Các Kỹ thuật Phân hoạch Lồi cho Hoạch định Chuyển động Robot

Nhóm 2 Khoa

Ngày 18 tháng 10 năm 2025

Mục tiêu Buổi trình bày

Các câu hỏi cần làm rõ:

- Phân hoạch lồi là gì? Tại sao nó quan trọng trong hoạch định chuyển động?
- Độ đo xấp xỉ của một vùng lồi được đo lường ra sao?
- Trong bài báo, Thuật toán IRIS được sử dụng cụ thể ở đâu, trong trường hợp nào? Có những thuật toán nào khác được dùng cho từng trường hợp không?
- Các phương pháp được áp dụng trong môi trường 2D như thế nào?

Mục tiêu chính

Buổi trình bày hôm nay sẽ tập trung giải đáp các câu hỏi trên, cung cấp cái nhìn tổng quan về các kỹ thuật phân hoạch lồi và ứng dụng của chúng.

Nội dung

Định nghĩa Phân hoạch Lồi

Định nghĩa

Phân hoạch lồi (Convex Decomposition) là quá trình chia một hình học phức tạp (không lồi) thành một tập hợp các **vùng con lồi** sao cho hợp của chúng khôi phục lại hình ban đầu và các vùng con không chồng lấn nhau (ngoại trừ biên chung).

Công thức

Với một miền hình học $P \subset \mathbb{R}^n$, một *phân hoạch lồi* của P là tập hợp $\{P_1, P_2, \ldots, P_k\}$ sao cho:

$$P = \bigcup_{i=1}^{k} P_i, \quad P_i \cap P_j = \partial P_i \cap \partial P_j \text{ với mọi } i \neq j,$$

và mỗi P_i là một tập lồi. ∂P_i là biên của vùng P_i.

Mục tiêu và Điều kiện Tồn tại

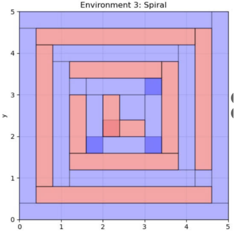
Mục tiêu của Phân hoạch Lồi:

- Tối thiểu hóa số lượng vùng lồi
- Giảm thiểu chồng chéo
- Tránh vùng quá “mảnh” hoặc “dài”

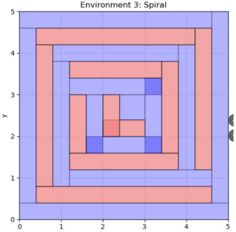
Định lý: Tồn tại Phân hoạch Lồi

Bất kỳ đa giác đơn nào đều có thể được phân hoạch thành tập hợp các vùng lồi.
Lý do: Mọi đa giác đơn có thể tam giác hóa, mỗi tam giác là tập lồi.
Hạn chế: Tạo ra số lượng lớn các thành phần, không thực tế cho ứng dụng.

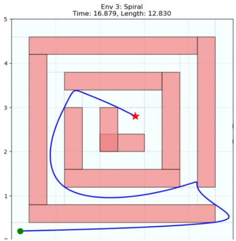
Ví dụ về Phân hoạch Lồi Trong 2D

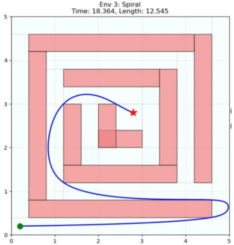


(a) Case 1



(b) Case 2





Phân loại Phương pháp

Exact Convex Decomposition (ECD)

- Thành phần hoàn toàn lồi
- Tái tạo chính xác
- NP-hard
- Số lượng vùng lớn
- Chủ yếu có trong lý thuyết

Approximate Convex Decomposition (ACD)

- Gần lồi với dung sai τ
- Hiệu quả tính toán cao
- Giảm số lượng vùng
- Kiểm soát mức chi tiết
- Ứng dụng thực tế

