

LỜI CAM ĐOAN

Chúng em xin cam đoan đây là luận văn nghiên cứu do chính nhóm thực hiện và chưa
được công bố trong các công trình khác. Ngoại trừ các tài liệu tham khảo, hình ảnh, thông
tin có trong danh mục tham khảo có trích dẫn bản quyền tác giả cụ thể.

LỜI CẢM ƠN

Qua hơn 4 năm học tập và rèn luyện dưới giảng đường Đại học, đặc biệt là thời gian thực hiện thực tập và luận văn tại trường, nhóm thực hiện đã học hỏi và có những tích lũy quý báu về kiến thức cũng như khả năng phân chia, thực hiện và hoàn thành công việc. Luận văn tốt nghiệp đã giúp cho nhóm có được cơ hội để vận dụng những kiến thức học hỏi được và áp dụng nó vào trong thực tiễn sinh động, cụ thể, qua đó làm nền tảng cho quá trình phát triển sau này.

Đề tài luận văn này được hoàn thành thuận lợi ngoài sự cố gắng của chính bản thân thành viên trong nhóm, còn nhờ sự giảng dạy tận tình của quý thầy cô trường Đại học Bách khoa thành phố Hồ Chí Minh. Vì vậy nhóm thực hiện xin chân thành gửi lời cảm ơn đến các thầy cô, đặc biệt là lời cảm ơn sâu sắc đến thầy Dương Tuấn Anh, cán bộ trực tiếp hướng dẫn và góp ý cho nhóm về nội dung cũng như phương thức thực hiện đề tài. Bên cạnh đó nhóm thực hiện cũng xin gửi lời cảm ơn đến toàn thể quý thầy cô giáo khoa Khoa học và Kỹ thuật Máy tính trường Đại học Bách khoa thành phố Hồ Chí Minh đã nhiệt tình giảng dạy cho nhóm trong suốt thời gian học tập tại trường.

Mặc dù có nhiều cố gắng trong việc làm luận văn tốt nghiệp, nhưng do thời gian hạn hẹp nên đề tài chắc chắn không thể tránh khỏi những thiếu sót trong quá trình thực hiện. Kính mong nhận được sự đóng góp phê bình của quý thầy cô, bạn bè để đề tài được hoàn chỉnh.

Sau cùng nhóm thực hiện xin kính chúc quý thầy cô dồi dào sức khỏe và thành công trong công việc.

TÓM TẮT

Nội dung chính của đề tài này là nghiên cứu cách tiếp cận các giải thuật giải hệ ràng buộc (CSP) , so sánh giải thuật Tabu Search và WSAT (Scandinavian Workshop on Algorithm Theory) trong việc sắp xếp thời khóa biểu trường trung học phổ thông với một lời giải ban đầu.

Luận văn được chia thành 4 phần cơ bản:

Phần I: Tổng quan về bài toán lập lịch, sắp xếp thời khóa biểu trong trường phổ thông với những ràng buộc cụ thể.

Phần II: Trình bày lý thuyết của bài toán giải hệ ràng buộc với những thuật toán được áp dụng cụ thể vào việc giải quyết bài toán trong thực tế.

Phần III: Trình bày về thuật toán Backtracking Look Ahead để đưa ra lời giải ban đầu cho bài toán.

Phần IV: Trình bày về hai giải thuật Tabu Search và WSAT Search để tối ưu lời bài toán. So sánh đánh giá tối ưu của từng giải thuật.

DANH MỤC CÁC TỪ VIẾT TẮT TRONG LUẬN VĂN

1. WSAT Scandinavian Workshop on Algorithm Theory
2. BC_FC Backtracking with Look Ahead (Backtracking Forward)
3. CSP Constraint Satisfaction Problem
4. SA Simulated Annealing

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU CHUNG VÀ MỤC TIÊU LUẬN VĂN TỐT NGHIỆP	9
CHƯƠNG 2: BÀI TOÁN XẾP THỜI KHÓA BIỂU	11
2.1. KHÁI NIỆM VỀ BÀI TOÁN LẬP LỊCH	11
2.2. BÀI TOÁN XẾP THỜI KHÓA BIỂU	11
2.2.1. Mô tả	11
2.2.2. Các ràng buộc trong việc lập thời khóa biểu	11
2.2.3. Tính tương đối của lời giải	12
2.2.4. Sắp thời khóa biểu bằng tay	13
2.3. TÓM TẮT	13
CHƯƠNG 3: LÝ THUYẾT VỀ HỆ GIẢI RÀNG BUỘC	14
3.1. TỔNG QUAN VỀ BÀI TOÁN GIẢI HỆ RÀNG BUỘC	14
3.2. BÀI TOÁN CSP	14
3.2.1. Định nghĩa	14
3.2.2. Độ phức tạp của bài toán CSP	16
3.2.3. Các bài toán CSP thông dụng	16
3.2.4. Ứng dụng trong thực tế	16
3.3. CÁC GIẢI THUẬT TRÊN BÀO TOÁN CSP	17
3.3.1. Tìm kiếm có hệ thống (Systematic search)	17
3.3.2. Tìm kiếm không hệ thống (non-Systematic search)	17
3.4. GIỚI THIỆU GIẢI THUẬT BÀI TOÁN CSP	18
3.4.1. Giải thuật vét cạn (Generate and test)	18
3.4.2. Giải thuật backtracking	18
3.4.3. Giải thuật tương thích (Consistency algorithms)	19
3.4.4. Giải thuật Look ahead (Backtracking + Forward Checking)	22
3.4.5. Các kỹ thuật được sử dụng để tăng tốc độ tìm kiếm	24
3.5. CÁC GIẢI THUẬT ĐỂ GIẢI BÀI TOÁN TỐI ƯU TỐ HỢP	26
3.5.1 Tìm kiếm cục bộ (Local Search)	26
3.5.2. Hàm chi phí (Cost Function)	27

3.5.3. Giải thuật tìm kiếm cục bộ tổng quát (General Local Search Algorithm)	27
3.5.4. Giải thuật leo đồi (Hill-Climbing)	27
3.5.4. Tabu Search.....	29
3.5.5. Giải thuật mô phỏng luyện kim (Simulated annealing)	32
3.5.6. Giải thuật Scandinavian Workshop on Algorithm Theory (WSAT)	37
CHƯƠNG 4: THIẾT KẾ VÀ HIỆN THỰC CHƯƠNG TRÌNH.....	39
4.1. THIẾT KẾ CƠ SỞ DỮ LIỆU.....	39
4.1.1. Sơ đồ quan hệ thực thể (ERD)	39
4.1.2. Sơ đồ cơ sở dữ liệu quan hệ trên MS SQL Server 2005	40
4.2. LUU ĐÒ GIẢI THUẬT TRONG CHƯƠNG TRÌNH.....	41
4.2.1. Lưu đồ giải thuật BC_FC.....	41
4.2.2. Lưu đồ giải thuật Tabu	42
4.2.3. Lưu đồ giải thuật WSAT Search	43
4.3. SƠ ĐỒ LỚP CHƯƠNG TRÌNH DEMO (CLASS DIAGRAM)	44
4.4. CÁCH SẮP XẾP ĐIỂM VÀ BIẾN TÓI ƯU GIẢI THUẬT BC_FC	44
4.5. CÁCH TÍNH ĐIỂM HÀM ĐÁNH GIÁ TRONG LOCAL SEARCH	45
4.6. CHẠY DEMO, ĐÁNH GIÁ ĐỘ HIỆU QUẢ CỦA HAI GIẢI THUẬT TABU SEARCH VÀ WSAT SEARCH.....	49
4.6.1. Bảng phân công.....	49
4.6.2. Chạy demo chương trình	50
4.7. SO SÁNH ĐÁNH GIÁ HIỆU QUẢ CỦA HAI GIẢI THUẬT	53
4.6.1 So sánh hiệu quả hai giải thuật dựa trên thực tế kết quả sắp xếp trong chương trình.....	53
4.7.1. Biểu đồ đánh giá hiệu quả giải thuật Tabu Search.....	55
4.7.2. Biểu đồ đánh giá hiệu quả giải thuật WSAT Search.....	56
4.7.3. Biểu đồ đánh giá chung hai giải thuật	57
4.7.4. So sánh đánh giá độ hiệu quả hai giải thuật.....	57
CHƯƠNG 5: KẾT LUẬN.....	59
5.1. ĐÁNH GIÁ.....	59

5.2. NỘI DUNG ĐÃ ĐƯỢC GIẢI QUYẾT	59
THIẾT KẾ CÁU TRÚC DỮ LIỆU:	61
Các cấu trúc dữ liệu dành cho việc hiện thực giải thuật BC_FC:	61
Hiện thực các phương thức chính trong các Class:	63
Cấu trúc giải thuật Local Search	67
Các phương thức chính trong giải thuật Local Search	69
PHỤC LỤC	61
TÀI LIỆU THAM KHẢO	76

DANH MỤC HÌNH

Hình 3-1: Minh họa bài toán tô màu	15
Hình 3-2: Đồ thị ràng buộc	20
Hình 4-1: Lược đồ quan hệ	39
Hình 4-2: Các bản quan hệ trong MS SQL Server 2005	40
Hình 4-3: Lưu đồ giải thuật BC_FC	41
Hình 4-4: Lưu đồ giải thuật Tabu Search	42
Hình 4-5: Lưu đồ giải thuật WSAT	43
Hình 4-6: Bảng phân công giảng dạy giáo viên	49
Hình 4-7: Giao diện chính của chương trình demo	50
Hình 4-8: Giao diện Phần trước khi sắp thời khóa biểu	51
Hình 4-9: Giao diện chương trình sau khi sắp xếp thời khóa biểu	52
Hình 4-10: Thời khóa biểu sắp xếp theo lớp của hai giải thuật	53
Hình 4-11: Thời khóa biểu sắp xếp theo giáo viên của hai giải thuật	54
Hình 4-12: Biểu đồ đánh giá giải thuật Tabu Search	55
Hình 4-13: Biểu đồ đánh giá giải thuật WSAT Search	56
Hình 4-14: Biểu đồ đánh giá chung của hai giải thuật	57

CHƯƠNG 1: GIỚI THIỆU CHUNG VÀ MỤC TIÊU LUẬN VĂN TỐT NGHIỆP

Công việc xếp Thời Khóa Biểu đã và luôn là một bài toán khó nhưng rất quan trọng trong mọi nhà trường bởi Thời khóa biểu là bộ xương sống thời gian của toàn bộ công việc HỌC của học sinh và DẠY của giáo viên. Bài toán xếp thời khóa biểu là không phải là bài toán mới, nó là bài toán cũ nhiều người đã dày công và tìm kiếm những lời giải hay nhất, tốt nhất để giải quyết cho mọi trường hợp có thể có. Nó cũng như các bài toán lập lịch nói chung là những bài toán phức tạp (NP-Complete) . Chính vì thế ngay từ khi những máy tính đầu tiên mới ra đời người ta đã nghĩ đến việc áp dụng máy tính để giải quyết bài toán này.

Trong các trường trung học ở Việt Nam hiện nay thì việc xếp thời khóa biểu đa số được thực hiện bằng tay. Đây là một công việc khó khăn và phức tạp khi trường có số lượng giáo viên và lớp nhiều. Vì vậy việc xếp thời khóa biểu bằng chương trình trên máy tính sẽ rất thuận lợi cho các trường học, giảm căng thẳng vào mỗi đầu năm học cho những người xếp lịch. Đồng thời sẽ tạo ra thời khóa biểu có lợi cho học sinh và cả giáo viên.

Có rất nhiều giải thuật có thể dùng để xếp lịch hay thời khoá biểu như: tô màu đồ thị (graph coloring) , lập trình ràng buộc (constraint prorgaming) , giải thuật di truyền (genetic algorithms) , BackTracking... Các giải thuật này đều chưa đưa ra kết quả tối ưu, nhưng chúng ta có thể xếp một lịch biểu tốt nhất trong các lịc biểu có thể.

Giải pháp được sử dụng trong đề tài này là kết hợp giải thuật Backtracking with Look Ahead và giải thuật Tabu Search hoặc WSAT search để xếp thời khóa biểu cho một trường phổ thông trung học. Giải thuật Backtracking with Look Ahead sẽ tiến hành xếp các môn học vào các tiết trong thời khóa biểu của các lớp tương ứng mà vẫn thỏa mãn các ràng buộc cứng. Giải thuật Tabu Search hoặc WSAT sẽ tối ưu hóa thời khóa biểu vừa được xếp theo hướng thỏa mãn càng nhiều ràng buộc mềm càng tốt (có lợi đối với giáo viên và học sinh) nhưng vẫn đảm bảo thỏa mãn các ràng buộc cứng.

Mục tiêu của đề tài

- Giới thiệu về bài toán xếp thời khóa biểu trường phổ thông và các ràng buộc của bài toán.
- Trình bày ba giải thuật Backtracking with Look Ahead (BC_FC) , Tabu Search, WSAT Search.
- Áp dụng giải thuật BC_FC để giải quyết thỏa mãn ràng buộc cứng của bài toán.
- Áp dụng giải thuật Tabu Search và WSAT để giải quyết các ràng buộc mềm của bài toán.
- So sánh và đánh giá hiệu quả và thời gian chạy của hai giải thuật Tabu Search và WSAT Search.

- Xây dựng chương trình bằng ngôn ngữ lập trình C# dựa trên cơ sở dữ liệu SQL Server 2005.

Nội dung của chính của bảng báo cáo

Chương 2: Giới thiệu về bài toán xếp thời khóa biểu cho trường phổ thông. Nội dung chính của chương trình sẽ giới thiệu về cách thức tổ chức giảng dạy của một trường phổ thông trung học ở Việt Nam. Dựa vào cách thức tổ chức đó sẽ đưa ra những yếu tố quyết định đến việc hình thành thời khóa biểu. Những yếu tố này chính là ràng buộc, là cơ sở quyết định tính khả thi của bài toán thời khóa biểu. Tất cả các ràng buộc của bài toán sẽ được trình bày ở chương này.

Chương 3: Giới thiệu ba giải thuật Backtracking with Look Ahead, Tabu Search và WSAT Search. Chương này sẽ trình bày các quy luật hoạt động chính của ba giải thuật Backtracking with Look Ahead, Tabu Search và WSAT Search, các thông số cần thiết của từng giải thuật. Mỗi tương quan giữa bài toán xếp thời khóa biểu với hai giải thuật cũng được trình bày trong chương này. Ngoài ra, trong phần giới thiệu về quy luật hoạt động của từng giải thuật có kèm theo một số heuristic để giúp giải thuật đạt hiệu quả cao nhất.

Chương 4: Trình bày phần thiết kế và thực hiện chương trình. Đây là phần chính của luận văn. Chương này sẽ trình bày về cách thiết kế cơ sở dữ liệu, các cấu trúc dữ liệu được sử dụng trong chương trình và các kỹ thuật để chuyển đổi từ bài toán xếp thời khóa biểu sang dạng của các giải thuật. Giao diện và cách sử dụng chương trình cũng được trình bày trong chương này.

Chương 5: Kết luận và hướng phát triển chương trình. Chương này sẽ trình bày các kết quả của quá trình thực hiện đề tài, các ưu khuyết điểm của chương trình và đề xuất hướng phát triển trong tương lai.

CHƯƠNG 2: BÀI TOÁN XẾP THỜI KHÓA BIỂU

Chương này giới thiệu các đặc trưng của từng bài toán xếp thời khóa biểu cho trường phổ thông trung học cũng như các ràng buộc cần được giải quyết.

2.1. KHÁI NIỆM VỀ BÀI TOÁN LẬP LỊCH

Lập lịch là bài toán thường gặp trong các trường học cũng như trong các tổ chức sản xuất để phân phối các tài nguyên như máy móc, vật tư, công cụ lao động, ... theo thời gian nhằm thực hiện một loạt công việc của tổ chức, trong đó phải đạt được các mục tiêu và thỏa mãn các ràng buộc khác nhau.

Để lịch tạo ra là đúng đắn thì nó phải thỏa mãn một số ràng buộc bắt buộc gọi là ràng buộc cứng. Ngoài ra còn có một số ràng buộc nếu không thỏa mãn sẽ ảnh hưởng tới tính khả thi của lịch gọi là ràng buộc mềm.

Một lịch biểu là tốt nhất khi nó thỏa mãn tất cả các ràng buộc cả cứng lẫn mềm nhưng trong thực tế hầu như không bao giờ có một lịch biểu tối ưu. Mục tiêu của chúng ta là tạo ra lịch biểu có tính khả thi tức là thỏa mãn tất cả các ràng buộc cứng và thỏa mãn một số ràng buộc mềm theo các tiêu chí đánh giá.

2.2. BÀI TOÁN XẾP THỜI KHÓA BIỂU

2.2.1. Mô tả

Xếp thời khóa biểu là một trong những việc quan trọng trong công tác tổ chức quản lý, giảng dạy và học tập tại các trường học từ phổ thông đến đại học. Việc xếp thời khóa biểu là một bài toán thuộc dạng NP-Complete, các giải thuật thông thường hầu như không giải quyết.

Thời khóa biểu là một lịch trình ấn định thứ tự tiến hành và thời gian của các bài giảng cho các lớp học trong tuần ở mỗi học kỳ. Ngoài ra thời khóa biểu còn phải thỏa mãn yêu cầu về việc sử dụng có hiệu quả cơ sở vật chất của nhà trường đồng thời tạo điều kiện thuận lợi nhất trong việc học tập và giảng dạy.

2.2.2. Các ràng buộc trong việc lập thời khóa biểu

a. *Ràng buộc cứng*: Là các ràng buộc phải thỏa mãn:

- Một lớp học không thể có nhiều hơn một giáo viên cùng dạy trong một tiết.
- Trong một tiết học, một giáo viên không dạy nhiều hơn một tiết.
- Không xếp giáo viên vào lớp học không được phân công dạy.
- Số tiết học của các môn phải bằng số tiết phân phối cho môn học lớp đó.
- Một lớp học không học một môn học quá 2 tiết trong một buổi học.

