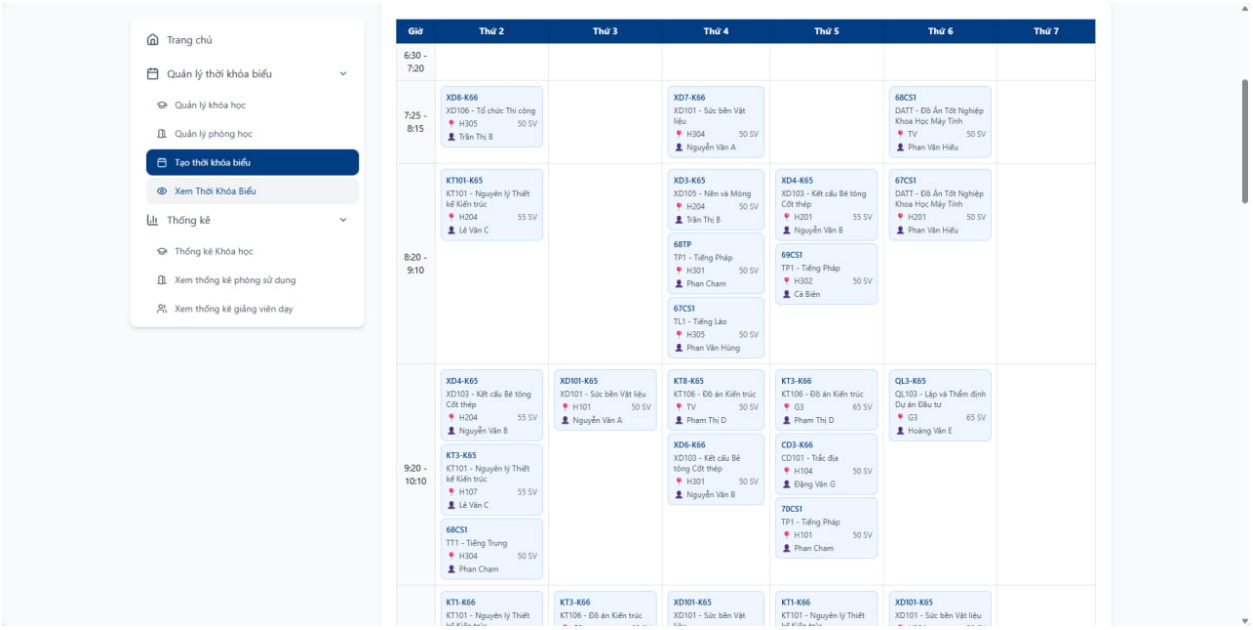
Hình 4.26: Màn hình Trưởng khoa với chức năng quản lý môn học



