**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**ĐỒ ÁN TỐT NGHIỆP**

**Phát triển và ứng dụng thuật toán giải bài toán phân công giảng dạy**

**NGUYỄN ĐÌNH THẮNG**

thang.nd153508@sis.hust.edu.vn

**Ngành Công nghệ thông tin**

**Chuyên ngành Khoa học máy tính**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Phạm Quang Dũng  Chữ ký của GVHD |
| **Bộ môn:** | Khoa học máy tính |
| **Viện:** | Công nghệ thông tin và Truyền thông |
| **HÀ NỘI, 12/2019** | |

**PHIẾU GIAO NHẬN ĐỒ ÁN TỐT NGHIỆP**

1. Thông tin về sinh viên

Họ và tên sinh viên: Nguyễn Đình Thắng

Điện thoại liên lạc: 0334492548 Email: thang.nd153508@sis.hust.edu.vn

Lớp: CNTT 2.01 K60 Hệ đào tạo: Đại học chính quy

Đồ án tốt nghiệp được thực hiện tại: Bộ môn Khoa học máy tính – Viện Công nghệ Thông tin và Truyền thông

Thời gian làm ĐATN: Từ ngày 22/08/2019 đến 22/12/2019

2. Mục đích nội dung của ĐATN

Phát triển và ứng dụng thuật toán giải bài toán phân công giảng dạy tại Viện Công nghệ thông tin và Truyền thông, Trường Đại học Bách Khoa Hà Nội.

3. Các nhiệm vụ cụ thể của ĐATN

* Khảo sát và tìm hiểu bài toán phân công giảng dạy tại Viện Công nghệ thông tin và Truyền thông, Trường Đại học Bách Khoa Hà Nội.
* Tìm hiểu phương pháp tìm kiếm cục bộ dựa trên ràng buộc.
* Đề xuất và cài đặt thuật toán giải bài toán phân công giảng dạy.
* Tìm hiểu công nghệ java Spring, ReactJS.
* Thiết kế và xây dựng phần mềm ứng dụng hỗ trợ người dùng, tạo dữ liệu và chạy thuật toán phân công giảng dạy.

4. Lời cam đoan của sinh viên:

Tôi – *Nguyễn Đình Thắng* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *TS. Phạm Quang Dũng*.

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

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Tác giả ĐATN  *Nguyễn Đình Thắng* |

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

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Giáo viên hướng dẫn  *TS. Phạm Quang Dũng* |

**Lời cảm ơn**

Đầu tiên, em xin gửi những lời cảm ơn chân thành nhất đến các thầy cô trường đại học Bách Khoa Hà Nội, đặc biệt là các thầy cô trong viện Công nghệ Thông Tin và Truyền Thông đã truyền cho em những kiến thức đáng giá trong thời gian học tập tại trường. Đồng thời em cũng xin gửi lời cảm ơn đặc biệt đến TS. Phạm Quang Dũng, người đã tận tình chỉ dẫn và cho những lời khuyên cũng như những kinh nghiệm quý báu, luôn nhắc nhở, động viên và đã cung cấp cho em dữ liệu thực tế để thực hiện đồ án này.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình và bạn bè đã luôn bên cạnh và động viên, giúp em hoàn thành đồ án tốt nghiệp này.

Sinh viên: Nguyễn Đình Thắng, 20153508, lớp CNTT2.01-K60.

**Tóm tắt nội dung đồ án**

Đồ án phát triển thuật toán giải quyết bài toán phân công giảng dạy và đưa vào ứng dụng. Phương pháp giải quyết bài toán là phát biểu bài toán dưới dạng một bài toán tối ưu tổ hợp và giải bài toán bằng phương pháp tìm kiếm cục bộ dựa trên ràng buộc. Sau quá trình thực hiện, kết quả của đồ án là đã đề xuất được một thuật toán hiệu quả và xây dựng được một hệ thống web service ứng dụng thuật toán đề xuất để giải các bài toán phân công giảng dạy. Kết quả của đồ án có tính thực tế cao, đáp ứng yêu cầu để có thể áp dụng vào hỗ trợ công việc phân công giảng dạy của Viện Công nghệ thông tin và Truyền thông. Hướng phát triển tiếp theo của đồ án là xây dựng các bộ dữ liệu của toàn bộ các khoa viện trong Trường và chỉnh sửa thuật toán sao cho phù hợp với thực tế của từng khoa viện.

**MỤC LỤC**

[CHƯƠNG 1. GIỚI THIỆU CHUNG 1](#_Toc28305782)

[1.1 Hoàn cảnh và động lực thực hiện đồ án 1](#_Toc28305783)

[1.2 Mô tả bài toán 1](#_Toc28305784)

[1.3 Cơ sở lý thuyết 2](#_Toc28305785)

[1.3.1 Bài toán tối ưu tổ hợp 2](#_Toc28305786)

[1.3.2 Một số bài toán tối ưu tổ hợp 2](#_Toc28305787)

[1.3.3 Các hướng tiếp cận giải bài toán tối ưu tổ hợp 3](#_Toc28305788)

[1.3.4 Tìm kiếm cục bộ dựa trên ràng buộc 3](#_Toc28305789)

[1.4 Thư viện OpenCBLS 6](#_Toc28305790)

[1.4.1 Biến quyết định 6](#_Toc28305791)

[1.4.2 Bất biến 6](#_Toc28305792)

[1.4.3 Hàm 7](#_Toc28305793)

[1.4.4 Ràng buộc 7](#_Toc28305794)

[1.4.5 Tìm kiếm 8](#_Toc28305795)

[CHƯƠNG 2. THUẬT TOÁN ĐỀ XUẤT 9](#_Toc28305796)

[2.1 Mô hình hóa bài toán 9](#_Toc28305797)

[2.2 Sơ đồ chung thuật toán 11](#_Toc28305798)

[2.3 Pha khởi tạo lời giải ban đầu 11](#_Toc28305799)

[2.3.1 Chiến lược khởi tạo ngẫu nhiên 11](#_Toc28305800)

[2.3.2 Chiến lược khởi tạo tham lam tối thiểu hóa số vi phạm 12](#_Toc28305801)

[2.3.3 Chiến lược khởi tạo tham lam tối đa hóa độ phù hợp 12](#_Toc28305802)

[2.4 Pha tìm kiếm lời giải tối ưu 13](#_Toc28305803)

[2.4.1 Chiến lược tìm kiếm chia thành từng bước độc lập 14](#_Toc28305804)

[2.4.2 Chiến lược tìm kiếm tích hợp nhiều bước 23](#_Toc28305805)

[2.5 Mô tả các lớp đã cài đặt thêm 25](#_Toc28305806)

[2.5.1 Bổ sung thêm vào lớp VarIntLS 25](#_Toc28305807)

[2.5.2 Tạo mới lớp SumPriority 26](#_Toc28305808)

[2.5.3 Tạo mới lớp Balance 27](#_Toc28305809)

[2.5.4 Tạo mới lớp TabuSearchM 27](#_Toc28305810)

[CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG 29](#_Toc28305811)

[3.1 Phân tích chức năng 29](#_Toc28305812)

[3.1.1 Biểu đồ usecase 29](#_Toc28305813)

[3.1.2 Đặc tả usecase 29](#_Toc28305814)

[3.2 Phân tích cấu trúc 31](#_Toc28305815)

[3.3 Thiết kết kiến trúc 34](#_Toc28305816)

[3.4 Thiết kế dữ liệu 36](#_Toc28305817)

[3.5 Thiết kế giao diện 39](#_Toc28305818)

[CHƯƠNG 4. KẾT QUẢ THỬ NGHIỆM 44](#_Toc28305819)

[4.1 Tạo bộ dữ liệu 44](#_Toc28305820)

[4.1.1 Dữ liệu thô ban đầu 44](#_Toc28305821)

[4.1.2 Tiền xử lý 45](#_Toc28305822)

[4.2 Phân tích kết quả thử nghiệm 47](#_Toc28305823)

[CHƯƠNG 5. KẾT LUẬN 54](#_Toc28305824)

[5.1 Kết luận 54](#_Toc28305825)

[5.2 Hướng phát triển của đồ án trong tương lai 54](#_Toc28305826)

[TÀI LIỆU THAM KHẢO 55](#_Toc28305827)

**DANH MỤC HÌNH VẼ**

[Hình 3.1 Sơ đồ Usecase của hệ thống 29](#_Toc28305828)

[Hình 3.2 Biểu đồ lớp phân tích Quản lý danh sách bài toán 31](#_Toc28305829)

[Hình 3.3 Biểu đồ lớp phân tích Xử lý dữ liệu đầu vào 32](#_Toc28305830)

[Hình 3.4 Biểu đồ lớp phân tích Tạo lời giải cho bài toán 32](#_Toc28305831)

[Hình 3.5 Biểu đồ lớp phân tích Thêm lời giải thủ công 33](#_Toc28305832)

[Hình 3.6 Biểu đồ lớp phân tích Phân tích lời giải bài toán 33](#_Toc28305833)

[Hình 3.7 Biểu đồ lớp phân tích Xuất lời giải bài toán 34](#_Toc28305834)

[Hình 3.8 Biểu đồ kiến trúc phân tầng của hệ thống 34](#_Toc28305835)

[Hình 3.9 Sơ đồ triển khai của hệ thống 35](#_Toc28305836)

[Hình 3.10 Sơ đồ thiết kế cơ sở dữ liệu của hệ thống 36](#_Toc28305837)

[Hình 3.11 Giao diện đăng nhập 39](#_Toc28305838)

[Hình 3.12 Giao diện trang bắt đầu 40](#_Toc28305839)

[Hình 3.13 Giao diện trang danh sách lớp 40](#_Toc28305840)

[Hình 3.14 Giao diện trang danh sách học phần cho giảng viên 41](#_Toc28305841)

[Hình 3.15 Giao diện trang lời giải của bài toán 41](#_Toc28305842)

[Hình 3.16 Giao diện trang chi tiết một lời giải 42](#_Toc28305843)

[Hình 3.17 Hộp thoại thêm bài toán 42](#_Toc28305844)

[Hình 3.18 Hộp thoại tạo lời giải mới 43](#_Toc28305845)

[Hình 3.19 Hộp thoại thêm lời giải thủ công 43](#_Toc28305846)

[Hình 4.1 Biểu đồ phân bố số tín chỉ của một giảng viên – thuật toán - 20182 49](#_Toc28305847)

[Hình 4.2 Biểu đồ phân bố số tín chỉ của một giảng viên – thủ công - 20182 49](#_Toc28305848)

[Hình 4.3 Biểu đồ phân bố số tín chỉ của một giảng viên – thuật toán - 20191 51](#_Toc28305849)

[Hình 4.4 Biểu đồ phân bố số tín chỉ của một giảng viên – thủ công - 20191 51](#_Toc28305850)

[Hình 4.5 Biểu đồ phân bố số tín chỉ của một giảng viên – thuật toán - 20192 53](#_Toc28305851)

[Hình 4.6 Biểu đồ phân bố số tín chỉ của một giảng viên – thủ công - 20192 53](#_Toc28305852)

**DANH MỤC BẢNG BIỂU**

[Bảng 2.1Mô tả ký hiệu trong mô hình hóa bài toán 9](#_Toc28305853)

[Bảng 3.1 Mô tả bảng user 37](#_Toc28305854)

[Bảng 3.2 Mô tả bảng input\_request 37](#_Toc28305855)

[Bảng 3.3 Mô tả bảng invalid\_class 37](#_Toc28305856)

[Bảng 3.4 Mô tả bảng parameter 38](#_Toc28305857)

[Bảng 3.5 Mô tả bảng solution 39](#_Toc28305858)

[Bảng 4.1 Mô tả các trường dữ liệu danh sách lớp ban đầu 44](#_Toc28305859)

[Bảng 4.2 Mô tả dữ liệu phân công học phần và giảng viên 45](#_Toc28305860)

[Bảng 4.3 Mô tả danh sách lớp sau xử lý 46](#_Toc28305861)

[Bảng 4.4 Mô tả danh sách học phần cho giảng viên sau xử lý 47](#_Toc28305862)

[Bảng 4.5 Phân tích kết quả xếp thủ của công kỳ 20182 48](#_Toc28305863)

[Bảng 4.6 Phân tích kết quả thuật toán của học kỳ 20182 48](#_Toc28305864)

[Bảng 4.7 Phân tích kết quả xếp thủ của công kỳ 20191 50](#_Toc28305865)

[Bảng 4.8 Phân tích kết quả thuật toán của học kỳ 20191 50](#_Toc28305866)

[Bảng 4.9 Phân tích kết quả xếp thủ của công kỳ 20192 52](#_Toc28305867)

[Bảng 4.10 Phân tích kết quả thuật toán của học kỳ 20192 52](#_Toc28305868)

# GIỚI THIỆU CHUNG

Phần này trình bày về hoàn cảnh, động lực thực hiện đồ án và cơ sở lý thuyết được sử dụng trong đồ án này.

## Hoàn cảnh và động lực thực hiện đồ án

Hiện nay, công nghệ thông tin đang được ứng dụng rộng rãi vào trong các lĩnh vực của đời sống, giúp cải thiện và nâng cao chất lượng công việc hàng ngày. Đặc biệt trong lĩnh vực quản lý, rất nhiều công việc đang dần được hỗ trợ và thay thế dần bởi các phần mềm quản lý. Tuy nhiên, vẫn còn một số công việc vẫn đang được thực hiện bởi con người như các bài toán lập lịch, sắp xếp nguồn lực. Những công việc này nếu phải thực hiện bằng con người sẽ mất rất nhiều thời gian của người quản lý, đồng thời, kết quả cũng không được tối ưu. Hiện nay, tại các khoa viện của trường Đại học Bách Khoa Hà Nội, công việc phân công giảng viên cho các lớp học vẫn được thực hiện thủ công, dẫn đến mất nhiều thời gian và gây nhàm chán cho người quản lý.

Nhận thấy bài toán sắp xếp giảng viên vào các lớp học là một vấn đề thực tế rất đáng được giải quyết và phù hợp để áp dụng những kiến thức được đã học trong trường. Vì vậy trong đồ án này, chúng tôi nghiên cứu phát triển thuật toán và xây dựng phần mềm ứng dụng hỗ trợ người dùng thực hiện việc phân công giảng dạy một cách thuận tiện.

## Mô tả bài toán

Phòng đào tạo nhận thông tin đăng ký học tập của sinh viên và và quyết định số lượng, địa điểm và thời gian học của lớp học sẽ được mở. Sau đó, phòng đào tạo gửi thông tin của các lớp học về các khoa viện trong trường để các khoa viện tự sắp xếp giảng viên. Các khoa viện dựa vào thông tin về các học phần mà giảng viên của viện có thể giảng dạy để phân công.

