**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH**


# - IPL ALGORITHM -

## Nghiên cứu ứng dụng thuật toán IPL và cây nhị phân để tối ưu sắp xếp phòng học trong thi cử

**Hội đồng:** Khoa học Máy tính

**Giảng viên hướng dẫn:** TS. Lê Xuân Bách

**Sinh viên:** Lê Minh Hiếu

**Mã số sinh viên:** 2550053

TP. HỒ CHÍ MINH, THÁNG 01 /2026

# Contents

# 1 Yêu cầu bài toán

Bài toán được xét trên một tập dữ liệu gồm các phòng thi, trong đó mỗi phòng thi được đặc trưng bởi các thông số sau: *sức chứa tối đa*, *số lượng thí sinh tham gia thực tế*, và *số lượng chỗ trống còn lại*.

Mục tiêu của bài toán là xây dựng một thuật toán, gọi là *thuật toán Merging*, nhằm giảm thiểu số lượng phòng thi được sử dụng trong khi vẫn đảm bảo tất cả thí sinh được bố trí hợp lệ.

Cụ thể, *thuật toán Merging* hoạt động dựa trên việc xét từng cặp phòng thi $A$ và $B$. Nếu số lượng thí sinh của phòng $B$ nhỏ hơn hoặc bằng số chỗ trống còn lại của phòng $A$, thì phòng $B$ có thể được *gộp* (merge) vào phòng $A$. Khi đó, toàn bộ thí sinh của phòng $B$ sẽ được chuyển sang phòng $A$, và phòng $B$ sẽ được xem là *đóng*, tức không còn được sử dụng cho bất kỳ mục đích nào khác và bị loại bỏ hoàn toàn khỏi tập dữ liệu. Việc này giúp giảm số lượng phòng thi đang hoạt động đi một đơn vị.

Một phòng thi có thể tiếp nhận (merge) nhiều phòng khác nhau, với điều kiện *sức chứa còn lại* cho phép. Phòng thi được giữ lại để tổ chức thi thực tế được gọi là *phòng mở*, trong khi các phòng bị gộp sẽ bị loại bỏ khỏi hệ thống.

Do đó, yêu cầu của bài toán là thiết kế một thuật toán hiệu quả nhằm xác định cách gộp các phòng thi sao cho *số lượng phòng mở còn lại là nhỏ nhất*, đồng thời xác định *giá trị tối ưu của số phòng thi cần sử dụng* tương ứng.

# 2 Thuật toán đề xuất

## 2.1 Lập trình tuyến tính nguyên(IPL)

Lập trình tuyến tính nguyên (*Integer Linear Programming – ILP*) là một lớp bài toán tối ưu hóa, trong đó hàm mục tiêu và các ràng buộc đều có dạng tuyến tính, đồng thời các biến quyết định bị ràng buộc phải nhận giá trị nguyên.

ILP thường được sử dụng để mô hình hóa các bài toán tối ưu có bản chất rời rạc, nơi các quyết định mang tính lựa chọn nhị phân hoặc số lượng nguyên, chẳng hạn như chọn hoặc không chọn một đối tượng, hoặc kích hoạt hay vô hiệu hóa một phương án.

Cụ thể, một bài toán ILP thỏa mãn các đặc điểm sau:

- Hàm mục tiêu có dạng tuyến tính theo các biến quyết định,
- Tập các ràng buộc được biểu diễn bằng các bất đẳng thức hoặc đẳng thức tuyến tính,
- Các biến quyết định bị ràng buộc thuộc tập số nguyên.

## 2.2 Dạng toán học tổng quát

Một bài toán ILP chuẩn có thể được biểu diễn dưới dạng:

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m, \\ & x_j \in \mathbb{Z}, \quad j = 1, \ldots, n. \end{aligned}$$

Trong đó:

- $x_j$ là biến quyết định, biểu diễn các lựa chọn cần được xác định trong bài toán,
- $c_j$ là hệ số của hàm mục tiêu, phản ánh mức độ đóng góp hoặc chi phí tương ứng với biến $x_j$,
- $a_{ij}$ là hệ số ràng buộc, thể hiện mức độ ảnh hưởng của biến $x_j$ lên ràng buộc thứ $i$,
- $b_i$ là hằng số xác định giới hạn của ràng buộc thứ $i$.