Các Khái niệm Cơ bản

Định nghĩa các thành phần

- **Vỏ Lồi (H_P):** Tập lồi nhỏ nhất chứa P

$$P \text{ lồi} \Leftrightarrow P = H_P$$

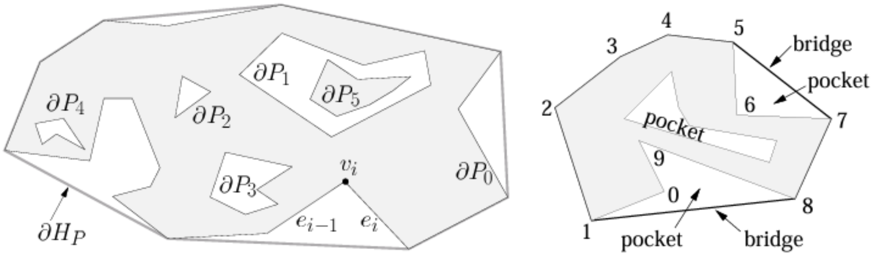
- **Khía (Notches):** Đỉnh không trên H_P, góc trong $> 180°$
- **Cầu (Bridges):** Cạnh của ∂H_P nối hai đỉnh không liền kề

$$\text{BRIDGES}(P) = \partial H_P \setminus \partial P$$

- **Túi (Pockets):** Chuỗi cạnh không thuộc ∂H_P

$$\text{POCKETS}(P) = \partial P \setminus \partial H_P$$

Minh họa Các Khái niệm



Hình: Vỏ lồi, khía, cầu và túi trong đa giác không lồi

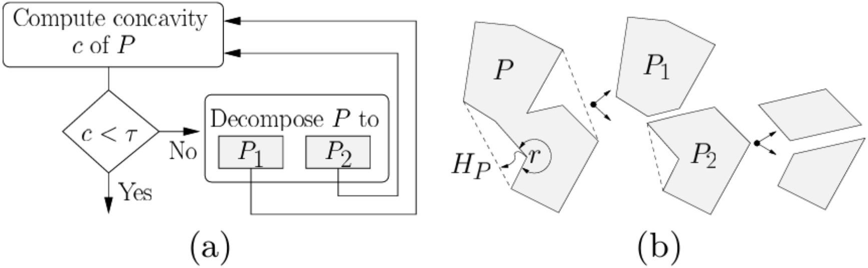
Thuật toán ACD - Ý tưởng

Chiến lược chia để trị đệ quy:

1. **Đo độ lõm:** Xác định đỉnh lõm có độ lõm lớn nhất
2. **Kiểm tra dung sai:** Nếu độ lõm $< \tau$, kết thúc
3. **Giải quyết đỉnh lõm:** Thêm đường cắt, chia đa giác
4. **Đệ quy:** Áp dụng cho các thành phần mới

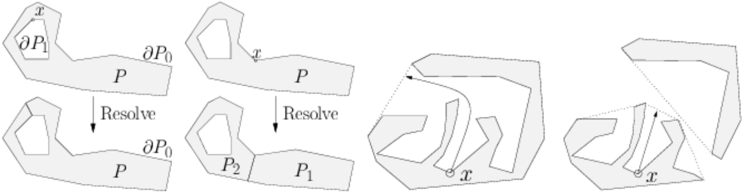
Độ phức tạp: $O(nr^2)$, với n là số đỉnh, r là số đỉnh lõm

Minh họa thuật toán



Hình: Quá trình đệ quy tiếp tục cho đến khi đạt đến giới hạn độ lõm đầu vào của người dùng τ.

Ví dụ về Giải quyết Độ lõm



(a) (b) (c)

Các trường hợp xử lý:

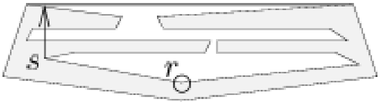
(a) Gộp biên vào đa giác

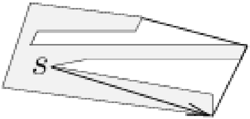
(b) Tách đa giác thành hai

(c) Độ lõm thay đổi sau phân rã

Quá trình tiếp tục đến khi độ lõm $\leq \tau$

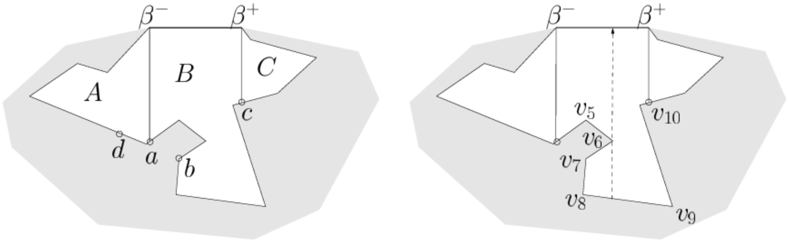
Các Thước đo Độ lõm





SL-Concavity: Khoảng cách Euclid đến cầu nối (bridge).