Đầu vào của bài toán sẽ bao gồm:

* Thông tin của các lớp học cần phân công giảng viên: học phần, loại lớp, thời gian.
* Thông tin về các học phần mà từng giảng viện có thể giảng dạy, độ ưu tiên của giảng viên đối với từng học phần và số tín chỉ tối đa của giảng viên.

Đầu ra của bài toán là danh sách sắp xếp giảng viên cho các lớp học.

Bài toán đặt ra một số ràng buộc cần phải thỏa mãn như:

* Giảng viên sẽ chỉ được dạy các lớp của các học phần mà giảng viên có thể dạy.
* Các lớp được phân cho cùng một giảng viên sẽ không bị trùng lịch học với nhau.
* Tổng số tín chỉ được phân cho một giảng viên phải nhỏ hơn số tín chỉ tối đa của giảng viên đó.
* Mỗi lớp học sẽ có duy nhất một giảng viên.

## Cơ sở lý thuyết

### Bài toán tối ưu tổ hợp

Tối ưu tổ hợp là một trong những lĩnh vực non trẻ và phát triển mạnh mẽ của toán rời rạc. Nó đã trở thành một lĩnh vực khoa học được nghiên cứu vào khoảng cách đây 60 năm. [1]

Tối ưu tổ hợp được bắt nguồn từ các ngành khoa học nghiên cứu về tổ hợp, vận trù học và lý thuyết tính toán. Một trong những động lực quan trọng thúc đẩy sự phát triển của tối ưu tổ hợp là hàng loạt các bài toán trong thực tiễn ứng dụng có thể phát biểu dưới dạng các bài toán tối ưu tổ hợp.

Một cách tổng quát, bài toán tối ưu tổ hợp có thể phát biểu như sau:

Cho một bộ ba , trong đó là tập hữu hạn phương án (hay lời giải), là hàm mục tiêu xác định trên và là tập các ràng buộc phải thỏa mãn. Mỗi lời giải thỏa mãn các ràng buộc được gọi là phương án (hay lời giải) chấp nhận được. Mục đích của ta là tìm được phương án tối ưu hóa toàn cục hàm mục tiêu . Chẳng hạn với bài toán cực tiểu hóa thì với mọi phương án chấp nhận được . [2]

### Một số bài toán tối ưu tổ hợp

Trong đời sống và đặc biết trong các hệ thống thông tin ta thường gặp các bài toán tối ưu tổ hợp quan trọng. Sau đây là một số ví dụ của các bài toán tối ưu tổ hợp kinh điển:

- Bài toán người du lịch (TSP) [3]: cho tọa độ của thành phố, tìm đường đi ngắn nhất đi qua tất cả các thành phố đúng một lần duy nhất.

- Bài toán đóng thùng [4]: cho tập vật với mỗi vật có kích thước , xếp chúng vào ít thùng có kích thước nhất có thể.

- Bài toán quy hoạch nguyên tuyến tính: Tối đa hóa một tổ hợp tuyến tính của các biến nguyên thỏa mãn các ràng buộc có dạng:

- Bài toán lập lịch công việc: cho một tập các công việc cần thực hiện và một tập các công cụ mà các công việc cần để có thể thực hiện được, lập lịch thực hiện và các công cụ cho từng công việc để tối thiểu hóa thời gian từ khi bắt đầu tới khi tất cả các công việc đều được hoàn thành.

### Các hướng tiếp cận giải bài toán tối ưu tổ hợp

Các hướng tiếp cận phổ biến để giải bài toán tối ưu tổ hợp là: phương pháp nhánh cận, siêu phẳng cắt, tìm kiếm địa phương, thuật toán tham lam, thuật toán gần đúng.

### Tìm kiếm cục bộ dựa trên ràng buộc

Tìm kiếm cục bộ dựa trên ràng buộc [5] đã được áp dụng thành công để giải rất nhiều bài toán trong công nghiệp. Đây cũng là thuật toán được lựa chọn để giải quyết bài toán Phân công giảng dạy.

#### Sơ đồ tổng quát

1. Định nghĩa

Tìm kiếm cục bộ là thuật toán lặp thực hiện việc di chuyển từ một lời giải đến một lời giải mới trong một cấu trúc lân cận xác định trước. Thuật toán dừng khi đến được tối ưu cục bộ.

Ưu điểm của tìm kiếm cục bộ là dễ cài đặt, đảm bảo tìm được tối ưu cục bộ với chi phí thời gian thấp và không cần mô hình chính xác của bài toán.

Nhược điểm của thuật toán là cho lời giải chất lượng kém do gặp bế tắc khi đạt tối ưu cục bộ chất lượng tồi.

1. Sơ đồ tổng quát

Sơ đồ tổng quát của thuật toán Tìm kiếm cục bộ dựa trên ràng buộc:

1. **Khởi tạo**: Chọn lời giải xuất phát , tính giá trị hàm mục tiêu
2. **Sinh lân cận**: Chọn lân cận và tìm lời giải trong lân cận này với giá trị hàm mục tiêu
3. **Kiểm tra chấp nhận**: Kiểm tra xem có chấp nhận di chuyển từ sang . Nếu chấp nhận thì thay bởi ; trái lại giữ nguyên là lời giải hiện tại.
4. **Kiểm tra dừng**: Kiểm tra điều kiện dừng. Nếu điều kiện dừng thỏa mãn thì kết thúc thuật toán và đưa ra lời giải tốt nhất tìm được; trái lại quay lại bước 2.

Tại phần khởi tạo, lời giải S có thể được chọn ngẫu nhiên trong không gian lời giải của bài toán hoặc là được chọn bởi một thuật toán tham lam. Khởi tạo bằng thuật toán tham lam phù hợp sẽ giúp cho quá trình tìm kiếm nhanh đến được tối ưu cục bộ tốt hơn.

Hàm sinh lân cận thường được xác định bởi việc sử dụng khái niệm dịch chuyển, cho phép thay đổi một hoặc nhiều hơn thuộc tính của lời giải để sinh ra lời giải mới. Lời giải được gọi là tối ưu cục bộ trong hàm lân cận , nếu với mọi . Lân cận càng lớn thì càng khó khảo sát nhưng cho lời giải tối ưu cục bộ với chất lượng tốt hơn. Việc xác định lân cận cần được cân nhắc giữa chất lượng lời giải và thời gian khảo sát.

Trong bước kiểm tra chấp nhận, việc quyết định xem có chấp nhận di chuyển từ sang hay không có nhiều cách. Sau đây là các ví dụ về các thuật toán cùng sử dụng sơ đồ tìm kiếm cục bộ nhưng khác khác nhau tại bước kiểm tra chấp nhận.

- Thuật toán iterative improvement: Lời giải sẽ chỉ được chọn khi

- Thuật toán threshold accepting: Lời giải sẽ được chọn chỉ khi . Đầu tiên có thể được đặc giá trị lớn, rồi sau đó giảm dần.

- Thuật toán Simulated Annealing: Xác định ; Nếu , thì việc di chuyển sang luôn được chấp nhận; Nếu , thì việc chuyển sang được chấp nhận với xác suất:

Trong đó là thông số nhiệt độ, nó được thay đổi trong quá trình thực hiện thuật toán. Thông thường là có giá trị lớn khi bắt đầu và sau đó được giảm dần đến 0 tại những bước cuối.

#### Thuật toán leo đồi – Hill Climbing

Thuật toán leo đồi là một kỹ thuật tối ưu trong nhóm các thuật toán tìm kiếm cục bộ hay được sử dụng. Đây là một thuật toán lặp bắt đầu với một lời giải ngẫu nhiên cho bài toán, sau đó cố gắng tìm một lời giải tốt hơn bằng cách cải thiện lời giải hiện có. Quá trình lặp diễn ra cho đến khi không thể tìm được một lời giải nào tốt hơn. [6]

Các bước của thuật toán được mô tả như sau:

1. Chọn ngẫu nhiên một điểm trong không gian tìm kiếm.
2. Xét tất cả các phương án trong lân cận của phương án hiện tại.
3. Chọn phương án với chất lượng tốt nhất trong lân cận và chuyển đến phương án mới này.
4. Lặp lại bước 2 đến 4 cho đến khi tất cả các lân cận đều có chất lượng tồi hơn phương án hiện tại.
5. Trả lại phương án hiện tại như lời giải tìm được.

Mã giả:

Thuật toán 1.1 Hill-Climbing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | **function** Hill-Climbing (bài toán): Trả về một lời giải | | | | |
| *2* |  | **input**: bài toán | | | |
| *3* |  | Khởi tạo lời giải hiện tại, lời giải tiếp theo ; | | | |
| *4* |  | Gán lời giải hiện tại lời giải ngẫu nhiện của bài toán ; | | | |
| *5* |  | **for** | | | |
| *6* |  |  | Gán lời giải tiếp theo một lân cận tốt nhất của lời giải hiện tại ; | | |
| *7* |  |  | **if** lời giải tiếp theo tồi hơn lời giải thiện tại **then** | | |
| *8* |  |  |  | **return** lời giải hiện tại ; |
| *9* |  |  | **endif** | | |
| *10* |  |  | Gán lời giải hiện tại lời giải tiếp theo ; | | |
| *11* |  | **endfor** | | | |
| *12* |  | **return** lời giải hiện tại ; | | | |

#### Tabu Search

Thuật toán leo đồi ở trên rất dễ đưa ra lời giải tối ưu cục bộ tồi do chỉ chấp nhận lời giải tiếp theo tốt hơn lời giải hiện tại. Một thuật toán khác giải quyết được vấn đề này là thay vì chỉ chấp nhận lời giải mới tốt hơn, thuật toán sẽ chấp nhận lời giải tốt nhất trong số các lân cận của lời giải hiện tại. [7]

Tuy nhiên, thuật toán này gặp một vấn đề là lặp lại lời giải đã đi qua, làm tăng thời gian tìm được lời giải tối ưu cục bộ. Để ngăn việc này xảy ra, cách đơn giản là ghi lại các phương án đã được thăm để ngăn cấm việc thăm lại các phương án đó trong tương lai. Tuy nhiên, việc sử dụng bộ nhớ để lưu lại toàn bộ các phương án đã thăm là không khả thi vì số lượng phương án đã thăm là rất lớn và bản than mỗi phương án cũng chiếm kích thước bộ nhớ không nhỏ. Trên thực tế, một hướng giải quyết được đề xuất là dùng bộ nhớ để lưu trữ thông tin về một số hữu hạn bước di chuyển cục bộ gần đây nhất để tạo ra phương án thay vì lưu trữ bản thân phương án đó.

Bộ nhớ lưu trữ này được gọi là danh sách tabu hay danh sách cấm và số hữu hạn bước di chuyển cục bộ gần đây nhất được lưu trong danh sách tabu được gọi là độ dài của danh sách tabu.

Với hướng giải quyết này, một bước di chuyển cục bộ có mặt trong danh sách tabu không đồng nghĩa phương án mà bước di chuyển cục bộ đó sinh ra (tại thời điểm bước di chuyển cục bộ đó được sử dụng) đã được viếng thăm. Chính vì lý do đó, trong quá trình khảo sát láng giềng, một bước di chuyển cục bộ tuy nằm trong danh sách tabu nhưng việc ứng dụng nó sẽ sinh ra phương án tốt hơn phương án tốt nhất đã tìm thấy trước đó vẫn sẽ được xem xét. Ngoài ra thuật toán tìm kiếm tabu cũng thường được khởi động lại khi số bước lặp liên tiếp trong đó thuật toán không tìm ra được phương án tốt hơn phương án tốt nhất đã tìm được lớn hơn hoặc bằng một tham số tìm kiếm.

## Thư viện OpenCBLS

OpenCBLS (https://github.com/dungkhmt/OpenCBLS) [8] là một thư viện mã nguồn mở được xây dựng bằng ngôn ngữ Java. Thư viện có chức năng mô hình hóa và tìm kiếm lời giải cho bài toán tối ưu tổ hợp bằng phương pháp tìm kiếm cục bộ dựa trên ràng buộc. Thư viện OpenCBLS được xây dựng dựa trên cơ sở kiến trúc tìm kiếm cục bộ dựa trên ràng buộc của Van Hentenryck và Micheal năm 2005.

### Biến quyết định

Biến quyết định là đối tượng dùng để mô hình hóa phương án của bài toán. Trong đó biến quyết định được khai báo như sau:

VarIntLS x = new VarIntLS(mgr, min, max);

Trong đó, “mgr” là đối tượng của lớp LocalSearchManager có tác dụng quản lý và duy trì mối quan hệ giữa các đối tượng trong mô hình, “min” và “max” xác định miền giá trị của biến quyết định x. Trong quá trình tìm kiếm cục bộ, các biến liên tục được thay đổi giá trị để sinh ra các phương án mới. Giá trị của các hàm và mức độ vi phạm các ràng buộc xác định trên các biến quyết định cũng được thay đổi. Phương thức setValuePropagate(int val) của biến quyết định x sẽ gán giá trị “val” cho biến đồng thời kích hoạt thủ tục để cập nhật lại giá trị các hàm, mức độ vi phạm ràng buộc được phụ thuộc vào biến x.

### Bất biến

Bất biến là khái niệm dùng để chỉ các đối tượng duy trì các thuộc tính (tổng giá tị các biến quyết định, tập các biến quyết định có giá trị lớn nhất, …) trong mô hình bài toán. Các đối tượng này được định nghĩa dựa trên các biến quyết định. Trong quá trình tìm kiếm cục bộ, giá trị của các biến quyết định được thay đổi và các thuộc tính được duy trì trong các bất biến cũng được cập nhật theo. Các đối tượng bất biến đều thi hành Invariant sau đây:

1. Public interface Invariant {
2. public VarIntLS[] getVariables();
3. public void propagateInt(VarIntLS x, int val);
4. public void initPropagate();
5. public LocalSearchManager getLocalSearchManager();
6. }

Hàm getVariables() trả về tập các biến quyết định tham gia vào định nghĩa bất biến. Hàm propagateInt(VarIntLS x, int Val) sẽ cập nhật lại các thuộc tính duy trì trong bất biến khi biến quyết định x được gán giá trị val. Hàm initPropagate() thực hiện khởi tạo các cấu trúc dữ liệu duy trì trong bất biến khi mô hình được đóng (hàm close() của đối tượng LocalSearchManager được gọi).

