**Ảnh có chứa hình vuông

Mô tả được tạo tự động**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🕯✡🕮🕮✡🕯**

****

**GVHD: ThS. Lê Minh Tân**

**Đề tài**

**ỨNG DỤNG THUẬT TOÁN MIN-CONFLICTS ĐỂ GIẢI QUYẾT BÀI TOÁN SẮP XẾP CHỖ NGỒI CHO HỌC SINH NÓI CHUYỆN TRONG LỚP**

**MÔN HỌC: TRÍ TUỆ NHÂN TẠO**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Nhóm sinh viên thực hiện :*** | |  | |  | | |
| Hoàng Mai Hiếu | 21110882 | |  | |  |
| Trương Nguyễn Thùy Trang | 21110691 | |  | |  |
| Đặng Công Tuấn | 21110709 | |  | |  |
| Nguyễn Văn Tín | 21110931 | |  | |  |
| Nguyễn Thành Lộc | 21110897 | |  | |  |
| Trần Hoàng Phúc  Nguyễn Hoàng Phương  Mai Đình Hồng Sơn | 21110606  21110609  21110465 | |  | |  |

**GVHD: ThS. Lê Minh Tân**

**TP Hồ Chí Minh , tháng 5 năm 2023**

**LỜI CẢM ƠN**

Lời đầu tiên, chúng em muốn gửi lời cảm ơn sâu sắc đến thầy Lê Minh Tân vì đã chịu trách nhiệm hướng dẫn và hỗ trợ tôi trong quá trình nghiên cứu và viết bài về Min-conflicts algorithm - một thuật toán tìm kiếm cục bộ cho các bài toán ràng buộc (CSPs).

Quá trình nghiên cứu này đã giúp tôi hiểu rõ hơn về những khía cạnh quan trọng của lý thuyết và thực tiễn trong lĩnh vực trí tuệ nhân tạo và học máy, cũng như rèn luyện kỹ năng tư duy logic và sáng tạo.

Thầy đã cung cấp cho tôi những kiến thức và kinh nghiệm quý báu về Min-conflicts algorithm, cũng như hướng dẫn chúng em trong quá trình tìm hiểu và nghiên cứu thuật toán này. Những đóng góp và sự hỗ trợ của thầy thực sự đã giúp tôi hoàn thành bài viết này một cách tốt nhất.

Một lần nữa, chúng em xin chân thành cảm ơn thầy Lê Minh Tân vì sự hỗ trợ và giúp đỡ của thầy trong quá trình nghiên cứu và viết bài về Min-conflicts algorithm. Kính chúc thầy sức khỏe, hạnh phúc thành công trên con đường sự nghiệp giảng dạy.

**BẢNG PHÂN CÔNG THÀNH VIÊN THAM GIA VIẾT BÁO CÁO CUỐI KỲ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | MSSV | Họ và tên | Nội dung công việc | Tỉ lệ hoàn thành | Điểm |
| 1 | 21110882 | Hoàng Mai Hiếu | Làm nội dung, viết báo cáo, thuyết trình | 100% |  |
| 2 | 21110691 | Trương Nguyễn Thùy Trang | Làm nội dung, viết báo cáo, thiết kế slide | 100% |  |
| 3 | 21110709 | Đặng Công Tuấn | Làm nội dung, viết báo cáo, code demo | 100% |  |
| 4 | 21110931 | Nguyễn Văn Tín | Làm nội dung, viết báo cáo, thuyết trình | 100% |  |
| 5 | 21110897 | Nguyễn Thành Lộc | Làm nội dung, viết báo cáo, code demo | 100% |  |
| 6 | 21110606 | Trần Hoàng Phúc | Làm nội dung, viết báo cáo | 100% |  |
| 7 | 21110609 | Nguyễn Hoàng Phương | Làm nội dung, viết báo cáo, code demo | 100% |  |
| 8 | 21110465 | Mai Đình Hồng Sơn | Làm nội dung, viết báo cáo | 100% |  |

**Nhóm sinh viên thực hiện:** Nhóm 10 – Lớp Thứ 7, tiết 09 – 12

**Nhận xét của giảng viên**

*Ngày tháng năm 2023*

*Giảng viên chấm điểm*

**ĐÁNH GIÁ**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Nội dung | **Điểm** | Cơ sở đánh giá |
| Báo cáo cuối kỳ | | | |
| 1 | Hình thức: Bìa, lời cảm ơn, lời mở đầu, mục lục, canh lề, đúng font, trang đánh giá & chấm điểm, chú thích, danh mục tài liệu tham khảo, ít lỗi chính tả,... | …/2 |  |
| 2 | Trình bày đầy đủ nội dung. Lựa chọn ứng dụng phù hợp với đề tài. | …/3 |  |
| 3 | Chạy được ít nhất 1 demo phù hợp nội dung báo cáo. | …/2 |  |
| 4 | Lập luận thuyết phục. Đưa ra đồ thị, thống kê, phân tích, kết luận đánh giá đa chiều. | .../1.5 |  |
| 5 | Tham khảo nhiều tài liệu nghiên cứu. Các tài liệu được đánh số, chú thích rõ ràng. | .../1.5 |  |
|  | Tổng: | .../10 |  |
|  | **Trưởng nhóm ký, ghi họ tên** | **Giảng viên ký, ghi họ tên** | |

Ghi chú: ……………………………………………………………………………..