Hàm mục tiêu nhằm tối ưu (tối thiểu hóa hoặc tối đa hóa) một đại lượng tuyến tính của các biến quyết định, trong khi các ràng buộc đảm bảo nghiệm thu được thỏa mãn các điều kiện hợp lệ của bài toán.

# 3 Mô hình hóa phương pháp IPL cho bài toán chia phòng

**Tập chỉ số.**
Xét tập phòng thi $I = \{1, 2, \ldots, n\}$, trong đó mỗi chỉ số $i \in I$ tương ứng với một phòng.
**Dữ liệu (tham số đầu vào).**
Với mỗi $i \in I$:

- $a_i \in \mathbb{Z}_{>0}$: sức chứa tối đa của phòng $i$.
- $b_i \in \mathbb{Z}_{\geq 0}$: số thí sinh hiện đang được xếp tại phòng $i$ (xem như một "nhóm" không tách).

Giả thiết dữ liệu hợp lệ: $0 \leq b_i \leq a_i,\ \forall i \in I$.
**Biến quyết định.**

- $y_j \in \{0, 1\},\ \forall j \in I$:

$$y_j = \begin{cases} 1, & \text{nếu phòng } j \text{ được giữ lại (tiếp tục sử dụng),} \\ 0, & \text{nếu phòng } j \text{ bị đóng.} \end{cases}$$

- $x_{ij} \in \{0, 1\},\ \forall i, j \in I$:

$$x_{ij} = \begin{cases} 1, & \text{nếu toàn bộ thí sinh của phòng } i \text{ được chuyển và ngồi tại phòng } j \text{ sau khi gộp,} \\ 0, & \text{ngược lại.} \end{cases}$$

**Hàm mục tiêu.**
Mục tiêu là tối thiểu hóa số phòng được giữ lại (tương đương tối đa hóa số phòng bị đóng):

$$\min \sum_{j \in I} y_j. \tag{1}$$

**Các ràng buộc.**

$$\sum_{j \in I} x_{ij} = 1, \qquad \forall i \in I, \tag{C1}$$

$$x_{ij} \leq y_j, \qquad \forall i, j \in I, \tag{C2}$$

$$x_{jj} = y_j, \qquad \forall j \in I, \tag{C3}$$

$$\sum_{i \in I} b_i \, x_{ij} \leq a_j \, y_j, \qquad \forall j \in I. \tag{C4}$$

**Diễn giải.**

Ràng buộc (C1) đảm bảo mỗi "nhóm" $i$ được gán đúng một phòng đích (không bị mất và không bị tách). Ràng buộc (C2) yêu cầu chỉ được gán vào các phòng được giữ lại. Ràng buộc (C3) thể hiện điều kiện "không dùng phòng đã đóng làm trung gian": nếu phòng $j$ được giữ ($y_j = 1$) thì nhóm $j$ bắt buộc ở lại chính phòng $j$; nếu phòng $j$ bị đóng ($y_j = 0$) thì nhóm $j$ phải chuyển sang phòng khác. Ràng buộc (C4) đảm bảo tổng số thí sinh được gán vào mỗi phòng giữ lại không vượt quá sức chứa của phòng đó.

# 4 Ý tưởng giải bài toán IPL

## 4.1 Lý luận thực tế

Bài toán được giải bằng cách xây dựng một mô hình Lập trình tuyến tính nguyên (ILP) kết hợp với chiến lược tìm kiếm theo nhánh và cận (*Branch-and-Bound*). Quy trình giải có thể được mô tả ở mức khái niệm như sau.

**Bước 1: Xác định cận dưới và cận trên**

Trước hết, bài toán xác định cận dưới (*Lower Bound – LB*) và cận trên (*Upper Bound – UB*) cho số lượng phòng thi cần sử dụng.

Cận dưới được ước lượng dựa trên tổng số thí sinh và sức chứa của các phòng, bằng cách sắp xếp các phòng theo sức chứa giảm dần và lần lượt cộng dồn cho đến khi tổng sức chứa vượt quá tổng số thí sinh. Giá trị này biểu diễn số phòng tối thiểu có thể cần thiết trong điều kiện lý tưởng.