- Nếu một lớp học phải học một môn học 2 tiết trong một buổi thì hai tiết này phải liên tục và không có giờ ra chơi xen vào giữa.
- Các tiết trùng (nếu có) trong tuần thì phải được dồn về một tiết ở cuối tuần hay cuối buổi (tùy trường).
- Xếp giáo viên chủ nhiệm vào lớp do giáo viên đó phụ trách.
- Không xếp giáo viên dạy vào ngày chuyên môn của môn học.
- Các ngày chuyên môn theo quy định của Sở Giáo dục và Đào tạo là:
 - Môn Toán, Địa lý, Lịch Sử: Thứ 3
 - Môn Văn, Lý, Hóa: Thứ 4
 - Môn Ngoại ngữ, Sinh Vật: Thứ 6
 - Môn GD&CD, Kỹ thuật, Thể dục: Thứ 7.

b. *Các ràng buộc mềm*: Là các ràng buộc thỏa càng nhiều càng tốt.

- Trong một buổi học, một lớp học không thể học toàn môn xã hội hay môn tự nhiên.
- Các môn học cần được rải đều ra các ngày trong tuần.
- Ông nên xếp giáo viên ngày quá nhiều hay quá ít tiết.
- Nên xếp mỗi giáo viên có ít nhất một ngày trùng trong tuần (có thể ngày chuyên môn).
- Ưu tiên cho giáo viên nữ, con nhỏ, không nên xếp vào tiết đầu, tiết cuối của buổi sáng hay tiết đầu của buổi chiều.

Vậy khi xếp thời khóa biểu, người xếp sẽ có bảng phân công giảng dạy, danh sách chủ nhiệm, các yêu cầu của giáo viên.

2.2.3. Tính tương đối của lời giải

Mỗi thời khóa biểu được gọi là tối ưu nếu thỏa mãn tất cả các ràng buộc nhưng trong thực tế hầu như không thể tìm được nên chúng ta chỉ tìm một thời khóa biểu tốt nhất có thể đạt được. Tùy theo giải thuật mà ta chọn sẽ cho ta được những thời khóa biểu khác nhau. Khác về hiệu quả lẫn cả thời gian chạy giải thuật. Việc tìm được phương án để giải quyết tốt cho bài toán là rất quan trọng. Một thời khóa biểu tốt là một thời khóa biểu thỏa mãn tất cả các ràng buộc cứng và một số ràng buộc mềm phù hợp yêu cầu của việc lập thời khóa biểu.

Khi giải quyết tranh chấp phải dựa vào một số thông tin ưu tiên của giáo viên, việc này tương đối phức tạp. Ví dụ: Các giáo viên phải dạy nhiều lớp hoặc nhiều tiết trong tuần phải được ưu tiên xếp trước.

2.2.4. Sắp thời khóa biểu bằng tay

- Phân phối các tiết trong tuần:
- Tổng số tiết trong tuần là $5*6 = 30$ tiết.
- Tiết sinh hoạt chủ nhiệm là cố định.
- Còn lại 29 tiết để xếp lịch học.
- Giáo viên giảng dạy theo bảng phân công của Ban giám hiệu.
- Người xếp thời khóa biểu không được tùy ý sửa đổi bảng phân công.
- Ưu tiên cho các ràng buộc cứng và khó thì xếp trước, ràng buộc mềm thì xếp sau.
- Xếp những môn nhiều tiết trước.
- Xếp các giáo viên nhiều tiết trước.
- Tất cả các ràng buộc cứng phải thỏa, ràng buộc mềm thỏa càng nhiều càng tốt.

Do những giới hạn về khả năng của con người nên những công việc này tỏ ra khó khăn khi số lượng giáo viên và lớp học của các trường khá nhiều. Vì vậy việc xếp thời khóa biểu bằng tay thường mất rất nhiều thời gian và công sức nhưng không cho được kết quả mong muốn.

2.3. TÓM TẮT

Qua chương này chúng ta đã biết các đặc trưng của bài toán xếp thời khóa biểu cũng như các yêu cầu về ràng buộc và tính chất của từng ràng buộc trong bài toán.

CHƯƠNG 3: LÝ THUYẾT VỀ HỆ GIẢI RÀNG BUỘC

Chương này đề cập đến nền tảng lý thuyết của ba giải thuật cũng như phương pháp áp dụng lý thuyết này để giải các bài toán trong thực tế.

3.1. TỔNG QUAN VỀ BÀI TOÁN GIẢI HỆ RÀNG BUỘC

Trong một bài toán giải hệ ràng buộc chúng ta có một tập các biến, miền cho từng biến và một tập các ràng buộc. Mỗi ràng buộc liên quan đến một tập các biến và giới hạn sự kết hợp của các giá trị gán cho biến đó. Mục tiêu của bài toán là tìm ra một cách gán giá trị cho mỗi biến mà thỏa mãn toàn bộ ràng buộc.

Bài toán CSP có thể chia làm hai loại chính: thỏa mãn ràng buộc (Satisfiability problems) và bài toán tối ưu hóa (Optimization problems).

Ràng buộc cũng có thể chia làm hai loại là ràng buộc cứng và ràng buộc mềm.

Bài toán CSP được ứng dụng trong nhiều lĩnh vực như:

- Machine vision
- Sắp xếp (Scheduling) .
- Temporal reasoning.
- Floor plan design.
- Mô tả các mạch analog (Diagnosis of analog) .
- Lập kế hoạch về tài chính (Finacial planning) .
- Thiết kế kỹ thuật dựa trên nền tảng đã có (constraint- based engineering design) .

3.2. BÀI TOÁN CSP

3.2.1. Định nghĩa

Constraint Satisfaction Problem (CSP) bao gồm một bộ ba $P = (V, D, C)$ trong đó:

$V = \{ V_1, V_2, \dots, V_n \}$ là tập hợp các biến

$D = \{ D_{v1}, D_{v2}, \dots, D_{vn} \}$ là tập hợp miền trị của các biến

$C = \{ C_{i,j,k}, \dots, C_{i,j,m}, \dots, C_n \}$ là tập hợp các ràng buộc trên các biến, nó giới hạn các khả năng lựa chọn giá trị cho biến.

Biến (Variables) : là các đại lượng cần được gán giá trị, ví dụ như vị trí các quân cờ trong bài toán N-Queens.

Miền trị (domains) : là tập hợp các giá trị có thể gán được cho một biến, ví dụ như tập giá trị của con Hậu trong bài toán N-Queens.

Ràng buộc (Constraint) : là điều kiện mà một hay nhiều biến phải thỏa, ví dụ như ràng buộc của bài toán N-Queens là hai quân Hậu không được thấy nhau.

Mục tiêu khi giải bài toán CSP là tìm giá trị gán cho các biến sao cho thỏa mãn tất cả các ràng buộc.

Ví dụ về bài toán CSP:

Có bốn biến A, B, C, D. Miền giá trị các biến là : { 0,1,2,3,4,5,6} . Tìm các giá trị của A,B,C,D thỏa : $A = D$, $B = 2*D$, $C > D$, $B > C$.

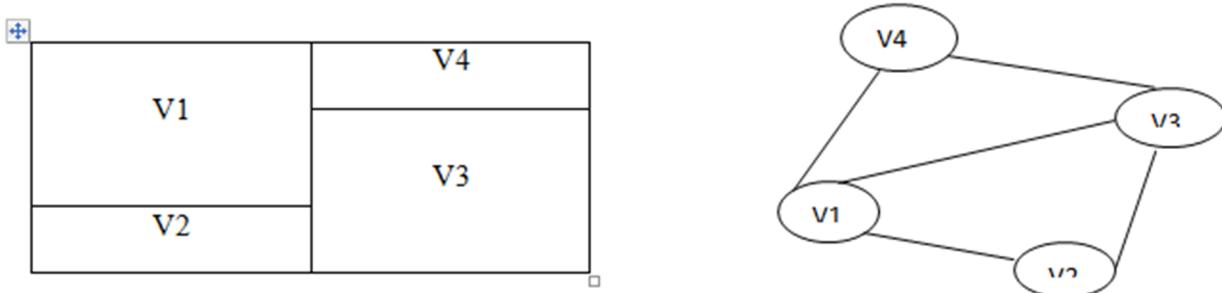
Đây là một bài toán CSP, trong đó:

- Biến V = {A, B, C, D}
- Miền trị DA = { 0,1,2,3,4,5,6} , DB = {0,1,2,3,4,5,6} , DC = {0,1,2,3,4,5,6}
,
- D = {0,1,2,3,4,5,6} .
- Ràng buộc C = { A = D, C > D, B=2 *D, B> C}

Ví dụ mô tả CSP cho bài toán tô màu bản đồ

Bản đồ có bốn vùng khác nhau được tô màu khác nhau , mỗi vùng có thể có một trong ba màu sau: đỏ, xanh đậm và xanh dương. Ta có thể mô tả bài toán tô màu bản đồ như sau: mỗi vùng trên bản đồ là một biến được biểu diễn bằng một nút là {đỏ, xanh đậm và xanh dương} .

Với các vùng kề nhau trên bản đồ ta biểu diễn bằng các ràng buộc không cho phép hai biến tương ứng có vùng giá trị màu.



Hình 3-1: Minh họa bài toán tô màu

Phân dạng bài toán CSP

Các bài toán CSP gồm hai loại chính:

- Bài toán thỏa mãn ràng buộc (Satisfiability Problems) : mục tiêu của bài toán này gán giá trị cho tất cả các biến sao cho thỏa mãn tất cả các ràng buộc.

- Bài toán tối ưu (Optimization problems) : mỗi giá trị được gán cho biến được đánh giá theo mức chi phí hay giá trị. Mục tiêu của bài toán là tìm ra kết quả có tổng chi phí thấp nhất hay tổng giá trị cao nhất.

3.2.2. Độ phức tạp của bài toán CSP

Tất cả các bài toán CSP đều có dạng NP-Complete hay NP-Hard, có độ phức tạp hàm mũ.

Vì bài toán CSP là thuộc dạng NP- Complete nên có bốn phương pháp chính như sau:

Cố gắng tìm ra giải thuật có thể áp dụng cho đa số trường hợp dù một số trường hợp đặc biệt thì giải thuật có độ phức tạp là hàm mũ.

- Cố gắng tìm ra các giải thuật hiệu quả cho các trường hợp đặc biệt.
- Cố gắng tìm ra các giải thuật xấp xỉ tối ưu.
- Phát triển các giải thuật song song và phân bố.

3.2.3. Các bài toán CSP thông dụng

Bài toán tô màu đồ thị: tô màu sao cho giữa hai đỉnh nối nhau không cùng màu.

Bài toán sắp xếp quân Hậu (bài toán N- Queens) ...

3.2.4. Ứng dụng trong thực tế

- Được ứng dụng rất rộng rãi trong công nghiệp như: hỗ trợ ra quyết định, bài toán về thiết kế robot.
- Tô màu bản đồ, xử lý ảnh 2D.
- Các bài toán về lập kế hoạch hay xếp lịch như: xếp lịch thi, xếp thời khóa biểu, xếp lịch công tác,...
- Các bài toán vận chuyển, tìm đường,...

Phân loại ràng buộc:

Phân loại theo số biến tham gia vào ràng buộc:

- Đơn biến (unary constraint)
- Song biến (binary constraint)
- N biến (n-arg constraint), ràng buộc n biến có biểu diễn bằng ràng buộc song biến.

Phân loại theo mức độ quan trọng:

- Ràng buộc cứng (hard-constraint) : kết quả có thể vi phạm ràng buộc này.

- Ràng buộc mềm (soft - constraint) : kết quả có thể vi phạm ràng buộc này nhưng hạn chế số lượng vi phạm thấp nhất.

Phân loại bài toán CSP:

Phân loại theo miền giá trị:

Hữu hạn : đa số bài toán là thuộc dạng này trong đó có bài toán có thời khóa biểu thuộc dạng này.

Vô hạn.

Phân loại theo số biến:

Cố định hoặc thay đổi. Bài toán xếp thời khóa biểu có số biến cố định.

Trong trường hợp bài toán CSP không có kết quả do ràng buộc quá cứng (Over-constrained) ta có thể hiệu chỉnh bài toán để có kết quả bằng các cách sau:

- Mở rộng miền trị.
- Nới lỏng ràng buộc.
- Loại bỏ biến.
- Loại bỏ ràng buộc.

3.3. CÁC GIẢI THUẬT TRÊN BÀI TOÁN CSP

Phân loại

Có hai phương pháp để giải quyết bài toán CSP:

3.3.1. Tìm kiếm có hệ thống (Systematic search)

Gán từng giá trị cho các biến đến khi tất cả các biến này đều được gán và thỏa mãn tất cả ràng buộc. Phương pháp này cố gắng gán tất cả giá trị có thể cho biến.

3.3.2. Tìm kiếm không hệ thống (non-Systematic search)

Từ một kết quả ban đầu giải thuật tìm một kết quả lân cận tốt hơn, quá trình tiếp tục cho đến khi tìm được kết quả tối ưu cục bộ (local optimun) .

Phương pháp tìm kiếm	Hệ thống	Không hệ thống
Các giải thuật tiêu biểu	Vét cạn (Generation and test) , backtracking + forward checking,...	Leo đồi (Hill-climbing), Tabu search, cực tiểu xung đột (min-conflic) , giải thuật di truyền (genetic algorithm)

Bảng 3.1: Các phương pháp tìm kiếm.

Chương này xếp thời khóa biểu sử dụng phương pháp backtracking + forward checking (BC_FC) để sinh ra kết quả ban đầu, sau đó dùng giải thuật Tabu Search, WSAT Search để hiệu chỉnh, tối ưu bài toán sắp xếp thời khóa biểu.

3.4. GIỚI THIỆU GIẢI THUẬT BÀI TOÁN CSP

3.4.1. Giải thuật vét cạn (Generate and test)

Giải thuật này có khả năng giải quyết bất kỳ bài toán CSP nào.

Nguyên tắc:

- Tất cả các kết hợp có thể sẽ được sinh ra và kiểm tra có thỏa mãn tất cả các ràng buộc hay không.
- Kết hợp đầu tiên thỏa mãn tất cả các ràng buộc là kết quả.

Ví dụ:

Xếp 5 môn A,B,C,D,E vào các tiết {1,2,3,4,5,6} . Các kết hợp có thể là $6^5 = 7776$ kết hợp.

- Ưu điểm: dễ hiểu và hiện thực nhất nhưng chỉ thích hợp cho các bài toán có không gian tìm kiếm nhỏ.
- Nhược điểm: Không có định hướng trong việc gán giá trị cho biến và không phát hiện sự bất tương thíc trong tương lai.

3.4.2. Giải thuật backtracking

Nguyên tắc : Các biến được xếp thứ tự sau đó lần lượt gán giá trị cho các biến, trước khi gán giá trị kiểm tra giá trị đó có vi phạm ràng buộc với các giá trị đã gán hay không.Nếu có vi phạm thì kiểm tra giá trị khác. Nếu tất cả các giá trị đều không thỏa mãn thì quay lùi lại biến gần nhất.

- **Giải thuật:**

Procedure BACKTRACKING:

```
i = 1
Di':=Di
While 1 ≤ i ≤ n
    Giá trị ban đầu Vi := SELECTVALUE
    if Vi rỗng then
        i:=i-1;
    else
        i:=i+1
```

```
Di' := Di
End while
if i=0
Return “mâu thuẫn”
else
Return giá trị được gán của { V1, V2, ..., Vn}
End procedure

Procedure SELECTVALUE
    While Di' không rỗng
        Chọn một giá trị a bất kỳ thuộc Di', xóa a khỏi Di'
        If a phù hợp với < a1, a2, ...m ai-1>
            Return a
        End while
        Return null
    End Procedure
```

Ưu điểm: Cải tiến so với giải thuật vét cạn, không gian tìm kiếm giảm đáng kể.

Nhược điểm:

- Thrashing: các biến gán giá trị không thích hợp do cùng nguyên nhân có thể do ràng buộc đơn biến hay song song.
- Công việc dư thừa (Redundant work) : kiểm tra và gán giá trị cho các biến mà sau đó chắc chắn sẽ xung đột.
- Không có phương pháp phát triển trước sự xung đột.

3.4.3. Giải thuật tương thích (Consistency algorithms)

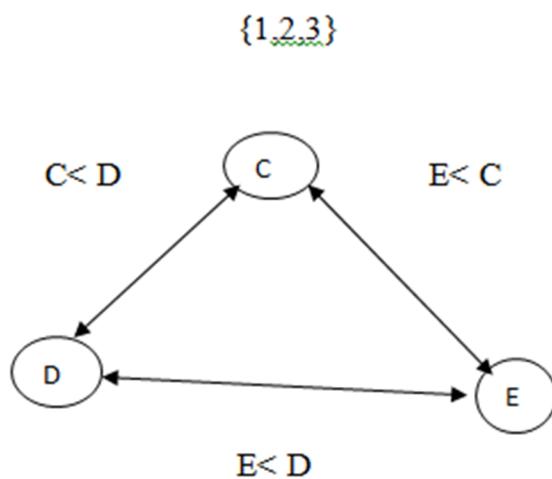
Giải thuật Backtracking hiệu quả hơn giải thuật vét cạn nhưng độ phức tạp của nó cũng là một hàm mũ. Một trong những nguyên nhân khiến giải thuật Backtracking không hiệu quả là vấn đề “thrashing”. Thrashing có nghĩa là việc tìm kiếm trong những phần khác nhau của không gian bài toán thất bại do cùng một nguyên nhân.

Ví dụ:

Xét hai biến C, D. Gán C=4 là không tương thích với mọi giá trị của D vì Dd=1,2,3,4 và C< D. Trong trường hợp giải thuật Backtracking thì điều này lặp lại trong mỗi lần ta gán cho A, B hay E. Để tránh điều này ta có thể xóa hòn 4 ra khỏi Dc một lần duy nhất.

Bài toán CSP có thể được biểu diễn dưới dạng đồ thị ràng buộc. Mỗi biến tương ứng một đỉnh của đồ thị với miền trị kèm theo. Mỗi ràng buộc P (X, Y) tương ứng hai cạnh $\langle X, Y \rangle$ và $\langle Y, X \rangle$ của đồ thị.

Ví dụ:



Hình 3-2: Đồ thị ràng buộc

Một node trong đồ thị ràng buộc được gọi là có miến tị tương thích nếu không có giá trị nào vi phạm các ràng buộc đơn vị. Ví dụ: $D_b = \{1, 2, 3, 4\}$ là một miền trị không tương thích nếu có ràng buộc $B \# 3$.