**MỤC LỤC**

[**NỘI DUNG** 2](#_Toc134914005)

[**I. GIỚI THIỆU** 2](#_Toc134914006)

[1. Khái quát về Constraint Satisfaction Problems (CSPs) 2](#_Toc134914007)

[2. Local Search 2](#_Toc134914008)

[3. Tại sao cần sử dụng Local Search for CSPs 3](#_Toc134914009)

[4. Giới thiệu về Min-conflicts algorithm 4](#_Toc134914010)

[**II. THUẬT TOÁN MIN-CONFLICTS** 6](#_Toc134914011)

[1. Các thành phần chính của thuật toán Min-conflicts 6](#_Toc134914012)

[2. Cơ chế hoạt động của thuật toán 7](#_Toc134914013)

[3. Giải thích chi tiết các bước của thuật toán 8](#_Toc134914014)

[**III. ỨNG DỤNG CỦA MIN-CONFLICTS ALGORITHM** 10](#_Toc134914015)

[1. Giải bài toán sắp xếp học sinh nói chuyện trong lớp 10](#_Toc134914016)

[2. Các ứng dụng thực tiễn khác của Min-conflicts algorithm 12](#_Toc134914017)

[**IV. ĐÁNH GIÁ THUẬT TOÁN** 13](#_Toc134914018)

[**V. ƯU ĐIỂM VÀ HẠN CHẾ CỦA THUẬT TOÁN** 14](#_Toc134914019)

[1. Ưu điểm 14](#_Toc134914020)

[2. Hạn chế 15](#_Toc134914021)

[3. Các phương pháp giải quyết các hạn chế 16](#_Toc134914022)

[**VI. CÁC BIẾN THỂ CỦA THUẬT TOÁN MIN-CONFLICTS** 17](#_Toc134914023)

[1. Hybrid Min-conflicts 17](#_Toc134914024)

[2. Min-conflicts-Random-Restart 17](#_Toc134914025)

[3. Min-conflicts Tabu Search 18](#_Toc134914026)

[4. Min-conflicts with Constraint Weighting 19](#_Toc134914027)

[**TỔNG KẾT** 20](#_Toc134914028)

[1. Kết luận về Min-conflicts Algorithm 20](#_Toc134914029)

[2. Hướng phát triển và ứng dụng trong tương lai 20](#_Toc134914030)

[**TÀI LIỆU THAM KHẢO** 22](#_Toc134914031)

**LỜI MỞ ĐẦU**

Các bài toán ràng buộc (CSPs) là một trong những lĩnh vực quan trọng và thú vị nhất của trí tuệ nhân tạo, đóng vai trò quan trọng trong nhiều lĩnh vực ứng dụng như lập lịch, lập kế hoạch, điều phối và tổ chức các tài nguyên, và nhiều hơn nữa. Tuy nhiên, việc giải quyết các bài toán này thường rất phức tạp và đòi hỏi nhiều thời gian và tài nguyên. Vì vậy, các nhà nghiên cứu đã phát triển nhiều thuật toán để giải quyết các bài toán này, trong đó Min-conflicts algorithm được coi là một trong những thuật toán tìm kiếm cục bộ hiệu quả nhất.

Min-conflicts algorithm tập trung vào việc tối ưu hóa các ràng buộc còn lại, thay vì giải quyết toàn bộ bài toán. Thuật toán này sử dụng một chiến lược tìm kiếm địa phương để tìm ra giải pháp tốt nhất cho các bài toán ràng buộc.

Trong bài viết này, chúng ta sẽ tìm hiểu chi tiết về Min-conflicts algorithm - thuật toán tìm kiếm cục bộ cho các bài toán ràng buộc (CSPs). Bài viết sẽ bao gồm lịch sử ra đời, sự phát triển, các thành phần chính, cách hoạt động của thuật toán, ứng dụng của nó trong các bài toán thực tế, cùng với một số ví dụ minh họa và phân tích độ phức tạp của thuật toán.

Hy vọng rằng bài viết sẽ giúp bạn hiểu rõ hơn về Min-conflicts algorithm và cách áp dụng nó để giải quyết các bài toán ràng buộc của riêng bạn.

# **NỘI DUNG**

## **I. GIỚI THIỆU**

### **1. Khái quát về Constraint Satisfaction Problems (CSPs)**

Constraint Satisfaction Problems (CSPs) một bài toán thỏa mãn ràng buộc trong lĩnh vực trí tuệ nhân tạo và toán học. Một bài toán CSP được mô tả bằng một tập các biến, mỗi biến có miền giá trị và một tập các ràng buộc áp đặt hạn chế đối với các giá trị mà một biến có thể nhận trong miền giá trị của chúng. Mục tiêu của CSPs là tìm ra một bộ giá trị để gán cho các biến sao cho tất cả các ràng buộc được thỏa mãn.[1]

CSPs được ứng dụng rộng rãi trong các lĩnh vực như lập lịch, quản lý tài nguyên, mạng máy tính, thiết kế mạch,…

Một số thuật toán phổ biến được sử dụng để giải quyết CSPs như tìm kiếm cục bộ (Local Search), thuật toán quay lui (Backtracking Search), lan truyền ràng buộc (Constraint Propagation), các thuật toán học tăng cường,…

### **2. Local Search**

Local Search (tìm kiếm cục bộ) là một phương pháp tìm kiếm giải pháp cho các bài toán tối ưu, trong đó ta cần tìm giải pháp tốt nhất cho một tập hữu hạn các giải pháp có thể.

Trong bài toán thỏa mãn ràng buộc, các thuật toán tìm kiếm cục bộ hoạt động dựa trên nguyên tắc cải thiện, lặp đi lặp lại phép gán cho các biến cho đến khi tất cả ràng buộc được thỏa mãn. Phương pháp này duyệt qua các giải pháp láng giềng và chọn ra giải pháp tốt hơn, do đó có tên là tìm kiếm cục bộ.

Tất cả các thuật toán tìm kiếm cục bộ đều dùng một hàm để đánh giá chất lượng của phép gán, ví dụ như số ràng buộc vi phạm trong phép gán. Giá trị này được gọi là chi phí của phép gán. Mục tiêu của tìm kiếm cục bộ là tìm ra một phương án có chi phí tối thiểu, đó là một giải pháp nếu có tồn tại.