+ Nhanh
– Không đảm bảo tính đơn điệu



(a) (b)

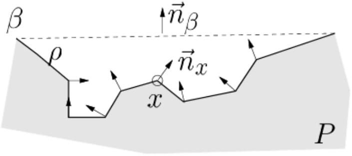
SP-Concavity: Đường đi ngắn nhất trong túi (pocket) đến cầu nối tương ứng.

− Chậm hơn

\+ Đảm bảo tính đơn điệu

\+ Xử lý tất cả đặc điểm lõm quan trọng

Các Thước đo Độ lõm



(a) (b)

H-Concavity: Lai ghép SL và SP - Mặc định dùng SL, chuyển sang SP khi phát hiện không đơn điệu (kiểm tra: $n_\beta \cdot n_i < 0$).

IRIS - Ý tưởng Chính

IRIS: Iterative Regional Inflation by Semidefinite Programming
Câu hỏi cốt lõi: Với điểm mầm (seed), vùng lồi lớn nhất là gì?

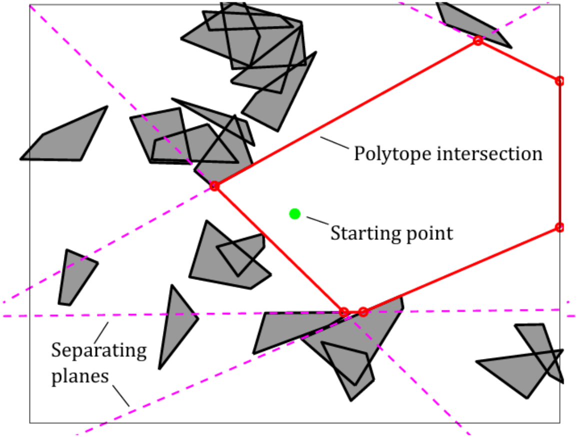
Quy trình lặp 4 bước:

1. Khởi tạo hình cầu rất nhỏ tại seed
2. Tìm siêu phẳng phân cách (QP)
3. Tìm ellipsoid nội tiếp cực đại (SDP)
4. Lặp lại đến hội tụ

Output: Một vùng lồi lớn nhất từ seed point

IRIS Bước 2: Tìm Siêu phẳng Phân cách

Mục tiêu: Tìm siêu phẳng phân tách ellipsoid với obstacles



Giải pháp - Thuật toán tham lam:

1. Bắt đầu từ obstacle gần ellipsoid nhất
2. Tìm điểm trên obstacle gần nhất
3. Tính siêu phẳng $a_j^\top x \geq b_j$
4. Kiểm tra tất cả obstacles khác o_k:
   Nếu $a_j^\top v \geq b_j \; \forall v \in o_k$
   $\Rightarrow$ Loại o_k (đã bị phân tách)
5. Lặp lại cho obstacles còn lại

Ký hiệu: o_j: obstacle j
v: đỉnh của obstacle
a_j, b_j: tham số siêu phẳng

IRIS - Tìm obstacle Gần nhất

Biến đổi không gian:

$\Rightarrow$ Biến đổi về không gian quả cầu đơn vị:

- Ellipsoid: $\mathcal{E} = \{C\tilde{x} + d : \|\tilde{x}\|_2 \leq 1\}$
- Quả cầu: $\tilde{\mathcal{E}} = \{\tilde{x} : \|\tilde{x}\| \leq 1\}$
- Obstacle: $\tilde{o}_j = \text{ConvexHull}(\tilde{v}_{j,1}, \ldots, \tilde{v}_{j,m})$
- Với $\tilde{v}_{j,k} = C^{-1}(v_{j,k} - d)$

Khi này khoảng cách đến ellipsoid tương đương với khoảng cách đến gốc tọa độ trong không gian biến đổi.