Một cạnh $\langle X, Y \rangle$ gọi là tương thích nếu với mọi giá trị của X trong D_x tồn tại giá trị Y trong D_y sao cho ràng buộc $P(X, Y)$ thỏa mãn. Một đồ thị gọi là tương thích cạnh nếu mọi cạnh của nó đều tương thích.

Nếu một cạnh $\langle X, Y \rangle$ không tương thích thì mọi giá trị X trong D_x mà không có giá trị Y tương thích phải xóa khỏi D_x để đảm bảo tính tương thích của cạnh $\langle X, Y \rangle$.

Giải thuật tương thích cạnh AC-3 tạo ra đồ thị ràng buộc thỏa mãn tính tương thích cạnh.

Nguyên tắc: tạo ra một queue chứa tất cả các cạnh của đồ thị (một ràng buộc $P(X, Y)$ sẽ tạo ra hai cạnh $\langle X, Y \rangle$ và $\langle Y, X \rangle$; lặp cho tới khi queue rỗng, một cạnh được lấy khỏi queue và xét. Nếu nó không tương thích thì làm nó tương thích và thêm tất cả các cạnh có thể trở nên không tương thích vào queue.

• Giải thuật

Input:

Tập hợp các biến X
Tập hợp các miền trị D_x
Ràng buộc P_x trên biến X.
Ràng buộc P_{xy} trên biến X, Y.

Output: Đồ thị ràng buộc tương thích cạnh.

For mỗi biến X do:

$D_x := \{ x \in D_x / P_x(x) \} //$ sử dụng các ràng buộc đơn biến

$Q := \{ < X, Y > / P_{xy} \text{ là ràng buộc song biến} \} .$

Repeat:

Chọn một cạnh bất kỳ $< X, Y >$ thuộc Q

$Q := Q - \{ < X, Y > \}$

$ND_x := D_x - \{ x / x \in D_x \text{ và không có giá trị } Y \in D_y \text{ nào để}$

$P_{xy}(X, Y)$ thỏa mãn}

If $ND_x \neq D_x$ then

$Q := Q - \{ < Z, X > / Z \neq Y \}$

Until Q is empty

Ví dụ : Áp dụng giải thuật AC -3 cho đồ thị ràng buộc trên:

Ban đầu queue có các cạnh sau:

$C < D$ (cạnh 1)

$D > C$ (cạnh 2)

$E < C$ (cạnh 3)

$C > E$ (cạnh 4)

$E > D$ (cạnh 5)

$D > E$ (cạnh 6)

Xét cạnh 1: loại 4 khỏi miền trị của C. $D_C = \{1, 3\}$, cần thêm cạnh 3 nhưng cạnh đã có trong Q.

Xét cạnh 2: Loại 1 khỏi miền trị của D. $D_d = \{2,3,4\}$, cần thêm cạnh 5 nhưng cạnh 5 đã có trong Q.

Xét cạnh 3: Loại 3,4 khỏi miền trị của E. $D_e = \{1,2\}$, cần thêm cạnh 6 nhưng cạnh 6 đã có trong Q.

Xét cạnh 4: không thay đổi

Xét cạnh 5: không thay đổi

Xét cạnh 2: Loại 2,3 khỏi miền trị của D. $D_d = \{4\}$, cần thêm cạnh 5 vào Q.

Xét cạnh 5: không thay đổi.

Lúc này queue rỗng. Vậy miền trị của các biến như sau:

$$D_c = \{3\} \quad D_d = \{4\} \quad D_e = \{1,2\}$$

Kết quả sau khi thực hiện kỹ thuật này cho tất cả các biến.

Có một miền trị rỗng: bài toán không có lời giải.

Mỗi miền trị chỉ còn có một giá trị : đó cũng là kết quả bài toán.

Không có miền trị rỗng và có ít nhất một miền trị có hơn một giá trị . Khi đó ta lại chia các miền trị đó ra và áp dụng tiếp kỹ thuật tương thích cho hai bài toán CSP chỏ hơn.Kết hợp giải thuật AC-3 và giải thuật chia miền trị có thể giải quyết bài toán CSP.

Độ phức tạp của giải thuật AC-3:

Với kích thước miền trị của các biến là d và có e ràng buộc thì độ phức tạp của giải thuật AC-3 là $O(ed^3)$.

3.4.4. Giải thuật Look ahead (Backtracking + Forward Checking)

Giải thuật Forward Checking có thể xem là một giải thuật cải tiến của giải thuật Backtracking cơ bản bằng cách loại khỏi các giá trị thuộc miền của các biến chưa gán giá trị mà không tương thích với giá trị vừa gán cho biến hiện tại. Điều này loại bỏ tất cả các vi phạm có thể xảy ra trên các biến đã gán giá trị.

Việc sắp xếp biến có ảnh hưởng lớn đến kích thước của không gian tìm kiếm nên việc tìm ra một giải thuật sắp xếp biến hiệu quả là một việc hết sức quan trọng trong việc giải quyết bài toán CSP.

Một heuristic thông dụng là sử dụng kích thước của miền trị D' để lựa chọn biến kế tiếp.

Ngoài ra, Những thông tin có được về kích thước của miền trị các biến trong giai đoạn look- ahead có thể được sử dụng để lựa chọn giá trị. Giải thuật sắp xếp giá trị nhìn trước (look- ahead value ordering) dựa trên ý tưởng forward checking: LVO lựa chọn từng giá trị cho biến hiện tại và kiểm tra sự tác động đến miền trị của các biến

phía sau. Sau đó giải thuật LVO sử dụng một hàm heuristic để đánh giá xếp hạng các giá trị.

Một heuristic khác gọi là cực tiểu xung đột (min-conflict) thì xét từng giá trị trong D' của biến hiện tại và kết hợp nó với giá trị của các biến tương lai, đếm số kết hợp không tương thích. Các giá trị sẽ được chọn theo thứ tự có số xung đột thấp nhất.

- Giải thuật

```
Procedure Backtracking-with-look-ahead
     $D'_i := D_i$  với  $1 \leq i \leq n$ 
     $i := 1$ 
    While  $1 < i < n$ 
        Gán  $V_i :=$  Selectvalue-forward-checking
        If  $V_i$  rỗng then
             $i := i - 1;$ 
            Reset  $D_k'$  với  $k > i$  về giá trị trước khi  $V_i$  được gán giá trị
        Else  $i := i + 1;$ 
    End while
    If  $I = 0$  then
        Return “mâu thuẫn”
    Else return giá trị của  $\{V_1, V_2, \dots, V_n\}$ 
End procedure

Procedure Selectvalue-forward-checking
    While  $D'_i$  không rỗng
        Chọn một giá trị  $a$  bất kỳ thuộc  $D'_i$ , loại nó khỏi  $D'_i$ 
        Empty_domain = false:
        For tất cả  $k_i < k \leq n$ 
            For tất cả  $b$  thuộc  $D_k'$ 
                If  $b$  không tương thích với  $< a_1, \dots, a_{i-1}, \dots >$  và  $V_i = a$  then
                    Loại bỏ khỏi  $D_k'$ 
            End for
        If  $D_k'$  rỗng then
```

```
Empty_domain = true;  
End for  
If Empty_domain = true then  
    Reset các giá trị  $D_k'$  về trạng thái trước khi a được gán.  
Else return a  
End while  
Return null  
End procedure
```

3.4.5. Các kỹ thuật được sử dụng để tăng tốc độ tìm kiếm

3.4.5.1. Kỹ thuật sắp xếp biến (Variable Ordering)

Thứ tự các biến có ảnh hưởng lớn đến hiệu quả của giải thuật tìm kiếm.

Một số heuristic cho việc sắp xếp biến

Phương pháp	Mô tả
Sắp thứ tự tĩnh (Static ordering)	Sắp xếp giá trị các biến trước khi quá trình tìm kiếm bắt đầu và thứ tự này không thay đổi sau này.
Sắp thứ tự động (Dynamic ordering)	Lựa chọn biến kế tiếp phụ thuộc vào trạng thái hiện tại.
Least-constraining	Ưu tiên cho các biến có miền trị nhỏ nhất.
Most-constrained 1	Ưu tiên cho các biến tham gia vào nhiều ràng buộc nhất.
Most-constrained 2	Ưu tiên cho các biến tham gia vào nhiều ràng buộc với các biến đã gán nhất.
Stable set	Chọn tập biến mà giữa chúng không có ràng buộc nào. Tập này được gán giá trị sau cùng.
Cycle cutset	Xóa một số ràng buộc trong đồ thị ràng buộc thành cây ràng buộc.

Bảng 3.2: Các phương pháp sắp xếp biến

3.4.5.2. Kỹ thuật sắp xếp trị (Value ordering)

Thứ tự lựa chọn giá trị cho các biến là rất quan trọng vì khi ta chọn đúng giá trị cho biến thì không phải quay lui. Một trong những quy tắc đầu tiên là chọn trị cho lượng giá trị của các biến chưa khởi tạo là lớn nhất. Một heuristic cho kỹ thuật này

Phương pháp	Mô tả
Least constraining	Chọn giá trị làm cho số lượng lựa chọn cho các biến còn lại là lớn nhất
Succeed-first	Chọn giá trị dễ thành công nhất, tùy thuộc bản chất bài toán
Easiest	Đánh giá mức độ khó của ràng buộc

Bảng 3.3: Các phương pháp sắp xếp trị

Hầu hết các giải thuật CSP đều dựa trên backtracking với việc tìm cách gán các giá trị lần lượt cho các biến.

Ngược lại các giải thuật tu chỉnh lặp tạo ra một phép gán đầy đủ cho các biến nhưng phép gán này chưa thỏa mãn các ràng buộc. Sau đó thực hiện tu chỉnh lặp cho đến khi các ràng buộc được thỏa mãn.

3.5. CÁC GIẢI THUẬT ĐỂ GIẢI BÀI TOÁN TỐI ƯU TỔ HỢP

3.5.1. Tìm kiếm cục bộ (Local Search)

Tìm kiếm cục bộ là một trong những kỹ thuật tìm kiếm chung cho những bài toán tìm kiếm và tối ưu. Các ứng dụng của giải thuật tìm kiếm cục bộ cho bài toán tối ưu được nghiên cứu từ những năm 1960. Kể từ đó, tầm quan trọng của đề tài này được phát triển đáng kể trong các lĩnh vực vận trù học (Operations Research), khoa học máy tính (Computer Science) và trí tuệ nhân tạo (AI).

Các giải thuật tìm kiếm cục bộ là không toàn diện (non-exhaustive) tức là chúng không bảo đảm sẽ tìm được một lời giải tối ưu, nhưng chúng ta sẽ tìm kiếm một cách không hệ thống (non-systematically) cho tới khi thỏa mãn một điều kiện dừng được chỉ định.

Tuy nhiên, những kỹ thuật này lại rất được chú ý bởi tính hiệu quả và khả năng ứng dụng rộng rãi của chúng.

a. Bài toán tối ưu tổ hợp (Combinatorial Optimization Problems)

Chúng ta định nghĩa một hiện thể π của bài toán tối ưu kết hợp Π là một bộ ba $< S, F, f >$. Trong đó S là một tập hữu hạn các lời giải, $F \subseteq S$ là một tập các lời giải khả thi và $f: S \rightarrow \mathbb{R}$ biểu thị một hàm mục tiêu đánh giá chất lượng của mỗi lời giải trong tập S .

Vấn đề được đặt ra là tìm một lời giải tối ưu toàn diện, tức là tìm một phần tử $x^* \in F$ sao cho $f(x^*) \leq f(x)$ với mọi giá trị $x \in F$.

Theo sự sắp đặt này được gọi là tập khả thi, các phần tử của nó được gọi là lời giải khả thi.

b. Bài toán tìm kiếm (Search Problems)

Cho một cặp $< S, F >$, trong đó S là tập lời giải, $F \subseteq S$ là tập lời khả thi. Một bài toán tìm kiếm chính là một bài toán tìm ra lời giải khả thi, tức là một phần tử $x^* \in F$.

Có 3 thực thể chính trong giải thuật tìm kiếm cục bộ : không gian tìm kiếm (search space) quan hệ kế cận (neighborhood relation) và hàm chi phí (cost function) .

c. Không gian tìm kiếm (Search Space)

Cho bài toán tối ưu kết hợp Π , chúng ta sẽ kết hợp mỗi hiện thể π của nó với một không gian tìm kiếm S_π với những thuộc tính sau:

Mỗi phần tử $s \in S_\pi$ biểu diễn một phần tử $x \in S$.

Ít nhất có một phần tử tốt nhất trong F được biểu diễn trong S_π .

Thông thường, không gian tìm kiếm S_π và tập các lời giải S của bài toán như nhau, nhưng có một số trường hợp hai thực thể này sẽ khác nhau.

d. Quan hệ kế cận (Neighborhood Relation)

Cho bài toán Π , một hiện thể π và một không gian tìm kiếm S cho nó, chúng ta gán cho mỗi phần tử $s \in S$ một tập $N(s)$ là những lời giải lân cận của s . $N(s)$ được gọi là tập hợp kế cận (neighbor) của s .

Đối với mỗi lời giải, $N(s)$ sẽ được định nghĩa không tường minh bằng cách suy ra từ một tập hợp các bước chuyển có thể. Những bước chuyển này sẽ định nghĩa các sự dịch chuyển giữa các lời giải.

3.5.2. Hàm chi phí (Cost Function)

Cho một không gian tìm kiếm S đối với một hiện thế π của bài toán Π , chúng ta định nghĩa hàm chi phí f , nó kết hợp mỗi một phần tử $s \in S$ với một giá trị $f(s)$ để đánh giá chất lượng của lời giải. Trên thực tế, miền giá trị của hàm f là một tập hợp có thứ tự như là tập các số tự nhiên hoặc tập các số thực không dấu.

Hàm chi phí được dùng để hướng việc tìm kiếm về các lời giải tốt của không gian tìm kiếm. Nó cũng được dùng để lựa chọn sự di chuyển tại mỗi bước của việc tìm kiếm.

3.5.3. Giải thuật tìm kiếm cục bộ tổng quát (General Local Search Algorithm)

Một giải thuật tìm kiếm cục bộ bắt đầu từ một lời giải ban đầu $s_0 \in S$ và lặp lại cuộc khảo sát trong không gian tìm kiếm, sử dụng các bước chuyển được kết hợp với các định nghĩa kế cận. Tại mỗi bước nó tạo ra một bước chuyển giữa một lời giải s đến một lời giải s' là kế cận của nó. Khi giải thuật tạo ra một bước chuyển từ lời giải s sang s' thì chúng ta nói rằng bước chuyển m tương ứng đó đã được chấp nhận, và chúng ta viết:

$$s' = s \oplus m.$$

Việc lựa chọn các bước chuyển được dựa trên giá trị của hàm chi phí.

Đối với một số kỹ thuật, việc khởi tạo lời giải ban đầu s_0 là một phần của giải thuật. Với những phương pháp khác, s_0 có thể được khởi tạo một cách tự do bởi một số giải thuật hoặc được tạo ra một cách ngẫu nhiên.

Điều kiện dừng là một phần của kỹ thuật đặc trưng, nó dựa trên các chất lượng được chỉ định mà lời giải đạt được hoặc dựa trên một số lần lặp lớn nhất nào đó.

- Giải thuật

Procedure Loacal Search (MaxCost, MaxMoves)

Begin

Gán $s = \{ (v_i, d_{ij}) / d_{ij} \in D_i \text{ cho tất cả } v_i \in V \}$ là lời giải ban đầu

While $f(s) > \text{MaxCost}$ and $\text{TotalMoves} < \text{MaxMoves}$ do

Begin

← GennerateLocalMoves ($s, \text{TotalMoves}$)

If $M' \neq$ then MakeLocalMoves ($s, M', \text{TotalMoves}$)

End

End

M là tập tất cả các bước chuyển cục bộ từ lời giả s, $M' \subseteq M$

Thủ tục *GenarateLocalMoves* trả về một tập tất cả các bước chuyển tốt nhất M' trong tập lân cận của s.

Các hàm *GenaralLocalMoves* và *MakeLocalMoves* sẽ được định nghĩa chi tiết tùy theo từng phương pháp tìm kiếm cục bộ cụ thể.

Có 3 kỹ thuật tìm kiếm cục bộ phổ biến nhất đó là: leo đồi (*Hill-Climbing*) , tìm kiếm Tabu (*Tabu search*) và mô phỏng luyễn kim (*Simulated Annealing*) .

3.5.4. Giải thuật leo đồi (Hill-Climbing)

Giải thuật leo đồi là cơ sở cho tất cả các giải thuật tìm kiếm cục bộ. Mặc dù rất đơn giản nhưng nó vẫn chứng tỏ được sức mạnh và tính hiệu quả khi giải quyết các bài toán với không gian tìm kiếm rộng.

Thuật ngữ “leo đồi” xuất phát từ việc cải tiến được lặp lại, nghĩa là tại mỗi bước tìm kiếm, một bước chuyển làm cải thiện giá trị của hàm chi phí sẽ được chọn. Vì thế nó chỉ chấp nhận các bước chuyển làm cải thiện hoặc giữ nguyên giá trị của hàm chi phí.

Giải thuật leo đồi có các dạng sau:

Steepest Hill-Climbing (SHC) : thủ tục chỉ chấp nhận bước chuyển m nếu và chỉ nếu nó cải thiện được kết quả của hàm chi phí. Do đó, việc tìm kiếm sẽ sớm đạt được giá trị nhỏ nhất cục bộ.

Random Hill-Climbing (RHC) : thủ tục được trình bày trong giải thuật bên dưới.

Min Conflict Hill-Climbing (MCHC) : việc lựa chọn một bước chuyển được chia làm hai phần:

- Tìm ngẫu nhiên một biến v của lời giải s hiện hành bao gồm cả những biến có ít nhất một sự vi phạm ràng buộc.

- Chọn giữa các bước chuyển trong tập N(s) chỉ làm thay đổi giá trị của biến v một bước chuyển tạo ra sự vi phạm ràng buộc ít nhất.