Cận trên được xác định thông qua một phương án khả thi bất kỳ thỏa mãn các ràng buộc của bài toán, dù chưa tối ưu. Phương án này cung cấp một giới hạn trên cho số phòng cần sử dụng. Khoảng cách giữa LB và UB càng nhỏ thì không gian tìm kiếm càng được thu hẹp, từ đó làm giảm chi phí tính toán.

**Bước 2: Tìm kiếm nghiệm tối ưu**

Từ khoảng giá trị $[LB, UB)$, thuật toán tiến hành kiểm tra khả năng tồn tại nghiệm thỏa mãn các ràng buộc với từng số lượng phòng được mở. Quá trình này được thực hiện thông qua việc phân nhánh trên các biến quyết định nhị phân, tương ứng với việc mở hoặc đóng từng phòng thi.

Tại mỗi nhánh, các ràng buộc của bài toán được áp dụng đồng thời với các cơ chế suy diễn và lan truyền ràng buộc (*constraint propagation*) nhằm loại bỏ các cấu hình không khả thi. Các nhánh mà cận dưới cục bộ vượt quá cận trên hiện tại sẽ bị loại bỏ khỏi không gian tìm kiếm.

Quá trình phân nhánh và cắt tỉa được lặp lại cho đến khi xác định được số lượng phòng nhỏ nhất thỏa mãn toàn bộ các ràng buộc của bài toán, từ đó thu được nghiệm tối ưu.

## 4.2 Pseudo-code

**Chiến lược giải ILP theo hai giai đoạn**

Quá trình giải bài toán ILP được tổ chức thành hai giai đoạn chính. Giai đoạn thứ nhất sử dụng bài toán nới lỏng tuyến tính (LP relaxation) nhằm xác định cận dưới và thu thập thông tin heuristic phục vụ cho việc phân nhánh. Giai đoạn thứ hai thực hiện tìm kiếm trên cây nhị phân kết hợp với các cơ chế cắt tỉa nhằm xác định nghiệm tối ưu.

**Giai đoạn 1: LP Relaxation và sinh heuristic phân nhánh**

Tại gốc của cây tìm kiếm, bài toán ILP ban đầu được nới lỏng bằng cách loại bỏ ràng buộc nguyên trên các biến quyết định, cho phép chúng nhận giá trị thực. Bài toán LP relaxation thu được được giải để xác định một nghiệm tối ưu phân số.

Giá trị tối ưu của bài toán LP relaxation được sử dụng như một cận dưới toàn cục (*Lower Bound – LB*) cho bài toán ILP. Đồng thời, nghiệm phân số này cung cấp thông tin về mức độ "gần nguyên" của từng biến quyết định.

Dựa trên nghiệm LP relaxation, các biến có giá trị phân số được sử dụng để xây dựng heuristic phân nhánh. Cụ thể, các biến có giá trị gần 0 hoặc 1 được ưu tiên trong quá trình phân nhánh, do chúng mang lại khả năng nhanh chóng dẫn đến nghiệm nguyên hoặc loại bỏ các nhánh không khả thi. Thông tin này được sử dụng để ước lượng xác suất lựa chọn nhánh trong cây tìm kiếm nhị phân.

**Giai đoạn 2: Tìm kiếm trên cây nhị phân và cắt nhánh**

Từ nghiệm ban đầu, thuật toán tiến hành xây dựng cây tìm kiếm nhị phân bằng cách lần lượt cố định các biến quyết định về các giá trị nguyên khả dĩ. Mỗi nút trong cây tương ứng với một bài toán con, trong đó một tập con các biến đã được cố định.

Thứ tự phân nhánh được quyết định dựa trên heuristic thu được từ giai đoạn LP relaxation. Tại mỗi nút, bài toán LP relaxation tương ứng được giải để xác định cận dưới cục bộ. Nếu cận dưới này vượt quá cận trên hiện tại (*Upper Bound – UB*), nút đó sẽ bị loại bỏ khỏi không gian tìm kiếm.

Ngoài ra, các ràng buộc của bài toán cùng với các quy tắc lan truyền ràng buộc (*constraint propagation*) được áp dụng nhằm phát hiện sớm các trường hợp không khả thi và cắt bỏ các nhánh tương ứng. Quá trình phân nhánh và cắt tỉa được lặp lại cho đến khi xác định được nghiệm nguyên tốt nhất, tương ứng với số lượng phòng thi tối ưu.