Có hai loại thuật toán tìm kiếm cục bộ. Đầu tiên là các thuật toán tham lam (greedy) hay không ngẫu nhiên (non-randomized). Các thuật toán này tiến hành thay đổi phương án hiện tại bằng cách luôn cố gắng giảm (hoặc ít nhất là không tăng) chi phí của nó. Vấn đề chính của các thuật toán này là sự hiện diện có thể có của các cao nguyên (plateaus), là những vùng mà không có phương án nào làm giảm chi phí. Nhóm thuật toán tìm kiếm cục bộ thứ hai được phát triển để giải quyết vấn đề này. Các thuật toán này thoát khỏi các cao nguyên bằng cách thực hiện các bước di chuyển ngẫu nhiên, vì thế được gọi là thuật toán tìm kiếm cục bộ ngẫu nhiên.[2]

Trong các thuật toán tìm kiếm cục bộ, thuật toán Min-conflicts là một thuật toán tìm kiếm cục bộ ngẫu nhiên được sử dụng để giải quyết các bài toán thỏa mãn ràng buộc.

### **3. Tại sao cần sử dụng Local Search for CSPs**

Thuật toán Local Search là một kỹ thuật giải quyết bài toán tối ưu bằng cách tìm kiếm trạng thái tốt nhất trong không gian tìm kiếm của bài toán. Trong trường hợp của Constraint Satisfaction Problems (CSPs), bài toán đặt ra là tìm một phân bố giá trị cho tất cả các biến sao cho thỏa mãn tất cả các ràng buộc. Vì vấn đề CSPs có thể có không gian tìm kiếm rất lớn, sử dụng thuật toán Local Search để tìm kiếm các giải pháp tiềm năng là một phương pháp hữu hiệu và có thể tìm được kết quả tốt trong thời gian ngắn.

Các thuật toán Local Search thường bao gồm việc xây dựng một trạng thái bắt đầu ngẫu nhiên, sau đó thực hiện các bước tìm kiếm để cải thiện trạng thái hiện tại, thông qua việc di chuyển tới các trạng thái kế tiếp mà có độ hàm mục tiêu tốt hơn. Trong trường hợp của CSPs, các bước tìm kiếm này thường liên quan đến việc chọn một biến và gán giá trị mới cho biến đó.

Sử dụng Local Search cho CSPs có nhiều lợi ích, bao gồm khả năng tìm kiếm các giải pháp trong thời gian hữu hạn, khả năng xử lý các ràng buộc mạnh và yếu, và khả năng xử lý các ràng buộc với các biến không phải là nhị phân. Tuy nhiên, các thuật toán Local Search cũng có thể gặp phải các vấn đề như rơi vào các địa điểm cực tiểu hoặc đạt đến các plateau, nơi các trạng thái mới không cải thiện được hàm mục tiêu. Do đó, các kỹ thuật như Tabu Search và Constraint Weighting cũng được sử dụng để giải quyết các vấn đề này.

### **4. Giới thiệu về Min-conflicts algorithm**

**4.1. Lịch sử ra đời của thuật toán**

Trước đây, các bài toán thỏa mãn ràng buộc đã được đề cập đến trong lĩnh vực trí tuệ nhân tạo và toán học, nhưng mãi đến đầu những năm 1990, quy trình giải các bài toán CSP lớn mới được hệ thống hóa dưới dạng thuật toán. Ban đầu, Mark Johnson của Space Telescope Science Institute (STScI) đã tìm kiếm một phương pháp để lên lịch quan sát trên kính viễn vọng không gian Hubble. Trong quá trình này, ông ấy đã tạo ra một mạng lưới thần kinh có khả năng giải bài toán N-quân hậu (cho 1024 quân hậu). Steven Minton và Andy Philips đã phân tích thuật toán mạng thần kinh và chia nó thành 2 giai đoạn: (1) phép gán ban đầu sử dụng một thuật toán tham lam và (2) là giai đoạn giảm thiểu xung đột (sau này được gọi là Min-conflicts). Sau đó, một bài báo được công bố ở AAAI-90 [4], Philip Lard đã cung cấp phân tích toán học của thuật toán.

Từ đó, Mark Johnston và nhân viên của STScI đã sử dụng thuật toán Min-conflicts để lập lịch cho thời gian thực hiện thí nghiệm của các nhà thiên văn học trên vệ tinh Hubble.[3]

**4.2. Sự phát triển của thuật toán**

Thuật toán Min-conflicts đã trải qua nhiều sự phát triển để trở thành một công cụ quan trọng trong giải quyết CSPs. Ban đầu, thuật toán này được giới thiệu như một phương pháp đơn giản để giải quyết các bài toán liên quan đến ràng buộc. Tuy nhiên, với tính đơn giản, dễ hiểu và hiệu quả trong thực tế, thuật toán Min-conflicts đã được nghiên cứu và phát triển rộng rãi trong giải quyết các bài toán thực tế như lập lịch, tối ưu hóa quy trình sản xuất, quản lý tài nguyên và nhiều lĩnh vực khác.

Thuật toán Min-conflicts được phát triển để giải quyết các bài toán thỏa mãn ràng buộc (CSPs) liên quan đến một số lượng lớn các ràng buộc và biến.

Các phương pháp tìm kiếm CSP truyền thống có thể rất kém hiệu quả, đặc biệt khi có nhiều ràng buộc và biến liên quan. Thuật toán Min-conflicts được phát triển để cung cấp một cách hiệu quả hơn để giải quyết các loại vấn đề này.

Nhìn chung, thuật toán Min-conflicts được phát triển để cung cấp một cách hiệu quả và hiệu quả hơn để giải quyết các CSP, đặc biệt là những bài toán có số lượng lớn các ràng buộc và biến. Việc tập trung vào tìm kiếm cục bộ và giảm thiểu xung đột đã khiến nó trở thành một phương pháp phổ biến để giải quyết nhiều vấn đề tối ưu hóa.[4]

**4.3. Ý tưởng của thuật toán**

Thuật toán Min-conflicts là một thuật toán tìm kiếm hoặc phương pháp heuristic để giải các bài toán thỏa mãn ràng buộc.

Đúng như tên gọi của thuật toán, thuật toán Min-conflicts dựa trên ý tưởng giảm thiểu xung đột vi phạm giữa các biến. Thuật toán tìm ra giải pháp bằng cách giải quyết từng xung đột thông qua tìm kiếm và thay thế giá trị của biến sao cho số lượng xung đột là ít nhất. Thuật toán tiếp tục lặp lại quá trình này cho đến khi không còn xung đột nào trong bài toán hoặc đạt đến một số giới hạn vòng lặp tối đa được xác định trước đó. Mục tiêu của thuật toán là tìm ra một giải pháp thỏa mãn tất cả các ràng buộc trong bài toán.

