**TRƯỜNG ĐẠI HỌC VINH**

**VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

A blue and white logo with a globe and bird

AI-generated content may be incorrect.

**ĐỒ ÁN HỌC PHẦN**

**PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN**

**TÊN ĐỀ TÀI:**

**GIẢI BÀI TOÁN XẾP BA LÔ ĐA CHIỀU BẰNG THUẬT TOÁN NHÁNH CẬN**

**NHÓM : 11**

|  |  |
| --- | --- |
| **GVHD:** | **TS. Nguyễn Thị Uyên** |
| **Nhóm SVTH**: | 1. Nguyễn Danh Hiếu – 225748020110096 – NT  2.Lê Đức Anh – 225748020110057  3. Trần Đại Hiệp – 225748020110176  4.Hà văn Hiệp - 225748020110228 |

**Nghệ An, 5/2024**

**LỜI CẢM ƠN**

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến các thầy, các cô trong Trường Đại học Vinh, đặc biệt là các thầy, các cô khoa viện kỹ thuật và công nghệ, những người đã trực tiếp giảng dạy, giúp đỡ chúng em trong những tháng học tập ở trường, đã tạo điều kiện cho chúng em được học tập, rèn luyện các kĩ năng cần thiết cho công việc sau này. Trong thời gian thực hiện đồ án biết bao nhiêu khó khăn, nhờ có sự giúp đỡ tạo điều kiện của Trường Đại học Vinh, sự giúp đỡ của các bạn và đặc biệt là sự hướng dẫn trực tiếp và chỉ bảo tận tình của GVHD cô Nguyễn Thi Uyên chúng em đã hoàn thành đề tài cùng với bản báo cáo đúng thời gian quy định. Mặc dù có nhiều cố gắng nhưng do kiến thức còn hạn chế nên bài báo cáo của chúng em cũng không tránh khỏi những thiếu sót nhất định. Chính vì vậy chúng em rất mong nhận được những ý kiến đóng góp, bổ sung của các thầy giáo, cô giáo và các bạn để đề tài của chúng em được hoàn thiện hơn. Cuối cùng, nhóm chúng em xin gửi lời chào, lời chúc sức khỏe đến toàn thể thầy cô giáo, đặc biệt em xin gửi lời cảm ơn sâu sắc tới cô Nguyễn Thi Uyên đã hướng dẫn em suốt quá trình làm đồ án. Nhóm chúng em xin chân thành cảm ơn!

**MỤC LỤC**

[MỞ ĐẦU 4](#_Toc198579469)

[CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN VÀ THUẬT TOÁN NHÁNH CẬN 5](#_Toc198579470)

[1.1. Giới thiệu bài toán xếp ba lô đa chiều 5](#_Toc198579471)

[1.2. Các phương pháp giải bài toán xếp ba lô đa chiều (MKP) 7](#_Toc198579472)

[1.3 Nghiên cứu liên quan 9](#_Toc198579473)

[1.4. Thuật toán Nhánh cận và cách tiếp cận bài toán 10](#_Toc198579474)

[1.5. bài toán xếp ba lô đa chiều bằng thuật toán Nhánh cận 11](#_Toc198579475)

[1.6. Mã giả thuật toán Nhánh cận cho MKP 11](#_Toc198579476)

[1.7. Giải bài toán xếp ba lô đa chiều bằng thuật toán nhánh cận 13](#_Toc198579477)

[1.̃8. Phân tích độ phức tạp của thuật toán 15](#_Toc198579478)

[1.9. Một số ứng dụng liên quan đến đề tài 16](#_Toc198579479)

[CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ BÀI TOÁN ỨNG DỤNG 18](#_Toc198579480)

[2.1. Mô tả bài toán ứng dụng 18](#_Toc198579481)

[2.2. Phân tích bài toán xếp ba lô cổ điểm và bài toán xếp ba lô đa chiều 18](#_Toc198579482)

[2.3. Bối cảnh bài toán 20](#_Toc198579483)

[2.4. Giải bài toán ứng dụng 22](#_Toc198579484)

[CHƯƠNG 3: CÀI ĐẶT VÀ THỰC NGHIỆM ĐÁNH GIÁ 27](#_Toc198579485)

[3.1 Môi trường cài đặt 27](#_Toc198579486)

[3.2. Chi tiết cài đặt thuật toán 27](#_Toc198579487)

[3.3 Bộ dữ liệu thử nghiệm 33](#_Toc198579488)

[3.4 Đánh giá kết quả thực nghiệm 35](#_Toc198579489)

[3.5. So sánh với các phương pháp khác 37](#_Toc198579490)

[KẾT LUẬN 39](#_Toc198579491)

[TÀI LIỆU THAM KHẢO 40](#_Toc198579492)

**DANH MỤC HÌNH ẢNH**

[*Hình 1.1. Minh họa bài toán ba lô đa chiều* 5](#_Toc198733391)

[*Hình 1.2: Quản lý tài nguyên và đầu tư dự án* 15](#_Toc198733392)

[*Hình 1.3: Lập lịch và tối ưu hóa sản xuất* 15](#_Toc198733393)

[*Hình 3.1: Lớp Project – Biểu diễn đối tượng dự án* 26](#_Toc198733394)

[*Hình 3.2: Lớp Node – Biểu diễn trạng thái trong cây tìm kiếm* 28](#_Toc198733395)

[*Hình 3.3: thuật toán nhánh cận trong HomeController* 28](#_Toc198733396)

[*Hình 3.4: Kết quả thực nghiệm bộ dữ liệu một* 32](#_Toc198733397)

[*Hình 3.5: Kết quả thực nghiệm bộ dữ liệu hai* 33](#_Toc198733398)

**DANH MỤC BẢNG**

[*Bảng 1.1: So sánh và lý do chọn thuật toán nhánh cận* 8](#_Toc198733412)

[*Bảng 2.1: Bảng so sánh bài toán gốc và bài toán ứng dụng* 19](#_Toc198733413)

[*Bảng 2.2: Giải bài toán ứng dụng* 20](#_Toc198733414)

[*Bảng 3.1. Bộ dữ liệu thử nghiệm một* 30](#_Toc198733415)

[*Bảng 3.. Bộ dữ liệu thử nghiệm hai* 31](#_Toc198733416)

# **MỞ ĐẦU**

Bài toán xếp ba lô từ lâu đã trở thành một trong những bài toán tối ưu hóa tổ hợp kinh điển trong khoa học máy tính, với nhiều ứng dụng thực tiễn trong các lĩnh vực như quản lý tài nguyên, lập lịch, và vận tải. Nếu như bài toán xếp ba lô cổ điển chỉ tập trung vào một ràng buộc đơn giản (thường là trọng lượng), thì bài toán xếp ba lô đa chiều (Multidimensional Knapsack Problem - MKP) mở rộng phạm vi bằng cách đưa vào nhiều ràng buộc đồng thời như trọng lượng, thể tích, chi phí, hoặc các yếu tố khác. Sự phức tạp này không chỉ phản ánh sát hơn các tình huống thực tế mà còn biến MKP thành một bài toán NP-khó, đòi hỏi các phương pháp giải tiên tiến để đạt được hiệu quả cao.

Trong số các kỹ thuật giải quyết bài toán tối ưu hóa tổ hợp, thuật toán Nhánh cận (Branch and Bound) nổi bật nhờ khả năng tìm kiếm toàn cục một cách thông minh. Bằng cách chia không gian nghiệm thành các nhánh và sử dụng các cận trên, cận dưới để loại bỏ những nhánh không tiềm năng, Nhánh cận giúp giảm thiểu thời gian tính toán mà vẫn đảm bảo tìm được nghiệm tối ưu. Với đặc điểm này, thuật toán trở thành một lựa chọn lý tưởng để giải bài toán xếp ba lô đa chiều, đồng thời là chủ đề chính của đồ án học phần lần này.

Đề tài "Giải bài toán xếp ba lô đa chiều bằng thuật toán Nhánh cận" được thực hiện nhằm mục đích nghiên cứu lý thuyết về MKP, phân tích cách áp dụng thuật toán Nhánh cận, và kiểm chứng hiệu quả của nó qua một bài toán cụ thể. Đồ án không chỉ giúp củng cố kiến thức về phân tích và thiết kế thuật toán mà còn mở ra cơ hội khám phá các ứng dụng thực tiễn của MKP trong đời sống. Nội dung đồ án bao gồm ba chương chính: mô tả bài toán, phân tích và thiết kế thuật toán, cùng với cài đặt và đánh giá thực nghiệm.

# **CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN VÀ THUẬT TOÁN NHÁNH CẬN**

## **1.1. Giới thiệu bài toán xếp ba lô đa chiều**

Bài toán xếp ba lô (Knapsack Problem) là một trong những bài toán tối ưu hóa tổ hợp kinh điển trong khoa học máy tính, nghiên cứu vận trù, và lý thuyết tối ưu hóa. Bài toán được đặt tên dựa trên tình huống thực tế: một người cần chọn các vật phẩm để đặt vào ba lô có sức chứa giới hạn, sao cho tổng giá trị của các vật phẩm được chọn là tối đa mà không vượt quá giới hạn tài nguyên.