### Hàm

Hàm là khái niệm chỉ các đối tượng duy trì các thuộc tính của mô hình bài toán tối ưu tổ hợp, cung cấp các phương thức cho phép truy vấn sự thay đổi của các thuộc tính này khi biến quyết định thay đổi giá trị. Các hàm trong thư viện OpenCBLS đều thi hành interface sau đây:

1. public interface IFunction extends Invariant {
2. public int getMinValue();
3. public int getMaxValue();
4. public int getValue();
5. public int getAssignDelta(VarIntLS x, int val);
6. public int getSwapDelta(VarIntLS x, VarIntLS y);
7. }

Hàm getMinValue() và getMaxValue() trả về giá trị nhỏ nhất và giá trị lớn nhất mà hàm có thể đạt tới, giá trị này phụ thuộc vào miền giá trị biến quyết định đã tham gia định nghĩa hàm. Hàm getValue() trả về giá trị hiện tại của hàm. Các hàm getAssignDelta(VarIntLS x, int val) và getSwapDelta(VarIntLS x, VarIntLS y) trả về sự thay đổi giá trị của hàm khi biến x được gán giá trị val và 2 biến x và y hoán đổi giá trị cho nhau. Ví dụ một số hàm:

FuncPlus (FuncMinus): biểu diễn đối tượng là tổng (hiệu) của 2 đối tượng khác, 2 đối tượng này có thể là VarIntLS hoặc IFunction.

Sum: biểu diễn đối tượng là tổng một mảng các đối tượng VarIntLS hoặc IFunction.

ConditionalSum: Biểu diễn đối tượng là tổng của các phần tử trong một mảng thỏa mãn điều kiện. Hàm tạo ConditionalSum(VarIntLS[] x, int[] w, int val) sẽ khởi tạo một đối tượng biểu diễn tổng các phần tử của w[i] với x[i] có giá trị bằng val.

### Ràng buộc

Ràng buộc là khái niệm chỉ đối tượng duy trì các thuộc tính về số vi phạm của lời giải trong mô hình bài toán tối ưu tổ hợp. Đối tượng này cung cấp phương thức truy vấn giá trị số vi phạm của lời giải hiện tại. Các ràng buộc trong thư viện OpenCBLS đề thi hàng interface sau đây:

1. public interface IContraint extends Invariant {
2. public int violations();
3. public int violations(VarIntLS x);
4. public int getAssignDelta(VarIntLS x, int val);
5. public int getSwapDelta(VarIntLS x, VarIntLS y);
6. }

Hàm violations() trả về tổng số vi phạm ràng buộc. Hàm violations(VarIntLS x) trả về tổng số vi phạm ràng buộc mà biến x đóng góp vào. Các hàm getAssignDelta(VarIntLS x, int val) và getSwapDelta(VarIntLS x, VarIntLS y) trả về sự thay đổi số vi phạm ràng buộc khi biến x được gán giá trị val và 2 biến x, y hoán đổi giá tị cho nhau. Ví dụ một số ràng buộc:

- LessOrEqual, LessThan, IsEqual, NotEqual, Implicate: biểu diễn đối tượng thể hiện các ràng buộc . AND, OR biểu diễn đối tượng thể hiện toán tử giữa hai hay nhiều ràng buộc.

- AllDifferent: biểu diễn đối tượng thể hiện ràng buộc đôi một khác nhau giữa các phần tử trong một mảng.

- MultiKnapsack: biểu diễn đối tượng thể hiện ràng buộc về sức chứa các đối tượng được xếp vào các vật chứa khác. Hàm tạo có dạng MultiKnapsack(VarIntLS[] x, int[] w, int[] c), trong đó w thể hiện trọng lượng của các đối tượng sẽ được xếp vào các vật chứa mà sức chứa được thể hiện ở mảng c, biến quyết định x[i] thể hiện chỉ số của vật chứa mà vật i được xếp vào.

### Tìm kiếm

Tìm kiếm là các lớp biểu diễn các phương thức tìm kiếm cục bộ dựa trên ràng buộc. Mô tả các phương thức tìm kiếm trong lớp TabuSearch của thư viện OpenCBLS như sau:

Phương thức search(IConstraint C, int tabulen, int maxTime, int maxIter, int maxStable) thực hiện chiến lược tìm kiếm tabu với mục tiêu tối thiểu hóa vi phạm ràng buôc C với số bước lặp tối đa maxIter, thời gian tối đa maxTime (giây), độ dài danh sách tabu là tabulen và toán tử di chuyển cục bộ là phép gán.

Phương thức searchMaintainConstraintsMinimize(IFunction f, IConstraint C, int tabulen, int maxTime, int maxStable) thực hiện chiến lược tìm kiếm tabu với mục tiêu tối thiểu hóa hàm f trong đó duy trì vi phạm ràng buộc C (có nghĩa khi khảo sát láng giềng, thuật toán bỏ qua các phương án là số vi phạm tăng lên) với số bước lặp tối đa maxIter, thời gian tối đa maxTime (giây), trong đó tabulen là độ dài danh sách tabu và toán tử di chuyển cục bộ là phép gán.

# THUẬT TOÁN ĐỀ XUẤT

Chương này mô tả chi tiết thuật toán được đề xuất để giải bài toán phân công giảng dạy.

## Mô hình hóa bài toán

Việc đầu tiên để giải bài toán phân công giảng dạy là phải phát biểu bài toán dưới dạng bài toán tối ưu tổ hợp, hay mô hình hóa bài toán. Sau đây là mô hình toán học của bài toán phân công giảng dạy:

**Mô tả các ký hiệu**:

Bảng 2.1Mô tả ký hiệu trong mô hình hóa bài toán

|  |  |
| --- | --- |
| **Ký hiệu** | **Mô tả** |
|  | Tập hợp các cặp hai lớp bị trùng lịch học và không thể cùng phân công cho một giảng viên. |
|  | Số tín chỉ học của lớp học thứ . |
|  | Số tín chỉ học tối đa mà giảng viên thứ có thể giảng dạy trong kỳ học này. |
|  | Độ phù hợp giữa giảng viên và lớp học thứ , giá trị này càng cao thì độ phù hợp giữa giảng viên và lớp học càng cao. |
|  | Hệ số dùng để cân bằng số lượng tín chỉ được phân cho các giảng viên sao cho không có giảng viên nào bị phân quá nhiều lớp hay quá ít lớp. |

**Biến**:

* thể hiện giảng viên mà lớp học thứ được phân công vào.
* Miền giá trị là tập hợp chỉ số của các giảng viên có thể dạy được lớp thứ i.

**Ràng buộc**:

|  |  |  |
| --- | --- | --- |
|  |  | Ràng buộc 2.1 |
|  |  | Ràng buộc 2.2 |
|  |  | Ràng buộc 2.3 |

**Hàm mục tiêu**:

|  |  |  |
| --- | --- | --- |
|  |  | Ràng buộc 2.4 |
|  |  | Ràng buộc 2.5 |

Cách mô hình hóa bài toán như trên được giải thích như sau:

Ràng buộc 2.1 thể hiện cho ràng buộc giảng viên được phân cho lớp học thứ phải nằm trong danh sách các giảng viên có thể giảng dạy lớp học thứ .

Ràng buộc 2.2 thể hiện cho ràng buộc hai lớp bị trùng lịch học không được phân cho cùng một giảng viên.

Ràng buộc 2.3 thể hiện cho ràng buộc số tín chỉ được phân công cho một giảng viên không được vượt quá số tín chỉ tối đa mà giảng viên đấy có thể giảng dạy trong kỳ học hiện tại. sẽ có giá trị bằng 0 nếu giảng viên thứ không được phân cho lớp thứ và có giá trị bằng 1 trong trường hợp ngược lại.

Ràng buộc 2.4 là hàm mục tiêu tối đa hóa tổng độ phù hợp giữa giảng viên và lớp học được phân công. là một ánh xạ giữa thứ tự lớp học và thứ tự giảng viên với độ phù hợp (ưu tiên) của giảng viên với lớp học.

Ràng buộc 2.5 là hàm mục tiêu cân bằng lại số tín chỉ được phân công cho các giảng viên. Biểu thức là tổng số tín chỉ của các lớp đã được phân công cho giảng viên thứ . Hệ số là số tín chỉ mong muốn mà mọi giảng viên đều đạt được. Nếu tổng số tín chỉ được phân công cho giảng viên thứ lớn hơn thì mô hình cần giảm số lượng tín chỉ phân cho giảng viên và ngược lại thì mô hình cần tăng số lượng tín chỉ phân cho giảng viên. Vì vậy, mô hình cần tối thiểu hóa trị tuyệt đối của biểu thức . Tuy nhiên, để mô hình có thể ưu tiên tối ưu lại các môn được sắp xếp cho giảng viên có số tín cách xa điểm hơn số với các giảng viên đã có số tín gần với điểm , ta bổ sung thêm hệ số và thu được ràng buộc 2.5 như trên. Ví dụ về tác dụng của hệ số như sau: Giả sử 2 giảng viên cùng có thể dạy được lớp thứ i có số tín chỉ là 2, lớp này chưa được phân công giảng viên. Số tín chỉ đã được phân công của 2 giảng viên lần lượt là 8 và 0. Số tín chỉ mong muốn các giảng viên đạt được . Giá trị hạm mục tiêu lúc này bằng 12. Nếu không sử dụng hệ số , giá trị của hàm mục tiêu sẽ được giảm một lượng bằng nhau nếu nếu ta phân công lớp học i cho một trong số hai giảng viên. Tuy nhiên trong thực tế, ta cần phân công lớp học này cho giảng viên đang có số tín chỉ bằng 0. Khi thay đổi hàm mục tiêu để sử dụng hệ số như công thức trong ràng buộc 2.5, giá trị hàm sẽ giảm một lượng bằng 3 nếu phân công lớp cho giảng viên có 8 tín chỉ và giảm một lượng bằng 768 nếu phân công lớp cho giảng viên đang có 0 tín chỉ. Như vậy, để tối thiểu hóa hàm mục tiêu mô hình sẽ phân bổ lớp thứ cho giảng viên có số tín chỉ bằng 0.

## Sơ đồ chung thuật toán

Thuật toán đề xuất bao gồm 2 pha: khởi tạo lời giải ban đầu và tìm kiếm lời giải tối ưu. Dựa vào đặc điểm của dữ liệu và yêu cầu của bài toán, pha một sẽ khởi tạo một lời giải ban đầu giúp quá trình tìm kiếm đưa ra được lời giải tối ưu cục bộ tốt với thời gian nhanh hơn. Pha hai sẽ dựa vào lời giải từ pha một để tìm kiếm lời giải tối ưu thỏa mãn các ràng buộc và các hàm mục tiêu.

Thuật toán 2.1 Phân công giảng dạy

|  |  |  |
| --- | --- | --- |
| *1* | **function** teacherClassAssignment(Bài toán): Lời giải | |
| *2* |  | Khởi tạo lời giải ban đầu ; |
| *3* |  | Tìm kiếm lời giải tối ưu ( ); |
| *4* |  | **return** lời giải tối ưu |

## Pha khởi tạo lời giải ban đầu

Lời giải của bài toán được mô hình hóa bằng biến mảng một chiều . Với giá trị là số thứ tự của giảng viên được phân công dạy môn thứ . Để bắt đầu thuật toán, ta cần khởi tạo giá trị bằng một trong 3 chiến lược được đề xuất dưới đây.

### Chiến lược khởi tạo ngẫu nhiên

Đối với các hướng tiếp cận giải bài toán tối ưu tổ bằng tìm kiếm địa phương thì chiến lược chọn lời giải ban đâu một cách ngẫu nhiên rất hay được sử dụng. Tuy góp phần làm thay đổi kết quả giữa các lần chạy, nhưng nó tăng khả năng tìm được lời giải tối ưu toàn cục của bài toán trong nhiều lần chạy khác nhau.

Với bài toán phân công giảng dạy, biến sẽ được khởi tạo ngẫu nhiên giá trị trong miền giá trị .

Mã giả:

Thuật toán 2.2 Khởi tạo ngẫu nhiên

|  |  |  |  |
| --- | --- | --- | --- |
| *1* | **function** randomInitialize( *D*: mảng miền giá trị ): lời giải *x* | | |
| *2* |  | Tạo mảng x kích thước *N* ; | |
| *3* |  | **for** *i = 1: N* | |
| *4* |  |  | chọn ngẫu nhiên một phần tử trong |
| *5* |  | **endfor** | |
| *6* |  | **return** *x* ; | |

### Chiến lược khởi tạo tham lam tối thiểu hóa số vi phạm

Với chiến lược khởi tạo ngẫu nhiên, giá trị của các biến được khởi tạo mà không qua tâm đến các ràng buộc, nên thời gian tìm kiếm lời giải tối ưu là khá dài. Để giảm thời gian tìm kiếm lời giải tối ưu, thuật toán đề xuất chiến lược khởi tạo tham làm nhằm tối thiểu hóa số phi phạm ràng buộc ban đầu.

Mã giả:

Thuật toán 2.3 Khởi tạo tối thiểu vi phạm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | **function** minViolationInitialize(*D*: mảng miền giá trị) : lời giải *x* | | | | |
| *2* |  | **for** | | | |
| *3* |  |  | Khởi tạo tập rỗng ; | | |
| *4* |  |  | **for** các phần tử thuộc | | |
| *5* |  |  |  | **if** số vi phạm nhỏ hơn so với các phần tử trong | |
| *6* |  |  |  |  | Làm rỗng và thêm phần tử này vào ; |
| *7* |  |  |  | **else** | |
| *8* |  |  |  |  | Thêm phần tử này vào ; |
| *9* |  |  |  | **endif** | |
| *10* |  |  | **endfor** | | |
| *11* |  |  | chọn ngẫu nhiên phần tử thuộc ; | | |
| *12* |  | **endfor** | | | |
| *13* |  | **return** *x* ; | | | |

### Chiến lược khởi tạo tham lam tối đa hóa độ phù hợp

Ngoài chiến lược khởi tạo tham làm tối thiểu hóa số vi phạm ràng buộc, thuật toán cũng đề xuất chiến lược khởi tạo tham làm tối đa hóa độ phù hợp của giảng viên với lớp học. Do trong thực tế, lớp được phân cho một giảng viên thường cũng có độ phù hợp (hay ưu tiên) cao với giảng viên đó.