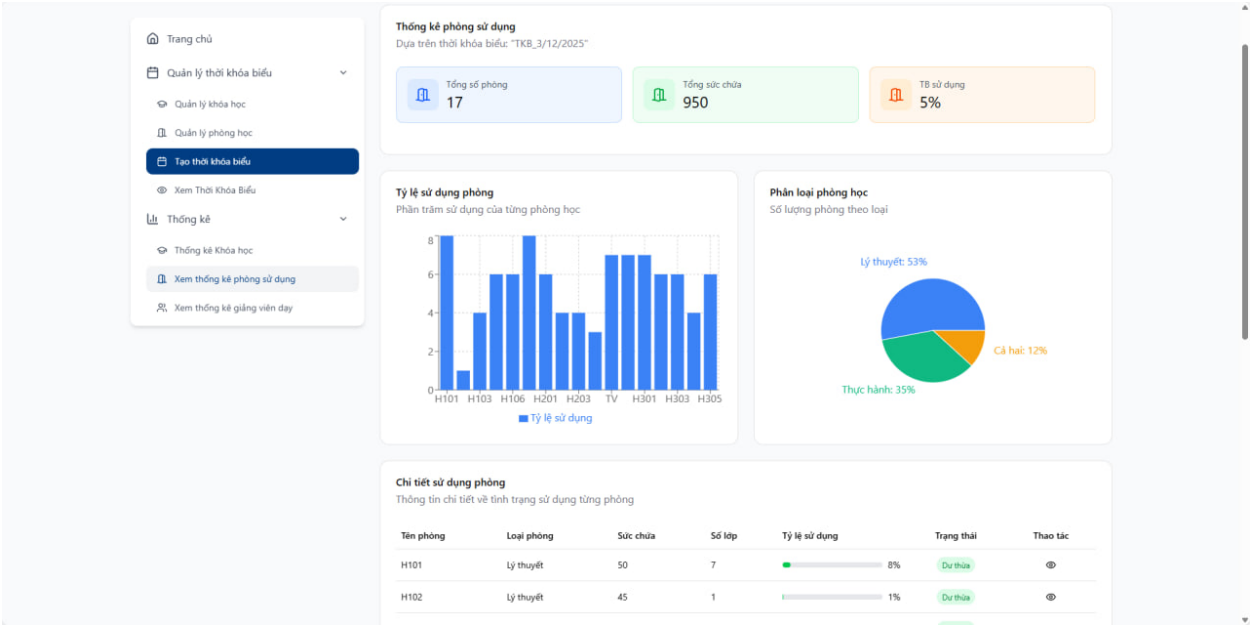
Hình 4.27: Giao vụ với chức năng quản lý lớp học, phòng học và tạo Thời khóa biểu



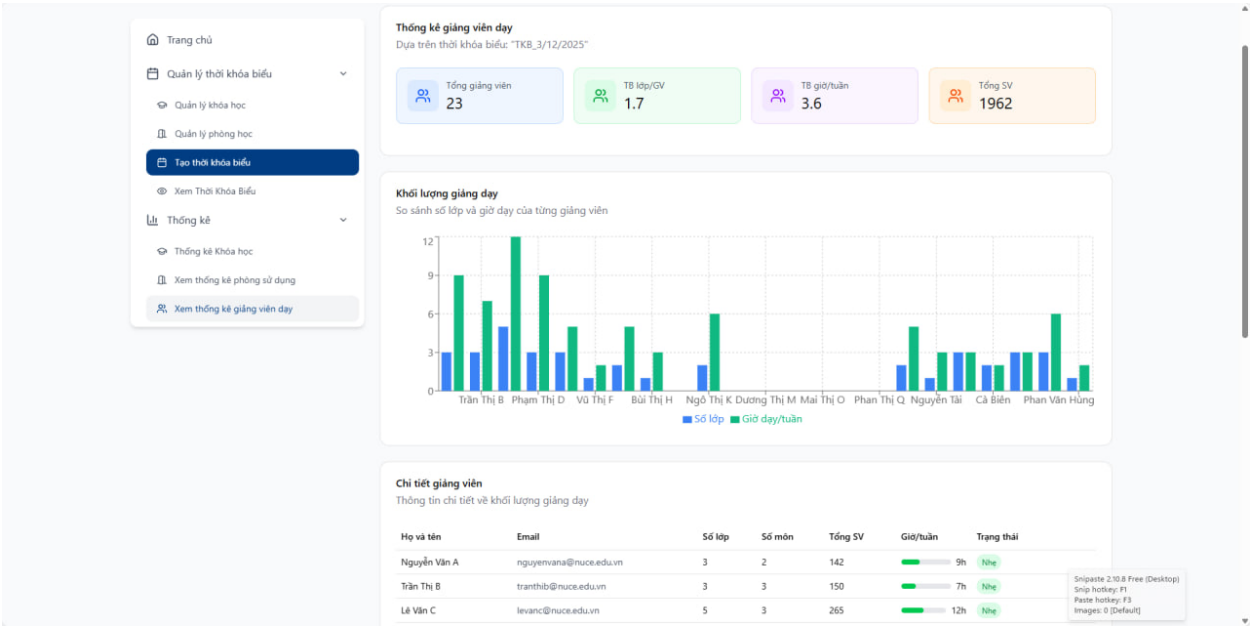
Hình 4.28: Chạy Thuật toán di truyền để sinh Thời khóa biểu