**II. THUẬT TOÁN MIN-CONFLICTS**

1. **Các thành phần chính của thuật toán Min-conflicts**

**1.1. Trạng thái khởi tạo**

Trạng thái khởi tạo là một phần của bài toán được tối ưu hóa, ví dụ như một lịch làm việc, một bảng xếp hạng hoặc một bảng phân phối công việc. Trạng thái khởi tạo có thể được tạo ra bằng cách sử dụng một số phương pháp khác nhau, chẳng hạn như ngẫu nhiên hoặc sử dụng heuristics.

**1.2. Danh sách các ràng buộc**

Danh sách các ràng buộc là tập hợp các điều kiện phải được đáp ứng trong trạng thái tối ưu. Ví dụ, trong bài toán lập lịch, mỗi công việc cần được thực hiện trong một khoảng thời gian nhất định và không thể chạy đồng thời với công việc khác.

**1.3. Hàm lựa chọn biến**

Hàm lựa chọn biến được sử dụng để chọn một biến nào đó trong trạng thái tối ưu để thay đổi giá trị của nó. Hàm này thường được xây dựng để chọn biến có nhiều xung đột nhất với các ràng buộc hiện tại.

**1.4. Hàm lựa chọn giá trị mới**

Hàm lựa chọn giá trị mới được sử dụng để chọn giá trị mới cho biến được chọn bởi hàm lựa chọn biến. Hàm này thường được xây dựng để chọn giá trị mới là giá trị làm giảm số lượng xung đột với các ràng buộc hiện tại.

**1.5. Hàm đếm số lượng xung đột**

Hàm đếm số lượng xung đột được sử dụng để đếm số lượng ràng buộc không được đáp ứng trong trạng thái hiện tại. Hàm này thường được sử dụng để đánh giá chất lượng của một trạng thái.

**1.6. Số lần lặp tối đa**

Đây là giới hạn số lần lặp tối đa để tìm kiếm giải pháp. Nếu không tìm thấy giải pháp trong số lần lặp này, thuật toán sẽ dừng lại và trả về kết quả tốt nhất tìm được.

1. **Cơ chế hoạt động của thuật toán**

**2.1. Khởi tạo giá trị ban đầu**

Trong quá trình khởi tạo, ta cần xác định giá trị ban đầu cho các biến trong bài toán. Việc khởi tạo giá trị ban đầu cho các biến của bài toán có thể được chọn ngẫu nhiên.  Lựa chọn ngẫu nhiên có thể giúp cho thuật toán Min-conflicts khám phá được nhiều lời giải tiềm năng và tăng tính thực tế của bài toán.

**2.2. Lựa chọn biến và giá trị**

Các biến sẽ được chọn ngẫu nhiên để thực hiện cập nhật giá trị. Khi đã chọn được biến , thuật toán sẽ tìm kiếm giá trị mới cho biến này bằng cách chọn giá trị có số lượng xung đột (conflict) thấp nhất với các biến khác.

**2.3. Kiểm tra ràng buộc và cập nhật trạng thái**

Sau khi đã chọn được giá trị mới cho biến, ta cần thực hiện kiểm tra ràng buộc để đảm bảo giá trị mới không vi phạm các ràng buộc của bài toán. Nếu giá trị mới không vi phạm các ràng buộc, ta cần lưu giá trị mới vào các biến có liên quan để đưa ra một trạng thái mới hợp lệ cho bài toán. Sau khi cập nhật trạng thái, ta cần kiểm tra xem trạng thái mới có đáp ứng được kết quả tốt hơn so với trạng thái trước đó hay không.

**2.4. Kiểm tra điều kiện dừng**

Thực hiện đến khi đạt được giá trị tối ưu hoặc dừng lại sau khi đã lặp lại số bước tối đa cho phép. Việc đạt được giá trị tối ưu đồng nghĩa với việc tất cả các ràng buộc trong bài toán đều được đáp ứng.

1. **Giải thích chi tiết các bước của thuật toán**

**hàm**MIN-CONFLICTS(*csp, max\_steps*)**trả về**một giải pháphoặc *thất bại*

**đầu vào*:*** *csp:* một vấn đề thỏa mãn ràng buộc

*max\_steps:* số lần lặp tối đa được phép

*current* ←trạng thái khởi tạo ban đầu cho các biến trong csp

**vòng lặp từ***i* ← *1***đến***max\_steps***thực hiện**

**nếu***current* là một giải pháp cho *csp***thì trả về***current*

**gán** *var* ←Một biến xung đột được chọn ngẫu nhiên từ *csp*.VARIABLES

**gán***value* ← giá trị *v* cho biến *var*  để giảm thiểu xung đột trong hàm CONFLICTS(csp, var, v, current)

**gán***var* ← *value* trong *current*

trả về *thất bại*

Trong thuật toán Min-conflicts, nhận đầu vào là hai biến csp và max\_steps trong đó:

csp: viết tắt của “constraint satisfaction problem” - một bài toán trong lý thuyết ràng buộc, mô tả việc tìm kiếm giải pháp thỏa mãn các ràng buộc. Bài toán này thường được mô tả dưới dạng một tập hợp các biến cần được gán giá trị sao cho các ràng buộc giữa chúng được thỏa mãn. Ví dụ về một bài toán hạn chế ràng buộc có thể là lịch thi cho học sinh, trong đó các ràng buộc bao gồm các môn học phải được thi vào các thời điểm cụ thể, mỗi môn học chỉ thi một lần, và không thể có hai môn học cùng thi vào cùng thời điểm.

max\_steps: là một số nguyên dương đại diện cho số bước tối đa cho phép được thực hiện trong thuật toán. Nếu số bước này được vượt qua mà không tìm được phân bổ nào thỏa mãn các ràng buộc của bài toán csp, thuật toán sẽ kết thúc và trả về kết quả không thành công (failure).