Giải thuật Giải thuật leo đồi được hiện thực như sau:

```
Procedure GernerateLocalMover (s, TotalMoves)
begin
    M' ← Ø, BestCost ← f(s)
    for each vi ∈ V do if vi bị vi phạm ràng buộc then
        begin
            dcurr ← miền trị hiện thời của vi
            for mỗi d ∈ Di | d ≠ dcurr do
                begin
                    m ← {vi, d}
                    if f(s ⊕ m) < BestCost then
                        begin
                            BestCost ← f(s ⊕ m);
                            M' ← Ø;
                        end
                end
            end
        if M' = Ø then TotalMovers ← MaxMovers
        return M'
    end
```

Vấn đề chính của giải thuật leo đồi là nó đi dần đến lời giải có chi phí nhỏ nhất trong không gian tìm kiếm. Nếu không có một bước chuyển đơn nào có thể cải tiến cho một lời giải, quá trình tìm kiếm sẽ bị khóa, ngay cả khi nó chưa đạt được chi phí nhỏ nhất toàn cục.

3.5.4. Tabu Search

Tabu search là một kỹ thuật cải thiện lời giải được đề nghị đầu tiên bởi Glover vào cuối những năm 1980, và sau đó nó đã được ứng dụng để giải quyết nhiều bài toán trong các lãnh vực khác nhau.

a. Giải thuật Tabu search cơ bản

Bắt đầu từ một lời giải ban đầu giải thuật Tabu Search thăm dò lặp lại nhiều lần một tập con V của tập lân cận N của lời giải hiện tại, các thành phần của tập V sẽ được gán một giá trị tối thiểu của hàm mục tiêu (Objective function) và trở nên lời giải hiện tại một cách độc lập với các yếu tố mà giá trị của nó là tốt hơn hay tệ hơn giá trị trong s.

Để ngăn chặng việc lặp vòng (Cycling), ta sử dụng một danh sách gọi là *Tabu List*, là một danh sách của các bước chuyển mà các bước sau đó bị cấm. Đây là một

danh sách của việc đảo ngược của một bước chuyển được chấp nhận cuối cùng k (k là tham số của phương thức) và nó hoạt động như là một hàng đợi (queue) kích thước cố định.

Điều đó có nghĩa là khi một bước chuyển mới được thêm vào thì bước chuyển cũ hơn sẽ bị loại bỏ.

Thủ tục sẽ dừng khi hoặc là số lần lặp quy định đã đạt mà lời giải không có sự cải tiến nào hoặc là giá trị của hàm mục đích trong trạng thái hiện tại đã đạt bằng với cản dưới.

- Giải thuật

```
procedure GenerateLocalMoves (s, TotalMoves)  
begin  
    M'  $\leftarrow \emptyset$ ; BestCost  $\leftarrow \infty$  // giá trị BestCost ban đầu bằng vô cùng (hoặc  
    bằng hàm đánh giá lời giải ban đầu.  
    for each  $v_i \in V$  do  
        begin  
            dcurr  $\leftarrow$  miền trị hiện thời của  $v_i$   
            for each  $d \in D_i | d \neq dcurr$  do  
                begin  
                    m  $\leftarrow \{v_i, d\}$   
                    if m nằm ngoài Tabu List and  $f(s \oplus m) \leq \text{BestMove}$  then  
                        begin  
                            if  $f(s \oplus m) < \text{BestMove}$  then  
                                begin  
                                    BestCost  $\leftarrow f(s \oplus m)$  ; M'  $\leftarrow \emptyset$   
                                end  
                                M'  $\leftarrow M' \cup m$   
                            end  
                        end  
                    end  
                end  
            end  
        end
```

```
return M'  
end  
procedure MakeLocalMoves (s, M', TotalMoves)  
begin  
    lấy ngẫu nhiên một m trong M'  
    s ← s⊕m; TotalMoves ← TotalMoves + 1;  
    update tabu list  
end
```

b. Một số phương pháp cải tiến

Điều kiện mong muốn (Aspiration Level Conditions)

Vì một bước chuyển, không phải là lời giải, được đưa vào trong danh sách Tabu cho nên bước chuyển đó sẽ bị cấm, thậm chí ngay cả khi bước chuyển đó có thể đưa đến lời giải. Nó cách khác, thì mô hình giải thuật Tabu Search không chỉ không trở lại nhưng lời giải trước đó mà còn có thể bỏ sót những lời giải tốt vì sự đơn giản của nó. Chính vì thế mà có một cơ chế gọi là tiêu chuẩn ước muôn (Aspiration criterion) mà có thể ghi đè lại trạng thái cấm của một bước chuyển. Nếu trong một lời giải s, một bước chuyển m có thể tạo ra một sự cải thiện lớn lao giá trị của hàm chi phí thì lời giải s⊕m được chấp nhận như là một lời giải mới mà không quan tâm đến trạng thái bị cấm của nó.

Một cách rõ ràng hơn, chúng ta định nghĩa một hàm ước muôn (Aspiration function) A mà với mỗi giá trị v của hàm mục đích f nó sẽ trả về một giá trị v', biểu diễn giá trị mà giải thuật mong muốn đạt được từ giá trị v. Với một lời giải hiện tại s, hàm mục đích f, và một lời giải lân cận s' được lấy ra thông qua bước chuyển m, nếu $f(s') \leq A(f(s))$ thì s' có thể được chấp nhận, thậm chí m có trong Tabu List.

Tìm kiếm tương thích

Vào năm 1960, Battiti đã mở rộng ý tưởng chiều dài danh sách Tabu cố định, bằng cách làm cho chiều dài của danh sách Tabu có thể co giãn một cách thích hợp tùy theo trạng thái hiện tại của quá trình tìm kiếm. Nếu kích thước của Tabu List được co giãn một cách hợp lý thì giải thuật sẽ rất hiệu quả. Cách làm này tuy sẽ giải quyết được một số vấn đề nhưng cũng để lại một vấn đề khác. Việc tìm ra một cơ chế co giãn kích thước của Tabu List là rất phức tạp.

Vào năm 1995, Hertz et al. đã đề nghị một cách khác đó là giá trị của hàm chi phí sẽ được gán cho lời giải tại những thời điểm khác nhau là khác nhau tùy thuộc vào trạng thái hiện tại của quá trình tìm kiếm.

Như vậy ý tưởng cơ bản của giải thuật Tabu Search đó là tìm kiếm và dùng một danh sách Tabu để tránh việc lặp vòng. Bên cạnh đó cũng dùng một số phương pháp cải tiến để giải thuật hoạt động hiệu quả hơn. Tham số điều khiển chính của giải thuật chính là chiều dài của Tabu List k, hàm ước muôn A, số yếu tố của tập hợp V của các lời giải lân cận được kiểm tra tại mỗi lần lặp, và Tsmax là số lần lặp tối đa mà không có sự cải tiến của hàm mục tiêu

3.5.5. Giải thuật mô phỏng luyện kim (Simulated annealing)

a. Khái niệm về mô phỏng luyện kim

Kỹ thuật này mô phỏng việc làm nguội vật chất trong một bồn nóng. Quá trình này gọi là luyện. Nếu nung nóng một chất rắn qua điểm nóng chảy và sau đó làm nguội nó, các thuộc tính về cấu trúc của chất rắn phụ thuộc vào tốc độ làm nguội. Nếu chất lỏng được làm nguội đủ chậm, những tinh thể lớn sẽ được định hình.

Giải thuật mô phỏng quá trình làm nguội bằng cách làm giảm nhiệt độ của hệ thống một cách từ từ cho đến khi nó đông tụ vững chắc (trạng thái đóng băng) Khi áp dụng vào bài toán tối ưu, mô phỏng luyện kim có thể tìm kiếm các lời giải khả thi và sau đó “hội tụ” về một lời giải tối ưu.

b. Sự khác nhau giữa mô phỏng luyện kim và leo đồi

Leo đồi là một kỹ thuật tìm kiếm cục bộ, thường gặp khó khăn trong một số bài toán khi bị khóa ở chi phí cục bộ nhỏ nhất (lớn nhất) bởi vì nó luôn luôn chọn bước chuyển trong không gian tìm kiếm có thể cải thiện kết quả hàm chi phí. Cho nên nếu không chọn được bước chuyển nào tốt hơn thì nó sẽ không chuyển qua trạng thái mới, ngay cả khi chưa tìm được lời giải tối ưu toàn cục.

Không giống như leo đồi, mô phỏng luyện kim chọn bước chuyển ngẫu nhiên từ một quan hệ kế cận. Nếu bước chuyển là tốt hơn vị trí hiện tại thì mô phỏng luyện kim luôn luôn chọn nó. Nếu bước chuyển là xấu hơn (chất lượng kém hơn), nó sẽ chấp nhận dựa trên một xác suất nhất định.

Mô phỏng luyện kim đã giải quyết được vấn đề của giải thuật leo đồi, nó sẽ không bị xó ở chi phí cục bộ nhỏ nhất, nó có thể chấp nhận một bước chuyển có chất lượng xấu hơn để thoát ra khỏi lời giải có chi phí cục bộ nhỏ nhất và tìm đến lời giải có chi phí nhỏ nhất toàn cục.

c. Điều kiện chấp nhận một bước chuyển

Theo quy luật củ a trạng thái nhiệt động lực học tại nhiệt độ t, xác suất của công việc tăng mực năng lượng lên một khoảng σE được cho bởi công thức:

$$P(\sigma E) = \exp(-\sigma E / kt) \quad (1)$$

Với k là hằng số Boltzmann.

Theo đó, giải thuật mô phỏng luyện kim sẽ tính toán mức năng lượng mới của hệ thống. Nếu năng lượng giảm thì hệ thống sẽ chuyển sang trạng thái mới. Nếu năng lượng tăng thì trạng thái mới được chấp nhận theo một xác suất được trả về từ công thức trên.

Tại mỗi giá trị nhiệt độ, hệ thống sẽ thực hiện một số lần lặp nhất định, sau đó nhiệt độ sẽ được giảm xuống một giá trị thấp hơn. Việc này được lặp lại cho đến khi hệ thống đạt trạng thái đóng băng (ổn định, không thay đổi năng lượng).

Công thức (1) được sử dụng một cách trực tiếp trong giải thuật mô phỏng luyện kim với hằng số Boltzmann được lược bỏ đi bởi vì nó chỉ được giới thiệu trong biểu thức đối với các dạng vật chất khác nhau. Do đó, xác suất chấp nhận một trạng thái xấu hơn được tính theo công thức:

$$P = \exp(-c/t) > r$$

Với c: mức thay đổi của hàm chi phí

t: nhiệt độ hiện tại

r: một số ngẫu nhiên nằm trong khoảng (0,1)

d. Quan hệ giữa luyện kim vật lý và mô phỏng luyện kim

Sự ánh xạ từ quá trình luyện kim sang giải thuật mô phỏng luyện kim được thể hiện ở bảng sau:

Luyện kim	Bài toán tối ưu tổ hợp
Trạng thái hệ thống	Lời giải khả thi
Năng lượng	Chi phí lời giải
Thay đổi trạng thái	Lời giải lân cận
Nhiệt độ	Biến điều kiện
Trạng thái đóng băng	Lời giải sau cùng

Với bảng ánh xạ trên, bất kỳ một bài toán tối ưu nào cũng có thể được đưa về giải thuật mô phỏng luyện kim để giải quyết.

e. Giải thuật mô phỏng luyện kim

Để áp dụng giải thuật mô phỏng luyện kim, ta phải mô tả bài toán ở dạng tìm kiếm cục bộ bằng cách định nghĩa các thành phần sau:

- Một không gian lời giải.
- Một cấu trúc kế cận.
- Một hàm chi phí.

Thực ra việc này chính là định nghĩa một bối cảnh lời giải sẽ được tìm kiếm bằng cách di chuyển từ trạng thái hiện tại sang một trạng thái kế cận. Giải thuật mô phỏng luyện kim được hiện thực như sau:

Chọn lời giải ban đầu s_0

Chọn nhiệt độ ban đầu $t_0 > 0$

Chọn hàm giảm nhiệt độ α

Repeat

Repeat

Chọn lời giải ngẫu nhiên $s \in n(s_0)$ /* s là lời giải lân cận của s_0 */

$$\delta = f(s) - f(s_0)$$

If ($\delta < 0$) **then**

$$S_0 = s$$

Else

Sinh ra một số ngẫu nhiên $x \in [0;1]$

if $x < \exp(-\delta/t)$ **then**

$$S_0 = s$$

endif

endif

until iteration_count=nrep

$$t = \alpha(t)$$

until điều kiện dừng = true. /* s_0 gần như là lời giải tối ưu*/

Các bước chuyển sẽ được lấy ngẫu nhiên và tất cả các bước chuyển cải tiến hơn được chấp nhận một cách tự động. Các bước chuyển khác được chấp nhận với xác suất $\exp(-\delta/t)$, δ là sự thay đổi của hàm chi phí và t là thông số điều khiển.

Ta biết rằng chất lượng lời giải phụ thuộc vào sự điều chỉnh thông số nhiệt độ - lịch làm nguội. Điều này được xác định bởi:

- Nhiệt độ ban đầu t_0
- Điều kiện dừng
- Hàm giảm nhiệt độ α
- Số lần lặp lại mỗi giá trị nhiệt độ, nrep.

Các giá trị là đặc trưng cho mỗi bài toán vì chúng phải được lựa chọn dựa trên trạng thái của bối cảnh lời giải.

f. Lịch làm nguội (cooling Schedule)

Lịch làm nguội của giải tinh mô phỏng luyện kim gồm 4 thành phần:

- nhiệt độ bắt đầu.
- nhiệt độ kết thúc.
- Hàm giải nhiệt độ.
- Số lần lặp trong tại mỗi nhiệt độ.

Nhiệt độ ban đầu:

- Nhiệt độ ban đầu phải đủ nóng để cho phép dịch chuyển đến hầu hết các trạng thái lân cận. Nếu không thì lời giải cuối cùng sẽ giống hoặc rất gần với lời giải ban đầu.
- Tuy nhiên nếu nhiệt độ ban đầu quá cao, quá trình tìm kiếm có thể chuyển đến bất cứ lời giải kế cận nào và do đó sẽ làm biến đổi quá trình tìm kiếm ngẫu nhiên. Khi đó, quá trình tìm kiếm sẽ là ngẫu nhiên cho đến khi nhiệt độ giảm xuống đủ để thực hiện giải thuật mô phỏng luyện kim.
- Vấn đề là xác định đúng nhiệt độ ban đầu. Hiện tại, chưa có phương pháp nào để tìm ra được một nhiệt độ bắt đầu phù hợp cho một bài toán tổng quát.
- Nếu ta biết khoảng cách lớn nhất (sự chênh lệch của hàm chi phí) giữa một lời giải lân cận và một lời giải khác thì thông tin này để tính nhiệt độ ban đầu.
- Một phương pháp khác là khởi động với nhiệt độ ban đầu rất cao và làm nguội một cách nhanh chóng cho đến khoảng 60% lời giải tệ nhất được chấp nhận. Đây sẽ là nhiệt độ ban đầu thật sự và bây giờ nó có thể được làm nguội chậm hơn.

Nhiệt độ kết thúc:

- Thông thường nhiệt độ được giảm cho tới khi nó đạt giá trị 0. Tuy nhiên, điều này có thể làm cho giải thuật chạy lâu hơn rất nhiều.
- Trong thực tế, không cần làm nhiệt độ đạt đến 0 bởi vì khi nó tiến gần 0, khả năng chấp nhận một bước chuyển xấu hơn gần như giống với khi nhiệt độ bằng 0.
- Do đó điều kiện dừng có thể là một nhiệt độ thấp phù hợp hoặc khi hệ thống “đóng băng” ở nhiệt độ hiện tại (không có một bước chuyển nào được chấp nhận)

Hàm giảm nhiệt độ:

- Một khi có nhiệt độ ban đầu và nhiệt độ kết thúc, ta cần chuyển từ nhiệt độ này qua nhiệt độ khác. Tức là, ta cần giảm nhiệt độ để cuối cùng có thể đạt đến điều kiện dừng.
- Một cách để làm giảm nhiệt độ là phương pháp tuyến tính:

$$T = t \cdot \alpha \text{ với } \alpha < 1$$

- Kinh nghiệm cho thấy ta nên chọn α khoảng 0.8 đến 0.99, giá trị càng cao thì kết quả thu được sẽ càng tốt. đương nhiên , nếu α có giá trị càng cao thì nhiệt độ càng lâu đạt đến điều kiện dừng.

Số vòng lặp tại mỗi giá trị nhiệt độ:

- Chúng ta cho phép thực hiện đủ số lần lặp ở mỗi nhiệt độ để hệ thống ổn định ở nhiệt độ đó. Đôi khi số vòng lặp tại mỗi nhiệt độ có thể là hàm mǔ so với kích thước bài toán. Bởi vì điều này là không thực tế, ta cần sắp xếp lại bằng cách:
 - Số vòng lặp lớn ở một vài giá trị nhiệt độ, hoặc
 - Số vòng lặp nhỏ ở nhiều giá trị nhiệt độ, hoặc
 - Một sự cân bằng giữa chúng
- Quyết định cuối cùng là ta phải có bao nhiêu vòng lặp tại mỗi nhiệt độ.
- Hằng số vòng lặp ở mỗi nhiệt độ là ngẫu nhiên.

g. những quyết định đặc trưng của bài toán:

Có một tập hợp các quyết định khác mà chúng ta cần phải thực hiện, các quyết định đó là đặc trưng cho mỗi bài toán cần phải giải quyết.

hàm chi phí:

- Với mỗi bài toán cụ thể thì sẽ phải có một vài cách để đo lường chất lượng lời giải. Khi định nghĩa hàm chi phí, chúng ta cần bảo đảm rằng nó miêu tả bài toán mà ta đang giải quyết.
- Một điều quan trọng nữa là hàm chi phí có thể được tính toán càng hiệu quả càng tốt bởi vì nó sẽ được tính ở mỗi lần lặp ở giải thuật. tuy nhiên, hàm chi phí thường là một nút thắt cổ chai (bottleneck) và đôi khi nó cần sử dụng các cách tính sau:
 - Ước lượng delta: ước lượng sự khác biệt giữa lời giải hiện tại và lời giải lân cận.
 - Ước lượng từng phần: hàm ước lượng đơn giản được sử dụng mà không đưa ra kết quả chính xác nhưng đưa ra một sự chỉ dẫn tốt về chất lượng lời giải.