Bài toán xếp ba lô đa chiều (Multidimensional Knapsack Problem - MKP) là một biến thể mở rộng của bài toán xếp ba lô cổ điển (0-1 Knapsack Problem), được định nghĩa lần đầu bởi nhà toán học George Dantzig vào năm 1957. Nếu bài toán cổ điển chỉ xem xét một ràng buộc duy nhất (thường là trọng lượng), MKP yêu cầu thỏa mãn đồng thời nhiều ràng buộc, chẳng hạn như trọng lượng, thể tích, chi phí, nhân sự, hoặc thời gian.

******

*Hình 1.1. Minh họa bài toán ba lô đa chiều*

**- Bài toán MKP được định nghĩa chính thức như sau:**

*Mục tiêu tối ưu hóa:*

*Ràng buộc:*

≤ ,

*Giải thích công thức:*

+ n: Số lượng vật phẩm (items) hoặc đối tượng (projects) có thể được chọn.

+ m: Số lượng ràng buộc (dimensions), ví dụ: trọng lượng, thể tích, chi phí....

+ Giá trị (lợi nhuận, profit) của vật phẩm thứ i. Đây là giá trị cần tối đa hóa.

+ ​: Lượng tài nguyên loại j cần thiết cho vật phẩm thứ i.

+ ​: Giới hạn tối đa của tài nguyên loại j

+ ​: Biến nhị phân, =1 nếu vật phẩm i được chọn, = 0 nếu không chọn.

***- Đầu vào và đầu ra của bài toán ba lô đa chiều***

*Đầu vào:*

+ n: Số lượng vật phẩm.

+ m: Số loại ràng buộc.

+ : Giá trị của vật phẩm thứ i.

+ ​: Tài nguyên loại j cần cho vật phẩm thứ i.

+ ​: Giới hạn tối đa của tài nguyên loại j.

*Đầu ra:*

+ Vector x = (, ) với = 1 nếu vật phẩm i được chọn, = nếu không.

+ Tổng giá trị tối đa ​, đại diện cho lợi nhuận hoặc giá trị tối ưu đạt được.

+ Danh sách các vật phẩm được chọn (tương ứng với các = 1), thỏa mãn tất cả các ràng buộc ≤ cho mọi j.

## **1.2. Các phương pháp giải bài toán xếp ba lô đa chiều (MKP)**

Để giải quyết bài toán xếp ba lô đa chiều (MKP), nhiều phương pháp thuật toán đã được nghiên cứu và ứng dụng. Mỗi phương pháp có những đặc điểm riêng biệt, thể hiện qua hiệu quả tính toán, khả năng mở rộng và độ chính xác của lời giải. Phần này trình bày các nhóm phương pháp phổ biến, bao gồm quy hoạch động, thuật toán tham lam, phương pháp metaheuristic và nhánh cận, đồng thời phân tích lý do chọn thuật toán phù hợp trong phạm vi đồ án này**.**

**1.2.1. Quy hoạch động**

Quy hoạch động là một trong những phương pháp truyền thống và hiệu quả cho bài toán ba lô cổ điển (0-1 Knapsack Problem). Phương pháp này xây dựng một bảng lưu trữ các trạng thái trung gian, trong đó mỗi trạng thái đại diện cho một tập hợp vật phẩm đã chọn và lượng tài nguyên đã sử dụng. Thuật toán lặp lại qua từng vật phẩm và cập nhật bảng trạng thái để đảm bảo rằng nghiệm tối ưu được bảo toàn tại mỗi bước.

Ưu điểm nổi bật của phương pháp quy hoạch động là khả năng đảm bảo tìm được nghiệm tối ưu toàn cục. Tuy nhiên, đối với bài toán MKP với nhiều ràng buộc tài nguyên (đa chiều), kích thước bảng lưu trữ tăng theo cấp số nhân của số ràng buộc và giới hạn tài nguyên. Cụ thể, độ phức tạp thời gian là O(n. và độ phức tạp không gian là () trong đó nnn là số vật phẩm, mmm là số ràng buộc, và ​ là giới hạn tài nguyên của ràng buộc thứ j. Do đó, phương pháp này không khả thi cho các bài toán MKP quy mô lớn.

**1.2.2. Thuật toán tham lam**

Thuật toán tham lam là phương pháp đơn giản, thường được sử dụng để đưa ra lời giải xấp xỉ nhanh cho các bài toán tổ hợp. Ý tưởng cơ bản là sắp xếp các vật phẩm theo một tiêu chí ưu tiên, chẳng hạn như tỷ lệ giá trị trên tổng lượng tài nguyên tiêu thụ Sau đó lần lượt chọn vật phẩm theo thứ tự cho đến khi không thể chọn thêm mà vẫn thỏa mãn tất cả các ràng buộc.

Phương pháp này có ưu điểm là tốc độ xử lý nhanh, dễ cài đặt và có độ phức tạp thời gian thấp, thường là O() do thao tác sắp xếp, với nnn là số vật phẩm. Tuy nhiên, thuật toán tham lam không đảm bảo tìm được nghiệm tối ưu và hiệu quả giải không cao trong những trường hợp có sự xung đột lớn giữa các ràng buộc.

**1.2.3. Thuật toán nhánh cận**

Thuật toán nhánh cận là một phương pháp tìm kiếm toàn cục có hiệu quả, đặc biệt phù hợp với bài toán tổ hợp quy mô vừa. Phương pháp này phân chia không gian nghiệm thành các nhánh nhỏ hơn bằng cách xét từng quyết định (chọn hoặc không chọn vật phẩm). Mỗi nhánh được đánh giá bằng cận trên (upper bound) và cận dưới (lower bound). Nếu cận trên của một nhánh không vượt qua nghiệm tốt nhất hiện tại, nhánh đó sẽ bị loại bỏ, giúp rút gọn đáng kể không gian tìm kiếm.

Ưu điểm của nhánh cận là khả năng đảm bảo nghiệm tối ưu nhờ việc đánh giá toàn bộ không gian nghiệm, nhưng vẫn có thể cắt bỏ các nhánh không tiềm năng. Tuy nhiên, trong trường hợp xấu nhất, độ phức tạp thời gian là O()và độ phức tạp không gian cũng ở mức tương tự, gây khó khăn khi xử lý các bài toán có số lượng vật phẩm lớn.

**1.2.4. So sánh và lý do chọn thuật toán nhánh cận**

Bảng dưới đây tổng hợp so sánh các phương pháp theo các tiêu chí: khả năng đảm bảo nghiệm tối ưu, hiệu quả với bài toán quy mô lớn, độ phức tạp thời gian và không gian:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuật toán** | **Đảm bảo tối ưu** | **Hiệu quả quy mô lớn** | **Độ phức tạp thời gian** | **Độ phức tạp không gian** |
| Quy hoạch động | Có | Không | O(n⋅∏bj) | O(∏bj) |
| Tham lam | Không | Có | O() | O(n) |
| Nhánh cận | Có | Trung bình | O() (xấu nhất) | O() (xấu nhất) |

*Bảng 1.1: So sánh và lý do chọn thuật toán nhánh cận*

## **1.3 Nghiên cứu liên quan**

Bài toán xếp ba lô đa chiều (Multidimensional Knapsack Problem - MKP) và các phương pháp giải quyết, bao gồm thuật toán Nhánh cận, đã thu hút sự chú ý của nhiều nhà nghiên cứu trong lĩnh vực khoa học máy tính, nghiên cứu vận trù và tối ưu hóa tổ hợp.

Bài toán xếp ba lô (Knapsack Problem) lần đầu tiên được định hình bởi nhà toán học George Dantzig vào năm 1957, với phiên bản cổ điển 0-1 Knapsack Problem. Tuy nhiên, phiên bản đa chiều (MKP) bắt đầu được nghiên cứu sâu rộng từ những năm 1980, khi các ứng dụng thực tiễn đòi hỏi các mô hình phức tạp hơn với nhiều ràng buộc đồng thời. Một trong những công trình nổi bật về MKP là bài báo của Fréville (2004), *"The Multidimensional Knapsack Problem: Structure and Algorithms"* [1]. Trong bài báo này, Fréville đã tổng hợp các đặc điểm cấu trúc của MKP, phân tích độ phức tạp tính toán, và đề xuất các chiến lược giải quyết, bao gồm cả phương pháp Nhánh cận và các thuật toán xấp xỉ.

Một nghiên cứu quan trọng khác là của Kellerer et al. (2004), được trình bày trong cuốn sách *"Knapsack Problems"* [2]. Các tác giả đã phân tích các biến thể của bài toán xếp ba lô, bao gồm MKP, và cung cấp một cái nhìn tổng quan về các phương pháp giải quyết, từ thuật toán chính xác (exact algorithms) như Nhánh cận và quy hoạch động, đến các thuật toán heuristic và metaheuristic như thuật toán di truyền (Genetic Algorithm) và tối ưu hóa đàn kiến (Ant Colony Optimization). Nghiên cứu này cũng nhấn mạnh rằng MKP là một bài toán NP-Complete, với độ phức tạp tăng theo cấp số nhân khi số chiều ràng buộc (mmm) tăng.