1. Ở bước khởi đầu, ta khởi tạo giá trị ban đầu của “current” một cách ngẫu nhiên. Mỗi biến trong “csp” được gán một giá trị ban đầu sao cho tất cả các ràng buộc đều được thỏa mãn.

2. Sau đó vòng lặp từ 1 đến max\_steps, thuật toán thực hiện lặp lại các bước sau đây "max\_steps" lần:

+Kiểm tra xem biến “current” có phải là một giải pháp cho bài toán hay không. Nếu “current” thỏa mãn tất cả các ràng buộc thì “current” là một giải pháp, thuật toán sẽ trả về “current” và kết thúc.

+Nếu giá trị “current” không phải là một lời giải, ngẫu nhiên chọn một biến “var” có xung đột trong tập các phần tử đang bị xung đột.

+Tìm một giá trị “v” cho biến “var” để giảm thiểu số lượng xung đột với các biến khác. Điều này được thực hiện bằng cách sử dụng hàm CONFLICTS, hàm có chức năng tính toán số lượng xung đột mà giá trị “v” của biến “var” trong “current” gây ra với các biến khác.

+Sau đó, giá trị mới trong giải pháp hiện tại (current) được gán cho biến “var”. Nghĩa là cập nhật lại trạng thái hiện tại của các biến, theo giá trị mới được chọn cho biến “var”. Lặp lại các bước trên cho đến khi đạt tới số lần lặp tối đa hoặc tìm được một lời giải.

3. Khi đã đạt tới số lần lặp tối đa mà không tìm được lời giải thì thuật toán trả về thất bại(failure).

**III. ỨNG DỤNG CỦA MIN-CONFLICTS ALGORITHM**

1. **Giải bài toán sắp xếp học sinh nói chuyện trong lớp**

**1.1. Giới thiệu về bài toán**

Bài toán: Cô Thu là GVCN của lớp 1C, hôm nay lớp có một tiết học dự giờ, vấn đề của cô gặp phải là có một vài bạn học sinh trong lớp thường xuyên nói chuyện với nhau trong giờ học, vì thế trong tiết học dự giờ ngày hôm nay cô phải sắp xếp lại lớp học sao cho các học sinh nói chuyện với nhau không ngồi cạnh nhau.

**1.2. Giải quyết bài toán bằng thuật toán Min-conflicts**

**Ảnh có chứa biểu đồ

Mô tả được tạo tự động**

Trong bài toán sắp xếp các học sinh nói chuyện trong lớp, chúng ta có thể định nghĩa các ràng buộc như sau:

Biến(X): là vị trí chỗ ngồi của các học sinh.

Miền giá trị(D): là tập các vị trí chỗ ngồi trong 1 lớp học vuông được khởi tạo

Ràng Buộc(C): không có một học sinh nào ngồi cạnh nhau

Ta có thể áp dụng giải thuật Min-conflicts giải quyết bài toán trên như sau:

B1: Khởi tạo các giá trị biến bằng cách chọn ngẫu nhiên chỗ ngồi cho từng học sinh (chỉ 1 học sinh ngồi trên 1 hàng ngang ) và kích thước lớp học.

B2: Lặp lại các bước sau cho đến khi giải pháp tối ưu được tìm thấy hoặc đạt đến một số giới hạn:

* B2.1: Kiểm tra xem biến “current\_state” có phải là một giải pháp cho bài toán hay không. Nếu “current\_state” thỏa mãn tất cả các ràng buộc thì “current\_state” là một giải pháp , sẽ trả về “current\_state” và kết thúc.
* B2.2: Nếu giá trị “current\_state” không phải là một lời giải,  chọn ngẫu nhiên một biến “var” có xung đột trong conflicted\_vars.
* B2.3: Tiếp tục duyệt qua tất cả các giá trị (value) của biến được chọn(var) ở bước 2.22 để tìm giá trị tối thiểu của num\_conflict (số lượng xung đột) và lưu giá trị đó vào min\_conflict và giá trị tương ứng vào min\_value.
* B2.4 : Sau khi vòng lặp ở bước 2.3 kết thúc, giá trị tốt nhất của biến var được gán vào self.current\_state để tạo ra trạng thái mới. Lặp lại các bước trên cho đến khi đạt tới số lần lặp tối đa hoặc tìm được một lời giải.

B3. Khi đã đạt tới số lần lặp tối đa mà không tìm được lời giải thì thuật toán trả về None.

1. **Các ứng dụng thực tiễn khác của Min-conflicts algorithm**

Thuật toán Min-conflicts được ứng dụng rộng rãi trong các bài toán tìm kiếm, lập lịch và quy hoạch tài nguyên. Dưới đây là một số ứng dụng thực tế của thuật toán Min-conflicts:

* Giải quyết bài toán sudoku: Bài toán Sudoku có thể được giải bằng cách sử dụng thuật toán Min-conflicts. Bài toán này có thể được xem như một bài toán tối ưu hóa với các ràng buộc và các giá trị cho các ô còn trống.
* Sắp xếp chỗ ngồi cho khách mời: Bài toán này yêu cầu phải sắp xếp chỗ ngồi cho các khách mời theo một số ràng buộc nhất định để đảm bảo sự hài lòng của các khách mời.
* Lập lịch sản xuất: Trong quá trình sản xuất, các bộ phận của hệ thống có thể xung đột với nhau. Thuật toán Min-conflicts có thể được sử dụng để tối ưu hóa việc lập lịch sản xuất để giảm thiểu xung đột và tăng hiệu quả sản xuất.
* Xếp lịch thi: Việc xếp lịch thi cho học sinh đôi khi là một bài toán phức tạp. Thuật toán Min-conflicts có thể được sử dụng để tối ưu hóa việc xếp lịch thi để giảm thiểu xung đột và đảm bảo rằng không có học sinh thi cùng một môn học vào cùng một thời điểm.
* Lập lịch đặt phòng khám: Thuật toán Min-conflicts có thể được sử dụng để tối ưu hóa việc lập lịch đặt phòng khám cho bệnh nhân để đảm bảo rằng các bệnh nhân không đến cùng lúc và không phải chờ đợi quá lâu.