Cấu trúc kế cận:

- Vấn đề là làm cách nào để chuyển từ một trạng thái tới một trạng thái khác. Nghĩa là ta phải xác định được tập kế cận. tức là ta đang ở một trạng thái xác định thì những trạng thái có thể chuyển đến là trạng thái nào.
- Một số kết quả nghiên cứu cho thấy cấu trúc kế cận nên đối xứng với nhau. Tức là nếu ta có thể chuyển từ trạng thái i sang trạng thái j thì cũng có thể chuyển từ trạng thái j sang trạng thái i.
- Một điều kiện yếu hơn cần phải có để nó chắc chắn hội tụ là: mỗi trạng thái phải có thể chuyển đến được từ mọi trạng thái khác.

h. Cải tiến quá trình thực hiện:

Ngay cả khi chúng ta tìm được một tập các thông số rất tốt cho giải thuật mô phỏng luyện kim, chúng ta cũng cần phải cải tiến quá trình thực thi của giải thuật bằng cách kết hợp với các kỹ thuật khác.

Khởi tạo:

Khi giải thuật mô phỏng luyện kim được bắt đầu, thông thường nó chọn một lời giải ngẫu nhiên, sau đó để quá trình luyện kim cải tiến trên lời giải đó. Tuy nhiên, tốt hơn ta nên bắt đầu với lời giải đã được xây dựng bằng heuristic.

VD: Trong bài toán sắp thời khóa biểu trường phổ thông trong luận văn này sử dụng BC_FC tạo lời giải ban đầu sau đó sẽ dùng Local Search (Tabu, WSAT) để tối ưu lời giải.

Kết hợp:

Ta có thể kết hợp hai giải thuật tìm kiếm cục bộ với nhau.

VD: Simulated Annealing và Shifting-Bottleneck được kết hợp với nhau để giải quyết bài toán sắp lịch bán hàng (job-shop scheduling). Simulated Annealing và Lin-Kernighan được kết hợp để giải quyết TSP

3.5.6. Giải thuật Scandinavian Workshop on Algorithm Theory (WSAT)

- Wsat giới hạn việc di chuyển lân cận bằng việc chọn ngẫu nhiên 1 ràng buộc vi phạm và sau đó chỉ xem xét giá trị của những biến mà ràng buộc thỏa mãn hoặc cải thiện.
- Với xác suất P, một biến được chọn ngẫu nhiên trong ràng buộc và giá trị của nó được thay đổi, mặc khác chi phí tốt nhất của di chuyển được chọn từ miền trị của các ràng buộc.
- WSAT chấp nhận tăng chi phí di chuyển dựa trên một ngưỡng xác suất và điều này tương tự như SA.
- Tuy nhiên có 3 sự khác nhau giữa WSAT và Simulated Annealing (SA):
 - Trong WSAT các biến chỉ liên quan đến những vi phạm ràng buộc được xem xét để thay đổi.
 - Giá trị xác suất P là cố định.
 - Hàm chi phí di chuyển trong WSAT trên cơ sở giảm thiểu số lượng vi phạm ràng buộc.
- Giải thuật

```
procedure GenerateLocalMoves (s, TotalMoves)
```

```
begin
```

```
    M'  $\leftarrow \emptyset$ ; BestCost  $\leftarrow \infty$ ;
```

```
    Chọn một biến bất kỳ vi phạm ràng buộc c;
```

```
    if p > (lấy ngẫu nhiên một số từ 0 đến 1) then
```

```
while M' = ∅ do
    begin
        chọn một biến bất kỳ trong Vars (c) và duy chuyển m cho biến này;
        if m cải thiện c then M' ← M' ∪ m
    end
    else
        for each Vnext ∈ Vars (c) do
            begin
                dcurr ← giá trị hiện tại of Vnext
                for each d ∈ Di | d ≠ dcurr do
                    begin
                        if f (s ⊕ m) ≤ BestCost and m cải thiện c then
                            begin
                                if f (s ⊕ m) < BestCost then
                                    begin
                                        BestCost ← f (s ⊕ m) ; M' ← ∅
                                    end;
                                    M' ← M' ∪ m
                                end
                            end
                        end
                    end
                end
            return M'
```

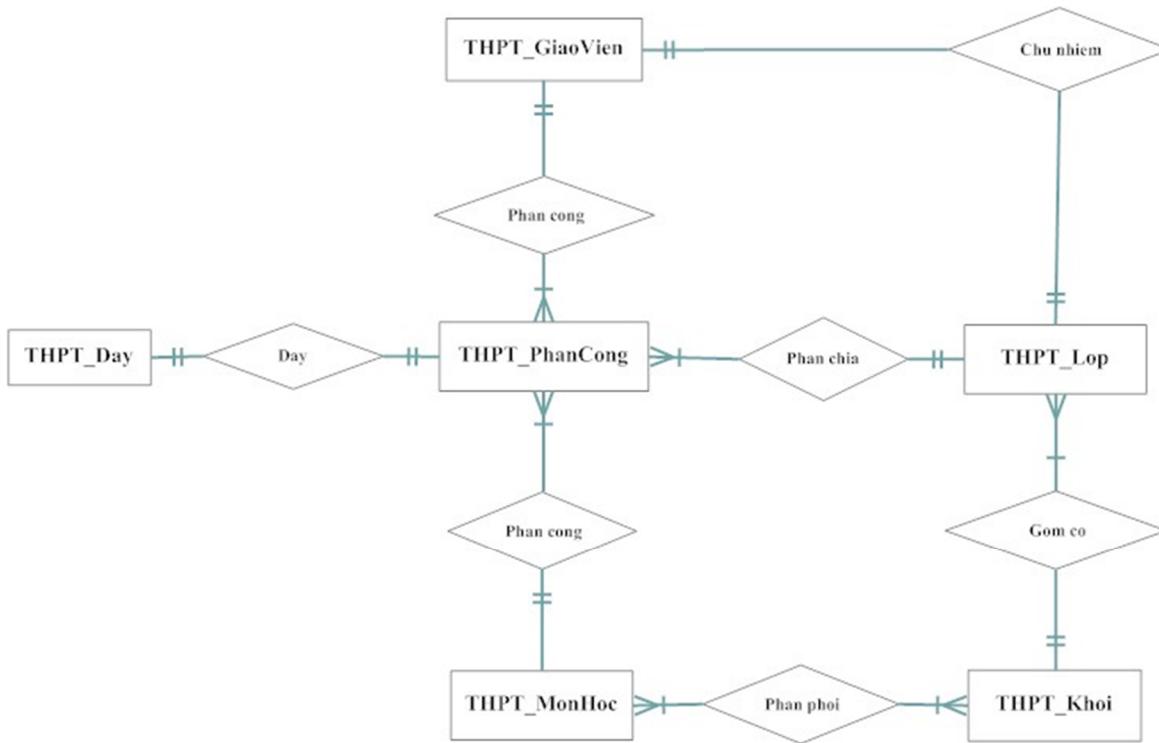
CHƯƠNG 4: THIẾT KẾ VÀ HIỆN THỰC CHƯƠNG TRÌNH

Chương này sẽ giới thiệu về cách thiết kế cơ sở dữ liệu, các cấu trúc dữ liệu chính dùng trong chương trình, cũng như các kỹ thuật được sử dụng để thiết kế và hiện thực. Nội dung của chương cũng sẽ giới thiệu các giải thuật sử dụng trong chương trình và cách hiện thực chúng. Giao diện và cách sử dụng chương trình cũng sẽ được giới thiệu.

4.1. THIẾT KẾ CƠ SỞ DỮ LIỆU

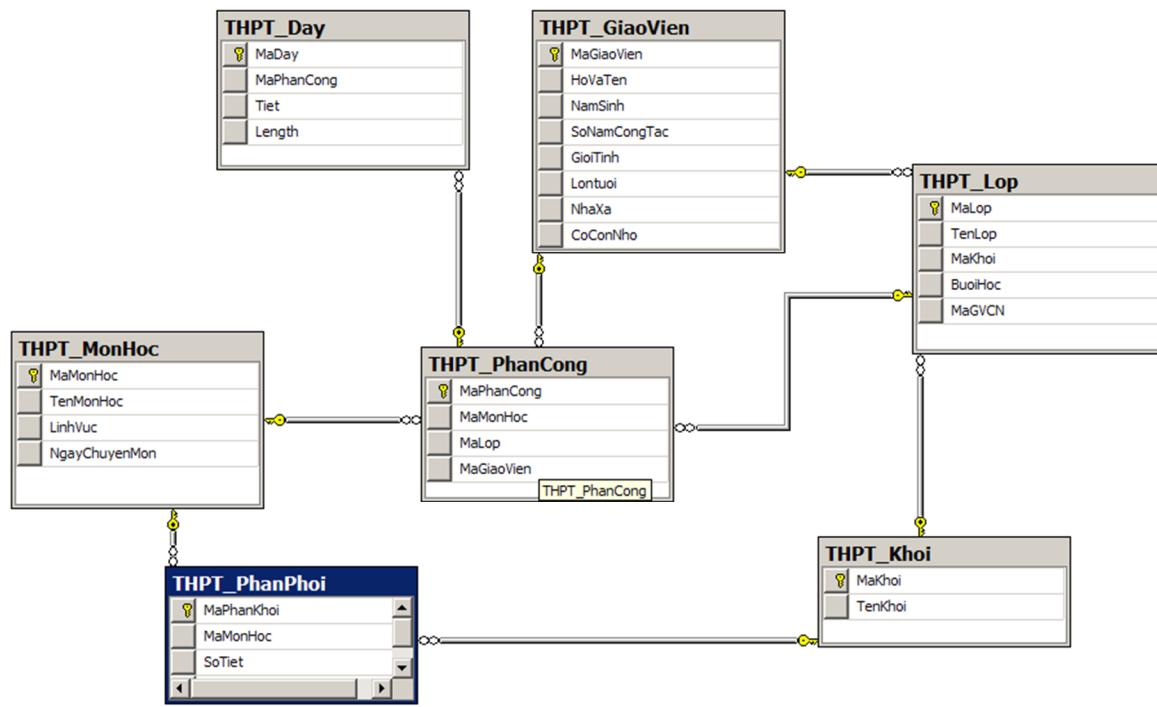
4.1.1. Sơ đồ quan hệ thực thể (ERD)

Chương trình có cơ sở dữ liệu khá đơn giản, mục này nêu ra các bảng, các thực thể, cũng như các quan hệ trong cơ sở dữ liệu.



Hình 4-1: Lược đồ quan hệ

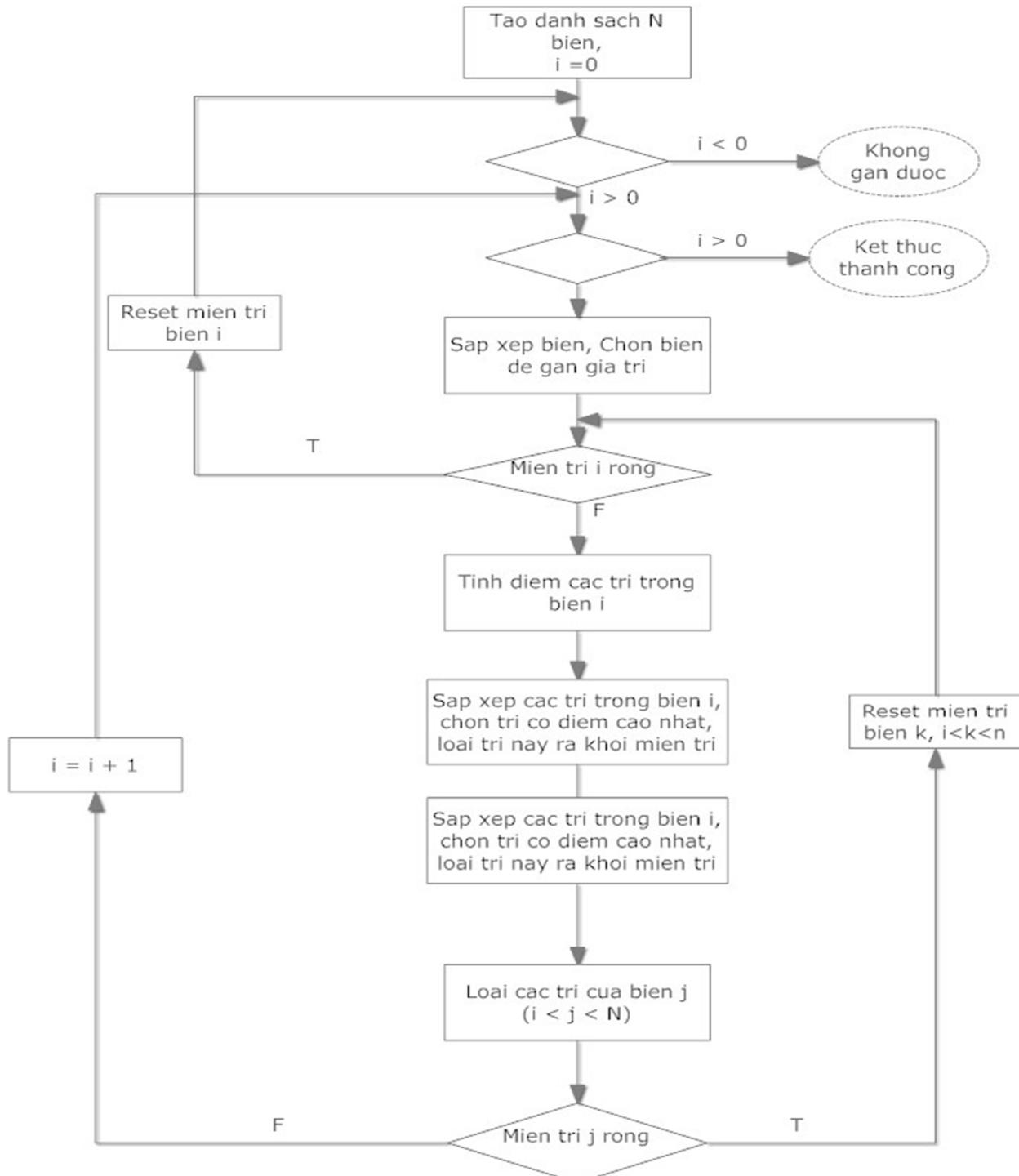
4.1.2. Sơ đồ cơ sở dữ liệu quan hệ trên MS SQL Server 2005



Hình 4-2: Các bản quan hệ trong MS SQL Server 2005

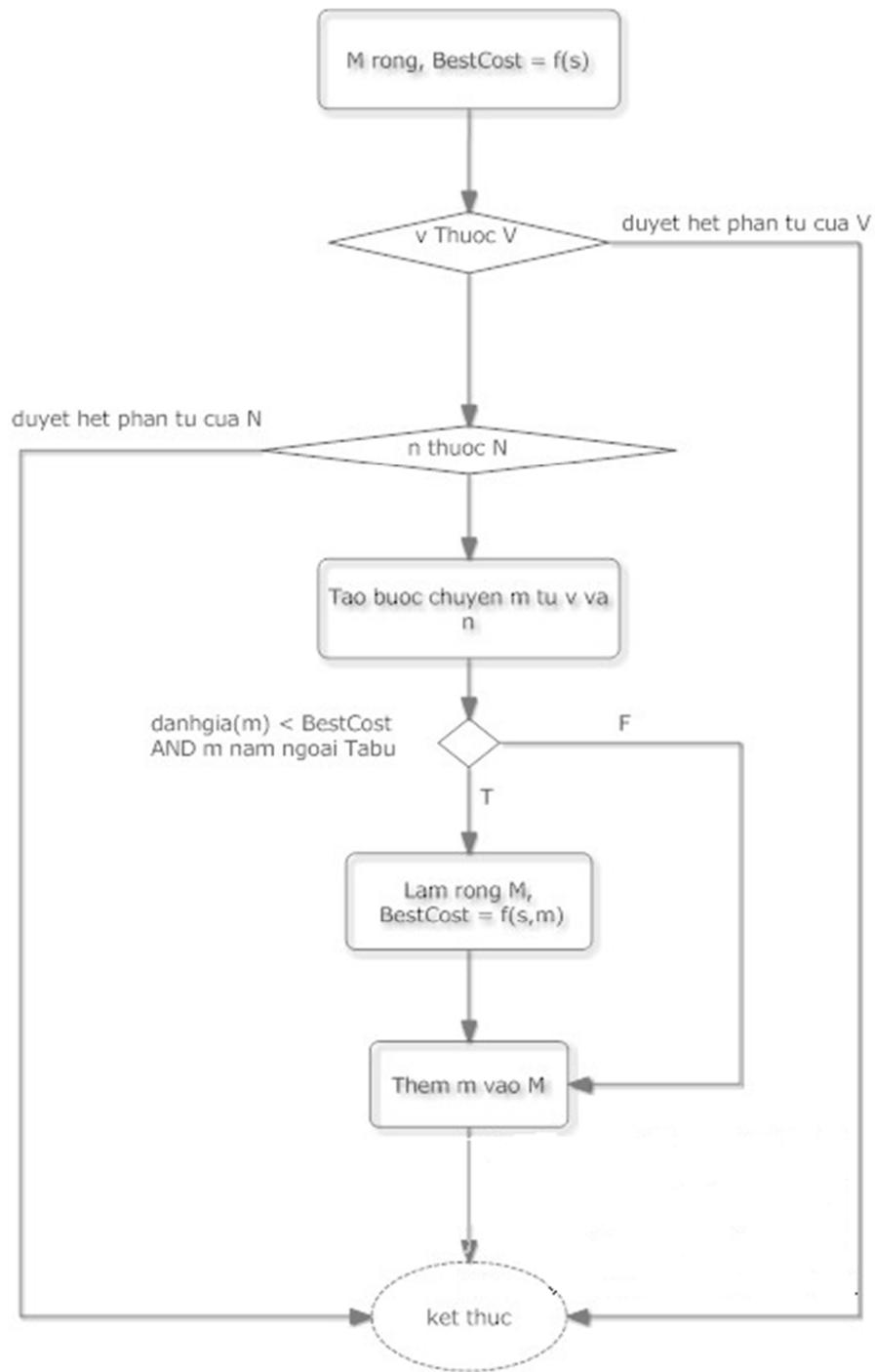
4.2. LUU ĐỒ GIẢI THUẬT TRONG CHƯƠNG TRÌNH