**Algorithm 1** Giải bài toán ILP theo hai giai đoạn: LP Relaxation và Branch-and-Bound

**Require:** Mô hình ILP $\mathcal{P}$ gồm các biến quyết định $x$ và tập ràng buộc $\mathcal{C}$; nghiệm khả thi ban đầu (nếu có)

**Ensure:** Nghiệm nguyên tối ưu $x^\star$ và giá trị hàm mục tiêu tương ứng $UB$

**Khởi tạo:**
$UB \leftarrow +\infty$, $x^\star \leftarrow$ rỗng
Tạo nút gốc $N_0$ với tập biến chưa cố định; $\mathcal{C}(N_0) \leftarrow \mathcal{C}$
Khởi tạo hàng đợi ưu tiên $\mathcal{Q} \leftarrow \{N_0\}$ (sắp xếp theo cận dưới nhỏ nhất)
**while** $\mathcal{Q}$ không rỗng **do**
  Lấy ra một nút $N$ từ $\mathcal{Q}$
  **Giai đoạn 1: LP relaxation để xác định cận và heuristic**
  Giải bài toán LP relaxation của $\mathcal{P}$ với các ràng buộc $\mathcal{C}(N)$
  **if** bài toán LP không khả thi **then**
    **Bỏ qua nút này** (cắt nhánh do không khả thi)
  **end if**
  Gọi $z_{LP}(N)$ là giá trị hàm mục tiêu của LP, $\hat{x}(N)$ là nghiệm LP thu được
  Đặt $LB(N) \leftarrow z_{LP}(N)$
  **if** $LB(N) \geq UB$ **then**
    **Bỏ qua nút này** (cắt nhánh theo cận)
  **end if**
  Trích xuất heuristic phân nhánh từ nghiệm $\hat{x}(N)$:
  Chọn một biến $x_k$ có giá trị phân số $\hat{x}_k(N) \notin \mathbb{Z}$ theo quy tắc heuristic
  Xác định thứ tự ưu tiên nhánh dựa trên độ gần 0 hoặc 1 của $\hat{x}_k(N)$
  **Kiểm tra tính nguyên**
  **if** $\hat{x}(N)$ là nghiệm nguyên hợp lệ **then**
    $UB \leftarrow z_{LP}(N)$
    $x^\star \leftarrow \hat{x}(N)$ (cập nhật nghiệm tốt nhất)
    **Tiếp tục vòng lặp**
  **end if**
  **Giai đoạn 2: Phân nhánh nhị phân và cắt tỉa**
  Tạo hai nút con bằng cách cố định biến $x_k$:
  $N^{(0)}$: thêm ràng buộc $x_k = 0$
  $N^{(1)}$: thêm ràng buộc $x_k = 1$
  Áp dụng lan truyền ràng buộc (*constraint propagation*) cho từng nút con
  Loại bỏ sớm các nút con không khả thi nếu phát hiện mâu thuẫn
  **if** $N^{(0)}$ còn khả thi **then**
    Đưa $N^{(0)}$ vào $\mathcal{Q}$ với độ ưu tiên dựa trên $LB$ và heuristic
  **end if**
  **if** $N^{(1)}$ còn khả thi **then**
    Đưa $N^{(1)}$ vào $\mathcal{Q}$ với độ ưu tiên dựa trên $LB$ và heuristic
  **end if**
**end while**
**return** $x^\star, UB$

## 4.3 Giải thích pseudo-code

Quy trình giải bài toán được triển khai theo cơ chế *Branch-and-Bound* trên mô hình ILP, trong đó *LP relaxation* được sử dụng để xây dựng cận dưới và trích xuất thông tin heuristic phục

vụ cho quyết định phân nhánh.

Tại nút gốc, bài toán ILP ban đầu (chưa cố định bất kỳ biến quyết định nào) được nới lỏng bằng cách loại bỏ ràng buộc nguyên, từ đó thu được bài toán *LP relaxation*. Việc giải bài toán LP này cho phép xác định một nghiệm tối ưu $\hat{x}$ (thường là nghiệm phân số) cùng với giá trị hàm mục tiêu tương ứng $z_{LP}$. Trong bài toán tối thiểu hóa, $z_{LP}$ được sử dụng như cận dưới (*Lower Bound – LB*) cho nghiệm tối ưu nguyên. Đồng thời, nghiệm $\hat{x}$ cung cấp cơ sở để xây dựng heuristic phân nhánh, cụ thể là lựa chọn biến phân nhánh và thứ tự ưu tiên xét nhánh tiếp theo.