Như vậy, thuật toán Min-conflicts có nhiều ứng dụng trong thực tế và được sử dụng để giải quyết các bài toán tối ưu hóa có ràng buộc.

**IV. ĐÁNH GIÁ THUẬT TOÁN**

Ảnh có chứa biểu đồ

Mô tả được tạo tự động

Thuật toán Min-conflicts là một thuật toán tìm kiếm địa phương đơn giản để giải quyết bài toán này. Nó hoạt động bằng cách thay đổi vị trí của các học sinh một cách ngẫu nhiên và giải quyết xung đột cho đến khi không còn xung đột nào còn lại.

Về biểu đồ thể hiện kết quả của thuật toán Min-conflicts, nó cho thấy số bước di chuyển học sinh ở các bàn học tăng theo cấp số nhân với số học sinh và xấp xỉ là một đường thẳng tuyến tính 2x/5 = y

Trong trường hợp này, biểu đồ cho thấy rằng với mỗi tăng thêm 25 học sinh, số bước di chuyển tăng thêm khoảng 10 bước. Điều này cho thấy rằng sự tăng trưởng của số bước di chuyển học sinh tăng theo một cách tương đối chậm

Điều này cũng có thể được giải thích bởi sự tăng trưởng một cách tuyến tính của số xung đột có thể được giải quyết trong mỗi bước. Tuy nhiên, số lượng xung đột sẽ tăng theo cấp số nhân với số lượng học sinh, vì vậy số bước di chuyển tăng một cách chậm chạp hơn khi số lượng học sinh tăng lên.

Vì vậy, kết quả của biểu đồ này có thể được xem như là một hướng dẫn thô sơ về cách tăng trưởng của số bước di chuyển trong khi giải quyết bài toán sắp xếp chỗ học sinh nói chuyện bằng thuật toán Min-conflicts. Tuy nhiên, để có một kết quả chính xác và chính thức, ta cần thực hiện nhiều lần giải quyết và đánh giá kết quả trung bình trên nhiều trường hợp khác nhau của bài toán.

**V. ƯU ĐIỂM VÀ HẠN CHẾ CỦA THUẬT TOÁN**

1. **Ưu điểm**

* Hiệu quả: Thuật toán Min-conflicts là một phương pháp hiệu quả để giải quyết các bài toán tối ưu hóa lớn, phức tạp. Nó có thể xử lý một lượng lớn các xung đột trong thời gian ngắn và giúp tối ưu hóa trạng thái hiện tại.
* Linh hoạt: Min-conflicts algorithm có tính linh hoạt cao, có thể dễ dàng thay đổi để phù hợp với nhiều loại bài toán khác nhau. Nó cũng có thể được sử dụng trong các bài toán tối ưu hóa động, trong đó các ràng buộc thay đổi theo thời gian.
* Tìm lời giải tốt nhất: Min-conflicts algorithm có thể tìm ra lời giải tốt nhất trong thời gian ngắn và đôi khi có thể tìm ra lời giải tối ưu trong thời gian chấp nhận được. Điều này là do thuật toán tập trung vào việc giải quyết các xung đột, giúp tối ưu hóa trạng thái hiện tại và tìm kiếm lời giải tốt hơn.
* Dễ dàng thực hiện: Thuật toán Min-conflicts có cấu trúc đơn giản và dễ thực hiện so với các thuật toán thỏa mãn ràng buộc khác. Nó có thể được thực hiện trên các máy tính cá nhân hoặc cả trên các hệ thống phân tán.

1. **Hạn chế**

**2.1. Không đảm bảo tìm được giải pháp tối ưu**

Thuật toán Min-conflicts không đảm bảo tìm được lời giải tối ưu cho mọi trường hợp. Thuật toán tập trung vào giải quyết các xung đột giữa các ràng buộc để tìm ra trạng thái tối ưu nhất có thể. Tuy nhiên, trong một số trường hợp, lời giải tối ưu có thể không nằm trong tập hợp các trạng thái mà thuật toán đã tìm được.

**2.2. Dễ bị mắc kẹt ở cục bộ**

Thuật toán Min-conflicts có thể mắc kẹt ở cục bộ do nó là một thuật toán tìm kiếm cục bộ, có nghĩa là nó có thể bị kẹt trong tối ưu cục bộ. Khi thuật toán không thể giải quyết được các xung đột nữa hoặc không còn cách nào để giảm số lượng xung đột, thuật toán sẽ kết thúc và trả về trạng thái hiện tại.

**2.3. Thời gian chạy không đồng đều**

Thuật toán Min-conflicts phụ thuộc vào trạng thái khởi tạo ban đầu của bài toán. Nếu trạng thái khởi tạo không tốt, thuật toán có thể rơi vào một chu kỳ vô hạn, không thể tìm ra lời giải tốt. Ngoài ra, thời gian chạy tăng khi số lượng ràng buộc tăng. Khi có nhiều ràng buộc hơn, số lượng xung đột cũng sẽ tăng, dẫn đến thời gian giải quyết bài toán càng lâu.

1. **Các phương pháp giải quyết các hạn chế**

* Kết hợp với các phương pháp tối ưu hóa khác:

Kết hợp Min-conflicts algorithm với các phương pháp tối ưu hóa khác để tăng độ chính xác và giảm thời gian chạy. Ví dụ như kết hợp với thuật toán khác để thực hiện khởi tạo ban đầu tốt hơn, hoặc kết hợp với thuật toán tìm kiếm đa dạng để đánh giá nhiều giải pháp hơn.

* Sử dụng các thuật toán khởi tạo ngẫu nhiên:

Để giảm phụ thuộc vào trạng thái khởi tạo, ta có thể sử dụng các thuật toán khởi tạo ngẫu nhiên để tạo ra nhiều trạng thái khởi tạo khác nhau. Khi đó, Min-conflicts sẽ được áp dụng trên nhiều trạng thái khác nhau, giúp tăng khả năng tìm ra lời giải tối ưu.