4.2.1. Lưu đồ giải thuật BC_FC



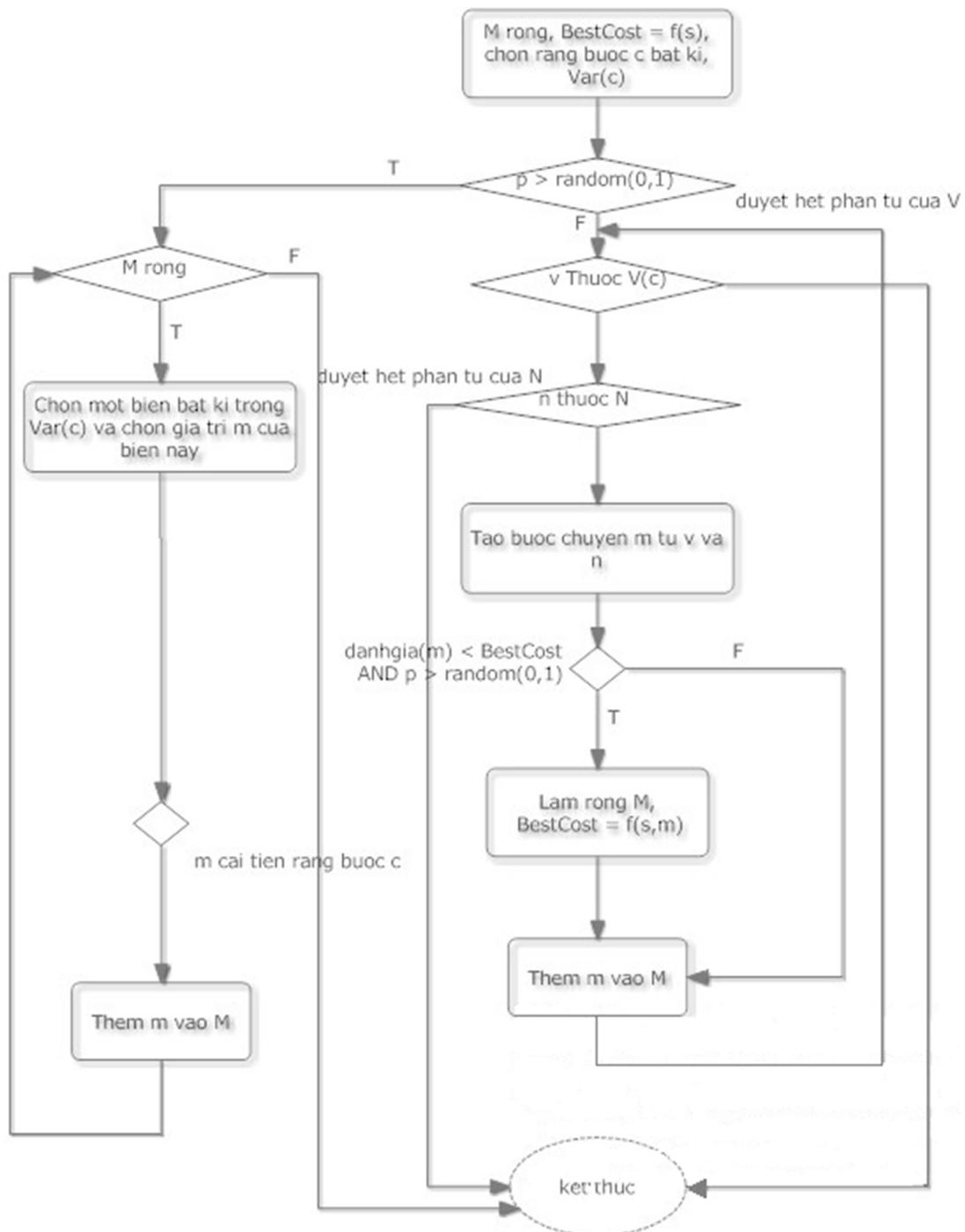
Hình 4-3: Lưu đồ giải thuật BC_FC

4.2.2. Lưu đồ giải thuật Tabu



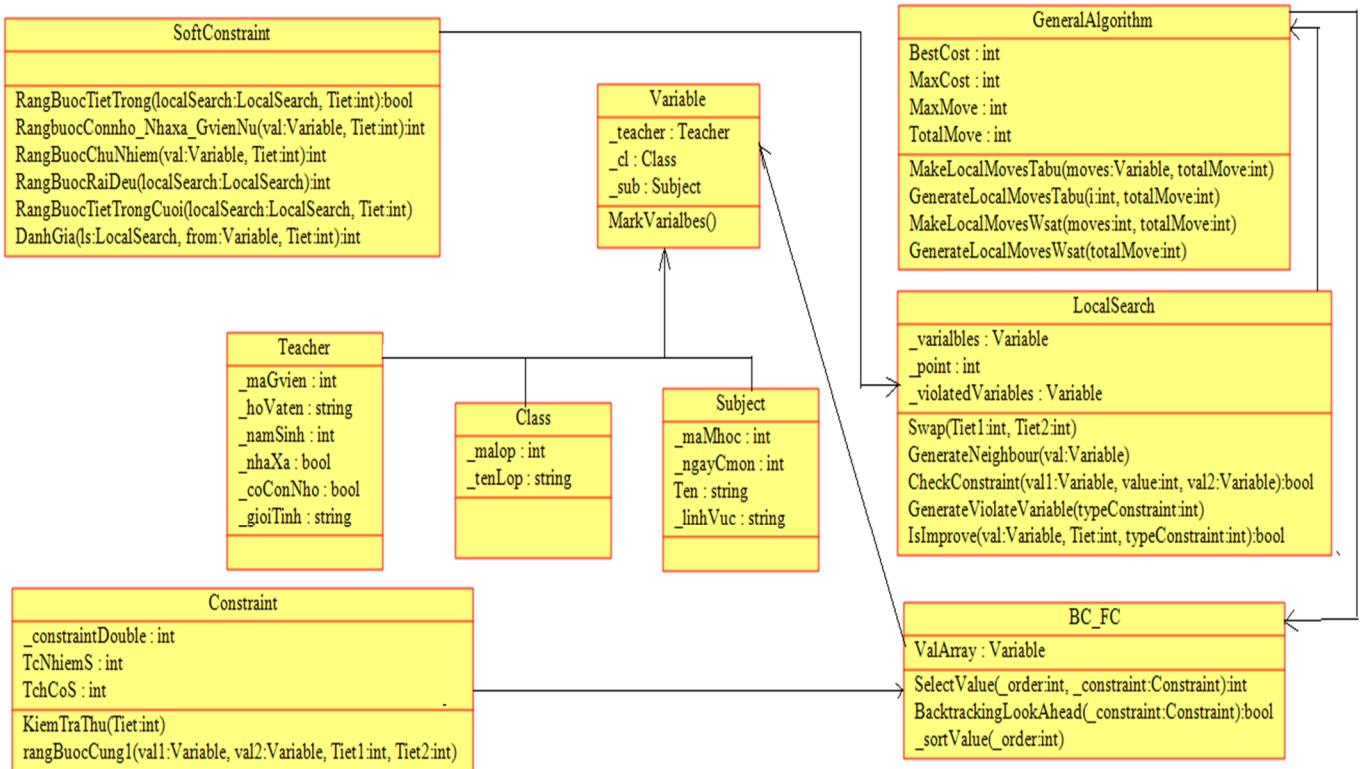
Hình 4-4: Lưu đồ giải thuật Tabu Search

4.2.3. Lưu đồ giải thuật WSAT Search



Hình 4-5: Lưu đồ giải thuật WSAT

4.3. SƠ ĐỒ LỚP CHƯƠNG TRÌNH DEMO (CLASS DIAGRAM)



Hình 4-6: Class Diagram

4.4. CÁCH SẮP XẾP ĐIỂM VÀ BIẾN TỐI ƯU GIẢI THUẬT BC_FC

a) Cách sắp xếp biến

Việc tìm cách sắp xếp biến cho giải thuật BC_FC cần phải dựa vào từng bài toán cụ thể, do đó khó có thể sắp xếp biến một cách tốt nhất, dựa vào kinh nghiệm và đặc điểm bài toán chúng em đưa ra cách tính điểm cho biến như sau:

$$D = 15 * L - G + 100 * M - 50 * C$$

D: điểm tính cho biến i.

L: số lần lặp.

G: số lớp giáo viên này dạy.

M: số phần tử của miền trị còn lại.

C: tổng số các trị thỏa mãn biến i.

Với cách tính điểm này, điểm càng nhỏ thì độ ưu tiên càng cao, vì vậy ưu tiên cho biến nào làm giảm miền trị các biến còn lại nhiều nhất. Sau đó đến biến nào có giáo viên dạy nhiều lớp nhất.

b) Cách sắp xếp trị

Cứ sau mỗi lần lập để tìm ra giá trị để gán ra trong miền trị, thì các trị trong miền trị được sắp xếp thứ tự, giá trị nào làm giảm miền trị của các biến tiếp theo càng nhỏ thì được ưu tiên sắp xếp trước.

Đây là cách tính mà chương trình chạy tương đối hiệu quả. Tuy nhiên việc đánh giá này vẫn chưa phải là cách đánh giá tốt nhất, vì thời gian cũng như điều kiện chúng em vẫn chưa so sánh với những cách đánh giá khác để tìm ra cách đánh giá hợp lý hơn nữa.

4.5. CÁCH TÍNH ĐIỂM HÀM ĐÁNH GIÁ TRONG LOCAL SEARCH

Giải thuật Tabu và WSAT dựa trên hàm chi phí, để di chuyển đến những lời giải lân cận tốt hơn.

a. Giải thuật tính hàm chi phí Cost ()

- Điểm phạt cho mỗi cách gán thể hiện mức độ ưu tiên của các ràng buộc, hay còn gọi là cách tính điểm phân cấp. Mục tiêu chính của cách tính này sẽ ưu tiên giải quyết các ràng buộc quan trọng trước.
- Để dễ phân biệt một cách hiệu quả ta sẽ tính điểm theo hàm mũ, tức là các mức độ chênh lệch điểm tùy theo hàm mũ.
- Điểm phạt cho mỗi cách gán như sau:
 - Nếu gán vào tiết giáo viên bận phạt 10000 điểm
 - Nếu tiết trống xen giữa các tiết học phạt 10000 điểm cho mỗi tiết trống.
 - Đối với ràng buộc rải đều tiết trống, tổng các tiết tổng trong một buổi sẽ bị phạt tổng số tiết trống trong ngày mũ 20.
 - Đối với ràng buộc chủ nhiệm, nếu giáo viên có tiết chủ nhiệm, nếu có tiết dạy ngày thứ 7 thì thưởng -10000 điểm.
 - Nếu gán vào tiết đầu:
 - Giáo viên nhà xa: phạt 100 điểm
 - Giáo viên có con nhỏ: phạt 1000 điểm
 - Giáo viên là phụ nữ: phạt 10 điểm
 - Nếu gán vào hai tiết cuối:
 - Giáo viên nhà xa: phạt 100 điểm
 - Giáo viên có con nhỏ: phạt 1000 điểm
 - Giáo viên là phụ nữ: phạt 100 điểm

Hàm **Danhgia ()** sẽ được tính như sau:

- Với mỗi cách gán đã có một điểm phạt theo đúng thang điểm như trên.

- Ngoài ra trong hàm **Danhgia()** còn ưu tiên điểm cho các ràng buộc như sau
 - Ràng buộc nhà xa, ràng buộc con nhỏ, ràng buộc lớn tuổi phân công vào tiết cuối hoặc tiết sáng thì hệ số ưu tiên là 10.
 - Ràng buộc phân công giáo viên có tiết chủ nhiệm dạy vào ngày thứ 7 hệ số ưu tiên là 12.
 - Ràng buộc tiết trống cuối, hệ số ưu tiên là 40.
- Cộng tất cả các giá trị trên tính được giá trị của hàm **Danhgia()**.

b. Điều kiện dừng của giải thuật

Đối với giải thuật Tabu Search và WSAT Search điều kiện dừng dựa vào Max Cost (giá trị hàm đánh giá lời giải bài toán đạt đến mức độ tối ưu nào đó) và Max Moves (số lần duy chuyển các giá trị của các biến).

Trong Luận Văn này chúng em dựa vào Max Moves là điều kiện dừng của giải thuật. Việc tìm kiếm, đánh giá Max Cost trong bài toán sắp xếp thời khóa biểu để đạt lời giải tối ưu rất là khó khăn vì số lượng biến rất lớn. Và phải dựa vào từng lớp, từng giáo viên.

c. Cách tìm lời giải lân cận

- Để có một lời giải lân cận cho thời khóa biểu hiện tại, ta hoán đổi các tiết học trong các Biến gán thỏa điều kiện:
- Xét trên thời khóa biểu trên một lớp.
- Các biến đổi không cùng một giáo viên.
- Các Biến gán cần đổi có thể chuyển đến tiết đích mà vẫn thõa các ràng buộc cứng của bài toán.
- Nếu tiết đích đã được gán thì Biến gán đó phải có thể chuyển về tiết của Biến gán cần đổi mà vẫn thõa các ràng buộc cứng của bài toán.

Hàm kiểm tra điều kiện hoán đổi của **Variable** gán và tiết cần chuyển đến.

Public bool CheckSwap (int perioD1, int perioD2)

perioD1: là tiết nguồn

perioD2 : là tiết đích

d. Các trường hợp các Tiết hoán vị được cho nhau

Quy ước:

T1: tiết đầu là tiết cần đổi.

T2: tiết tiếp theo của tiết cần đổi.

D1: tiết đầu là tiết đích.

D2: tiết đầu là tiết đích.

có: trong thời khóa biểu có tiết

trống: trong thời khó biếu còn trống

Các trường hợp đổi được gồm:

- ✓ TH1: T1 và T2 là tiết đôi.
D1 và D2 trống.

T1 và D1, T2 và D2 đổi được cho nhau.

- ✓ TH2: T1 và T2 là tiết đôi.
D1 trống, D2 có.

D2 và T2 đổi được cho nhau.

- ✓ TH3: T1 và T2 là tiết đôi.
D1 trống, D2 có.

D2 và T1 đổi được cho nhau.

- ✓ TH4: T1 và T2 là tiết đôi.
D1 có , D2 trống.

D1 và T1 đổi được cho nhau.

- ✓ TH5: T1 và T2 là tiết đôi.
D1 có, D2 trống.

D1 và T2 đổi được cho nhau.

- ✓ TH6: T1 và T2 là tiết đôi.
D1 và D2 là hai tiết đơn.

D2 và T2 đổi được cho nhau.

D1 và T1 đổi được cho nhau.

- ✓ TH7: T1 và T2 là tiết đôi.
D1 và D2 là hai tiết đơn.

D2 và T1 đổi được cho nhau.

D1 và T2 đổi được cho nhau.

- ✓ TH8: T1 và T2 là tiết đôi.
D1 và D2 là tiết đôi.

T1 và D1, T2 và D2 đổi được cho nhau.

- ✓ TH9: T1 và T2 là tiết đôi.
D1 trống, D2 trống.

T1 và D1, T2 và D2 đổi được cho nhau.

- ✓ TH10: T1 là tiết đơn.
D1 là tiết đơn.

T1 và D1 đổi được cho nhau.

- ✓ TH11: T1 có, T2 trống
D1 và D2 là tiết đôi

D1 và T1 đổi được cho nhau

- ✓ TH12: T1 và T2 là hai tiết đơn
T2 không bắt đầu tiết đôi
D1 và D2 là tiết đôi

D1 và T1 đổi được cho nhau
D2 và T2 đổi được cho nhau

Úng với mỗi trường ta sẽ có một hàm CheckSwap tương ứng

4.6. CHẠY DEMO, ĐÁNH GIÁ ĐỘ HIỆU QUẢ CỦA HAI GIẢI THUẬT TABU SEARCH VÀ WSAT SEARCH

4.6.1. Bảng phân công

Buổi Sáng	Toán	Lý	Hoa	Văn	Anh Văn	Sinh	Sử	Địa	GDCD	Kỹ Thuật	SHCN
10A8	Tuệ(6)	Sơn(4)	Mai(3)	Q.Hoa(4)	N.Thúy(3)	Quí(1)	Hữu(1)	Phuộc(1)	Trâm(1)	Kim(1)	Tuệ(1)
10A9	Hoa T(6)	Sơn(4)	Mai(3)	Q.Hoa(4)	N.Thúy(3)	Hằng(1)	Nguyệt(1)	Phuộc(1)	Trâm(1)	Kim(1)	Hoa T(1)
10A10	Dung(6)	Nhung(4)	Mai(3)	Hường(4)	Tú(5)	Hằng(1)	Hữu(1)	Manh(1)	Trâm(1)	Kim(1)	Mai(1)
10A11	Dung(6)	Huell(4)	Giang(3)	Lên V(4)	N.Thúy(3)	Hằng(1)	Hữu(1)	Manh(1)	Trâm(1)	Kim(1)	Dung(1)
10A12	Cuong(6)	Huell(4)	Giang(3)	Liên V(4)	Tú(5)	Hằng(1)	Nguyệt(1)	Manh(1)	Trâm(1)	Kim(1)	Lien V(1)
10A13	Tuệ(6)	Liên L(4)	Mai(3)	Tuòng(4)	Tâm A(3)	Hằng(1)	Nguyệt(1)	Phuộc(1)	Dung(1)	Kim(1)	Nguyệt(1)
11A1	Thảo(6)	Quang(4)	Mai(3)	H.Lan(4)	P.Lan(3)	Linh(1)	Mỹ(1)	Manh(2)	Dung(1)	Kim(1)	Quang(1)
11A2	Tâm T(6)	Sơn(4)	Giang(3)	Thúy(4)	P.Lan(3)	Linh(1)	Mỹ(1)	Manh(2)	Dung(1)	Kim(1)	Manh(1)
11A3	Tâm T(6)	Sơn(4)	Hồng(3)	Thúy(4)	P.Lan(3)	Linh(1)	Mỹ(1)	Manh(2)	Dung(1)	Kim(1)	Sơn(1)
11A4	Năng(6)	Quang(4)	Mai(3)	Cúc(4)	P.Lan(3)	Linh(1)	Mỹ(1)	Manh(2)	Dung(1)	Kim(1)	P.Lan(1)
11A5	Năng(6)	Quang(4)	Hồng(3)	Thanh(4)	P.Lan(3)	Linh(1)	Mỹ(1)	Phuộc(2)	Trâm(1)	Kim(1)	Phuộc(1)
11A6	Thảo(6)	Quỳnh(4)	Mai(3)	H.Lan(4)	Tâm A(3)	Quí(1)	Mỹ(1)	Manh(2)	Trâm(1)	Hoàng(1)	Quỳnh(1)
11A7	Cuong(6)	Sơn(4)	Hồng(3)	Hường(4)	Tâm A(3)	Quí(1)	Mỹ(1)	Phuộc(2)	Trâm(1)	Hoàng(1)	Hường(1)
11A8	Tâm T(6)	Quang(4)	Hồng(3)	Cúc(4)	P.Lan(3)	Quí(1)	Nguyệt(1)	Phuộc(2)	Trâm(1)	Hoàng(1)	Hồng(1)
11A9	Tâm T(6)	Quang(4)	Giang(3)	Thúy(4)	P.Lan(3)	Quí(1)	Nguyệt(1)	Phuộc(2)	Trâm(1)	Hoàng(1)	Tâm T(1)
11A10	Thảo(6)	Quang(4)	Giang(3)	Hường(4)	Tâm A(3)	Quí(1)	Mỹ(1)	Phuộc(2)	Dung(1)	Hoàng(1)	Giang(1)

Hình 4-7: Bảng phân công giảng dạy giáo viên

4.6.2. Chạy demo chương trình

Chương trình có ba phần chính gồm: Thông Tin Chung, Xếp Thời Khóa Biểu, Thông Kê

1. Thông Tin Chung

Gồm những thông tin về Lớp, Giáo viên, Môn học và cả Bảng Phân Công và Phân Phối để giúp việc xếp thời khóa biểu. Có thể thêm, xóa sửa cho phù hợp.