## **1.4. Thuật toán Nhánh cận và cách tiếp cận bài toán**

Thuật toán Nhánh cận (Branch and Bound) là một kỹ thuật tìm kiếm toàn cục được sử dụng để giải các bài toán tối ưu hóa tổ hợp, đặc biệt hiệu quả với các bài toán NP-khó như bài toán xếp ba lô đa chiều. Phương pháp này hoạt động bằng cách duyệt qua không gian nghiệm một cách có hệ thống, phân chia chúng thành các nhánh nhỏ hơn (nhánh) và đánh giá các nhánh này dựa trên các giới hạn (cận) để loại bỏ những nhánh không thể chứa nghiệm tối ưu, từ đó giảm thiểu số lượng nghiệm cần xem xét.

Thuật toán Nhánh cận (Branch and Bound) được giới thiệu lần đầu tiên bởi Land và Doig vào năm 1960 trong bài báo *"An Automatic Method of Solving Discrete Programming Problems"* [4]. Kể từ đó, Nhánh cận đã trở thành một phương pháp tiêu chuẩn để giải các bài toán tối ưu hóa tổ hợp, bao gồm cả MKP. Một trong những nghiên cứu quan trọng về ứng dụng Nhánh cận cho MKP là của Martello và Toth (1990), được trình bày trong cuốn sách *"Knapsack Problems: Algorithms and Computer Implementations"* [5]. Các tác giả đã đề xuất một chiến lược Nhánh cận hiệu quả cho MKP, sử dụng các cận trên dựa trên phép thư giãn tuyến tính (linear relaxation) và các kỹ thuật cắt tỉa thông minh để giảm không gian tìm kiếm.

Ngoài ra, nghiên cứu của Chu và Beasley (1998), *"A Genetic Algorithm for the Multidimensional Knapsack Problem"* [6], đã so sánh hiệu quả của Nhánh cận với các phương pháp metaheuristic. Mặc dù thuật toán di truyền của họ cho kết quả tốt hơn về thời gian chạy với các bài toán quy mô lớn, Nhánh cận vẫn được đánh giá cao vì khả năng đảm bảo nghiệm tối ưu toàn cục, đặc biệt với các bài toán có số chiều ràng buộc nhỏ.

## **1.5. bài toán xếp ba lô đa chiều bằng thuật toán Nhánh cận**

Cách tiếp cận bài toán xếp ba lô đa chiều bằng thuật toán Nhánh cận:

- Xác định nghiệm ban đầu: Bắt đầu với nghiệm rỗng, tức là chưa chọn bất kỳ đối tượng nào.

- Tính cận trên và cận dưới:

+ Cận trên (Upper Bound): Đánh giá giá trị tối đa có thể đạt được từ nghiệm hiện tại bằng cách giả định rằng có thể thêm một phần của các đối tượng còn lại (tương tự như bài toán xếp ba lô phân số), nhằm xác định mức trần của giá trị có thể đạt được.

+ Cận dưới (Lower Bound): Giá trị của nghiệm tốt nhất hiện có, thường được cập nhật khi tìm thấy một nghiệm khả thi mới với giá trị cao hơn.

+ Phân nhánh: Từ nghiệm hiện tại, tạo ra các nhánh con bằng cách xem xét việc chọn hoặc không chọn từng đối tượng tiếp theo.

+ Cắt tỉa: Loại bỏ các nhánh mà cận trên của chúng nhỏ hơn hoặc bằng cận dưới hiện tại, vì chúng không thể dẫn đến nghiệm tối ưu.

+ Lặp lại: Tiếp tục quá trình phân nhánh và cắt tỉa cho đến khi không còn nhánh nào để xem xét. Nghiệm tối ưu sẽ là nghiệm khả thi có giá trị cao nhất được tìm thấy trong quá trình này.

Phương pháp Nhánh cận đảm bảo tìm được nghiệm tối ưu bằng cách khám phá có hệ thống toàn bộ không gian nghiệm, nhưng nhờ việc cắt tỉa các nhánh không cần thiết, nó giúp giảm đáng kể số lượng nghiệm cần xem xét, đặc biệt hiệu quả đối với các bài toán có không gian nghiệm lớn như bài toán xếp ba lô đa chiều.

## **1.6. Mã giả thuật toán Nhánh cận cho MKP**

Dưới đây là mã giả chi tiết cho thuật toán Nhánh cận áp dụng cho MKP:

Input:

- n: Số vật phẩm

- v[i]: Giá trị vật phẩm i (i = 1, 2, ..., n)

- a[i][j]: Tài nguyên j cần cho vật phẩm i (j = 1, 2, ..., m)

- b[j]: Giới hạn tài nguyên j

Output:

- maxProfit: Tổng giá trị tối đa

- selected: Danh sách vật phẩm được chọn

***Thuật toán:***

1. Sắp xếp vật phẩm theo tỷ lệ hiệu quả:

2. Khởi tạo:

- queue: Hàng đợi ưu tiên lưu các node (sắp xếp theo cận trên giảm dần)

- root: Node gốc (Level = -1, Profit = 0, Resources = 0)

- maxProfit = 0

- bestSelection = [ ]

3. Tính cận trên cho root bằng phương pháp tham lam

4. Thêm root vào queue

5. While queue không rỗng:

- Lấy node u từ queue

- Nếu u.Bound ≤ maxProfit, bỏ qua

- Nếu u.Level = n, bỏ qua

- Tạo nhánh chọn vật phẩm u.Level + 1:

+ v = new Node(

Level = u.Level + 1,

Profit = u.Profit + v[u.Level + 1],

Resources = u.Resources + a[u.Level + 1])

+ Nếu v thỏa mãn tất cả ràng buộc:

+ Nếu v.Profit > maxProfit:

- Cập nhật maxProfit = v.Profit

- Cập nhật bestSelection = v.Selected

+ Tính v.Bound bằng phương pháp tham lam

+ Nếu v.Bound > maxProfit, thêm v vào queue

- Tạo nhánh không chọn vật phẩm u.Level + 1:

+ w = new Node(

Level = u.Level + 1,

Profit = u.Profit,

Resources = u.Resources)

+ Tính w.Bound

+ Nếu w.Bound > maxProfit, thêm w vào queue

6. Trả về maxProfit, bestSelection

## **1.7. Giải bài toán xếp ba lô đa chiều bằng thuật toán nhánh cận**

Trong thuật toán Nhánh cận (Branch and Bound) được triển khai để giải bài toán xếp ba lô đa chiều (Multidimensional Knapsack Problem - MKP), cận trên (Upper Bound) đóng vai trò quan trọng trong việc đánh giá tiềm năng của một nhánh. Cận trên là giá trị ước lượng tối đa mà một nhánh có thể đạt được nếu tiếp tục được khám phá. Hàm CalculateBound trong mã nguồn sử dụng phương pháp tham lam (Greedy Approach) để tính cận trên, nhưng để làm rõ cơ chế này, dưới đây là các bước chi tiết:

Để giải bài toán xếp ba lô đa chiều bằng thuật toán Nhánh cận, ta thực hiện các bước tổng quát như sau:

*Bước 1: Sắp xếp các đối tượng theo tỷ lệ hiệu quả.*

Công thức =

Thứ tự ưu tiên: Sắp xếp giảm dần theo tỷ lệ này để ưu tiên đối tượng hiệu quả nhất trước.

*Bước 2: Khởi tạo hàng đợi ưu tiên*

Nút gốc:

+ Chưa chọn đối tượng nào.

+ Tổng tài nguyên = 0, tổng giá trị = 0.

Cận trên (Bound): Tính bằng phương pháp tham lam:

+ Chọn lần lượt các đối tượng theo thứ tự đã sắp xếp đến khi vi phạm ràng buộc.

+ Nếu không thể chọn toàn bộ đối tượng tiếp theo, thêm một phần giá trị của nó (dựa trên tài nguyên còn lại).

*Bước 3: Phân nhánh và cắt tỉa*

- Lấy nút có cận trên cao nhất từ hàng đợi.

- Tạo hai nhánh con:

- Nhánh chọn đối tượng tiếp theo:

+ Cộng thêm tài nguyên và giá trị của đối tượng.

+ Kiểm tra ràng buộc: Nếu vượt giới hạn → cắt tỉa.

- Nhánh không chọn: Giữ nguyên trạng thái hiện tại.

- Tính cận trên mới cho các nhánh con:

+ Nếu cận trên ≤ giá trị nghiệm tốt nhất hiện có → cắt tỉa.

*Bước 4: Cập nhật nghiệm tốt nhất*

- Nếu một nhánh đã xét hết tất cả đối tượng và có giá trị > nghiệm hiện tại → cập nhật nghiệm tối ưu.

- Nếu còn đối tượng chưa xét → thêm nhánh vào hàng đợi.

*Bước 5: Kết thúc thuật toán*

- Dừng khi hàng đợi rỗng hoặc cận trên tất cả nhánh còn lại ≤ nghiệm hiện tại.

- Nghiệm tốt nhất tìm được là tối ưu toàn cục.

## **1.̃8. Phân tích độ phức tạp của thuật toán**

**- Độ phức tạp thời gian**

Trường hợp xấu nhất: Thuật toán Nhánh cận phải xét toàn bộ không gian nghiệm, tức là tất cả tổ hợp chọn hoặc không chọn nnn đối tượng, dẫn đến tổ hợp. Với mỗi tổ hợp:

+ Kiểm tra m ràng buộc (trong bài toán này, m=2) có độ phức tạp O(n).