Mã giả:

Thuật toán 2.4 Khởi tạo tối đa hóa độ phù hợp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | **function** maxPriorityInitialize(*D*: mảng miền giá trị): lời giải *x* | | | | |
| *2* |  | **for** | | | |
| *3* |  |  | Khởi tạo tập rỗng ; | | |
| *4* |  |  | **for** các phần tử thuộc | | |
| *5* |  |  |  | **if** độ phù hợp cao hơn so với các phần tử trong | |
| *6* |  |  |  |  | Làm rỗng và thêm phần tử này vào ; |
| *7* |  |  |  | **else** | |
| *8* |  |  |  |  | Thêm phần tử này vào ; |
| *9* |  |  |  | **endif** | |
| *10* |  |  | **endfor** | | |
| *11* |  |  | chọn ngẫu nhiên phần tử thuộc ; | | |
| *12* |  | **endfor** | | | |
| *13* |  | **return** *x* ; | | | |

## Pha tìm kiếm lời giải tối ưu

Pha này nhận đầu vào là lời giải ban đầu đã được khởi tạo từ bước trên, từ đó tìm kiếm lời giải tối ưu thỏa mãn các ràng buộc và tối ưu các hàm mục tiêu của bài toán. Thuật toán đề xuất hai chiến lược tìm kiếm là chia thành 3 bước độc lập và tích hợp giữa tìm lời giải chấp nhận được và tối ưu độ phù hợp. Từng bước trong 2 chiến lược này được thực hiện với thuật toán tìm kiếm tabu với sơ đồ chung sau đây. Một số ký hiệu như sau:

* maxIter: số vòng lặp tìm kiếm tối đa
* iter: vòng lặp hiện tại
* : lời giải hiện tại
* : lời giải lân cận của
* : lời giải tốt nhất từng ghi nhận

Thuật toán 2.5 tìm kiếm với danh sách cấm tabu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | **function** tabuSearch(Lời giải *x*) : Lời giải | | | | |
| *2* |  | Khởi tạo danh sách tabu ; | | | |
| *3* |  | Tạo lời giải tốt nhất ; | | | |
| *4* |  | **while** (**not** thỏa mãn điều kiện dừng) | | | |
| *5* |  |  | Sinh lân cận ; | | |
| *6* |  |  | **if** tập lân cận rỗng **then** | | |
| *7* |  |  |  | Làm mới danh sách tabu ; | |
| *8* |  |  | **else** | | |
| *9* |  |  |  | Thực hiện bước di chuyển ; | |
| *10* |  |  |  | Cập nhật danh sách tabu và lời giải tốt nhất ; | |
| *11* |  |  |  | **if** thỏa mãn điều kiện làm mới tabu **then** | |
| *12* |  |  |  |  | Làm mới danh sách tabu ; |
| *13* |  |  |  | **endif** | |
| *14* |  |  | **endif** | | |
| *15* |  | **endwhile** | | | |
| *16* |  | **return** ; | | | |

Sau đây là phần trình bày chi tiết tường thành phần của sơ đồ thuật toán chung của từng bước tìm kiếm với 2 chiến lược.

### Chiến lược tìm kiếm chia thành từng bước độc lập

Ý tưởng chính của chiến lược này là chia nhỏ quá trình tìm kiếm lời giải tối ưu thành ba bước độc lập và liên tiếp: tìm lời giải chấp nhận được, tối ưu độ phù hợp và cân bằng số tín chỉ. Lời giải đầu ra của bước trước là đầu vào cho bước phía sau.

Chiến lược này tuân theo nguyên tắc là các bước phía sau sẽ không được làm giảm độ tốt của lời giải từ các bước trước tạo ra. Vì vậy, thứ tự thực hiện của các bước sẽ thể hiện được độ quan trọng của chúng trong bài toán. Trong bài toán tối ưu tổ hợp nói chung và bài toán phân công giảng dạy nói riêng thì phần tìm lời giải thỏa mãn các ràng buộc của bài toán luôn là ưu tiên hàng đầu. Sau đó, mới đến tối ưu lần lượt các hàm mục tiêu của bài toán.

Thuật toán 2.6 chiến lược tìm kiếm chia thành từng bước

|  |  |  |
| --- | --- | --- |
| *1* | **function** discreteStrategySearch*(Lời giải x) : Lời giải* | |
| *2* |  |  |
| *3* |  | Tìm lời giải chấp nhận được () ; |
| *4* |  | Tối ưu độ phù hợp () ; |
| *5* |  | Cân bằng số lượng tín chỉ (); |
| *6* |  | **return** ; |

Ba hàm tìm kiếm ở các dòng 3, 4, 5 của thuật toán đề tuân theo thuật toán tabuSearch ở trên. Sau đây sẽ trình bày về chi tiết sự khác nhau của từng bước trong các hàm tìm kiếm.

#### Bước tìm lời giải chấp nhận được

Bước này của thuật toán đi tìm một lời giải thỏa mãn tất cả các ràng buộc của toán phân công giảng day. Như đã trình bày ở trên, các hàm mục tiêu sẽ chưa được quan tâm tới. Sơ đồ thuật toán của của bước tìm lời giải chấp nhận được như được trình bày như hình sau:

1. Khởi tạo danh sách Tabu

Danh sách Tabu được sử dụng để giúp quá trình tìm kiếm của thuật toán không bị tối ưu cục bộ, bằng cách đánh dấu các bước di chuyển đã thực hiện. Thuật toán sẽ không lặp lại bước di chuyển hay chính là lặp lại việc gán cùng một giá trị cho một phần tử của mảng trong một số lượng bước lặp nhất định. Danh sách Tabu được cài đặt bằng mảng 2 chiều, với số hàng bằng với kích thước của mảng và số cột bằng với kích thước miền giá trị lớn nhất của các phần tử trong .

Mã giả:

Thuật toán 2.7 khởi tạo danh sách cấm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1* | **function** InitializeTabu( *D* ): danh sách cấm tabu | | | |
| *2* |  |  | | |
| *3* |  |  | | |
| *4* |  | **for** | | |
| *5* |  |  | Nếu giá trị nhỏ nhất của *D(x(i))* nhỏ hơn minV thì cập nhật minV ; | |
| *6* |  |  | Nếu giá trị lớn nhất của *D(x(i))* lớn hơn maxV thì cập nhật maxV ; | |
| *7* |  | **endfor** | | |
| *8* |  | ; | | |
| *9* |  | Khởi tạo tabu với kích thước *N, V* ; | | |
| *10* |  | **for** | | |
| *11* |  |  | **for** | |
| *12* |  |  |  | ; |
| *13* |  |  | **endfor** | |
| *14* |  | **endfor** | | |
| *15* |  | **return** tabu ; | | |

1. Thuật toán sinh lân cận

Thuật toán sinh lân cận dựa vào lời giải hiện tại để sinh ra các lời giải lân cận của lời giải đó. Thuật toán đề xuất định nghĩa lân cận của một lời giải

là

Với và là một giá trị thuộc miền giá trị .

Một lời giải lân cận được cho vào tập các lân cận của lời giải hiện cần thỏa mãn điều kiện bước di chuyển từ sang không bị cấm bởi danh sách Tabu hoặc là giá trị vi phạm của lân cận này phải nhỏ hơn giá trị vi tốt nhất được ghi nhận.

Mã giả:

Thuật toán 2.8 sinh lân cận trong tìm lời giải chấp nhận được

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *1* | **function** genNeighborFeasibleSolutionSearch(): tập bước di chuyển *M* | | | | | |
| *2* |  | Khởi tạo tập các bước di chuyển tạo lân cận ; | | | | |
| *3* |  | Khởi tạo *minDelta* ; | | | | |
| *4* |  | **for** | | | | |
| *5* |  |  | **for** | | | |
| *6* |  |  |  | Gán *delta* số vi phạm thay đổi khi gán *v* cho ; | | |
| *7* |  |  |  | **if** *tabu[i][v]* *iter* **or** *x* tốt hơn **then** | | |
| *8* |  |  |  |  | **if** *delta* < *minDelta* **then** | |
| *9* |  |  |  |  |  | Gán *minDelta* *delta* ; |
| *10* |  |  |  |  |  | Làm rỗng *M* và thêm cặp *(i, v)* vào *M* ; |
| *11* |  |  |  |  | **else** | |
| *12* |  |  |  |  |  | Thêm cặp *(i, v)* vào *M* ; |
| *13* |  |  |  |  | **endif** | |
| *14* |  |  |  | **endif** | | |
| *15* |  |  | **endfor** | | | |
| *16* |  | **endfor** | | | | |
| *17* |  | **return** *M* ; | | | | |

1. Thực hiện bước di chuyển

Sau khi có được một tập các lời giải lân cận, một lời giải sẽ được chọn ngẫu nhiên trong tập lân cận này để thay cho lời giải hiện tại của bài toán. Nếu lời giải mới tốt hơn lời giải tốt nhất hiện có, thì thuật toán sẽ cập nhật lời giải tốt nhất bằng lời giải mới. Để quá trình giải không bị rơi vào tối ưu cục bộ, thuật toán ghi nhận lại số lần di chuyển liên tiếp mà lời giải tốt nhất không bị thay đổi. Giá trị này được sử dụng để khởi động lại quá trình tìm kiếm khi vượt quá một ngưỡng đã đặt ra. Đồng thời, danh sách Tabu cũng được cập nhật để ghi nhận bước di chuyển vừa thực hiện.

Mã giả:

Thuật toán 2.9 thực hiện bước di chuyển trong tìm lời giải chấp nhận được

|  |  |  |  |
| --- | --- | --- | --- |
| *1* | **function** moveFeasibleSolution(): | | |
| *2* |  | Lấy ngẫu nhiên một cặp trong tập lân cận ; | |
| *3* |  | Gán *Tabu[i][v] iter* + độ dài của danh sách *Tabu* ; | |
| *4* |  | Gán ; | |
| *5* |  | **if** tốt hơn **then** | |
| *6* |  |  | Gán ; |
| *7* |  |  | Cập nhật số lần không đổi bằng 0 ; |
| *8* |  | **else** | |
| *9* |  |  | Tăng số lần không đổi thêm 1 ; |
| *10* |  | **endif** | |
| *11* |  | **return**  ; | |

1. Làm mới danh sách Tabu

Trong quá trình tìm kiếm, lời giải của bài toán dễ bị rơi vào tối ưu cục bộ. Để thoát ra, thuật toán sử dụng phương pháp làm mới lại danh sách Tabu và tìm một lời giải mới thay thế cho lời giải hiện tại. Tại bước này của thuật toán, danh sách tabu sẽ được gán lại giá trị như lúc mới khởi tạo.

Mã giả:

Thuật toán 2.10 làm mới tabu trong tìm lời giải chấp nhận được

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1* | **function** restartTabuFeasibleSolution( *tabu, , T: int*) : void | | | |
| *2* |  | Gán giá trị các phần tử của tabu bằng ; | | |
| *3* |  | **for** | | |
| *4* |  |  | Khởi tạo tập rỗng ; | |
| *5* |  |  | **for** : | |
| *6* |  |  |  | *delta* độ vị phạm thay đổi khi gán ; |
| *7* |  |  |  | Nếu *delta* ngưỡng *T* thì thêm vào tập ; |
| *8* |  |  | **end for** | |
| *9* |  |  | **if** **then** | |
| *10* |  |  |  | Gán một giá trị ngẫu nhiên trong ; |
| *11* |  |  | **endif** | |
| *12* |  | **endfor** | | |

Ngưỡng T ở đây có tác dụng giới hạn sự thay đổi số vi phạm ràng buộc của lời giải khi gán . Giá trị của T càng lớn thì, lời giải càng thay đổi mạnh so với . Trong bài toán phân công giảng dạy, thuật toán đề xuất hai giá trị cho T. Giá trị thứ nhất của T bằng 0 khi làm mới bình thường, giá trị thứ hai của T bằng 1 khi sau 4 lần làm mới mà lời giải tốt nhất không được thay đổi.

#### Bước tối ưu độ phù hợp

Trong thực tế, một giảng viên có thể giảng dạy nhiều học phần khác nhau, tương ứng với nhiều lớp học khác nhau trong một kỳ. Tuy nhiên, tùy theo hướng nghiên cứu, khả năng và nhu cầu của giảng viên mà đối với mỗi lớp học sẽ có độ phù hợp (ưu tiên) khác nhau. Thuật toán đề xuất bước tối đa hóa độ phù hợp giữa lớp học và giảng viên được phân công với nhau.

Đầu vào của bước tối đa hóa độ phù hợp chính là phương án chấp nhận được của bước đầu tiên. Phương pháp tìm kiếm TabuSearch vẫn được sử dụng trong bước này, nhưng được cài đặt với yêu cầu không làm tăng số vi phạm so với phương án chấp nhận được của bước đầu tiên.

1. Thuật toán sinh lân cận

Trong phần này, thuật toán đề xuất chọn ra các lân cận của lời giải hiện tại dựa theo các tiêu chí là:

* không làm tăng số vi phạm so với lời giải của bước tìm lời giải chấp nhận được.
* không nằm trong danh sách Tabu hoặc có giá trị độ phù hợp tốt hơn lời giải tốt nhất từng biết.

Thuật toán 2.11 sinh lân cận trong tối ưu độ phù hợp

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *1* | **function** genNeighboPrioritySearch( ): tập bước di chuyển *M* | | | | | |
| *2* |  | Khởi tạo tập lân cận rỗng ; | | | | |
| *3* |  | Khởi tạo *maxDelta* ; | | | | |
| *4* |  | **for** | | | | |
| *5* |  |  | **for** | | | |
| *6* |  |  |  | **if** số vi phạm tăng lên khi gán *v* cho **then** | | |
| *7* |  |  |  |  | **continue ;** | |
| *8* |  |  |  | **endif** | | |
| *9* |  |  |  | Gán *delta* giá trị độ phù hợp thay đổi khi gán *v* cho ; | | |
| *10* |  |  |  | **if** *tabu[i][v] iter* **or** x tốt hơn **then** | | |
| *11* |  |  |  |  | **if** *delta* > *maxDelta* **then** | |
| *12* |  |  |  |  |  | Gán *maxDelta* *delta* ; |
| *13* |  |  |  |  |  | Làm rỗng *M* và thêm cặp *(i, v)* vào *M* ; |
| *14* |  |  |  |  | **else** | |
| *15* |  |  |  |  |  | Thêm cặp *(i, v)* vào *M* ; |
| *16* |  |  |  |  | **endif** | |
| *17* |  |  |  | **endif** | | |
| *18* |  |  | **endfor** | | | |
| *19* |  | **endfor** | | | | |
| *20* |  | **return** *M* **;** | | | | |

1. Thực hiện bước di chuyển

Thuật toán 2.12 thực hiện bước di chuyển trong tối ưu độ phù hợp

|  |  |  |  |
| --- | --- | --- | --- |
| *1* | **function** movePrioritySearch(): | | |
| *2* |  | Lấy ngẫu nhiên một cặp trong tập lân cận ; | |
| *3* |  | Gán *Tabu[i][v] iter* + độ dài của danh sách *Tabu* ; | |
| *4* |  | Gán *v* ; | |
| *5* |  | **if** tốt hơn **then** | |
| *6* |  |  | Gán  ; |
| *7* |  |  | Cập nhật số lần không đổi bằng 0 ; |
| *8* |  | **else** | |
| *9* |  |  | Tăng số lần không đổi thêm 1 ; |
| *10* |  | **endif** | |
| *11* |  | **return** ; | |

1. Làm mới danh sách TABU

Trong bước này, thuật toán vẫn thực hiện 2 thao tác là làm rỗng danh sách Tabu và tìm lời giải mới giúp quá trình tìm kiếm thoát khỏi tối ưu cục bộ. Tuy nhiên, lời giải vẫn đảm bảo quy tắc không làm tăng số vi phạm.