Hình 4-8: Giao diện chính của chương trình demo

2. Xếp Thời Khóa Biểu

Là phần quan trọng nhất của chương trình này. Ở phần này ta sẽ có hai lựa chọn xếp thời khóa biểu tương ứng cho hai giải thuật TabuSearch và WSAT. Sau khi xếp xong thời khóa biểu ta có thể xem thời khóa biểu cho từng lớp, từng giáo viên và có thể so sánh hai phương án khác nhau của cùng một thời khóa biểu của lớp. Ta cũng có thể xem được thời gian chạy của từng giải thuật.

Chương trình trước khi xếp:



Hình 4-9: Giao diện Phần trước khi sắp thời khóa biểu

Chương trình sau khi chạy:



Hình 4-10: Giao diện chương trình sau khi sắp xếp thời khóa biểu

4.7. SO SÁNH ĐÁNH GIÁ HIỆU QUẢ CỦA HAI GIẢI THUẬT

4.6.1 So sánh hiệu quả hai giải thuật dựa trên thực tế kết quả sắp xếp trong chương trình

Kết quả sau khi chạy chương trình sắp xếp thời khóa biểu trên hai giải thuật.



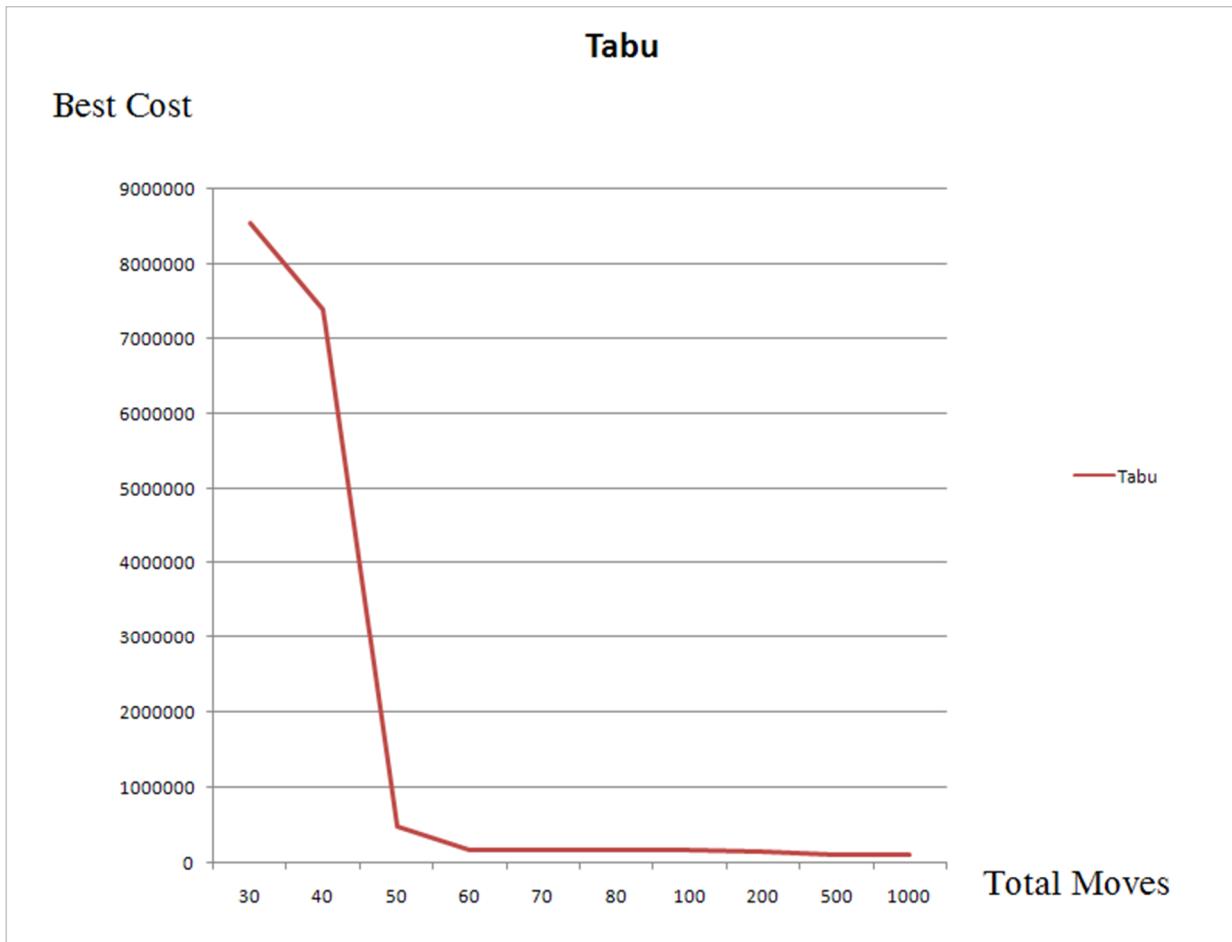
Hình 4-11: Thời khóa biểu sắp xếp theo lớp của hai giải thuật



Hình 4-12: Thời khóa biểu sắp xếp theo giáo viên của hai giải thuật

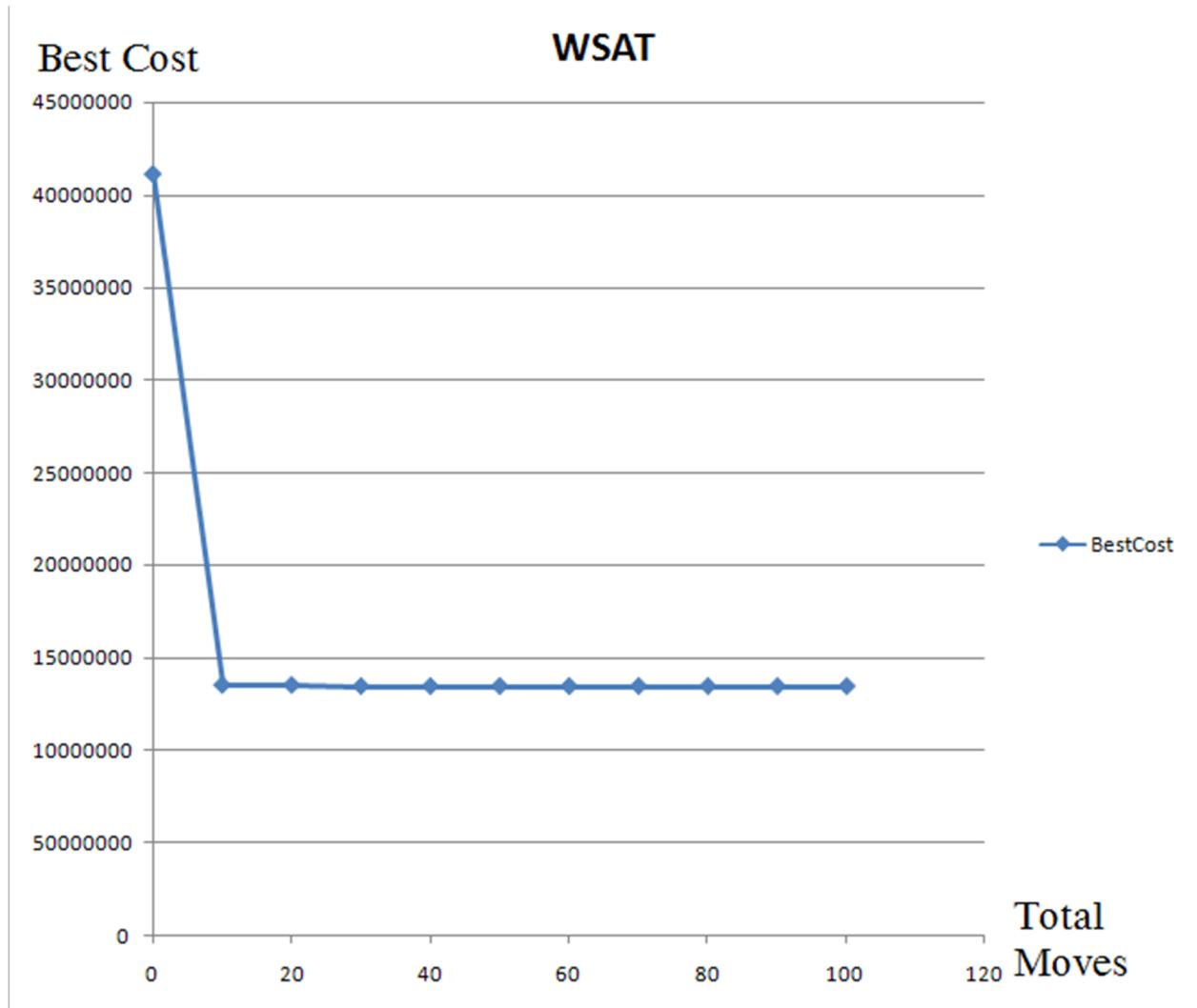
- Dựa vào kết quả thực tế của chương trình thì giải thuật Tabu Search chạy nhanh hơn rất nhiều so với giải thuật WSAT Search.
- Mặc dù giải thuật WSAT chạy thời gian lâu hơn, tuy nhiên độ hiệu quả cũng thấp hơn so với giải thuật Tabu Search. Trong hình 4.16 nhận thấy rằng giải thuật Tabu Search rải đều ngày dặm hơn so với WSAT Search.
- Kết luận: đối với bài toán sắp xếp thời khóa biểu, chúng ta nên dùng giải thuật Tabu Search hơn là WSAT Search cả về phương diện tối ưu lời giải lẫn thời gian chạy.

4.7.1. Biểu đồ đánh giá hiệu quả giải thuật Tabu Search



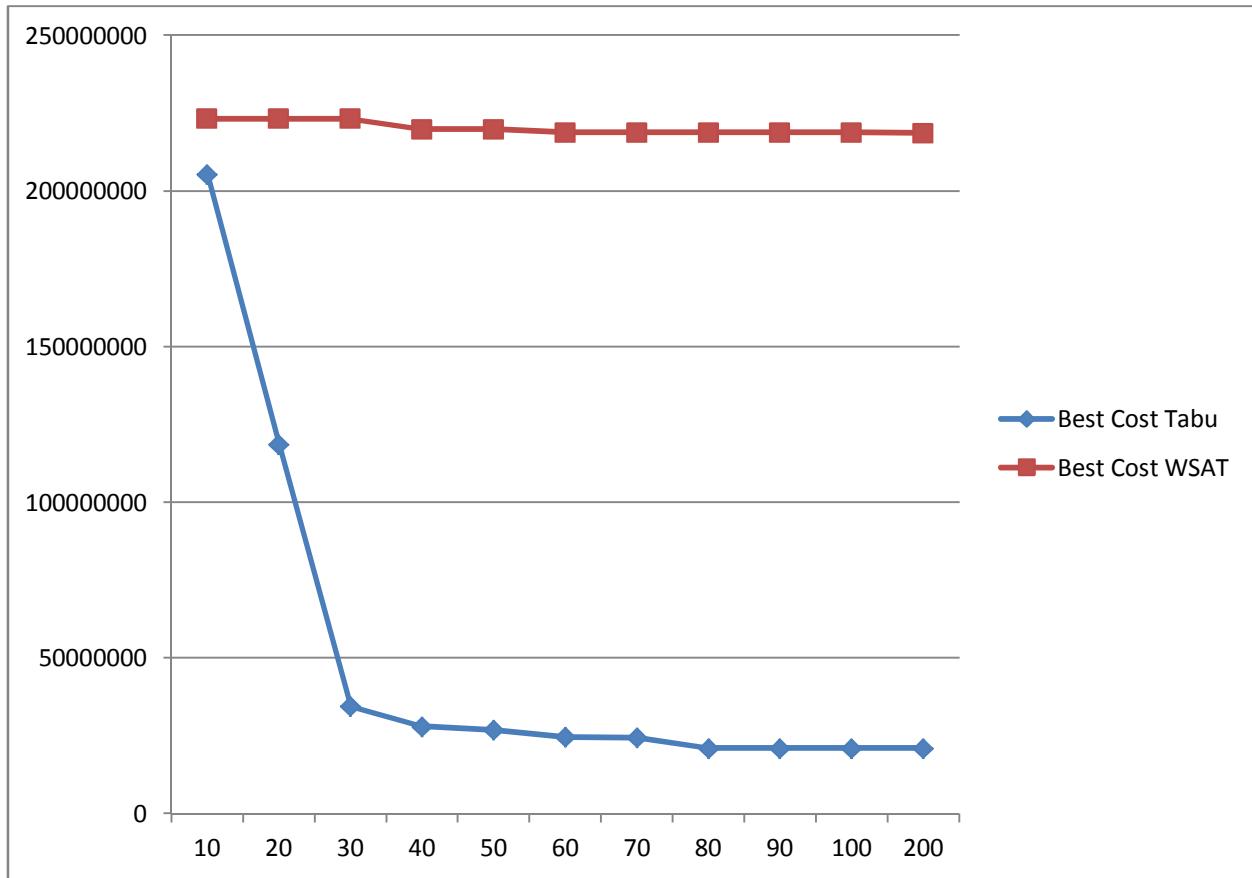
Hình 4-13: Biểu đồ đánh giá giải thuật Tabu Search

4.7.2. Biểu đồ đánh giá hiệu quả giải thuật WSAT Search



Hình 4-14: Biểu đồ đánh giá giải thuật WSAT Search

4.7.3. Biểu đồ đánh giá chung hai giải thuật



Hình 4-15: Biểu đồ đánh giá chung của hai giải thuật

4.7.4. So sánh đánh giá độ hiệu quả hai giải thuật

Giải thuật Tabu là một dạng của giải thuật leo đồi, thường gặp khó khăn trong một số bài toán khi bị kẹt tại chi phí cục bộ nhỏ nhất (hoặc lớn nhất) bởi vì nó luôn chọn bước chuyển trong không gian tìm kiếm có thể cải thiện hàm chi phí. Bởi vì khi không chọn được bước chuyển tốt hơn thì nó sẽ không bao giờ chuyển sang trạng thái mới, ngay cả khi chưa tìm được lời giải tối ưu toàn cục.

Không giống như Tabu. Giải thuật WSAT lựa chọn một bước chuyển ngẫu nhiên từ một quan hệ kế cận. Nếu bước chuyển tốt hơn thì WSAT luôn luôn chọn nó. Nếu bước chuyển xấu hơn thì WSAT sẽ chấp nhận nó dựa trên một xác suất nhất định. WSAT đã giải quyết được vấn đề của giải thuật Tabu vì nó sẽ không bị kẹt ở chi phí cục bộ nhỏ nhất bởi vì nó có thể chấp nhận một bước chuyển xấu hơn để thoát ra lời giải có chi phí cục bộ nhỏ nhất và tìm đến lời giải có chi phí cục bộ nhỏ nhất toàn cục.

Giải thuật tabu chủ yếu dựa vào độ dài của TabuList. Nếu độ dài này quá lớn sẽ làm cho nhiều bước chuyển bị loại bỏ làm ảnh hưởng đến chất lượng của lời giải và dễ

dàng dẫn đến tình trạng bị kẹt tại chi phí cục bộ nhỏ nhất. Nếu độ dài này quá lớn sẽ làm cho các bước chuyển có thể lập đi lập lại nhiều lần do đó ảnh hưởng đến thời gian tìm kiếm.

WSAT chủ yếu dựa vào xác xuất p cho trước. Phải chọn p thích hợp giải thuật mới chạy hiệu quả. Nếu p càng nhỏ thì xác xuất chấp nhận những lời giải mới xấu hơn lời giải hiện tại rất ít do đó việc thoát ra khỏi điểm tối ưu cục bộ sẽ gặp nhiều khó khăn. Nếu ta chọn p quá lớn thì ảnh hưởng đến tốc độ lời giải. Lúc này giải thuật sẽ chạy gần như là giải thuật vét cạn nên thời gian chạy giải thuật là rất lâu. Cần lưu ý là khi $p = 0$ giải thuật WSAT sẽ trở thành giải thuật leo đồi vì bỏ qua bước chấp nhận lời giải xấu hơn, khi $p = 1$ thì giải thuật sẽ trở thành giải thuật vét cạn.

Trong luận văn này, ứng với đề tài xếp thời khóa biểu cho trường cấp 3 thì em thấy dùng giải thuật TabuSearch sẽ hiệu quả hơn. Vì trong đây các tập lân cận tương đối nhiều, khi ta dịch chuyển một trạng thái thì tập lân cận của bài toán sẽ dịch chuyển rất ít. Nên vấn đề cực tiểu cục bộ rất khó xảy ra. Nếu ta dùng WSAT thì do tập lân cận quá lớn sẽ làm cho giải thuật chạy rất lâu. Sau đây là số liệu thống kê sau khi chạy giải thuật

Kết quả này được so sánh trên thực nghiệm của demo. Trên thực tế chúng em vẫn chưa tìm thấy bài báo nào so sánh hiệu quả của hai thuật toán Tabu và WSAT trong bài toán sắp xếp thời khóa biểu.