+ Tính cận trên bằng phương pháp tham lam cũng có độ phức tạp O(n).

+ Do đó, độ phức tạp thời gian tổng cộng là: O( ⋅n).

**Độ phức tạp không gian:**

+ Hàng đợi ưu tiên: Lưu trữ các nút trong cây phân nhánh. Trong trường hợp xấu nhất, hàng đợi chứa O() nút, mỗi nút cần không gian O(n) để lưu trạng thái (mảng Selected, các giá trị tổng). Tổng không gian là: O( ⋅n).

+ Các biến phụ: Mảng lưu danh sách đối tượng và các biến tạm thời chiếm không gian O(n), không đáng kể.

+ Tổng cộng: Độ phức tạp không gian là: O( ⋅n).

Như vậy thuật toán Nhánh cận có độ phức tạp thời gian và không gian tăng theo cấp số nhân (O( ⋅n)). phản ánh bản chất NP-Complete của bài toán MKP. Tuy nhiên, cơ chế cắt tỉa giúp giảm số nhánh cần xét trong thực tế, làm cho thuật toán khả thi với các bài toán quy mô nhỏ.

## **1.9. Một số ứng dụng liên quan đến đề tài**

Bài toán xếp ba lô đa chiều (Multidimensional Knapsack Problem - MKP) và thuật toán Nhánh cận không chỉ mang tính lý thuyết mà còn có nhiều ứng dụng thực tiễn trong các lĩnh vực khác nhau, từ quản lý tài nguyên, lập kế hoạch sản xuất, đến tối ưu hóa trong công nghệ và kinh tế. Dưới đây là một số ứng dụng tiêu biểu của MKP và thuật toán Nhánh cận:

**1.9.1. Quản lý tài nguyên và đầu tư dự án**

MKP được sử dụng rộng rãi trong các bài toán phân bổ tài nguyên với nhiều ràng buộc đồng thời. Ví dụ, trong quản lý dự án (như bài toán ứng dụng trong đồ án này), một công ty cần lựa chọn các dự án đầu tư sao cho tối đa hóa lợi nhuận, đồng thời không vượt quá ngân sách, nhân sự, hoặc các nguồn lực khác như thời gian và thiết bị. Thuật toán Nhánh cận giúp tìm nghiệm tối ưu trong không gian nghiệm lớn, đặc biệt hiệu quả với các bài toán có quy mô vừa phải.



*Hình 1.2: Quản lý tài nguyên và đầu tư dự án*

**1.8.2. Lập lịch và tối ưu hóa sản xuất**

Trong ngành sản xuất, MKP có thể được áp dụng để tối ưu hóa việc phân bổ tài nguyên sản xuất (máy móc, nhân công, nguyên liệu) cho các đơn hàng. Ví dụ, một nhà máy cần quyết định sản xuất những sản phẩm nào để tối đa hóa lợi nhuận, trong khi phải tuân thủ các ràng buộc về công suất máy móc, thời gian hoàn thành, và nguồn nguyên liệu. Thuật toán Nhánh cận hỗ trợ việc cắt tỉa các phương án không khả thi, từ đó tìm ra kế hoạch sản xuất tối ưu.



*Hình 1.3: Lập lịch và tối ưu hóa sản xuất*

**1.9.3. Logistics và vận tải**

MKP được ứng dụng trong các bài toán xếp hàng hóa lên container hoặc xe tải, nơi cần tối ưu hóa giá trị hàng hóa (dựa trên lợi nhuận hoặc mức độ ưu tiên) trong khi tuân thủ các ràng buộc về trọng lượng, thể tích, và số lượng. Thuật toán Nhánh cận có thể giúp giảm không gian tìm kiếm, đảm bảo tìm được phương án xếp hàng tối ưu trong thời gian hợp lý.

## **CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ BÀI TOÁN ỨNG DỤNG**

## **2.1. Mô tả bài toán ứng dụng**

Bài toán xếp ba lô đa chiều (Multidimensional Knapsack Problem - MKP) được áp dụng trong đồ án này để giải quyết một vấn đề thực tiễn: một công ty cần lựa chọn các dự án đầu tư nhằm tối đa hóa lợi nhuận, đồng thời tuân thủ các ràng buộc về ngân sách và nhân sự cấp cao. Đây là một bài toán tối ưu hóa tổ hợp, trong đó mỗi dự án được xem như một vật phẩm với ba đặc điểm: chi phí (tài nguyên tài chính), số lượng nhân sự cần thiết, và lợi nhuận kỳ vọng. Mục tiêu là chọn một tập hợp dự án sao cho tổng lợi nhuận đạt tối đa, mà không vượt quá ngân sách và số nhân sự có sẵn.

## **2.2. Phân tích bài toán xếp ba lô cổ điểm và bài toán xếp ba lô đa chiều**

**2.2.1.Bài toán gốc (0-1 Knapsack Problem)**

-Đặc điểm:

Chỉ có một ràng buộc, thường là trọng lượng (W).

Mục tiêu: Tối đa hóa tổng giá trị (  ), với ( ).

Thuộc lớp NP-Complete, nhưng dễ giải hơn MKP nhờ số lượng ràng buộc đơn giản.

-Phương pháp giải phổ biến:

Quy hoạch động: Độ phức tạp ( O(n . W) ), hiệu quả với ( W ) nhỏ.

Tham lam: Dùng cho biến thể Fractional Knapsack, không tối ưu cho 0-1 Knapsack.

Nhánh cận: Ít sử dụng vì quy hoạch động hiệu quả hơn.

-Ứng dụng:

Chọn vật phẩm cho ba lô với trọng lượng giới hạn.

Phân bổ tài nguyên đơn lẻ, như chọn dự án với ngân sách cố định.

**2.2.2 Bài toán ứng dụng (MKP)**

**Đặc điểm:**

Có hai ràng buộc: chi phí ( C ) và nhân sự ( S ).

Mục tiêu: Tối đa hóa ( ), với () và( ).

Phức tạp hơn do phải cân bằng nhiều ràng buộc, thuộc lớp NP-Complete với độ phức tạp tăng theo số ràng buộc.

Cải tiến so với bài toán gốc:

Tính thực tiễn cao hơn:

Bài toán gốc chỉ xem xét một yếu tố (trọng lượng), ít phản ánh các tình huống thực tế. MKP với hai ràng buộc (chi phí, nhân sự) mô phỏng tốt hơn các kịch bản quản lý dự án, như phân bổ tài nguyên trong đầu tư hoặc sản xuất.

Tỷ lệ hiệu quả tổng quát:

Thay vì chỉ dùng (), bài toán ứng dụng sử dụng tỷ lệ (), cân nhắc đồng thời hai ràng buộc để ưu tiên dự án hiệu quả hơn.

Không gian nghiệm phức tạp:

MKP yêu cầu thuật toán xử lý không gian tìm kiếm lớn hơn, đòi hỏi cơ chế cắt tỉa thông minh như Nhánh cận, thay vì quy hoạch động vốn không khả thi do bộ nhớ lớn.

Hạn chế:

Độ phức tạp tính toán cao hơn, đặc biệt khi số dự án hoặc ràng buộc tăng.

Yêu cầu thuật toán phức tạp hơn (Nhánh cận) so với quy hoạch động cho bài toán gốc.

**2.2.4 Bảng so sánh bài toán gốc và bài toán ứng dụng**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **0-1 Knapsack Problem** | **Multidimensional Knapsack Problem (MKP)** |
| **Số ràng buộc** | 1 (trọng lượng W ) | 2 (chi phí C, nhân sự S) |
| **Mục tiêu** | Tối đa hóa: với | Tối đa hóa: với . |
| **Độ phức tạp** | NP-Complete, đơn giản hơn | NP-Complete, phức tạp hơn do nhiều ràng buộc |
| **Phương pháp giải chính** | - Quy hoạch động O(n.W)  - Tham lam (không tối ưu) - Nhánh cận (ít dùng) | - Nhánh cận (tối ưu, quy mô vừa) - Metaheuristic (quy mô lớn) - Quy hoạch động (không khả thi) |
| **Tỷ lệ hiệu quả** |  |  |
| **Ứng dụng** | - Chọn vật phẩm cho ba lô - Phân bổ tài nguyên đơn lẻ | - Quản lý dự án (tài nguyên đa chiều) - Logistics, sản xuất |
| **Tính thực tiễn** | Thấp, giới hạn ở tình huống đơn giản | Cao, mô phỏng tốt các kịch bản thực tế |
| **Hạn chế** | Ít phản ánh thực tế do chỉ một ràng buộc | Phức tạp hơn, độ phức tạp tính toán cao |

*Bảng 2.1: Bảng so sánh bài toán gốc và bài toán ứng dụng*

## **2.3. Bối cảnh bài toán**

Trong bối cảnh quản lý dự án, công ty có ngân sách giới hạn 10 triệu USD và tối đa 15 nhân sự cấp cao để phân bổ cho các dự án đầu tư. Mỗi dự án đòi hỏi một lượng chi phí và nhân sự nhất định, đồng thời mang lại lợi nhuận kỳ vọng. Bài toán này mô phỏng các tình huống thực tế, chẳng hạn như lựa chọn dự án phát triển sản phẩm, đầu tư cơ sở hạ tầng, hoặc nghiên cứu công nghệ, nơi công ty phải cân nhắc nhiều yếu tố đồng thời. So với bài toán xếp ba lô cổ điển (0-1 Knapsack Problem) chỉ có một ràng buộc (thường là trọng lượng), bài toán ứng dụng này phức tạp hơn do phải thỏa mãn hai ràng buộc (chi phí và nhân sự), khiến nó trở thành một phiên bản điển hình của MKP.