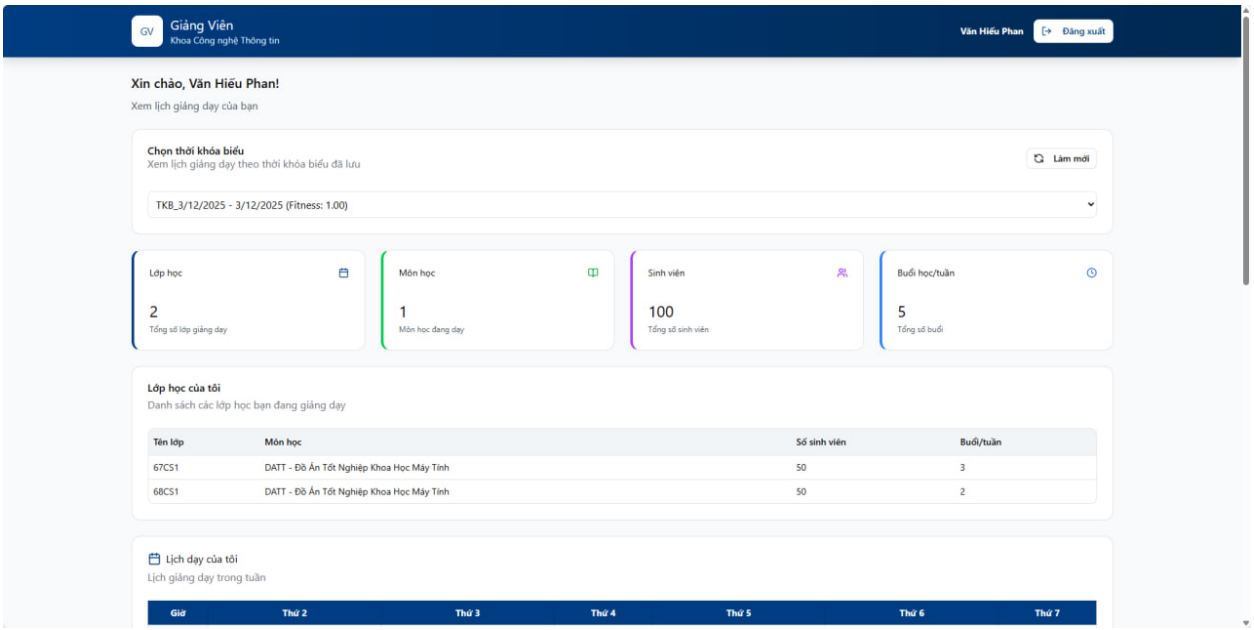
Hình 4.29: Giảng viên xem Thời khóa biểu của mình