CHƯƠNG 5:

KẾT LUẬN

5.1. ĐÁNH GIÁ

Qua quá trình thực hiện đề tài, đã đạt được những kết quả nhất định. Do đây là đề tài thiêng về tìm hiểu và đánh giá kết quả tìm được nên nó chưa mang lại ứng dụng thực tế cao. Song với mục đích và yêu cầu của đề tài, luận văn đã đáp ứng khá tốt những yêu cầu đưa ra.

Với kết quả của luận văn sẽ cho ta những phương án chọn lựa để giải quyết bài toán xếp thời khóa biểu tốt nhất. Với thời gian chỉ có 4 tháng tìm hiểu các giải thuật lẩn hiện thực và giải quyết bài toán thời khóa biểu nên những kết quả nghiên cứu trong đề tài này chưa phải tốt nhất, cần có những thay đổi hợp lý để có thể đưa vào áp dụng thực tế hoặc phát triển thêm.

5.2. NỘI DUNG ĐÃ ĐƯỢC GIẢI QUYẾT

- Về lý thuyết:
 - Tìm hiểu lý thuyết về bài toán xếp thời khóa biểu ở trường THPT Phan Bội Châu - Tam Kỳ - Quảng Nam.
 - Tìm hiểu lý thuyết giải thuật BackTracking with Look Ahead, tìm hiểu những phương pháp để tối ưu hóa giải thuật BackTracking with Look Ahead.
 - Tìm hiểu giải thuật TabuSearch và giải thuật WSAT.
 - So sánh sự khác biệt giữa hai giải thuật TabuSearch và WSAT.
- Về hiện thực chương trình:
 - Hiện thực giải thuật BC_FC và ứng dụng những phương pháp tối ưu hóa BC_FC để giải quyết những ràng buộc cứng của bài toán.
 - Hiện thực hai giải thuật TabuSearch và WSAT để giải quyết ràng buộc mềm.
 - Chương trình có quản lý môn học, giáo viên và lớp để hỗ trợ cho việc xếp thời khóa biểu.
 - Chương trình hỗ trợ phần so sánh và thông kê những kết quả của hai giải thuật TabuSearch và WSAT.
- Kết luận:

- Đối với bài toán sắp xếp thời khóa biểu thì giải thuật Tabu Search tốt hơn giải thuật WSAT Search cả về phương diện thời gian và phương diện hợp lý kết quả bài toán sắp xếp thời khóa biểu.

Vì vậy trong việc xây dựng chương trình sắp xếp thời khóa biểu, sắp xếp lịch thi ta nên chọn giải thuật Tabu Search là phương án phù hợp.

PHỤ LỤC

THIẾT KẾ CẤU TRÚC DỮ LIỆU:

Phần này sẽ giới thiệu các cấu trúc dữ liệu chính được sử dụng trong chương trình, các cấu trúc này chủ yếu phụ vụ cho các giải thuật sử dụng trong chương trình.

Các cấu trúc dữ liệu dành cho việc hiện thực giải thuật BC_FC:

Cấu trúc lớp học:

```
Class Day {  
    PhanCong _phanCong;  
    int _tiet;  
    int _maDay;  
    string _monhoc;  
}
```

Cấu trúc Môn học:

```
Class Lop_Hoc {  
  
    int _malop;  
    int[] TuNhien;  
    int[] XaHoi;  
    string _tenlop;  
    KhoiLop _khoilop;  
    int maCN;  
}
```

Cấu trúc Giáo viên:

```
Class Giao_Vien {  
  
    int _maGvien;  
    string _hoVaTen;  
    bool _nhaXa;  
    bool _coConNho;  
    string _gioiTinh;  
}
```

Cấu trúc Phân công:

```
Class Phan_Cong {  
    Subject _subject;  
    Class _lop;  
    int _maPhanCong;  
    Teacher _teacher;  
}
```

Cấu trúc Biến của giải thuật BC_FC:

```
Class Bien {  
    Teacher _teacher;  
    Class _cl;  
    Subject _sub;  
    ArrayList _domain;  
    ArrayList _removers;  
    int _period;  
    int _index;  
    int _point;  
    void RemoveDomain (int period, int indexOfRemover) ;  
    void Reset (int remover) ;  
    int GetRemovedPeriod (int value) ;  
    void FixDomain (Constraint constraint) ;  
}
```

Cấu trúc của ràng buộc cung:

```
Class Rang_Buoc_Cung {  
    ArrayList _constraintDouble;  
    const int TcNhiemS = 25;  
    const int TcNHiemC = 55;  
    const int TchCoS = 1;  
    const int TchCoC = 35;  
    int KiemTraThu (int value) ;  
    bool RangBuocTabu (int value1, int value2) ;  
    bool RangBuoc1 (Variable variable, int value) ;  
    bool RangBuoc2 (Variable val1, Variable val2, int value1, int value2) ;
```

}

Cấu trúc của giải thuật BC_FC:

```
Class BC_FC {  
    _variables = new ArrayList () ;  
    PhanCongDb phancong = new PhanCongDb () ;  
    PhanPhoiDb phanphoi = new PhanPhoiDb () ;  
    void SortValue (int i) ;  
    void UpdatePoint (int i) ;  
    GetNumRemove (int index, int value, Constraint constraint)  
    int SelectValue (int i, Constraint constraint) ;  
    bool BacktrackingLookAhead (Constraint constraint) ;  
}
```

Hiện thực các phương thức chính trong các Class:

Phương thức loại các giá trị không thỏa ràng buộc cứng ra khỏi domain

```
public void RemoveDomain (int period, int indexOfRemover)  
{  
    _domain.Remove (period) ;  
    _removers.Add (indexOfRemover*100 + period) ;  
}
```

Phương thức trả về tất cả các miền trị bị remove bởi một biến

```
public void Reset (int remover)  
{  
    int i = _removers.Count - 1;  
  
    if (_removers.Count == 0)  
    {  
        return;  
    }  
  
    while (i >= 0 && GetRemover ( (int) _removers[i] ) >= remover)  
    {
```

```
        _domain.Add (GetRemovedPeriod ( (int) _removers[i--] ) ) ;  
    }  
  
    int firstIndex = i + 1;  
    _removers.RemoveRange (firstIndex, _removers.Count -firstIndex)  
;  
}
```

Phương thức loại tất cả các trị không thỏa ràng buộc cứng của tất cả các biến

```
public void FixDomain (Constraint constraint)  
{  
    for (int j = 0; j < _domain.Count; j++)  
    {  
        if (constraint.RangBuoc1 (this, (int)  
_domain[j]) == false)  
        {  
            _domain.RemoveAt (j--) ;  
        }  
    }  
}
```

Phương thức chọn trị trong giải thuật BC_FC

```
public int SelectValue (int i, Constraint constraint)  
{  
    Variable variable = (Variable) _variables[i];  
    int index = 0;  
    while (variable.Domain.Count != 0 && index <  
variable.Domain.Count)  
    {  
        int value = (int) variable.Domain[index];  
        variable.RemoveDomain ( (int) variable.Domain[index], i) ;  
        bool emptyDomain = false;  
        Variable v;  
        for (int k = i + 1; k < _variables.Count; k++)  
        {
```

```
        v = (Variable) _variables[k];
        int cout = 0;
        while (cout < v.Domain.Count)
        {
            int b = (int) v.Domain[cout];

            if (constraint.RangBuoc2 (variable, v, value, b)
==false)
            {
                v.RemoveDomain (b, i) ;
                cout--;
            }
            cout++;
        }
        if (v.Domain.Count == 0)
            emptyDomain = true;
    }
    if (emptyDomain)
    {
        for (int k = _variables.Count-1 ; k > i; k--)
        {
            v = (Variable) _variables[k];
            v.Reset (i) ;
        }
    }
    else
        return value;

        index++;
    }
    return 0;
}
```

Phương thức hiện thực giải thuật BC_FC

```
public bool BacktrackingLookAhead (Constraint constraint)
```

```
{  
    int i = 0;  
    Variable v;  
    PointComparer point = new PointComparer () ;  
    while (i >= 0 && i < _variables.Count)  
    {  
        int value = SelectValue (i, constraint) ;  
        Variable var = ( Variable) _variables[i] ) ;  
        var.Period = value;  
        if (value == 0) // vi is null  
        {  
            i--;  
            for (int k = _variables.Count-1; k > i; k--)  
            {  
                v = ( Variable) _variables[k];  
                v.Period = 0;  
                v.Reset (i) ;  
            }  
        }  
        else  
        {  
            UpdatePoint (i) ;  
            _variables.Sort (i + 1, _variables.Count - i -  
1, point) ;  
            i++;  
            if (i < _variables.Count)  
                SortValue (i) ;  
        }  
    }  
  
    if (i == -1)  
        return false;  
  
    ClassComparer classComparer = new ClassComparer () ;
```

```
    _variables.Sort (classComparer) ;
    return true;
}
```

Các phương thức tối ưu giải thuật BC_FC

- Phương thức sắp xếp trị

```
private void SortValue (int i)
{
    ValueComparer comparer = new ValueComparer (i, this) ;
    Variable v = (Variable) _variables[i];
    v.Domain.Sort (comparer) ;
}
```

- Phương thức sắp xếp các biến

```
public void MarkVariable ()
{
    PhanCongDb pc = new PhanCongDb () ;
    int numClass = pc.GetNumClass (_teacher.MaGvien) ;
    _point = 15*_periodLoop + 100*_domain.Count -
    numClass - 50*_lengthPeriod;
}
```

Cấu trúc giải thuật Local Search

Cấu trúc lớp ràng buộc mềm:

```
class SoftConstraint
{
    public int Rangbuoc12 (Variable v, int period)
    public int RangBuoc20 (Variable v, int period
        public bool RangBuocTietTrong (LocalSearch
localSearch, int period)
        public int RangBuocRaiDeu (LocalSearch localSearch)
        public int RangBuocTietTrongCuoi (LocalSearch ls,
int period)
        public int DanhGia (LocalSearch ls, Variable fr, int
period)
}
```

Cấu trúc lớp Tìm kiếm cục bộ

```
public class LocalSearch
{
    private ArrayList _variables
    private int _maLop
    private int _maCn
    private int _point
    private ArrayList _violatedVariables;
    public Variable GetVariable (int period)
    public bool CheckSwap (int period1, int period2)
    public void Swap (int period1, int period2)
    public void GenerateNeighbour (Variable v)
        private bool CheckConstraint (Variable v1, int value, Variable v2)
        private int GetConstraintPoint (int typeConstraint)
    public bool IsImprove (int move, int typeConstraint)
        public bool IsImprove (Variable variable, int period, int typeOfContrainst)
}
```

Cấu trúc hiện thực giải thuật Tabu Search và WSAT Search

```
public class GeneralAlgorithm
{
    private ArrayList _localsearchs

    public int BestCost
    private const int MaxCost = 150000;
    private const int MaxMove = 100;
    private int point = 0;
    private double temp = 0;
        private int _totalMove;
    private int DanhGia (int move)
        public GeneralAlgorithm (Constraint constraint, Bcfc bcf)
}
```

```
#region Tabu Data
private ArrayList _tabuList;
private Bcfc _bcfc;
#endregion

#region WSAT Data
private double p = 0.12;
private ArrayList _violate;
#endregion

#region Tabu function
public ArrayList GenerateLocalMovesTabu (int i, ref int totalMove)
{
    public void MakeLocalMovesTabu (ArrayList moves, ref int totalMove)
    {
        public void LocalSearchTabu ()
    }
}

#region WSAT function
public ArrayList GenerateLocalMovesWsat (ref int totalMove)
{
    public void MakeLocalMovesWsat (ArrayList moves, ref int totalMove)
    {
        public void LocalSearchWsat ()
        GenerateViolatedLocalsearch (int typeConstraint)
    }
}
```

Các phương thức chính trong giải thuật Local Search

Tabu Search

```
public ArrayList GenerateLocalMovesTabu (int i, ref int totalMove)
{
    LocalSearch ls = (LocalSearch) _localsearchs[i];
    ArrayList moves = new ArrayList () ;
    int danhgia;
```

```
        foreach (Variable variable in ls.Variables)
    {
        ls.GenerateNeighbour (variable) ;
        int n = variable.Period;
        foreach (int m in variable.Neighbour)
        {
            int move = i * 10000 + n * 100 + m;
            danhgia = DanhGia (move) ;
            if (!_tabuList.Contains (move) && danhgia <= BestCost)
            {
                if (danhgia < BestCost)
                {
                    BestCost = danhgia;
                    moves.Clear () ;
                }
                moves.Add (move) ;
            }
        }
    }

    return moves;
}

public void MakeLocalMovesTabu (ArrayList moves, ref int totalMove)
{
    totalMove = totalMove + 1;
    int move = (int) moves[0];
    int index = move / 10000;
    int n = (move % 10000) / 100;
    int m = (move % 10000) % 100;
    LocalSearch ls = (LocalSearch) _localsearchs[index];
    ls.Swap (n, m) ;
```

```
_tabuList.Add (move) ;
if (_tabuList.Count == 20)
{
    _tabuList.RemoveAt (0) ;
}
}

public void LocalSearchTabu ()
{
    int totalMove = 0;
    _thongKe2 = new ArrayList () ;
    Random random = new Random () ;
    while (BestCost > MaxCost && totalMove < MaxMove)
    {
        int index = random.Next (0, _localsrchs.Count) ;
        ArrayList moves = GenerateLocalMovesTabu (index, ref totalMove) ;
        if (moves.Count != 0)
        {
            MakeLocalMovesTabu (moves, ref totalMove) ;
            _totalMove = totalMove;
            if ( (totalMove <= 100 && totalMove%10 == 0) || (totalMove > 100 && totalMove%100 == 0) )
            {
                Thongke tk = new Thongke () ;
                tk.TotalMove = totalMove;
                tk.BestCost = BestCost;
            }
        }
    }
}
```

WSAT Search

```
public ArrayList GenerateLocalMovesWsat (ref int totalMove)
{
    LocalSearch ls;
```

```
ArrayList moves = new ArrayList () ;
int danhgia;
Random random = new Random () ;
int typeConstraint = random.Next (1, 5) ;
temp = random.NextDouble () ;
GenerateViolatedLocalsearch (typeConstraint) ;

if (p > temp)
{
    while (moves.Count == 0)
    {
        if (_violate.Count == 0)
            break;
        int index = random.Next (0, _violate.Count) ;
        Variable v = (Variable) _violate[index];

        ls = SearchLs (v) ;
        ls.GenerateNeighbour (v) ;
        int neighbourIndex = random.Next (0,
v.Neighbour.Count) ;
        if (v.Neighbour.Count == 0)
            break;

        int m = (int)
v.Neighbour[neighbourIndex];
        int n = v.Period;
        int move = _localsearchs.IndexOf (ls) *10000 +
n*100 + m;

        if (ls.IsImprove (v, m, typeConstraint) )
        {
            moves.Add (move) ;
        }
    }
}
```

```
        _violate.Remove (v) ;
    }
}
else
{
    foreach (Variable variable in _violate)
    {
        ls = SearchLs (variable) ;
        ls.GenerateNeighbour (variable) ;
        int n = variable.Period;
        foreach (int m in variable.Neighbour)
        {
            int move = _localsearchs.IndexOf (ls) * 10000 + n *
100 + m;
            danhgia = DanhGia (move) ;

            if (danhgia <= BestCost && ls.IsImprove (variable, m,
typeConstraint) )
            {
                if (danhgia < BestCost)
                {
                    BestCost = danhgia;
                    moves.Clear () ;
                }
                moves.Add (move) ;
            }
        }
    }

    return moves;
}
```

```
public void MakeLocalMovesWsat (ArrayList moves, ref int totalMove)
{
    //move
    Random random = new Random () ;
    int mv = random.Next (0, moves.Count) ;
    int move = (int) moves[mv];
    int danhgia = DanhGia (move) ;
    if (danhgia <= point || p > temp)
    {
        int index = move / 10000;
        int n = (move % 10000) / 100;
        int m = (move % 10000) % 100;
        LocalSearch ls = (LocalSearch) _localsearchs[index];
        ls.Swap (n, m) ;
        point = danhgia;
    }
    totalMove = totalMove + 1;
}

public void LocalSearchWsat ()
{
    int totalMove = 0;
    Random random = new Random () ;
    _thongKe2 = new ArrayList () ;
    while (BestCost > MaxCost && totalMove < MaxMove)
    {
        int index = random.Next (0, _localsearchs.Count) ;
        ArrayList moves = GenerateLocalMovesWsat (ref
totalMove) ;
        if (moves.Count != 0)
        {
            MakeLocalMovesWsat (moves, ref totalMove) ;
            _totalMove = totalMove;
        }
    }
}
```

```
if (totalMove%10 == 0)
{
    Thongke tk = new Thongke () ;
    tk.TotalMove = totalMove;
    tk.BestCost = BestCost;

    _thongKe2.Add (tk) ;
}

}
```

TÀI LIỆU THAM KHẢO

- [1] Thầy Dương Tuấn Anh. *Tài liệu về Local Search*. Bài giảng cao học. Khoa khoa học và kỹ thuật máy tính trường Đại Học Bách Khoa, TP Hồ Chí Minh. 2010.
- [2] Đinh Phan Thúy Tâm, Nguyễn Thanh Huyền. *Ứng dụng giải thuật Tô Màu Đồ Thị và Mô Phỏng Luyện Kim để giải quyết bài toán xếp thời khóa biểu ở trường phổ thông*. Luận văn tốt nghiệp. Khoa Công nghệ thông tin trường Đại Học Bách Khoa, TP.Hồ Chí Minh, 2005.
- [3] Mai Hoàng Huy Huân. *Ứng dụng lập trình ràng buộc và tìm kiếm tabu để xếp thời khóa biểu trường phổ thông*. Luận văn tốt nghiệp. Khoa Công nghệ thông tin trường Đại Học Bách Khoa, TP.Hồ Chí Minh, 2005.
- [4] Asim YarKhan, Jack J.Dongarra. *Experiments with Scheduling Using Simulated Annealing in a Grid Environment*. Computer Science Department University of Tennessee, 2002.