**2.3.1. Định nghĩa bài toán ứng dụng**

**Đầu vào:**

- ( n ): Số lượng dự án (trong ví dụ: ( n = 5 ), tương ứng với các dự án A, B, C, D, E).

- ( ): Lợi nhuận kỳ vọng của dự án ( i ) (đơn vị: triệu USD).

- ( ): Chi phí cần thiết để triển khai dự án ( i ) (triệu USD).

- ( ): Số nhân sự cấp cao cần cho dự án ( i ) (người).

- ( C ): Ngân sách tối đa của công ty (10 triệu USD).

- ( S ): Số nhân sự tối đa (15 người).

**Đầu ra:**

- Vector (, , ..., ), với ( = 1 ) nếu dự án ( i ) được chọn, ( = 0 ) nếu không.

***Tổng lợi nhuận tối đa***

***Danh sách các dự án được chọn, thỏa mãn các ràng buộc:***

-Tổng chi phí:

-Tổng nhân sự:

## **2.4. Giải bài toán ứng dụng**

**2.4.1. Dữ liệu cụ thể:**

Dữ liệu đầu vào bao gồm 5 dự án với thông tin như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Dự án** | **Chi phí (triệu USD)** | **Nhân sự (người)** | **Lợi nhuận (triệu USD)** |
| A | 3 | 5 | 8 |
| B | 4 | 6 | 10 |
| C | 2 | 3 | 5 |
| D | 5 | 4 | 12 |
| E | 1 | 2 | 3 |

*Bảng 2.2: Giải bài toán ứng dụng*

**Đầu vào**

- Số dự án (( n )): Là số lượng các dự án có thể được chọn. Trong bài toán này, ( n = 5 ), tương ứng với các dự án A, B, C, D, E.

- Lợi nhuận ( ): Là giá trị kỳ vọng của mỗi dự án, đo bằng triệu USD. Đây là yếu tố cần tối đa hóa trong bài toán.

- Chi phí ( ): Là lượng tài chính cần thiết để triển khai dự án, cũng đo bằng triệu USD.

- Nhân sự ( ): Là số lượng nhân sự cấp cao cần thiết cho dự án, đo bằng số người.

- Ngân sách tối đa ( C ): Là giới hạn tài chính của công ty, ở đây là 10 triệu USD.

- Nhân sự tối đa ( S ): Là giới hạn về số lượng nhân sự, ở đây là 15 người.

Dữ liệu đầu vào được cung cấp dưới dạng bảng hoặc danh sách các dự án, trong đó mỗi dự án được mô tả bằng bộ ba ( , , ). Các giá trị được giả định là số nguyên dương để phù hợp với thực tế quản lý dự án.

**Đầu ra**

Vector lựa chọn ( x ): Một mảng nhị phân biểu thị các dự án được chọn (  = 1 ) hoặc không chọn ( = 0 ).

Tổng lợi nhuận tối đa: Giá trị ( ), đại diện cho lợi nhuận lớn nhất có thể đạt được.

Danh sách dự án được chọn: Tập hợp các dự án tương ứng với ( = 1 ), kèm thông tin chi phí và nhân sự sử dụng.

Tài nguyên sử dụng: Bao gồm tổng chi phí (( )) và tổng nhân sự (( )), để kiểm tra việc tuân thủ các ràng buộc.

**2.4.2. Giải bài toán ứng dụng**

***Bước 1: Sắp xếp theo hiệu suất***

Hiệu suất = Lợi nhuận / (Chi phí + Nhân sự):

|  |  |
| --- | --- |
| **Dự án** | **Hiệu suất** |
| D | 12 / (5+4) = 1.33 |
| A | 8 / (3+5) = 1.0 |
| B | 10 / (4+6) = 1.0 |
| C | 5 / (2+3) = 1.0 |
| E | 3 / (1+2) = 1.0 |

Thứ tự xét: D, A, B, C, E

***Bước 2: Khởi tạo Node gốc***

Level = -1

Profit = 0, Cost = 0, Personnel = 0

Bound (cận trên) = tổng lợi nhuận = 12 + 8 + 10 + 5 + 3 = 38

***Bước 3: Phân nhánh và cắt tỉa***

Node 0 (Xét D):

- Chọn D:

Profit = 12, Cost = 5, Personnel = 4 (Hợp lệ)

Xét tiếp: A (3,5), B (4,6), C (2,3), E (1,2)

Xét A:

Tổng: Cost = 5+3 = 8, Personnel = 4+5 = 9

Profit = 20

Xét tiếp phần của B:

Cost còn lại: 10-8 = 2, Personnel còn lại: 15-9 = 6

Tỷ lệ B: 10 / (4+6) = 1

max phần của B = min(2/4, 6/6) = 0.5 → thêm 10×0.5 = 5

→ Bound = 20 + 5 = 25 → Node D được đưa vào hàng đợi.

Không chọn D:

Profit = 0, Bound = 8 + 10 + 5 + 3 = 26

Node Không D (Bound = 26) → Xét A

- Chọn A:

Profit = 8, Cost = 3, Personnel = 5 (Hợp lệ)

Xét tiếp: B, C, E

- Chọn B:

Cost = 3+4 = 7, Personnel = 5+6 = 11 → OK

Profit = 18

- Chọn C:

Cost = 7+2 = 9, Personnel = 11+3 = 14 → OK

Profit = 23

- Chọn E:

Cost = 9+1 = 10, Personnel = 14+2 = 16 → Vượt

→ Có thể lấy 0.5 của E → thêm 1.5

→ Bound = 23 + 1.5 = 24.5 → Đưa vào hàng đợi

Không chọn A:

Profit = 0, Bound = B+C+E = 18 → Bỏ qua

Node D → chọn tiếp A

Profit = 20, Cost = 8, Personnel = 9 → OK

Chọn C:

Cost = 8+2 = 10, Personnel = 9+3 = 12 → OK

Profit = 25

E không thể thêm (Cost = 11 > 10) → Bound = 25

Cập nhật MaxProfit = 25, chọn D, A, C

Tiếp tục các nhánh còn lại...

Node A → chọn C → D:

A:(3,5,8)  
C:(2,3,5)  
D: (5,4,12)

- Tổng:

Cost = 10, Personnel = 12 → OK

Profit = 25

Không thể chọn E (Cost = 11 > 10)

→ MaxProfit vẫn = 25

***Bước 4: Kết luận nghiệm tối ưu:***

Dự án được chọn: A, C, D

Tổng lợi nhuận: 8 + 5 + 12 = 25 triệu USD

Tổng chi phí: 3 + 2 + 5 = 10 triệu USD

Tổng nhân sự: 5 + 3 + 4 = 12 người

# **CHƯƠNG 3: CÀI ĐẶT VÀ THỰC NGHIỆM ĐÁNH GIÁ**

## **3.1 Môi trường cài đặt**

Nhóm đã lựa chọn ngôn ngữ lập trình C# kết hợp với ASP.NET Core để triển khai thuật toán Nhánh cận cho bài toán xếp ba lô đa chiều dưới dạng một ứng dụng web. Việc sử dụng ASP.NET Core mang lại khả năng tương tác trực quan thông qua giao diện web, cho phép người dùng dễ dàng nhập số ràng buộc, dự án, và theo dõi kết quả tối ưu một cách trực tiếp mà không cần thao tác dòng lệnh.

Môi trường phát triển sử dụng là Visual Studio 2022, một IDE hiện đại hỗ trợ đầy đủ cho ASP.NET Core MVC, tích hợp công cụ thiết kế giao diện Razor, debug hiệu quả và khả năng quản lý gói NuGet tiện lợi. Dự án được triển khai theo mô hình MVC (Model–View–Controller), giúp tách biệt rõ ràng phần xử lý dữ liệu, giao diện và điều khiển luồng chương trình.

Các thử nghiệm được thực hiện trên máy tính có cấu hình: CPU Intel Core i5, RAM 8GB, chạy Windows 11, hoàn toàn đáp ứng tốt nhu cầu xử lý thuật toán và hiển thị web. Ngoài ra, ứng dụng không sử dụng cơ sở dữ liệu, mà xử lý trực tiếp dữ liệu người dùng nhập vào trình duyệt, giúp đơn giản hóa cài đặt và sử dụng. Với nền tảng ASP.NET Core, nhóm có thể mở rộng dễ dàng các tính năng như lưu lịch sử tính toán, xuất kết quả ra Excel, hoặc triển khai lên máy chủ để người dùng sử dụng trực tuyến.

## **3.2. Chi tiết cài đặt thuật toán**

Trong đồ án này, nhóm đã triển khai thuật toán Nhánh cận (Branch and Bound) để giải quyết bài toán xếp ba lô đa chiều Multidimensional Knapsack Problem – MKP, dưới dạng một ứng dụng web ASP.NET Core MVC. Việc chuyển thuật toán từ dạng console sang ứng dụng web giúp tăng tính tương tác, trực quan hóa quá trình giải bài toán và phù hợp hơn với các yêu cầu hiện đại trong phát triển phần mềm.