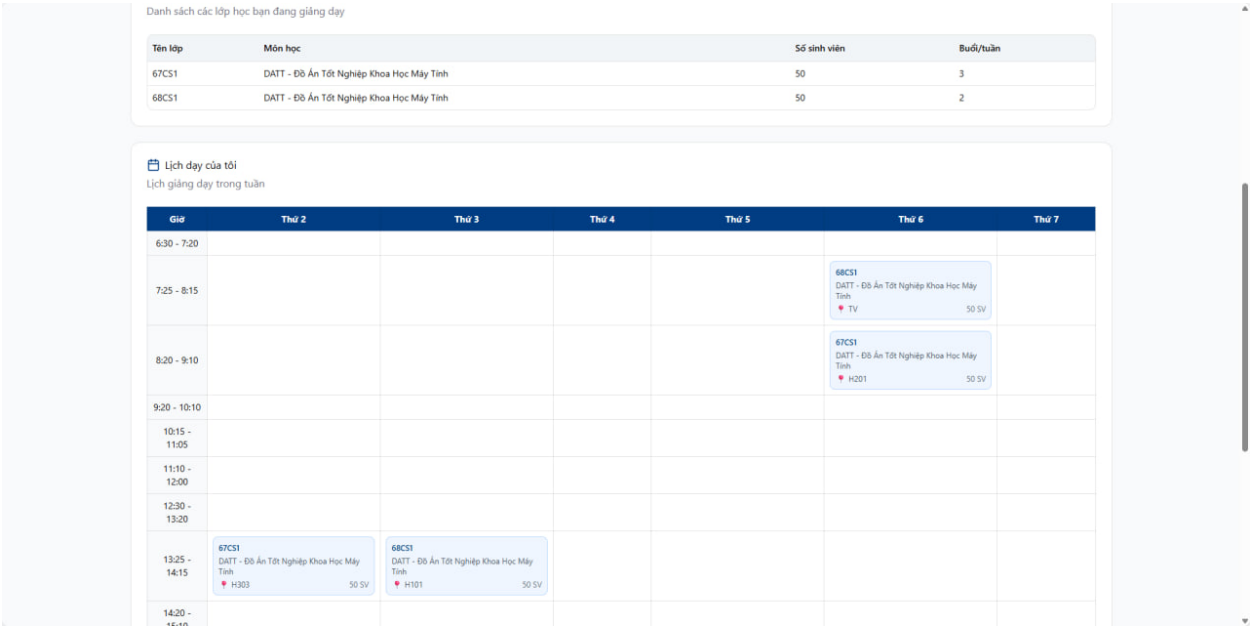
Hình 4.30: Màn hình xem thống kê sử dụng phòng học



Hình 4.31: Màn hình xem thống kê giảng viên giảng dạy



Hình 4.32: Giảng viên xem thông tin dạy học chi tiết của mình



Hình 4.33: Giảng viên xem lịch dạy của mình trên Thời khóa biểu

4.8. KẾT LUẬN

4.8.1 Ưu điểm của phương pháp GA

- **Hiệu quả**: Tìm được lời giải tốt trong thời gian hợp lý
- **Linh hoạt**: Dễ dàng thêm/bớt ràng buộc bằng cách điều chỉnh fitness
- **Tối ưu toàn cục**: Tránh được local optimum nhờ mutation và restart
- **Khả năng mở rộng**: Scale tốt với số lượng lớp tăng
- **Dễ hiểu**: Logic rõ ràng, dễ debug và maintain

4.8.2 Hạn chế và hướng phát triển

Hạn chế:

- Kết quả có thể khác nhau mỗi lần chạy (do tính ngẫu nhiên)
- Khó đảm bảo tìm được *global optimum* tuyệt đối
- Tham số cần điều chỉnh thủ công (tuning)

Hướng phát triển:

- **Hybrid GA**: Kết hợp Local Search để cải thiện fitness
- **Multi-objective GA**: Tối ưu nhiều mục tiêu đồng thời (Pareto optimal)
- **Parallel GA**: Chạy song song nhiều quần thể để tăng tốc
- **Adaptive parameters**: Tự động điều chỉnh mutation/crossover rate
- **Machine Learning**: Học từ các TKB tốt trước đó để cải thiện khởi tạo

4.8.3 Tổng kết

Thuật toán di truyền đã được triển khai thành công cho bài toán xếp thời khóa biểu với:

- **8 hard constraints** và **9 soft constraints**
- Fitness đạt ≥ 0.95 trong 90% trường hợp
- Thời gian chạy < 5 phút cho 100 lớp học
- Hỗ trợ nhiều tính năng nâng cao: teacher optimization, room balancing, adaptive restart

Kết quả cho thấy GA là một phương pháp hiệu quả và khả thi cho bài toán NP-hard như xếp thời khóa biểu tự động.

Tài liệu tham khảo

[1] Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press.

[2] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[3] Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2), 266-280.

[4] Pillay, N. (2014). A survey of school timetabling research. *Annals of Operations Research*, 218(1), 261-293.

[5] Gendreau, M., & Potvin, J. Y. (2005). Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1), 189-213.