Mã giả:

Thuật toán 2.13 làm mới tabu trong tối ưu độ phù hợp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | **function** restartTabuPrioritySearch( *tabu, , T: int*) : void | | | | |
| *2* |  | Gán giá trị các phần tử của *Tabu* bằng ; | | | |
| *3* |  | **for** | | | |
| *4* |  |  | Khởi tạo tập rỗng ; | | |
| *5* |  |  | **for** | | |
| *6* |  |  |  | *deltaV* độ vị phạm thay đổi khi gán ; | |
| *7* |  |  |  | *deltaP* độ phù hợp thay đổi khi gán ; | |
| *8* |  |  |  | **if** *deltaV* < 0 **and** *deltaP* ngưỡng *T* **then** | |
| *9* |  |  |  |  | Thêm vào tập ; |
| *10* |  |  |  | **endif** | |
| *11* |  |  | **endfor** | | |
| *12* |  |  | **if** **then** | | |
| *13* |  |  |  | Gán một giá trị ngẫu nhiên trong ; | |
| *14* |  |  | **endif** | | |
| *15* |  | **endfor** | | | |

#### Bước cân bằng số lượng tín chỉ

Sau khi có được lời giải của bước thứ hai, là một lời giải vừa thỏa mãn các ràng buộc vừa tối ưu về độ phù hợp giữa giảng viên và các lớp học được phân công. Thuật toán thực thiện bước cuối cùng là cân bằng lại số tín chỉ của các giảng viên với nhau, bằng cách tối thiểu hóa hàm mục tiêu trong ràng buộc 3.5 - đã được giải thích trong phần mô hình hóa.

1. Thuật toán sinh lân cận

Trong phần này, thuật toán đề xuất chọn ra các lân cận của lời giải hiện tại dựa theo các tiêu chí là:

* không làm tăng số vi phạm so với lời giải của bước tìm lời giải chấp nhận được.
* không làm giảm độ phù hợp của giảng viên với môn học so với lời giải tìm được ở bước tối ưu độ phù hợp.
* không nằm trong danh sách Tabu hoặc có giá trị độ phù hợp tốt hơn lời giải tốt nhất từng biết.

Mã giả:

Thuật toán 2.14 sinh lân cận trong cân bằng tín chỉ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *1* | **function** genNeighborRebalance ( ): tập bước di chuyển *M* | | | | | |
| *2* |  | Khởi tạo tập lân cận rỗng ; | | | | |
| *3* |  | Khởi tạo *minDelta* ; | | | | |
| *4* |  | **for** | | | | |
| *5* |  |  | **for** | | | |
| *6* |  |  |  | **if** số vi phạm tăng lên khi gán *v* cho **or** độ phù hợp giảm khi gán *v* cho **then** | | |
| *7* |  |  |  |  | **continue ;** | |
| *8* |  |  |  | **endif** | | |
| *9* |  |  |  | Gán *delta* giá trị hàm cân bằng thay đổi khi gán *v* cho ; | | |
| *10* |  |  |  | **if** *tabu[i][v] iter* **or** *x* tốt hơn **then** | | |
| *11* |  |  |  |  | **if** *delta* < *minDelta* **then** | |
| *12* |  |  |  |  |  | Gán *minDelta* *delta* ; |
| *13* |  |  |  |  |  | Làm rỗng *M* và thêm cặp *(i, v)* vào *M* ; |
| *14* |  |  |  |  | **else** | |
| *15* |  |  |  |  |  | Thêm cặp *(i, v)* vào *M* ; |
| *16* |  |  |  |  | **endif** | |
| *17* |  |  |  | **endif** | | |
| *18* |  |  | **endfor** | | | |
| *19* |  | **endfor** | | | | |
| *20* |  | **return** *M* ; | | | | |

1. Thực hiện bước di chuyển

Mã giả:

Thuật toán 2.15 thực hiện bước di chuyển trong cân bằng tín chỉ

|  |  |  |  |
| --- | --- | --- | --- |
| *1* | **function** moveRebalance(): | | |
| *2* |  | Lấy ngẫu nhiên một cặp trong tập lân cận ; | |
| *3* |  | Gán *Tabu[i][v] iter* + độ dài của danh sách *Tabu* ; | |
| *4* |  | Gán v ; | |
| *5* |  | **if** tốt hơn **then** | |
| *6* |  |  | Gán ; |
| *7* |  |  | Cập nhật số lần không đổi bằng 0 ; |
| *8* |  | **else** | |
| *9* |  |  | Tăng số lần không đổi thêm 1 ; |
| *10* |  | **endif** | |
| *11* |  | **return** ; | |

1. Làm mới danh sách TABU

Trong bước này, thuật toán vẫn thực hiện 2 thao tác là làm rỗng danh sách Tabu, tìm lời giải mới giúp quá trình tìm kiếm thoát khỏi tối ưu cục bộ và đảm bảo không làm tăng số vi phạm ràng buộc cũng như giảm độ phù hợp giữa giảng viện vào lớp học.

Mã giả:

Thuật toán 2.16 làm mới tabu trong cân bằng tín chỉ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | **function** restartTabuRebalance( *tabu, , T: int*) : void | | | | |
| *2* |  | Gán giá trị các phần tử của *Tabu* bằng ; | | | |
| *3* |  | **for** | | | |
| *4* |  |  | Khởi tạo tập rỗng ; | | |
| *5* |  |  | **for** | | |
| *6* |  |  |  | *deltaV* độ vị phạm thay đổi khi gán ; | |
| *7* |  |  |  | *deltaP* độ phù hợp thay đổi khi gán ; | |
| *8* |  |  |  | *deltaB* giá trị hàm cân bằng thay đổi khi gán ; | |
| *9* |  |  |  | **if** *deltaV* < 0 **and** *deltaP* > 0  **and** *deltaB* ngưỡng *T* **then** | |
| *10* |  |  |  |  | Thêm vào tập ; |
| *11* |  |  |  | **endif** | |
| *12* |  |  | **endfor** | | |
| *13* |  |  | **if** **then** | | |
| *14* |  |  |  | Gán một giá trị ngẫu nhiên trong ; | |
| *15* |  |  | **endif** | | |
| *16* |  | **endfor** | | | |

### Chiến lược tìm kiếm tích hợp nhiều bước

Chiến lược tìm kiếm chia thành các bước độc lập được trình bày ở trên tuy đã đưa ra được một lời giải tốt, thỏa mãn các ràng buộc vào tối ưu được các hàm mục tiêu. Tuy nhiên còn tồn tại hai vẫn đề là thời gian tìm kiếm dài do phải tìm kiếm ba lần và bước sau chỉ dựa vào một lời giải của bước trước.

Do vậy, thuật toán đề xuất thêm mới chiến lược tích hợp các bước lại với nhau trong một lần tìm kiếm. Cụ thể, trong bài toán này, chiến lược mới sẽ tích hợp hai bước là tìm lời giải chấp nhận được và tối ưu độ phù hợp giữa giảng viên và lớp học. Bước tối ưu độ cân bằng số tín chỉ giữa các giảng viên được giữ nguyên.

Mã giả:

Thuật toán 2.17 chiến lược tìm kiếm kết hợp

|  |  |  |
| --- | --- | --- |
| *1* | **function** combineStrategySearch*(Lời giải x) : Lời giải* | |
| *2* |  | Tìm phương án thỏa mãn ràng buộc kết hợp tối ưu độ phù hợp ; |
| *3* |  | Tối ưu độ cân bằng số tín chỉ ; |
| *4* |  | **return**  ; |

Sau đây sẽ trình bày các điểm khác biệt để cùng lúc tìm phương án thỏa mãn ràng buộc và tối ưu độ phù hợp trong một lần tìm kiếm.

1. Thuật toán sinh lân cận

Mã giả:

Thuật toán 2.18 sinh lân cận trong bước kết hợp

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *1* | **function** genNeighborFeasiblePrioritySearch ( ): tập bước di chuyển *M* | | | | | |
| *2* |  | Khởi tạo tập lân cận rỗng ; | | | | |
| *3* |  | Khởi tạo *minDelta* ; | | | | |
| *4* |  | Khởi tạo *maxDelta* ; | | | | |
| *5* |  | **for** | | | | |
| *6* |  |  | **for** | | | |
| *7* |  |  |  | *deltaV* độ vị phạm thay đổi khi gán ; | | |
| *8* |  |  |  | *deltaP* độ phù hợp thay đổi khi gán ; | | |
| *9* |  |  |  | **if** *tabu[i][v] iter* **or** *x* có số vi phạm nhỏ hơn  **or**  có độ phù hợp cao hơn **then** | | |
| *10* |  |  |  |  | **if** (*deltaP* == *maxDelta* **or** *deltaP* 0)  **and** *deltaV* == *minDelta* **then** | |
| *11* |  |  |  |  |  | Thêm cặp *(i, v)* vào *M* ; |
| *12* |  |  |  |  | **elseif** (*deltaP* > *maxDelta* **and** *deltaV* *minDelta*) **or** (*deltaP* *maxDelta* **and** *deltaV* < *minDelta*) **then** | |
| *13* |  |  |  |  |  | Gán *minDelta* *deltaV* ; |
| *14* |  |  |  |  |  | Gán *maxDelta* *deltaP* ; |
| *15* |  |  |  |  |  | Làm rỗng *M* và thêm cặp *(i, v)* vào *M* ; |
| *16* |  |  |  |  | **endif** | |
| *17* |  |  |  | **endif** | | |
| *18* |  |  | **endfor** | | | |
| *19* |  | **endfor** | | | | |
| *20* |  | **return** *M* ; | | | | |

1. Làm mới danh sách Tabu

Thuật toán 2.19 làm mới tabu trong bước kết hợp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | **function** restartTabuFeasiblePrioritySearch( *tabu, , T: int*) : void | | | | |
| *2* |  | Gán giá trị các phần tử của *Tabu* bằng | | | |
| *3* |  | **for** | | | |
| *4* |  |  | Khởi tạo tập rỗng ; | | |
| *5* |  |  | **for** | | |
| *6* |  |  |  | *deltaV* độ vị phạm thay đổi khi gán ; | |
| *7* |  |  |  | *deltaP* độ phù hợp thay đổi khi gán ; | |
| *8* |  |  |  | **if** *deltaP* > 0 **and** *deltaV* ngưỡng *T* **then** | |
| *9* |  |  |  |  | Thêm vào tập ; |
| *10* |  |  |  | **endif** | |
| *11* |  |  | **endfor** | | |
| *12* |  |  | **if** **then** | | |
| *13* |  |  |  | Gán một giá trị ngẫu nhiên trong ; | |
| *14* |  |  | **endif** | | |
| *15* |  | **endfor** | | | |

## Mô tả các lớp đã cài đặt thêm

Phần này trình bày mô tả các lớp đã cài đặt thêm các lớp đã được chỉnh sửa vào thư viện OpenCBLS để phục vụ giải bài toán phân công giảng dạy.

### Bổ sung thêm vào lớp VarIntLS

VarIntLS là lớp biểu diễn biến quyết định của bài toán. Ban đầu, lớp này được cài đặt với hai hàm tạo là:

* VarIntLS(LocalSearchManager mgr, int min, int max) có chức năng khởi tạo một biến quyết định nguyên có miền giá trị là min đến max và có giá trị ban đầu bằng min.
* VarIntLS(LocalSearchManager mgr, Set<Integer> domain) có chức năng khởi tạo một biến quyết định nguyên có miền giá trị thuộc tập domain và có giá trị ban đầu bằng giá trị nhỏ nhất trong domain.

Để phục vụ cho việc giải bài toán là có ba chiến lược khởi tạo biến quyết định đã được đề xuất ở phần trên thì lớp VarIntLS được bổ sung hàm tạo và các phương thức như sau:

* public VarIntLS(LocalSearchManager mgr,

IndexInput1D input,

int classIndex,

Tracker tracker,

ArrayList<Integer> domain,

ArrayList<Integer> priority,

String strategy)

Là hàm tạo có chức năng khởi tạo một biến quyết định nguyên có miền giá trị thuộc tập domain và có giá trị ban đầu được xác định bởi một trong ba chiến lược khởi tạo. Tham số input là đối tượng chứa thông tin đầu vào của bài toán. Tham số classIndex là chỉ số của lớp học mà biến quyết định đang được khởi tạo sẽ biểu diễn. Tracker là đối tượng chứa thông tin về giá trị của các biến quyết định đã được khởi tạo trước đó. Tham số priority biển diễn thông tin về độ phù hợp giữa giảng viên và lớp học. Chiến lược khởi tạo giá trị của biến quyết định được đánh dấu bằng tham số strategy.

* int randomTeacher(ArrayList<Integer> domain) trả về một giá trị ngẫu nhiên trong tập domain. Phương thức này dùng cho chiến lược khởi tạo ngẫu nhiên.
* int randomTeacherMaxPriority(ArrayList<Integer> domain,

ArrayList<Integer> priority)

Trả về một giá trị ngẫu nhiên trong số các phần tử thuộc tập domain có độ phù hợp cao nhất với biến quyết định dựa vào tập priority. Phương thức này dùng cho chiến lược khởi tạo tối đa độ phù hợp.

* int randomTeacherMinViolation(IndexInput1D input,

int classIndex,

ArrayList<Integer> domain,

Tracker tracker)

Trả về một giá trị ngẫu nhiên trong số các phần tử thuộc tập domain làm tăng số vi phạm ràng buộc ít nhất. Phương thức này dùng cho chiến lược khởi tạo tối thiểu số vi phạm.

### Tạo mới lớp SumPriority

SumPriority là lớp biểu diễn một hàm trong OpenCBLS. Hàm này có chức năng lưu giữ tổng độ phù hợp của các lớp học và các giảng viện được phân công dựa trên lời giải hiện tại.