**3.2.1. Biến thể bài toán và định dạng dữ liệu**

Bài toán được giải thuộc nhóm bài toán tối ưu tổ hợp, trong đó mỗi đối tượng (dự án) có thể được lựa chọn hoặc không lựa chọn (giá trị nhị phân 0 hoặc 1). Mỗi dự án có liên quan đến một hoặc nhiều ràng buộc – chẳng hạn như chi phí, nhân sự, thời gian, hoặc thiết bị – và khi được chọn, sẽ tiêu tốn các tài nguyên này ở các mức độ khác nhau. Mỗi ràng buộc đều có giới hạn tổng mà các dự án được chọn không được vượt quá. Mục tiêu của bài toán là tìm một tập hợp các dự án sao cho tổng lợi nhuận đạt được là lớn nhất, trong khi vẫn thỏa mãn toàn bộ các ràng buộc đã đặt ra.

Dữ liệu đầu vào của bài toán được nhập trực tiếp từ trình duyệt bởi người dùng. Người dùng có thể chỉ định số lượng ràng buộc (ví dụ: 2, 3, 4...) và số lượng dự án tùy ý. Đối với mỗi ràng buộc, người dùng nhập một giới hạn cụ thể, và đối với mỗi dự án, người dùng nhập tên dự án, các giá trị ràng buộc tương ứng, cùng với mức lợi nhuận kỳ vọng. Toàn bộ dữ liệu được gửi đến máy chủ thông qua HTTP POST và được xử lý bởi HomeController.

**3.2.2. Mô hình phần mềm ASP.NET Core MVC**

Ứng dụng được triển khai theo mô hình kiến trúc MVC (Model – View – Controller). Trong đó:

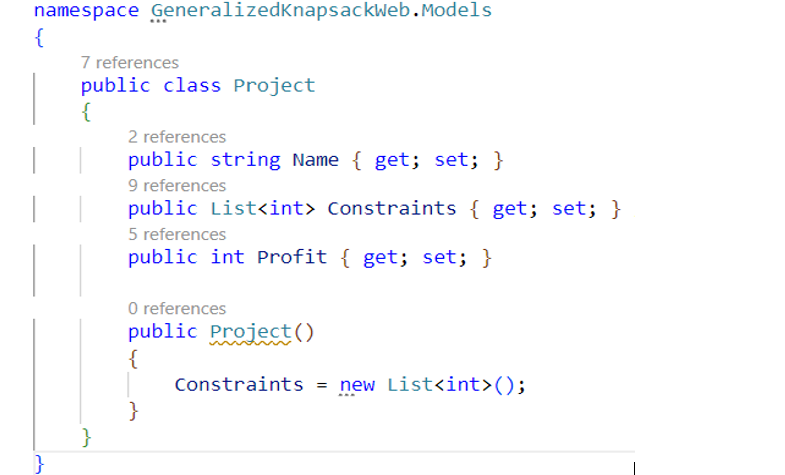
Model: biểu diễn cấu trúc dữ liệu của bài toán, bao gồm các lớp Project và Node.

View: được xây dựng bằng Razor Pages, đóng vai trò là giao diện người dùng để nhập liệu và hiển thị kết quả.

Controller: cụ thể là lớp HomeController, xử lý yêu cầu từ người dùng, thực thi thuật toán nhánh cận, và trả về kết quả.

Mô hình MVC giúp tổ chức mã nguồn rõ ràng, tách biệt phần xử lý dữ liệu, phần giao diện và phần điều khiển luồng, tạo điều kiện thuận lợi cho việc mở rộng, bảo trì và tái sử dụng mã.

**3.2.3. Lớp Project – Biểu diễn đối tượng dự án**

****

*Hình 3.1: Lớp Project – Biểu diễn đối tượng dự án*

Lớp Project được sử dụng để mô hình hóa thông tin của từng dự án đầu tư trong bài toán xếp ba lô đa chiều. Mỗi đối tượng Project đại diện cho một lựa chọn khả thi mà hệ thống cần đánh giá trong quá trình tối ưu hóa. Cấu trúc của lớp này bao gồm ba thuộc tính chính:

+ Name: tên của dự án, được sử dụng để phân biệt và hiển thị trong danh sách các lựa chọn. Đây là chuỗi định danh giúp người dùng nhận biết dễ dàng dự án nào đang được chọn.

+ Constraints: danh sách các giá trị ràng buộc của dự án, được thiết kế dưới dạng List<int>. Mỗi phần tử trong danh sách này đại diện cho một loại tài nguyên tiêu tốn khi thực hiện dự án, chẳng hạn như chi phí đầu tư, số lượng nhân sự cần thiết, thời gian triển khai, hoặc tài nguyên kỹ thuật cụ thể. Việc sử dụng danh sách động giúp lớp Project trở nên cực kỳ linh hoạt, cho phép hệ thống hỗ trợ bài toán với bất kỳ số lượng ràng buộc nào, do người dùng tùy ý nhập vào trong quá trình sử dụng giao diện web.

+ Profit: lợi nhuận dự kiến nếu dự án được lựa chọn. Đây là yếu tố then chốt mà thuật toán cần tối ưu hóa.

Thiết kế của lớp Project theo hướng tổng quát hóa giúp mở rộng dễ dàng phạm vi ứng dụng. Thay vì giới hạn trong các bài toán đơn giản với chỉ hai ràng buộc như chi phí và nhân sự, hệ thống có thể được áp dụng cho các bài toán thực tế phức tạp hơn với ba, bốn hoặc nhiều loại tài nguyên. Điều này rất quan trọng trong các mô hình hoạch định ngân sách, phân bổ tài nguyên, hoặc xây dựng chiến lược đầu tư, nơi mà mỗi phương án đều gắn với nhiều loại giới hạn cần cân nhắc.

**3.2.4. Lớp Node – Biểu diễn trạng thái trong cây tìm kiếm**



*Hình 3.2: Lớp Node – Biểu diễn trạng thái trong cây tìm kiếm*

Lớp Node là thành phần cốt lõi trong thuật toán nhánh cận, dùng để biểu diễn một trạng thái trong không gian tìm kiếm. Mỗi Node lưu trữ các thông tin sau:

+ Level: cấp độ hiện tại trong cây nhánh – tức là chỉ số của dự án đang xét.

+ TotalConstraints: tổng các giá trị ràng buộc đã sử dụng tính đến thời điểm node hiện tại.

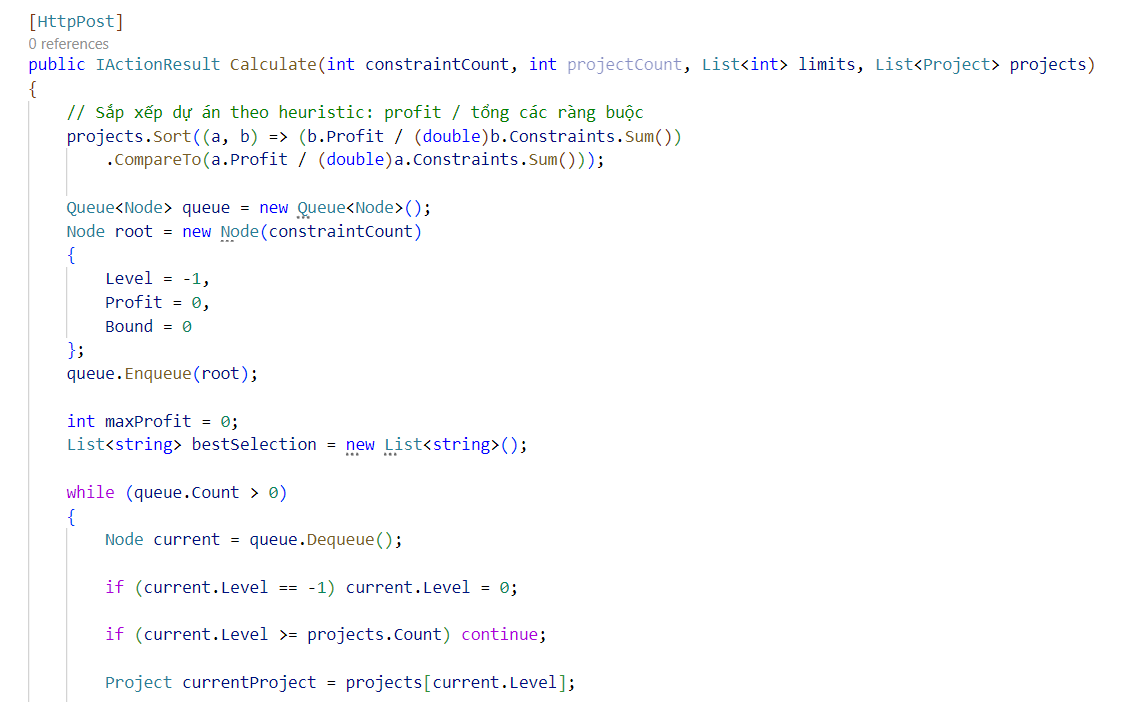
+ Profit: tổng lợi nhuận tương ứng với các dự án đã chọn.