Tại mỗi nút con trong cây tìm kiếm, do các quyết định phân nhánh trước đó áp đặt thêm các ràng buộc mới (ví dụ $x_k = 0$ hoặc $x_k = 1$), bài toán con tại nút đó có miền nghiệm khác so với nút cha. Vì vậy, *LP relaxation* phải được giải lại tại từng nút để xác định cận dưới cục bộ $LB(N)$ và cập nhật nghiệm $\hat{x}(N)$ phục vụ heuristic phân nhánh cho nút hiện tại.

Nếu tại một nút, nghiệm LP relaxation $\hat{x}(N)$ đồng thời là nghiệm nguyên và thỏa mãn các ràng buộc của bài toán, nghiệm này được xem như một nghiệm khả thi của ILP và được sử dụng để cập nhật cận trên hiện tại (*Upper Bound – UB*) cũng như nghiệm tốt nhất đang lưu giữ. Ngược lại, nếu nghiệm LP là phân số, thuật toán tiếp tục phân nhánh để tạo các bài toán con.

Cơ chế cắt tỉa được áp dụng dựa trên quan hệ giữa cận dưới và cận trên: nếu $LB(N) \geq UB$ thì nút $N$ không thể dẫn đến nghiệm tốt hơn nghiệm hiện tại và do đó bị loại bỏ khỏi không gian tìm kiếm. Bên cạnh đó, các kỹ thuật lan truyền ràng buộc (*constraint propagation*) được sử dụng nhằm suy diễn các hệ quả logic từ các ràng buộc đã cố định, qua đó phát hiện sớm các trường hợp không khả thi và cắt bỏ các nhánh tương ứng.

Quá trình phân nhánh, đánh giá cận và cắt tỉa được lặp lại cho đến khi không còn nút nào có khả năng cải thiện cận trên hiện tại, khi đó nghiệm lưu giữ tương ứng với nghiệm tối ưu của bài toán.

# 5 Ứng dụng thuật toán vào thực tiễn

Trong phạm vi nghiên cứu của ứng dụng này, do những hạn chế về thời gian cũng như mức độ triển khai thuật toán, báo cáo chưa tập trung vào việc hiện thực hóa thuật toán dưới dạng phần mềm hoàn chỉnh. Thay vào đó, nghiên cứu chủ yếu đề xuất và phân tích một thuật toán nhằm giải quyết bài toán tối ưu hóa việc phân bổ phòng thi.

Dữ liệu đầu vào được sử dụng trong quá trình đánh giá được lưu trữ trong tệp `output.xlsx`, bao gồm các thông tin cơ bản như mã phòng, sức chứa của từng phòng thi và số lượng thí sinh tham gia trong ngày thi tương ứng. Bên cạnh đó, một bảng tổng hợp phụ được xây dựng nhằm so sánh số lượng phòng trước và sau khi áp dụng thuật toán, trong đó số phòng ban đầu là 73 và số phòng sau tối ưu được giảm xuống còn 67.

Do giới hạn về thời gian thực nghiệm, thuật toán chưa được triển khai và chạy trực tiếp bằng chương trình chuyên dụng, mà được mô phỏng và thực thi thông qua một mô hình ngôn ngữ lớn (*Large Language Model – LLM*). Kết quả cho thấy số lượng phòng thi được giảm xấp xỉ 8,22%, phản ánh tiềm năng đáng kể của phương pháp đề xuất. Đặc biệt, khi kích thước tập dữ liệu tăng lên và số lượng phòng ban đầu lớn hơn, khả năng tận dụng việc gộp phòng được kỳ vọng sẽ mang lại hiệu quả tối ưu cao hơn.

Trong định hướng nghiên cứu tiếp theo, có thể xem xét việc phát triển một phần mềm chuyên dụng cho bài toán sắp xếp và tối ưu hóa phòng học. Mô hình này không chỉ dừng lại ở các ràng buộc liên quan đến phòng thi, mà còn có thể mở rộng để tích hợp thêm các yếu tố khác như môn học, mã lớp, lịch thi và các ràng buộc nghiệp vụ liên quan.