* Tối ưu hóa số lượng ràng buộc:

Để giảm thời gian chạy, ta có thể tối ưu hóa số lượng ràng buộc của bài toán. Một số ràng buộc có thể được thay đổi để giảm số lượng xung đột và giúp thuật toán giải quyết bài toán nhanh hơn.

* Sử dụng các biến phụ trợ:

Trong một số trường hợp, ta có thể sử dụng các biến phụ trợ để giải quyết các ràng buộc khó xử lý. Các biến này có thể giúp giảm số lượng xung đột và tăng khả năng tìm ra lời giải tối ưu.

* Tinh chỉnh các thông số của thuật toán:

Các thông số của Min-conflicts algorithm như số lần lặp hoặc số lần thay đổi giá trị của biến có thể được tinh chỉnh để tăng độ chính xác và giảm thời gian chạy.

**VI. CÁC BIẾN THỂ CỦA THUẬT TOÁN MIN-CONFLICTS**

**1. Hybrid Min-conflicts**

Thuật toán Hybrid Min-conflicts là một biến thể của thuật toán Min-conflicts kết hợp với các phương pháp tìm kiếm cục bộ khác để cải thiện hiệu quả giải quyết bài toán tối ưu. Thuật toán sử dụng một mô hình hồi quy để đánh giá mức độ vi phạm của các ràng buộc và thực hiện các chỉnh sửa để tối ưu hóa quá trình giải quyết.

Việc kết hợp giữa thuật toán Min-conflicts và một thuật toán khác cho phép tận dụng những ưu điểm của từng thuật toán để tìm giải pháp tối ưu nhất. Thuật toán Hybrid Min-conflicts cũng có tính linh hoạt trong việc thay đổi thuật toán khác để tìm kiếm giải pháp tối ưu hơn. Điều này giúp thuật toán có khả năng tìm kiếm giải pháp hiệu quả hơn với những bài toán khó.

Thuật toán Hybrid Min-conflicts khả năng giải quyết nhiều loại bài toán tối ưu khác nhau, bao gồm các bài toán về lập lịch, phân bổ tài nguyên... Có tốc độ hội tụ nhanh hơn so với các phương pháp giải quyết truyền thống và ta thể được tùy chỉnh để phù hợp với nhiều định dạng bài toán tối ưu.

**2. Min-conflicts-Random-Restart**

Thuật toán Min-conflicts-Random-Restart hoạt động bằng cách lựa chọn một giá trị ngẫu nhiên cho mỗi biến trong bài toán tối ưu, sau đó kiểm tra xem các ràng buộc có được đáp ứng hay không. Nếu các ràng buộc được đáp ứng, một giải pháp tối ưu tạm thời được tìm thấy. Tuy nhiên, nếu không có giải pháp tối ưu nào được tìm thấy, thuật toán sẽ tiến hành khởi động lại ngẫu nhiên một số lần để tìm kiếm giải pháp tối ưu hơn. Khi thuật toán khởi động lại, các giá trị của biến trong bài toán tối ưu được lựa chọn ngẫu nhiên một lần nữa, sau đó thuật toán kiểm tra xem các ràng buộc có được đáp ứng hay không. Quá trình này được tiếp tục cho đến khi thuật toán tìm thấy giải pháp tối ưu hoặc đạt đến giới hạn lần khởi động.

Ưu điểm của thuật toán Min-conflicts-Random-Restart là tránh kết quả tối ưu cục bộ vì thuật toán sử dụng kỹ thuật khởi động ngẫu nhiên để tránh rơi vào kết quả tối ưu địa phương, giúp tìm kiếm giải pháp tối ưu chính xác hơn. Thuật toán còn giới hạn thời gian để tìm kiếm giải pháp tối ưu, giúp đạt được kết quả nhanh hơn và tiết kiệm thời gian. Thuật toán Min-conflicts-Random-Restart là một trong những thuật toán đơn giản và dễ sử dụng để giải quyết các bài toán tối ưu hóa.

Tuy nhiên, nhược điểm của thuật toán này là chi phí tính toán có thể tăng lên do số lần khởi động lại, và kết quả tìm được có thể không phải là tối ưu toàn cục vì bị mắc kẹt trong các tối ưu cục bộ khác nhau.

**3. Min-conflicts Tabu Search**

Min-conflicts Tabu Search hoạt động bằng cách chọn ngẫu nhiên một biến trong giải pháp hiện tại và tìm cách thay đổi giá trị của biến đó để giảm thiểu xung đột. Sau đó, thuật toán kiểm tra xem thay đổi này có được cho phép bởi các ràng buộc của bài toán hay không. Nếu giải pháp mới thỏa mãn ràng buộc, Min-conflicts Tabu Search sẽ chấp nhận giải pháp mới và lưu lại nó trong danh sách Tabu để tránh lặp lại.

Nếu giải pháp mới không thỏa mãn ràng buộc, hoặc thuật toán thấy rằng việc thay đổi giá trị của biến này sẽ tạo ra xung đột khác với giải pháp hiện tại, Min-conflicts Tabu Search sẽ quay lại giải pháp cũ và chọn một biến khác để thay đổi giá trị.

Với sự kết hợp của cả hai phương pháp tìm kiếm cục bộ, Min-conflicts Tabu Search có khả năng tìm ra giải pháp tối ưu hơn so với các phương pháp tìm kiếm ngẫu nhiên. Ngoài ra, thuật toán cũng giảm thiểu khả năng rơi vào vòng lặp và tìm kiếm quá sâu trong không gian giải pháp.

Tuy nhiên Min-conflicts Tabu Search tìm kiếm giải pháp tốt nhất trong phạm vi giải pháp cục bộ và không đảm bảo tìm được giải pháp toàn cục tối ưu của bài toán.[5]

**4. Min-conflicts with Constraint Weighting**

Phương pháp Min-conflicts with Constraint Weighting, giá trị trọng số được gán cho từng ràng buộc trước khi bắt đầu thuật toán. Giá trị trọng số này sẽ quyết định thứ tự xử lý các ràng buộc. Giá trị này có thể được chọn bằng nhiều cách khác nhau, ví dụ như dựa trên một giá trị ngẫu nhiên hoặc được tính dựa trên các thông tin về bài toán.