Hàm tạo của lớp:

public SumPriority(VarIntLS[] X, HashMap<Integer, Integer>[] priority) nhận tham số X là mảng các biến quyết định biểu diễn các lớp cần phân công và priority là một HashMap giữa chỉ số của lớp và chỉ số của giảng viên với độ phù hợp giữa lớp và giảng viên.

### Tạo mới lớp Balance

Balance là một hàm trong kiến trúc của thư viện OpenCBLS. Hàm này có chức năng biểu diễn công thức sau

Ý nghĩa của hàm này là thể hiện mức độ tập trung của các phần tử trong mảng F đến giá trị , phục vụ cho hàm mục tiêu ở *Ràng buộc 2.5*.

Hàm tạo của lớp Balance:

public Balance(IFunction[] F, int target, float alpha) nhận tham số F là mảng các hàm thể hiện số lượng tín chỉ đã phân cho từng giảng viên, target là số tín chỉ mong muốn các giảng viên sẽ đạt được và alpha là một hệ số cân bằng nên nằm trong khoảng (1, 2).

### Tạo mới lớp TabuSearchM

Trong thư viện OpenCBLS đã có cài đặt sẵn một số phương thức sử dụng chiến tabusearch trong lớp TabuSearch. Tuy nhiên, để nâng phù hợp với bài toán và tăng hiệu năng, lớp này cần được chỉnh sửa lại và bổ sung thêm một số phương thức tìm kiếm mới. Vì vậy lớp TabuSearchM được tạo mới dựa trên lớp TabuSearch ban đầu với các phương thức mới. Các phương thức này được cài đặt theo thuật toán đã được đề xuất ở phần trên.

Phương thức searchInDomain(IConstraint S, ParameterModel paramModel, IFunction priorityF, IFunction balanceF) tìm kiếm lời giải chấp nhận được trong chiến lược 1.

Phương thức restartMaintainConstraintInDomain(VarIntLS[] x, IConstraint S, int[][] tabu, int threshold) thực hiện chức năng làm mới danh sách tabu cho pha tìm lời giải chấp nhận được.

Phương thức improveSumPriority(IConstraint S, IFunction priorityF, IFunction balanceF, int maxPriority, int tabulen) tìm kiếm lời giải nhằm nâng cao độ phù hợp giữa giảng viên vào lớp học dựa vào lời giải sau khi áp dụng phương thức searchInDomain.

Phương thức restartTabuPriority(VarIntLS[] x, IFunction priorityF, IConstraint S, int[][] tabu, int threshold) thực hiện chức năng làm mới danh sách tabu cho pha tối ưu độ phù hợp.

Phương thức rebalanceCredit(IFunction priorityF, IFunction balanceF, IConstraint S, int tabulen) tìm kiếm lời giải nhằm cân bằng số tín chỉ giữa các giảng viên dựa vào lời giải sau khi áp dụng phương thức improveSumPriority.

Phương thức restartTabuRebalance(VarIntLS[] x, IFunction balanceF, IFunction priorityF, IConstraint S, int[][] tabu, int threshold) thực hiện chức năng làm mới danh sách tabu cho pha tối ưu độ cân bằng.

Phương thức searchInDomainMaximize(IFunction priorityF, IFunction balanceF, IConstraint S, int tabulen, int maxTime, int maxIter, int maxStable, int maxPriority) thực hiện chức năng tìm kiếm tích hợp, vừa tìm lời giải chấp nhận được và tối ưu độ phù hợp giữa giảng viên và lớp học.

Phương thức restartMaintainConstraintInDomainMaximize(VarIntLS[] x, IFunction priorityF, IConstraint S, int[][] tabu, int threshold) thực hiện chức năng làm mới danh sách tabu cho phương thức searchInDomainMaximize.

# PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG

## Phân tích chức năng

### Biểu đồ usecase

A picture containing text, map

Description automatically generated

Hình 3.1 Sơ đồ Usecase của hệ thống

Hệ thống ứng dụng thuật toán giải bài toán phân công giảng dạy cần có các chức năng chính như trong hình. Đặc tả từng chức năng được trình bày ở phần sau.

### Đặc tả usecase

#### Quản lý danh sách bài toán

**Tên**: Quản lý danh sách bài toán

**Nội dung tóm tắt**: Người dùng quản lý danh sách các bài toán có trong hệ thống của mình.

**Kịch bản**:

1. Ca sử dụng bắt đầu khi khi người dùng muốn chỉnh sửa danh sách các bài toán phân công cần giải quyết.
2. Người dùng có thể bổ sung thêm bài toán hoặc xóa bài toán đã tồn tại.
3. Hệ thống thông báo lại chỉnh sửa đã được cập nhật.

#### Xử lý dữ liệu đầu vào

**Tên**: Xử lý dữ liệu đầu vào

**Nội dung tóm tắt**: Từ dữ liệu người dùng đưa vào, hệ thống sẽ xử lý để đưa ra dữ liệu đúng với định dạng đầu vào của thuật toán.

**Kịch bản**:

1. Ca sử dụng bắt đầu khi người dùng thêm mới một bài toán phân công giảng dạy.
2. Hệ thống sẽ xử lý dữ liệu đầu vào: tách các lớp không có giảng viên, tách các lớp chắc chắn sẽ bị chùng lịch học, tách các giảng viện không thể dạy một lớp nào cả.
3. Chuyển về định dạng dữ liệu của thuật toán.
4. Lưu đầu vào của bài toán là dữ liệu đã qua xử lý.

#### Tạo lời giải cho bài toán

**Tên**: Tạo lời giải cho bài toán

**Nội dung tóm tắt**: Tạo lời giải cho một bài toán phân công giảng dạy đã có trong danh sách.

**Kịch bản**:

1. Ca sử dụng bắt đầu khi người dùng muốn tạo một lời giải mới cho bài toán đã có trong danh sách.
2. Người dùng đưa vào các tham số của thuật toán.
3. Hệ thống giải bài toán và lưu lại kết quả.
4. Sau khi giải xong, hệ thống thông báo lại cho người dùng.

#### Thêm lời giải thủ công

**Tên**: Thêm lời giải thủ công

**Nội dung tóm tắt**: Thêm một lời giải được tạo ra bởi con người cho một bài toán đã có trong danh sách.

**Kịch bản**:

1. Ca sử dụng bắt đầu khi người dùng muốn đưa một lời giải được xếp bằng tay lên hệ thống.
2. Người dùng đưa lời giải xếp tay bằng một file excel lên hệ thống.
3. Hệ thống trích dữ liệu từ file excel và chuyển về định dạng lời giải của bài toán.
4. Hệ thống lưu lại lời giải tay và thông báo cho người dùng.

#### Phân tích lời giải của bài toán

**Tên**: Phân tích lời giải của bài toán

**Nội dung tóm tắt**: Phân tích lời giải của bài toán theo số vi phạm ràng buộc, giá trị của các hàm mục tiêu.

**Kịch bản**:

1. Ca sử dụng bắt đầu khi người dùng đã có lời giải của bài toán và muốn phân tích lời giải này.
2. Người dùng chọn lời giải cần phân tích.
3. Hệ thống trả về kết quả phân tích của lời giải

#### Xuất lời giải của bài toán

**Tên**: Xuất lời giải của bài toán

**Nội dung tóm tắt**: Xuất lời giải của bài toán cho người dùng sử dụng.

**Kịch bản**:

1. Ca sử dụng bắt đầu khi người dùng muốn lấy một lời giải của bài toán để sử dụng.
2. Người dùng chọn lời giải cần lấy.
3. Hệ thống trả về lời giải cho người dùng.

## Phân tích cấu trúc

Biểu đồ lớp phân tích của các ca sử dụng trong hệ thống được trình bày trong các phần sau.

1. Quản lý danh sách bài toán

A screenshot of a cell phone

Description automatically generated

Hình 3.2 Biểu đồ lớp phân tích Quản lý danh sách bài toán

1. Xử lý dữ liệu đầu vào

A screenshot of a cell phone

Description automatically generated

Hình 3.3 Biểu đồ lớp phân tích Xử lý dữ liệu đầu vào

1. Tạo lời giải cho bài toán

A screenshot of a cell phone

Description automatically generated

Hình 3.4 Biểu đồ lớp phân tích Tạo lời giải cho bài toán

1. Thêm lời giải thủ công

A screenshot of a cell phone

Description automatically generated

Hình 3.5 Biểu đồ lớp phân tích Thêm lời giải thủ công

1. Phân tích lời giải bài toán

A screenshot of a cell phone

Description automatically generated

Hình 3.6 Biểu đồ lớp phân tích Phân tích lời giải bài toán

1. Xuất lời giải của bài

A screenshot of a cell phone

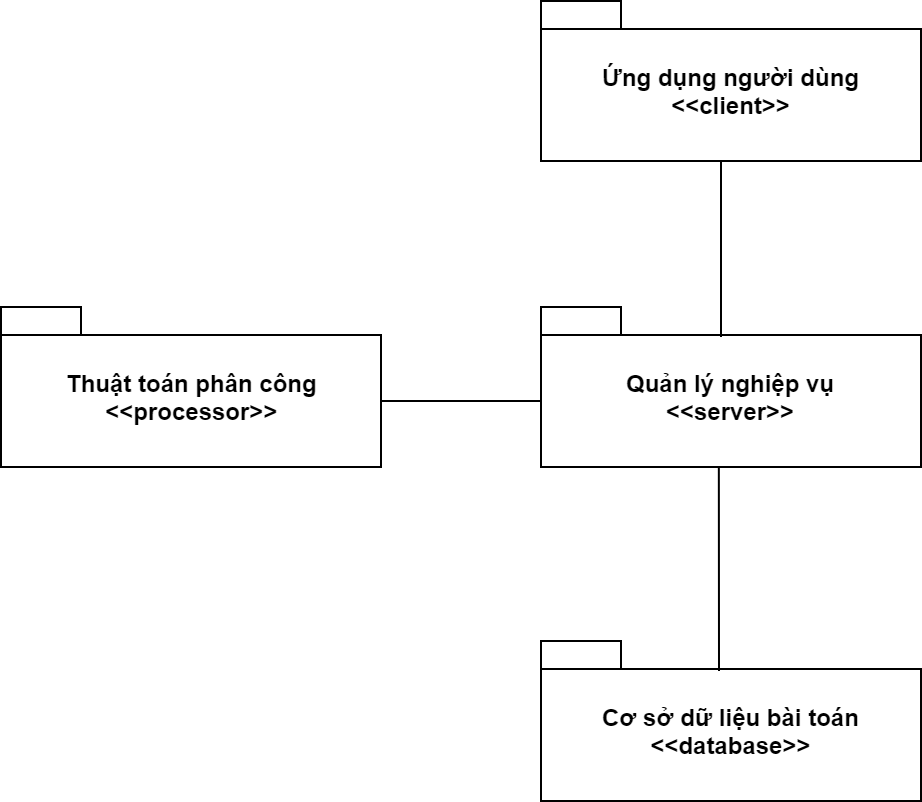
Description automatically generated

Hình 3.7 Biểu đồ lớp phân tích Xuất lời giải bài toán

## Thiết kết kiến trúc

1. Kiến trúc phân tầng

Hệ thống được chia nhỏ thành 4 thành phần con như sau:



Hình 3.8 Biểu đồ kiến trúc phân tầng của hệ thống

Trong đó, vai trò của từng thành phần như sau:

* Thuật toán phân công: xử lý dữ liệu đầu nhận và trả về lời giải của bài toán dưới dạng JSON.
* Quản lý nghiệp vụ: có vai trò điều phối giữa các thành phần của hệ thống, cung cấp các API ra bên ngoài, cung cấp dữ liệu cho thuật toán phân công và nhận lời giải.
* Cơ sở dữ liệu bài toán: cơ sở dữ liệu lưu trữ dữ liệu của hệ thống, bao gồm đầu vào bài toán, tham số, lời giải của bài toán.
* Ứng dụng người dùng: cung cấp giao diện sử dụng cho người quản lý, kết nối với thành phần quản lý nghiệp vụ qua các API được cung cấp.

1. Sơ đồ triển khai

Sau đây là sơ đồ triển khai thực tế của hệ thống, cung cấp các thông tin về ứng dụng và thành phần vật lý mà hệ thống sử dụng.

A picture containing screenshot

Description automatically generated

Hình 3.9 Sơ đồ triển khai của hệ thống

Người quản lý sẽ tương tác và sử dụng các chức năng của hệ thống thông qua trình duyệt web. Giao diện của trang web được quản lý trên một server chạy ReactJS. Server này có chức năng cung cấp giao diện sử dụng cho người dùng và kết nối đến một server khác quản lý nghiệp vụ của hệ thống thông qua các API được cung cấp.

Server quản lý nghiệp vụ chứa các thành phần nghiệp vụ và thuật toán xử lý. Thành phần Bài toán thuộc loại domain thể hiện thông tin về bài toán toán cần giải. Thuật toán thuộc loại thành phần processor, giải bài toán phân công. Thành phần lời giải bài toán thuộc loại domain thể hiện thông tin lời giải của bài toán phân công. Thành phần xử lý Phân tích thuộc loại processor có chức năng phân tích lời giải của bài toán. Thành phần quản lý bài toán kết nối các thành phân trên và cung cấp API ra bên ngoài cho client sử dụng. Thành phần này được xây dựng bằng framework Spring.

Hệ thống còn chứa một server cài đặt cơ sở dữ liệu của hệ thống. Cơ sở dữ liệu được triển khai bằng MySQL server.

## Thiết kế dữ liệu

Sơ đồ quan hệ giữa các bảng trong hệ thống được thể hiện trong hình dưới đây.