+ Bound: cận trên – một giá trị ước lượng lợi nhuận tối đa có thể đạt được từ node hiện tại nếu tiếp tục mở rộng các nhánh bên dưới.

+ SelectedProjects: danh sách tên các dự án đã được chọn tại node đó.

Lớp Node có constructor đầu vào là constraintCount – số lượng ràng buộc – để đảm bảo các danh sách được khởi tạo đúng kích thước. Điều này cho phép thuật toán hoạt động linh hoạt với mọi cấu hình của bài toán, kể cả khi số ràng buộc là 2, 3, hoặc lớn hơn.

**3.2.5. Cài đặt thuật toán nhánh cận trong Controller**

****

*Hình 3.3: thuật toán nhánh cận trong HomeController*

Thuật toán được triển khai trong phương thức Calculate của lớp HomeController. Khi người dùng nhấn nút “Tính toán”, dữ liệu đầu vào sẽ được gửi đến phương thức này. Trình tự các bước xử lý thuật toán như sau:

Bước 1: Sắp xếp các dự án theo tỷ lệ hiệu suất, được tính bằng công thức:  
Hiệu suất = Lợi nhuận / Tổng các ràng buộc.

Việc sắp xếp giúp thuật toán ưu tiên xét các dự án có hiệu quả cao trước, góp phần rút ngắn cây tìm kiếm.

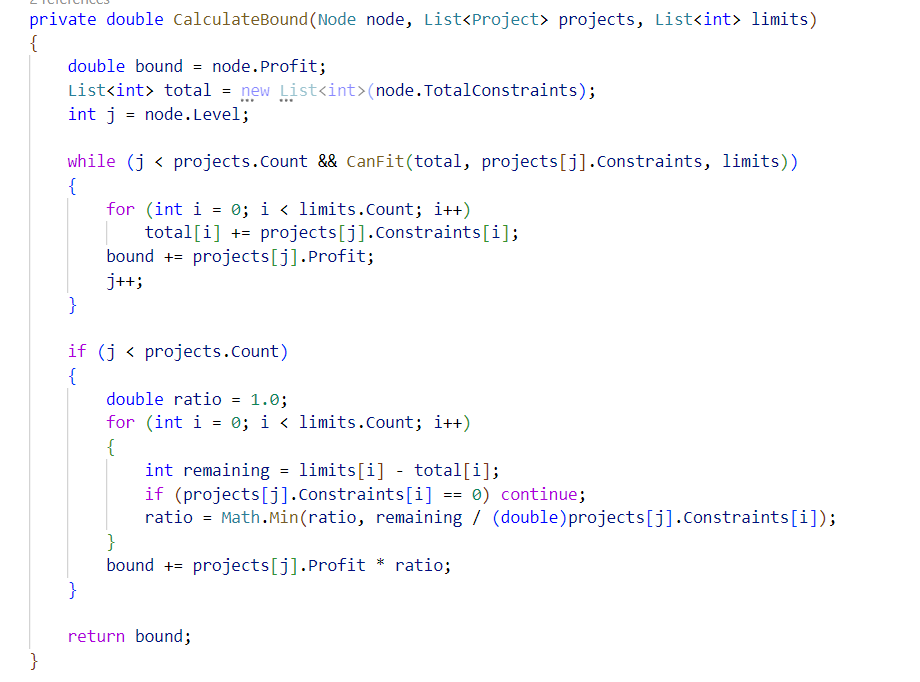
Bước 2: Khởi tạo node gốc của cây tìm kiếm, với Level = -1, Profit = 0, và các ràng buộc ban đầu là 0.

Bước 3: Duyệt theo chiều rộng (BFS). Mỗi node khi được lấy ra từ hàng đợi sẽ tạo ra hai nhánh:

Nhánh có chọn dự án hiện tại: cập nhật lại các ràng buộc, lợi nhuận, và danh sách dự án.

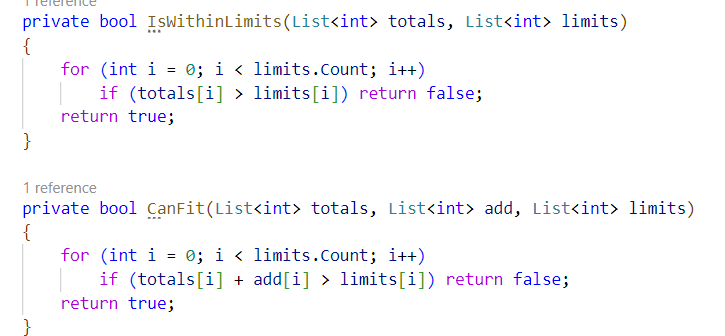
Nhánh không chọn dự án hiện tại: giữ nguyên trạng thái từ node cha.

Nếu node mới được tạo ra không vượt quá các giới hạn ràng buộc, nó sẽ được xét tiếp. Trong cả hai trường hợp, hệ thống sẽ tính Bound cho node đó bằng hàm CalculateBound.



Bước 4: Tính cận trên (Bound).

Cận trên được tính bằng cách cộng thêm lợi nhuận của các dự án còn lại mà vẫn có thể thêm vào mà không vượt quá giới hạn. Nếu không thể thêm toàn bộ dự án tiếp theo, thuật toán sẽ lấy một phần tỉ lệ lợi nhuận tương ứng (giống như phương pháp Fractional Knapsack) để đánh giá.



Bước 5: Cập nhật nghiệm tối ưu.

Nếu node nào có Profit lớn hơn giá trị hiện tại, hệ thống sẽ cập nhật lại maxProfit và lưu lại danh sách dự án tương ứng.

Cuối cùng, hệ thống hiển thị kết quả tối ưu trên giao diện web, bao gồm danh sách các dự án được chọn, tổng lợi nhuận đạt được, và tổng giá trị sử dụng của từng ràng buộc so với giới hạn cho phép.

## **3.3 Bộ dữ liệu thử nghiệm**

**3.3.1. Bộ dữ liệu 1 – Hai ràng buộc: Chi phí và Nhân sự**

Một bộ dữ liệu thử nghiệm được sử dụng để đánh giá thuật toán. Xét bài toán: Một công ty có ngân sách 10 triệu USD và 15 nhân sự cấp cao để phân bổ vào các dự án tiềm năng.

Dữ liệu đầu vào:

Dữ liệu bao gồm 5 dự án với thông tin như sau:

|  |  |  |
| --- | --- | --- |
| Dự án A: Chi phí 3 triệu USD | Nhân sự 5 người | Lợi nhuận 8 triệu USD. |
| Dự án B: Chi phí 4 triệu USD | Nhân sự 6 người | Lợi nhuận 10 triệu USD. |
| Dự án C: Chi phí 2 triệu USD | Nhân sự 3 người | Lợi nhuận 5 triệu USD. |
| Dự án D: Chi phí 5 triệu USD | Nhân sự 4 người | Lợi nhuận 12 triệu USD. |
| Dự án E: Chi phí 1 triệu USD | Nhân sự 2 người | Lợi nhuận 3 triệu USD. |

*Bảng 3.1. Bộ dữ liệu thử nghiệm một*

Bộ dữ liệu này đại diện cho một bài toán thực tế với quy mô nhỏ, giúp đánh giá hiệu quả của thuật toán Nhánh cận trong việc chọn dự án tối ưu.

**3.3.2. Bộ dữ liệu 2 – Ba ràng buộc: Chi phí, Nhân sự và Thời gian triển khai**

Trong bộ dữ liệu mở rộng thứ hai, nhóm đưa vào thêm một ràng buộc mới – thời gian triển khai dự án (đơn vị: tuần) – nhằm kiểm tra khả năng xử lý bài toán với số chiều tăng lên. Công ty vẫn có giới hạn về ngân sách và nhân sự, đồng thời cần đảm bảo tổng thời gian triển khai các dự án được chọn không vượt quá 18 tuần.

Thông tin chi tiết cho 5 dự án trong bộ dữ liệu này như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dự án** | **Chi phí**  **(triệu USD)** | **Nhân sự (người)** | **Thời gian (tuần)** | **Lợi nhuận**  **(triệu USD)** |
| A | 2 | 4 | 6 | 7 |
| B | 3.0 | 5 | 5 | 9 |
| C | 1 | 3 | 4 | 4 |
| D | 4.0 | 6 | 7 | 11 |
| E | 2.0 | 2 | 3 | 5 |

*Bảng 3.. Bộ dữ liệu thử nghiệm hai*

Giới hạn tổng cho ba ràng buộc trong bộ dữ liệu này được thiết lập như sau:

+ Ngân sách tối đa: 9 triệu USD

+ Tổng nhân sự tối đa: 15 người

+ Tổng thời gian triển khai: 18 tuần

Mục tiêu của thuật toán là lựa chọn tổ hợp các dự án sao cho tổng lợi nhuận đạt giá trị cao nhất, đồng thời không vượt quá giới hạn của cả ba ràng buộc.

Bộ dữ liệu này giúp kiểm tra hiệu quả hoạt động của hệ thống trong trường hợp phức tạp hơn – khi không gian tìm kiếm tăng kích thước do có thêm chiều ràng buộc. Qua thử nghiệm, hệ thống vẫn có thể xử lý tốt, trả về kết quả tối ưu nhanh chóng mà không cần can thiệp thủ công.