Khi bắt đầu tìm kiếm giải pháp, thuật toán Min-conflicts with Constraint Weighting sẽ tìm kiếm một giải pháp tốt nhất bằng cách xử lý lần lượt từng ràng buộc, từ các ràng buộc được gán trọng số nhỏ đến các ràng buộc được gán trọng số lớn. Quá trình này sẽ tối ưu hóa tập các giá trị biến đầu vào bằng cách giải quyết các ràng buộc một cách tối ưu. Nếu không thể tìm kiếm được giải pháp tốt nhất, thuật toán sẽ chạy lại với các giá trị trọng số khác nhau cho từng ràng buộc.

Min-conflicts with Constraint Weighting giảm số lượng giá trị phải kiểm tra bằng cách tập trung vào các ràng buộc có trọng số nhỏ hơn, từ đó tối ưu hóa quá trình tìm kiếm giải pháp tối ưu. Việc giảm số lượng giá trị phải kiểm tra cũng cắt giảm khi chi phí tính toán đối với thuật toán. Min-conflicts with Constraint Weighting là phương pháp đơn giản và dễ sử dụng, thường có hiệu quả trong việc giải quyết các bài toán tối ưu hóa đòi hỏi sự chính xác cao trong các ràng buộc.

Tuy nhiên phương pháp tên không phù hợp với những bài có quá nhiều ràng buộc hoặc có cấu trúc ràng buộc phức tạp. Phương pháp chỉ tìm được giải pháp tối ưu trong trường hợp các ràng buộc không quá phức tạp. Trong trường hợp các ràng buộc phức tạp hơn, phương pháp có thể không tìm được giải pháp tối ưu.[6]

**TỔNG KẾT**

1. **Kết luận về Min-conflicts Algorithm**

Min-conflicts algorithm là một thuật toán hiệu quả trong việc giải quyết các bài toán tối ưu hóa đa mục tiêu. Đặc biệt, thuật toán này có thể được áp dụng cho các bài toán liên quan đến các hệ thống phân cấp phân tán (CPSs). Việc sử dụng Min-conflicts algorithm giúp tối ưu hóa các giá trị của các biến trong hệ thống CPSs, từ đó giảm thiểu tổng số lần thay đổi giá trị của các biến trong hệ thống CPSs trong quá trình tối ưu hóa và tối ưu hóa hiệu suất của hệ thống.

Tuy nhiên, nhược điểm của Min-conflicts algorithm là không đảm bảo tìm ra giải pháp tối ưu nhất. Ngoài ra, nếu không có hàm mục tiêu rõ ràng, việc áp dụng thuật toán này cũng sẽ gặp khó khăn.

1. **Hướng phát triển và ứng dụng trong tương lai**

Trong tương lai, Min-conflicts algorithm có thể được phát triển để có thể tối ưu hơn trong việc giải quyết các bài toán phức tạp và có số lượng biến lớn hơn. Ngoài ra, việc áp dụng thuật toán này trong các lĩnh vực khác như điều khiển tự động, truy vấn dữ liệu, hoặc trò chơi cũng có thể đem lại nhiều kết quả tốt.

Các ứng dụng của Min-conflicts algorithm cũng có thể được tìm thấy trong các lĩnh vực như tự động hóa, điều khiển hệ thống và truyền thông không dây. Nó cũng có thể được sử dụng trong các hệ thống tự động điều khiển giao thông, hệ thống phát hiện xâm nhập, hệ thống đánh giá rủi ro và hệ thống điều khiển phòng cháy chữa cháy.

Vì vậy, Min-conflicts algorithm có tiềm năng phát triển và ứng dụng rộng rãi trong tương lai, đặc biệt là trong các lĩnh vực liên quan đến các hệ thống phân cấp phân tán.

**TÀI LIỆU THAM KHẢO**

[1] [Constraint satisfaction problem - Wikipedia](https://en.wikipedia.org/wiki/Constraint_satisfaction_problem#Solution)

[2] [Local search (constraint satisfaction) - Wikipedia](https://en.wikipedia.org/wiki/Local_search_(constraint_satisfaction)?fbclid=IwAR23AtPSDE7BJMpGx8QY9aWeU362RouBFhieyv0yrtUpbWCi1V28rmMXLqw#:~:text=In%20constraint%20satisfaction%2C%20local%20search,an%20assignment%20at%20each%20step)

[3] [Min-conflicts algorithm - Wikipedia](https://en.wikipedia.org/wiki/Min-conflicts_algorithm)

[4] [(PDF) The min-conflicts heuristic: Experimental and theoretical results (researchgate.net)](https://www.researchgate.net/publication/24322715_The_min-conflicts_heuristic_Experimental_and_theoretical_results?fbclid=IwAR30Wi6JGjtP3IamcNl_Mf4opQV4voADF2eWofH3O2ZIPNMB8KsnI_7bG8Y): Bài báo “The min-conflicts heuristic: Experimental and theoretical results” tháng 9 năm 1990, các tác giả Steven Minton, Andrew B. Philips, Mark Johnston, Philip Laird

[5] <http://www.math.ac.vn/publications/vjm/vjm_31/Pdf_files_4_2003/Duong.pdf>. Bài báo “Tabu Search Approach to the Solution of the General Lectures Scheduling Problem” ngày 22 tháng 9 năm 2002, các tác giả Do Xuan Duong , Pham Huy Dien.

[6] <https://hal.science/hal-02414832/document>. Bài báo “ Refining Constraint Weighting ” Ngày 5 tháng 1 năm 2021, các tác giả Hugues Wattez, Christophe Lecoutre, Anastasia Paparrizou, Sébastien Tabary.

[7] <https://utex.hcmute.edu.vn/mod/resource/view.php?id=729790>. Bài báo “Artificial Intelligence. A Modern Approach” các tác giả Stuart J. Russell and Peter Norvig.