A picture containing text, receipt

Description automatically generated

Hình 3.10 Sơ đồ thiết kế cơ sở dữ liệu của hệ thống

1. Bảng user

Chứa thông tin về người dùng sử dụng hệ thống. Mô tả các trường nhứ sau:

Bảng 3.1 Mô tả bảng user

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint(20) | Khóa chính, mã định danh của người dùng |
| name | varchar(255) | Tên người dùng |
| status | int(11) | Trạng thái người dùng. 1- hoạt động và 0 - không hoạt động |
| create\_at | datetime | Thời gian tạo |
| update\_at | datetime | Thời gian cập nhật gần nhất |

1. Bảng input\_request

Chứa thông tin về bài toán mà người dùng gửi lên hệ thống. Mô tả chi tiết bảng như sau:

Bảng 3.2 Mô tả bảng input\_request

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint(20) | Khóa chính, mã định danh của bài toán |
| name | varchar(255) | Tên của bài toán mà người dùng đặt |
| json\_input | longtext | Đầu vào bài toán dưới dạng JSON |
| status | int(11) | Trạng thái bài toán. 1-bình thường và 0-đã xóa |
| user\_id | bigint(20) | Khóa ngoài, mã định danh của người dùng |
| create\_at | datetime | Thời gian tạo |
| update\_at | datetime | Thời gian cập nhật gần nhất |

1. Bảng invalid\_class

Chứa thông tin về tập các lớp được tách riêng sau quá trình tiền xử lý của một bài toán. Mô tả chi tiết bảng như sau:

Bảng 3.3 Mô tả bảng invalid\_class

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint(20) | Khóa chính, định danh tập các lớp |
| json\_class | longtext | Thông tin tập các lớp dưới dạng JSON |
| is\_conflict | int(11) | 0-lớp không bị trùng lịch học, 1-lớp trùng lịch học |
| is\_non\_teacher | int(11) | 0-lớp có giảng viên có thể dạy, 1-lớp không có giảng viên có thể thể dạy |
| status | int(11) | Trạng thái của tập các lớp, chưa sử dụng |
| input\_request\_id | bigint(20) | Khóa ngoài, định danh của bài toán |
| create\_at | datetime | Thời gian tạo |
| update\_at | datetime | Thời gian cập nhật gần nhất |

1. Bảng parameter

Chứa thông tin mô tả tham số của thuật toán giải bài toán phân công. Mỗi một bộ tham số tương ứng với một lời giải của bài toán trong bảng solution. Mô tả chi tiết bảng như sau:

Bảng 3.4 Mô tả bảng parameter

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint(20) | Khóa chính, định danh của bộ tham số |
| max\_iter | int(11) | Số vòng lặp tối đa của bước đầu tiên |
| max\_stable | int(11) | Số lần giá trị tốt nhất không đổi tối đa bước đầu tiên |
| max\_time | int(11) | Thời gian tối đa giải bài toán |
| tabulen | int(11) | Độ dài danh sách tabu bước đầu tiên |
| max\_iter\_priority | int(11) | Số vòng lặp tối đa bước tối ưu phù hợp |
| max\_stable\_priority | int(11) | Số lần giá trị tốt nhất không đổi tối đa bước tối ưu phù hợp |
| tabulen\_priority | int(11) | Độ dài danh sách tabu bước tối ưu phù hợp |
| max\_iter\_balance | int(11) | Số vòng lặp tối đa bước tối ưu cân bằng |
| max\_stable\_balance | int(11) | Số lần giá trị tốt nhất không đổi tối đa bước tối ưu cân bằng |
| tabulen\_balance | int(11) | Độ dài danh sách tabu bước tối ưu cân bằng |
| target | int(11) | Số tín chỉ mục tiêu |
| init\_strategy | varchar(255) | Chiến lược khởi tạo biến |
| is\_combine | int(11) | Chiếu lược tìm kiếm, 1-tích hợp, 0-chia nhỏ |
| alpha | float | Hệ số cân bằng |
| status | int(11) | Trạng thái của tập các lớp, chưa sử dụng |
| create\_at | datetime | Thời gian tạo |
| update\_at | datetime | Thời gian cập nhật gần nhất |
| input\_id | bigint(20) | Khóa ngoài, định danh của bài toán |

1. Bảng solution

Chứa thông tin về một lời giải của bài toán. Mô tả chi tiết bảng như sau:

Bảng 3.5 Mô tả bảng solution

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint(20) | Khóa chính, định danh của lời giải |
| json\_solution | longtext | Lời giải của bài toán dưới dạng JSON |
| solver\_status | int(11) | Trạng thái giải bài toán, 0-đang xử lý, 1-đã xử lý xong, 2-đã được chỉnh sửa, 3-lời giải thủ công, 4-không xác định |
| parameter\_id | bigint(20) | Khóa ngoài, định danh bộ tham số tạo lời giải |
| input\_request\_id | bigint(20) | Khóa ngoài, định danh bài toán |
| status | int(11) | Trạng thái của tập các lớp, chưa sử dụng |
| create\_at | datetime | Thời gian tạo |
| update\_at | datetime | Thời gian cập nhật gần nhất |

## Thiết kế giao diện

1. Trang đăng nhập

A screenshot of a cell phone

Description automatically generated

Hình 3.11 Giao diện đăng nhập

1. Trang bắt đầu sau khi đăng nhập

A screenshot of a cell phone

Description automatically generated

Hình 3.12 Giao diện trang bắt đầu

1. Trang danh sách lớp

A screenshot of a cell phone

Description automatically generated

Hình 3.13 Giao diện trang danh sách lớp

1. Trang danh sách học phần của giảng viên

A screenshot of a cell phone

Description automatically generated

Hình 3.14 Giao diện trang danh sách học phần cho giảng viên

1. Trang lời giải

A screenshot of a cell phone

Description automatically generated

Hình 3.15 Giao diện trang lời giải của bài toán

1. Trang chi tiết một lời giải

A screenshot of a cell phone

Description automatically generated

Hình 3.16 Giao diện trang chi tiết một lời giải

1. Hộp thoại thêm bài toán

A screenshot of a cell phone

Description automatically generated

Hình 3.17 Hộp thoại thêm bài toán

1. Hộp thoại tạo lời giải

A screenshot of a cell phone

Description automatically generated

Hình 3.18 Hộp thoại tạo lời giải mới

1. Hộp thoại thêm lời giải thủ công

A screenshot of a cell phone

Description automatically generated

Hình 3.19 Hộp thoại thêm lời giải thủ công

# KẾT QUẢ THỬ NGHIỆM

## Tạo bộ dữ liệu

Bộ dữ liệu thực nghiệm là một phần quan trọng trong việc xây dựng mô hình thuật toán giải quyết bài toán tối ưu tổ hợp. Tùy vào đặc thù của dữ liệu thực nghiệm mà các tham số trong thuật toán sẽ được tinh chỉnh để quá trình tìm kiếm thoát khỏi tối ưu cục bộ tồi.

Bộ dữ liệu lớp học được lấy từ danh sách các lớp học Trường Đại học Bách khoa Hà Nội. Bộ dữ liệu này gồm 3 phần là kỳ học 20182, 20191 và 20192. Nhưng do các Khoa/Viện độc lập với nhau trong việc phân công giảng viên và dữ liệu về các học phần mà một giảng viên có thể giảng dạy mới chỉ thu thập được của Viện CNTT & TT nên thuật toán đã thử nghiệm trên dữ liệu 3 kỳ học 20182, 20191 và 20192 của Viện CNTT & TT.

### Dữ liệu thô ban đầu

Dữ liệu thô ban đầu là danh sách các lớp học trong một học kỳ của toàn Trường ĐHBK Hà Nội.

Các trường dữ liệu được mô tả như sau:

Bảng 4.1 Mô tả các trường dữ liệu danh sách lớp ban đầu

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Semester | text | Học kỳ |
| SchoolName | text | Tên Khoa/Viện |
| ClassID | text | Mã lớp học |
| RelatedClassId | text | Mã lớp có liên quan |
| CourseID | text | Mã học phần |
| ClassName | text | Tên lớp học |
| ClassNote | text | Các ghi chú liên quan tới lớp học |
| Program | text | Hệ đào tạo |
| Enrollment | number | Số lượng sinh viên đăng ký |
| MaxEnrollment | number | Số lượng sinh viên đăng ký tối đa |
| ClassType | text | Loại lớp học (lý thuyêt, bài tập, thí nghiệm, …) |
| SemesterType | text | Loại học kỳ (A, B, AB) |
| status | text | Trạng thái lớp (Mở, Hủy) |
| needTN | boolean | Lớp có lớp thí nghiệm không |
| Session | text | Thời khóa biểu |
| Department | text | Bộ môn |
| Teacher | text | Giảng viên được phân công |
| Emails | text | Email của giảng viên được phân |

Ngoài ra, đầu vào dữ liệu còn yêu cầu danh sách các môn học mà một giảng viên có thể giảng dạy. Dữ liệu này được thu thập từ kết quả phân công lớp học cho giảng viên từ học kỳ 20161 đến học kỳ 20181.

Bảng 4.2 Mô tả dữ liệu phân công học phần và giảng viên

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| course\_id | text | Mã học phần |
| course\_name | text | Tên học phần |
| class\_type | text | Loại lớp (lý thuyêt, bài tập, thí nghiệm, …) |
| credit\_num | text | Số tín chỉ |
| program | text | Hệ đào tạo |
| num\_teacher | number | Số giảng viên có thể dạy |
| teacher | text | Tên giảng viên |
| email | text | Email giảng viên |
| semeter | text | Các kỳ học đã dạy |

### Tiền xử lý

1. Danh sách lớp học

Từ hai phần dữ liệu ban đầu, quá trình tiền xử lý sẽ đưa ra hai dự liệu là danh sách các lớp cần phân công giảng viên và danh sách các học phần và giảng viên có thể giảng dạy học phần đấy.

Danh sách các lớp học được lọc theo các điều kiện sau:

* Thuộc Viện CNTT & TT
* Không thuộc hệ đào tạo Thạc sĩ
* Không phải các lớp đồ án, thực tập
* Trạng thái lớp là mở, không bị hủy bỏ

Danh sách các lớp gồm 3 phần riêng tương ứng với 3 học kỳ được bổ sung thêm số tín chỉ của từng lớp từ dữ liệu phân công lớp học thô ban đầu và được nhập bằng tay. Sau đó, số tín chỉ của các lớp thí nghiệm trong các học phần có cả lớp lý thuyết và lớp thí nghiệm được điều chỉnh xuống 1 tín cho phù hợp với thực tế. Ví dụ, học phần Tin học đại cương có mã học phần là IT1110 có số tín chỉ là 4, gồm có 1 lớp lý thuyết và 1 lớp thí nghiệm. Với thời lượng là 1 giờ học trên 1 tuần thì lớp thí nghiệm có số tín bằng 1 là phù hợp.

Bảng 4.3 Mô tả danh sách lớp sau xử lý

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| class\_id | text | Mã lớp |
| course\_id | text | Mã học phần |
| class\_name | text | Tên lớp |
| class\_type | text | Loại lớp |
| session | text | Lịch học |
| credit\_num | number | Số tín chỉ |

Sau đó, những lớp không thể tìm được giảng viên có thể giảng dạy sẽ được tách riêng và đánh dấu là không thể phân công.

Ngoài ra, những lớp được phát hiện là chắc chắn sẽ bị trùng lịch học cũng sẽ được tách riêng. Chúng là những lớp chỉ có một giảng viên duy nhất có thể giảng dạy được và đồng thời các lớp đó của cùng một giảng viên lại bị trùng lịch học với nhau. Trong một cặp lớp bị trùng lịch, một lớp sẽ được tách riêng và đánh dấu là chắc chắn bị trùng lịch học và một lớp sẽ được giữ lại để tiếp tục đưa vào thuật toán phân công giảng viên.

1. Danh sách học phần và giảng viên

Danh sách các học phần và giảng viên có thể giảng dạy học phần đấy được xử lý như sau:

* Bổ sung thêm dữ liệu từ kết quả phân công của từng học kỳ đang xét. Với danh sách học phần và giảng viên của kỳ 20182 sẽ bổ sung từ kết quả phân công 20182. Với danh sách học phần và giảng viên của kỳ 20191 sẽ bổ sung từ kết quả phân công 20182 và 20191. Tương tự, kỳ 20192 sẽ bổ sung kết quả phân công 20182, 20191 và 20192.
* Để đảm bảo về tính chính xác của các giảng viên có thể tham gia giảng dạy trong một kỳ. Danh sách học phần và giảng viên chỉ lọc lấy các giảng viên có trong danh sách phân công trong dữ liệu phân công thủ công.
* Bổ sung thêm độ ưu tiên giữa giảng viên và môn học dựa theo đánh giá của TS. Phạm Quang Dũng.
* Bổ sung số tín chỉ tối đa của các giảng viên là 15, riêng với các giảng viên dạy lớp thí nghiệm tin đại cương là 30.

Bảng 4.4 Mô tả danh sách học phần cho giảng viên sau xử lý

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| course\_id | text | Mã học phần |
| course\_name | text | Tên học phần |
| class\_type | text | Loại lớp |
| teacher | text | Tên giảng viên |
| email | text | Email của giảng viên |
| max\_credit | number | Số tín chỉ tối đa mà giảng viên có thể dạy |
| priority | number | Độ phù hợp của học phần với giảng viên |

## Phân tích kết quả thử nghiệm

Thuật toán đề xuất bao gồm hai pha: khởi tạo lời giải ban đầu và tìm kiếm lời giải tối ưu. Với pha khởi tạo lời giải ban đầu có 3 chiến lược được đề xuất là:

* Chiến lược khởi tạo ngẫu nhiên
* Chiến lược khởi tạo tham lam tối thiểu hóa số vi phạm
* Chiến lược khởi tạo tham lam tối đa hóa độ phù hợp

Và pha tìm kiếm lời giải tối ưu có 2 chiến lược được đề xuất là:

* Chiến lược tìm kiếm chia thành tường bước nhỏ
* Chiến lược tìm kiếm tích hợp nhiều bước

Vì vậy, trong phần phân tích kết quả thực nghiệm này, kết quả của 6 thuật toán tạo bởi 3 chiến lược khởi tạo và 2 chiến lược tìm kiếm sẽ được phân tích và so sánh với nhau. Ngoài ra, kết quả của thuật toán cũng được so sánh với kết quả phân công thủ công trong dữ liệu được cung cấp. Các tiêu chí đánh giá kết quả bao gồm:

* Số lượng lớp học không được phân công.
* Số lượng lớp học bị phân công trùng lịch học với lớp khác.
* Số lượng lớp học bị phân công không đúng theo chuyên môn của giảng viên.
* Số lượng giảng viên bị phân quá số tín chỉ.
* Độ phù hợp giữa lớp học và giảng viên được phân công.
* Độ cân bằng về số tín chỉ được phân công giữa các giảng viên.

1. Phân tích kết quả phân công của học kỳ 20182

Với dữ liệu học kỳ 20182, kết quả phân công thủ công có 2 lớp chưa được phân công giảng viên. Thống kê của lời giải thử công như sau:

Bảng 4.5 Phân tích kết quả xếp thủ của công kỳ 20182

|  |  |  |
| --- | --- | --- |
| **Số cặp lớp trùng lịch học** | **Số lớp phân không đúng chuyên môn** | **Tổng mức độ phù hợp** |
| 5 | 0 | 909 |

Dựa vào phân tích kết quả phân công thủ công, ta thấy vẫn còn tồn tại các cặp lớp bị trùng lịch được phân cho cùng một giảng viên.