## **3.4 Đánh giá kết quả thực nghiệm**

**3.4.1. Bộ dữ liệu 1 – Hai ràng buộc: Chi phí và Nhân sự**

Kết quả thực nghiệm cho thấy thuật toán Nhánh cận hoạt động hiệu quả với bộ dữ liệu thử nghiệm. Khi chạy chương trình với dữ liệu trên, thuật toán trả về nghiệm tối ưu: Dự án được chọn: A, C, D

Tổng lợi nhuận: 8 + 5 + 12 = 25 triệu USD

Tổng chi phí: 3 + 2 + 5 = 10 triệu USD

Tổng nhân sự: 5 + 3 + 4 = 12 người



*Hình 3.4: Kết quả thực nghiệm bộ dữ liệu một*

Đánh giá hiệu quả:

Thuật toán đã tìm được nghiệm tối ưu trong thời gian ngắn, nhờ cơ chế cắt tỉa các nhánh không khả thi hoặc không tiềm năng. Với 5 dự án, không gian tìm kiếm tối đa là = 32 tổ hợp, nhưng thuật toán chỉ xét một phần nhỏ các nhánh, giảm đáng kể thời gian tính toán. Kết quả cũng cho thấy thuật toán tận dụng tốt tài nguyên: sử dụng hết ngân sách và gần hết nhân sự, đồng thời đạt lợi nhuận cao nhất có thể.

**3.4.2. Bộ dữ liệu 2 – Ba ràng buộc: Chi phí, Nhân sự và Thời gian triển khai**

Bộ dữ liệu thử nghiệm thứ hai được mở rộng với một ràng buộc bổ sung là thời gian triển khai, bên cạnh hai ràng buộc ban đầu là chi phí và nhân sự. Điều này giúp kiểm tra khả năng mở rộng của thuật toán khi số chiều của bài toán tăng lên. Cụ thể, giới hạn tổng của ba ràng buộc được đặt như sau:

+ Ngân sách tối đa: 9 triệu USD

+ Tổng nhân sự tối đa: 15 người

+ Tổng thời gian triển khai tối đa: 18 tuần

Khi chạy chương trình với bộ dữ liệu này, thuật toán trả về nghiệm tối ưu như sau:



*Hình 3.5: Kết quả thực nghiệm bộ dữ liệu hai*

Đánh giá hiệu quả:

Thuật toán tiếp tục cho thấy khả năng xử lý hiệu quả khi không gian tìm kiếm tăng lên. Trong trường hợp có 3 ràng buộc, mỗi dự án được xét dưới nhiều điều kiện hơn, đồng nghĩa với việc số tổ hợp khả thi tăng đáng kể. Tuy nhiên, nhờ áp dụng cơ chế cắt tỉa thông minh dựa trên cận trên (bound), thuật toán đã loại bỏ sớm những nhánh không khả thi, giảm thiểu đáng kể số node cần xét.

Kết quả thu được thể hiện sự phân bổ tài nguyên hợp lý: cả ba ràng buộc đều được tận dụng ở mức gần tối đa mà không bị vi phạm. Điều này chứng minh rằng thuật toán không chỉ đúng về mặt lý thuyết mà còn thực sự hiệu quả khi triển khai với dữ liệu thực tế. Việc lựa chọn ba dự án P2, P3 và P5 cho thấy thuật toán có thể nhận diện được tổ hợp tối ưu giữa các phương án nhỏ, thay vì chọn những phương án có lợi nhuận lớn nhưng vượt ràng buộc.

**3.4.3. So sánh và đánh giá**

Khi so sánh giữa hai bộ dữ liệu thử nghiệm, có thể rút ra một số nhận xét như sau:

Độ chính xác: Trong cả hai trường hợp, hệ thống đều tìm được nghiệm tối ưu thỏa mãn toàn bộ ràng buộc, chứng minh tính đúng đắn của thuật toán.

Hiệu suất: Với quy mô dữ liệu nhỏ và trung bình, thời gian xử lý gần như tức thì (< 1 giây). Điều này cho thấy thuật toán Nhánh cận là lựa chọn hiệu quả cho các bài toán tối ưu hóa tổ hợp không quá lớn.

Khả năng mở rộng: Việc tăng số lượng ràng buộc từ 2 lên 3 không làm ảnh hưởng đến độ chính xác của thuật toán, cho thấy tính tổng quát cao của hệ thống. Việc sử dụng cấu trúc List<int> trong mô hình dữ liệu giúp hệ thống dễ dàng thích ứng với các bài toán có nhiều ràng buộc hơn trong tương lai (4, 5 hoặc nhiều hơn).

Tính ứng dụng thực tế: Hệ thống có thể dễ dàng áp dụng trong các bài toán ra quyết định của doanh nghiệp, như chọn danh mục đầu tư, phân bổ ngân sách, hoặc lập kế hoạch dự án theo nguồn lực.

## **3.5. So sánh với các phương pháp khác**

- Một cách tiếp cận khác của Nhánh cận là sử dụng chiến lược tìm kiếm theo chiều sâu (DFS), xét các nhánh theo thứ tự cố định mà không ưu tiên cận trên. Cách này có thể nhanh hơn với bài toán nhỏ, nhưng dễ bị chậm khi không gian nghiệm lớn, vì không tận dụng được cận trên để cắt tỉa hiệu quả.

- So với thuật toán quy hoạch động (Dynamic Programming), Nhánh cận có ưu điểm khi không gian nghiệm không quá lớn, vì không cần tạo bảng lưu trữ lớn.

- So với thuật toán tham lam (Greedy), Nhánh cận đảm bảo tìm được nghiệm tối ưu, trong khi tham lam chỉ cho nghiệm xấp xỉ. Thuật toán tham lam có thể chọn dự án theo tỷ lệ hiệu quả, nhưng thường không cân bằng tốt giữa các ràng buộc, dẫn đến kết quả kém hơn (ví dụ: chọn D, E, C nhưng bỏ qua A, tổng lợi nhuận thấp hơn).

# **KẾT LUẬN**

Qua quá trình nghiên cứu, thiết kế và hiện thực hóa, nhóm đã hoàn thành mục tiêu đề ra của đồ án là xây dựng một hệ thống giải bài toán xếp ba lô đa chiều 0-1 bằng thuật toán Nhánh cận. Cụ thể, các kết quả nổi bật mà nhóm đã đạt được bao gồm:

Tìm hiểu và phân tích thành công bài toán Multidimensional Knapsack Problem (MKP) – một bài toán tối ưu hóa tổ hợp phức tạp, thường gặp trong thực tế như lập kế hoạch đầu tư, phân bổ ngân sách, lựa chọn dự án.

Hiện thực thuật toán Nhánh cận với khả năng mở rộng linh hoạt theo số lượng ràng buộc, cho phép giải quyết cả các bài toán có 2, 3 hoặc nhiều chiều ràng buộc mà không cần thay đổi cấu trúc chương trình.

Cài đặt thuật toán theo hướng tối ưu, sử dụng chiến lược cắt tỉa hợp lý để rút ngắn không gian tìm kiếm, tăng hiệu suất xử lý mà vẫn đảm bảo tìm được nghiệm tối ưu.

Thử nghiệm thành công trên nhiều bộ dữ liệu khác nhau, bao gồm cả các bài toán có nhiều ràng buộc. Kết quả cho thấy hệ thống luôn tìm ra tổ hợp dự án có lợi nhuận cao nhất trong phạm vi giới hạn tài nguyên.

Triển khai hoàn chỉnh hệ thống dưới dạng ứng dụng web với kiến trúc rõ ràng, tách biệt giữa phần logic và phần giao diện, thuận tiện cho việc mở rộng hoặc tích hợp trong tương lai.

**DANH SÁCH CÔNG VIỆC NHÓM**

|  |  |  |
| --- | --- | --- |
| **STT** | **HỌ TÊN** | **CÔNG VIỆC** |
| 1 | Nguyễn Danh Hiếu -NT | Chương 1: 1.5, 1.6, 1.7  Chương 2: 2.4  Chương 3: 3.2 |
| 2 | Lê Đức Anh | Chương 1: 1.1, 1.2  Chương 2: 2.3.  Chương 3: 3.2, 3.5 |
| 3 | Trần Đại Hiệp | Chương 1: 1.4, 1.8  Chương 2: 2.2  Chương 3: 3.4 |
| 4 | Hà văn Hiệp | Chương 1: 1.3, 1.9  Chương 2: 2.1  Chương 3: 3.1 |

# **TÀI LIỆU THAM KHẢO**

[1] Fréville, A. (2004). The Multidimensional Knapsack Problem: Structure and Algorithms. INFORMS Journal on Computing, 16(3), 280-297.

[2] Kellerer, H., Pferschy, U., & Pisinger, D. (2004). Knapsack Problems. Springer.

[3] Puchinger, J., Raidl, G. R., & Pferschy, U. (2010). The Multidimensional Knapsack Problem: A Survey. European Journal of Operational Research, 206(1), 1-14.

[4] Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. Econometrica, 28(3), 497-520.

[5] Martello, S., & Toth, P. (1990). Knapsack Problems: Algorithms and Computer Implementations. Wiley.

[6] Chu, P. C., & Beasley, J. E. (1998). A Genetic Algorithm for the Multidimensional Knapsack Problem. Journal of Heuristics, 4(1), 63-86