Với tập dữ liệu 20182, chương trình tách riêng được 5 lớp chưa được phân công giảng viên như sau:

* 2 lớp học không có giảng viên để phân công
* 3 lớp học chắc chắn bị trùng lịch học với lớp khác

Sau đây là bảng thống kê kết quả sau khi chạy các chiến lược giải khác nhau.

Bảng 4.6 Phân tích kết quả thuật toán của học kỳ 20182

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Chiến lược khởi tạo** | **Chiến lược tìm kiếm** | **Số lớp trùng lịch học** | **Số giáo viên bị phân quá tín chỉ** | **Số lớp phân không đúng chuyên môn** | **Tổng mức độ phù hợp** | **Thời gian**  **(s)** |
| Ngẫu nhiên | Chia nhỏ | 0 | 0 | 0 | 918 | 101 |
| Tích hợp | 0 | 0 | 0 | 918 | 157 |
| Tối đa độ phù hợp | Chia nhỏ | 0 | 0 | 0 | 922 | 140 |
| Tích hợp | 2 | 0 | 0 | 924 | 161 |
| Tối thiểu số vi phạm | Chia nhỏ | 0 | 0 | 0 | 921 | 115 |
| Tích hợp | 0 | 0 | 0 | 923 | 149 |

Bảng trên cho thấy, các chiến lược của thuật toán đều cho kết quả tốt đối với dữ liệu đã tách riêng các lớp không thể công được. Kết quả phân công của thuật toán cũng có tổng độ phù hợp giữa giảng viên và lớp học cao hơn so với kết quả thủ công. Kết quả của chiến lược khởi tạo tối đa độ phù hợp kết hợp với chiến lược tìm kiếm tích hợp 2 bước tìm lời giải chấp nhận được và tối ưu độ phù hợp chứa hai lớp bị trùng lịch cùng phân cho một giảng viên. Nguyên nhân là do chiến lược tìm kiếm này chưa ổn định vì vừa phải tối ưu độ phù hợp trong khi tìm lời giải chấp nhận được. Tuy nhiên, kết quả cho thấy chiến lược tích hợp cho được lời giải có độ phù hợp cao hơn so với chiến lược chi nhỏ từng bước.

Để so sánh về mức độ cân bằng số tín chỉ được phân công của các giảng viên với nhau giữa lời giải của thuật toán và lời giải thủ công, ta sử dụng biểu đồ phân bố sau. Biểu đồ có trục x là số tín chỉ được phân cho một giảng viên và trục y là số lượng giảng viên, thể hiện sự phân bố của số tín chỉ trên một giảng viên.

A screenshot of a cell phone

Description automatically generated

Hình 4.1 Biểu đồ phân bố số tín chỉ của một giảng viên – thuật toán - 20182

Theo biểu đồ trên ta thấy, kết quả phân công của thuật toán có số tín chỉ trên một giảng viên tập trung trong khoảng từ 6 đến 11 tín chỉ và tập trung cao nhất ở mức 8 tín chỉ. Ngoài ra, còn một số lượng giảng viên được phân công đến 26 tín chỉ là do các giảng viên này thuộc Trung tâm máy tính, giảng dạy rất nhiều lớp thí nghiệm Tin học đại cương. Biểu đồ đã cho thấy số lượng tín chỉ được phân công tương đối đồng đều giữa các giảng viên.

Dưới đây là biểu đồ của kết quả phân công thủ công.

A screenshot of a cell phone

Description automatically generated

Hình 4.2 Biểu đồ phân bố số tín chỉ của một giảng viên – thủ công - 20182

Biểu đồ cho thấy số tín chỉ được phân cho các giảng viên bị phân bố dàn trải từ 2 đến 15. Đặc biệt một số thầy cô còn được phân công đến 34, 36 tín chỉ. Kết quả phân công thủ công chưa thực sự cân bằng giữa các giảng viên.

1. Phân tích kết quả phân công của học kỳ 20191

Với dữ liệu học kỳ 20191, kết quả phân công thủ công có 2 lớp chưa được phân công giảng viên. Thống kê của lời giải thử công như sau:

Bảng 4.7 Phân tích kết quả xếp thủ của công kỳ 20191

|  |  |  |
| --- | --- | --- |
| **Số cặp lớp trùng lịch học** | **Số lớp phân không đúng chuyên môn** | **Giá trị hàm mục tiêu độ phù hợp** |
| 6 | 0 | 885 |

Bảng trên cho thấy kết quả thủ công có nhiều lớp bị phân trùng lịch học với nhau.

Với tập dữ liệu 20191, chương trình tách riêng được 3 lớp chưa được phân công giảng viên do các lớp học chắc chắn bị trùng lịch học với lớp khác.

Sau đây là bảng thống kê kết quả sau khi chạy các chiến lược giải khác nhau.

Bảng 4.8 Phân tích kết quả thuật toán của học kỳ 20191

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Chiến lược khởi tạo** | **Chiến lược tìm kiếm** | **Số lớp trùng lịch học** | **Số giáo viên bị phân quá tín chỉ** | **Số lớp phân không đúng chuyên môn** | **Tổng mức độ phù hợp** | **Thời gian**  **(s)** |
| Ngẫu nhiên | Chia nhỏ | 2 | 0 | 0 | 903 | 142 |
| Tích hợp | 4 | 0 | 0 | 905 | 120 |
| Tối đa độ phù hợp | Chia nhỏ | 2 | 0 | 0 | 904 | 136 |
| Tích hợp | 4 | 0 | 0 | 905 | 130 |
| Tối thiểu số vi phạm | Chia nhỏ | 2 | 0 | 0 | 904 | 154 |
| Tích hợp | 4 | 0 | 0 | 905 | 128 |

Kết quả của tất cả các chiến lược đều vẫn còn tồn tại các lớp trùng lịch học. Tuy nhiên số lượng lớp này đã giảm nhiều so với kết quả thủ công. Ngoài ra độ phù hợp của giảng viên và lớp học cũng cao hơn so với kết quả thủ công.

Tiếp tục so sánh mức độ cân bằng số tín chỉ giữa các giảng viên qua các biểu đồ sau.

A screenshot of a cell phone

Description automatically generated

Hình 4.3 Biểu đồ phân bố số tín chỉ của một giảng viên – thuật toán - 20191

Biểu đồ trên cho thấy, với kết quả của thuật toán, số tín chỉ của một giảng viên tập trung trong khoảng từ 8 đến 13 tín và cao nhất là 15 tín chỉ. Độ cân bằng số lượng tín chỉ được phân cho các giảng viên là tương đối tốt.

A close up of a logo

Description automatically generated

Hình 4.4 Biểu đồ phân bố số tín chỉ của một giảng viên – thủ công - 20191

Biểu đồ của lời giải thủ công cho thấy sự cân bằng về số tín chỉ giữa các giảng viên không được tốt so với lời giải của thuật toán. Số tín chỉ của một giảng viên phân bố tương đối đều từ 6 đến 14 và có giá trị lớn nhất lên đến 20.

1. Phân tích kết quả phân công của học kỳ 20192

Với dữ liệu học kỳ 20192, kết quả phân công thủ công có 1 lớp chưa được phân công giảng viên. Thống kê của lời giải thử công như sau:

Bảng 4.9 Phân tích kết quả xếp thủ của công kỳ 20192

|  |  |  |
| --- | --- | --- |
| **Số cặp lớp trùng lịch học** | **Số lớp phân không đúng chuyên môn** | **Giá trị hàm mục tiêu độ phù hợp** |
| 7 | 0 | 909 |

Bảng trên cho thấy có đến 7 cặp lớp bị trùng lịch học trong lời giải thủ công, chứng tỏ lời giải vẫn chưa tốt.

Với tập dữ liệu 20192, không có lớp học nào phải tách riêng sau quá trình tiền xử lý. Sau đây là bảng thống kê kết quả sau khi chạy các chiến lược giải khác nhau.

Bảng 4.10 Phân tích kết quả thuật toán của học kỳ 20192

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Chiến lược khởi tạo** | **Chiến lược tìm kiếm** | **Số lớp trùng lịch học** | **Số giáo viên bị phân quá tín chỉ** | **Số lớp phân không đúng chuyên môn** | **Tổng mức độ phù hợp** | **Thời gian**  **(s)** |
| Ngẫu nhiên | Chia nhỏ | 0 | 0 | 0 | 513 | 26 |
| Tích hợp | 0 | 0 | 0 | 513 | 47 |
| Tối đa độ phù hợp | Chia nhỏ | 0 | 0 | 0 | 512 | 31 |
| Tích hợp | 0 | 4 | 0 | 518 | 49 |
| Tối thiểu số vi phạm | Chia nhỏ | 0 | 0 | 0 | 507 | 29 |
| Tích hợp | 0 | 4 | 0 | 518 | 48 |

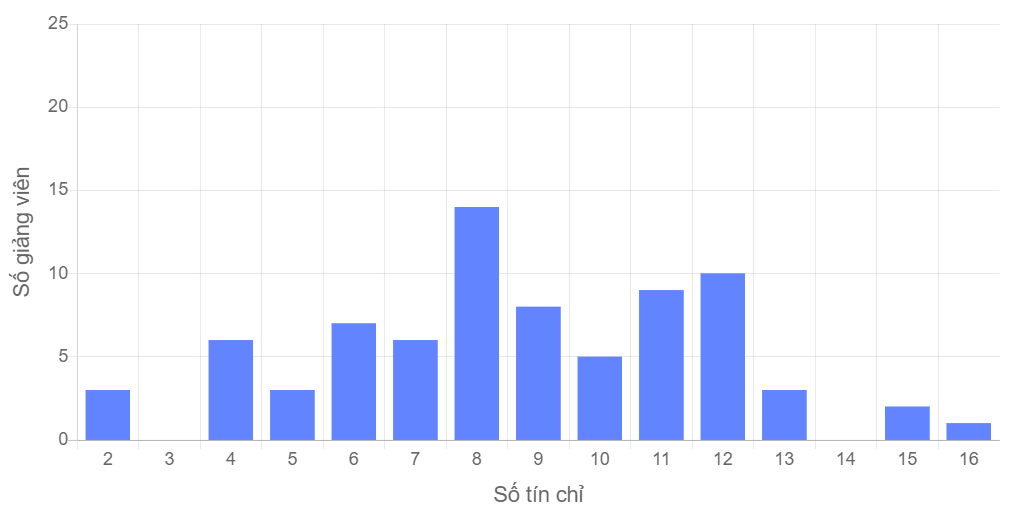
Bảng trên cho thấy các chiến lược của thuật toán đều đã phân công được các lớp học sao cho không có giảng viên nào bị phân các lớp trùng lịch học với nhau. Chiến lược tích hợp vẫn cho những lời giải có độ phù hợp cao nhưng còn vi phạm ràng buộc bài toán, ở đây là vi phạm về số lượng tín được phân công vượt mức tối đa.

A picture containing text

Description automatically generated

Hình 4.5 Biểu đồ phân bố số tín chỉ của một giảng viên – thuật toán - 20192

Với dữ liệu 20192, thuật toán đưa ra lời giải có mức độ cân bằng số tín chỉ giữa các giảng viên tương đối cao. Dựa vào biểu đồ ta có thể thấy, số tín chỉ được phân cho một giảng viên tập trung trong khoảng từ 8 đến 10 và cao nhất là 15.



Hình 4.6 Biểu đồ phân bố số tín chỉ của một giảng viên – thủ công - 20192

Trong khi đó, kết quả phân công thủ công có độ cân bằng tương đối thấp giữa các giảng viên. Với số tín được phân công cho một giảng viên nằm trong khoảng từ 4 đến 12 tín và cao nhất là 16.

# KẾT LUẬN

## Kết luận

Nội dung của đồ án đã đáp ứng được các yêu cầu đặt ra của đề bài. Với phần thuật toán phân công được đề xuất cho ra kết quả tương đối tốt so với kết quả phân công thủ công hiện tại trên ba bộ dữ liệu thực tế của ba kỳ học liên tiếp. Và phần hệ thống ứng dụng thuật toán phân công được xây dựng khá hoàn thiện. Tuy còn nhiều thiếu sót, những hệ thống cũng có thể đáp ứng một phần công việc thực tế trong quản lý phân công lớp học. Kết quả thu được của đồ án cho thấy các thuật toán gần đúng giải bài toán tối ưu tổ hợp đặc biệt là phương pháp tìm kiếm cục bộ dựa trên ràng buộc rất có hiệu quả trong thực tế. Với kết quả tốt sau khi thực nghiệm với các bộ dữ liệu thực tế, tính khả thi của đồ án là tương đối cao, có thể áp dụng vào thực tế để hỗ trợ việc phân công lớp học của Viện.

## Hướng phát triển của đồ án trong tương lai

Qua đề tài này, hy vọng kết quả của đồ án có thể ứng dụng vào việc hỗ trợ phân công giảng dạy của viện và xa hơn là phát triển thêm để ứng dụng cho cả Trường. Trong tương lai, em sẽ thử nghiệm thêm các bộ dữ liệu khác của các viện trong trường, để tìm cách tối ưu thêm về phương pháp và phân tích thêm ưu nhược điểm của phương pháp này. Ngoài ra, hệ thống phần mềm ứng dụng sẽ được bổ sung thêm tính năng chỉ sửa lời giải của thuật toán. Tính năng này cho phép người dùng điều chỉnh lại lời giải để phù hợp với yêu cầu thực tế.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | B. Korte và J. Vygen, Combinatorial Optimization theory and algorithms third edition, Springer, 2000. |
| [2] | T. N. Hà, Các bài toán tối ưu tổ hợp và tính toán mềm, luận án tiến sĩ CNTT 62.48.01.01, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, 2017. |
| [3] | D. L. Applegate, R. E. Bixby, V. Chvátal và W. J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2006. |
| [4] | E. G. Coffman, J. Csirik, D. S. Johnson và G. J. Woeginger, An Introduction to Bin packing, 2004. |
| [5] | L. Michel và P. V. Hentenryck, Constraint-Based Local Search, The MIT Press, 2005. |
| [6] | S. Edelkamp và S. Schrödl, Heuristic Search, Morgan Kaufmann, 2012. |
| [7] | F. &. L. M. &. M. R. Glover, Tabu Search, Springer, 1997. |
| [8] | P. Q. Dung, H. T. Trung, T. D. Hoang và N. T. Hoang, A Java library for Constraint-Based Local Search: Application to the master thesis defense timetabling problem, Proceedings of the Sixth International Symposium on Information and Communication Technology, 2015, Pages 67-74, 2015. |