Ký hiệu: C: ma trận xác định hình dạng ellipsoid, d: tâm ellipsoid, $v_{j,k}$: đỉnh thứ k của obstacle j trong không gian gốc, $\tilde{v}_{j,k}$: đỉnh sau biến đổi

IRIS - Bài toán QP cho Điểm Gần nhất

QP Problem

Tìm điểm gần nhất đến gốc tọa độ trong không gian biến đổi:

$$\begin{aligned} \min_{\tilde{x}, w} \quad & \|\tilde{x}\|^2 \\ \text{s.t.} \quad & [\tilde{v}_{j,1}, \tilde{v}_{j,2}, \ldots, \tilde{v}_{j,m}] w = \tilde{x} \\ & \sum_{i=1}^{m} w_i = 1 \\ & w_i \geq 0 \end{aligned}$$

$\tilde{x}$ là tổ hợp lồi của các đỉnh $\tilde{v}_{j,k}$ gần gốc nhất

Biến đổi ngược: $x^* = C\tilde{x}^* + d$ là điểm gần ellipsoid nhất trên obstacle o_j

Ký hiệu: $\tilde{x}$: điểm cần tìm trong không gian biến đổi, $w = [w_1, \ldots, w_m]$: hệ số tổ hợp lồi, $\tilde{v}_{j,k}$: đỉnh thứ k của obstacle j sau biến đổi, m: số đỉnh của obstacle

Biểu diễn ngược của Ellipsoid:

$$\mathcal{E} = \{x : (x - d)^\top C^{-1} C^{-\top} (x - d) \leq 1\}$$

Tính pháp tuyến

Gradient của hàm rào cản tại x^*:

$$a_j = \nabla_x [(x - d)^\top C^{-1} C^{-\top} (x - d)]|_{x^*}$$

$$= 2C^{-1} C^{-\top} (x^* - d)$$

Hệ số tự do:

$$b_j = a_j^\top x^*$$

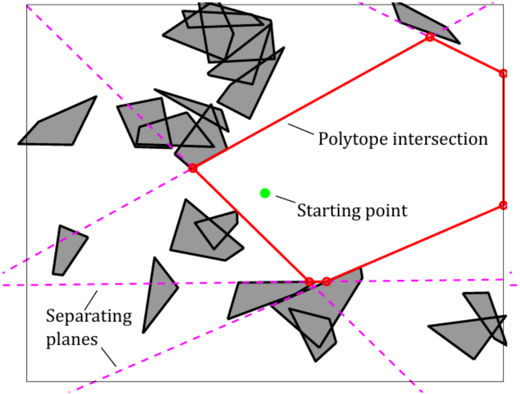
Kết quả:
Siêu phẳng $a_j^\top x = b_j$ tiếp xúc với $\mathcal{E}_{\alpha^*}$ tại x^* và phân tách $\mathcal{E}$ khỏi o_j

Chọn chiều: $a_j^\top x \geq b_j \ \forall x \in o_j$

IRIS Bước 2: Tìm Siêu phẳng

Ellipsoid Definition

$$\mathcal{E} = \{C\tilde{x} + d : \|\tilde{x}\|_2 \leq 1\}$$



Biến đổi không gian

Từ ellipsoid sang quả cầu đơn vị:

$$\tilde{x} = C^{-1}(x - d)$$

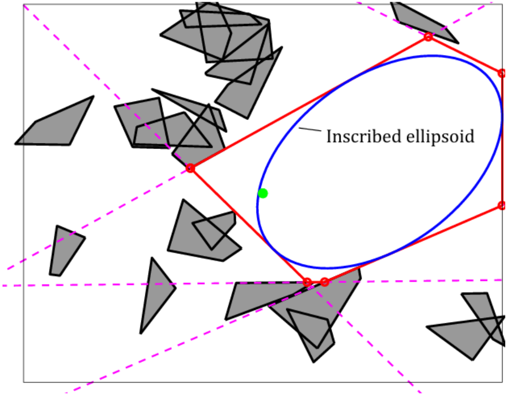
QP Problem

$$\begin{aligned} \min_{\tilde{x}, w} \quad & \|\tilde{x}\|^2 \\ \text{s.t.} \quad & [\tilde{v}_{j,i}]w = \tilde{x} \\ & w_i \geq 0 \end{aligned}$$

Pháp tuyến:

$$a_j = 2C^{-1}C^{-\top}(x^* - d)$$
$$b_j = a_j^\top x^*.$$

IRIS Bước 3: Ellipsoid Nội tiếp Cực đại



Đầu vào: Tập siêu phẳng $\{a_i^\top x \geq b_i\}_{i=1}^m$

SDP Problem

$$\begin{aligned} \max_{C,d} \quad & \log\det C \\ \text{s.t.} \quad & \|a_i^\top C\|_2 + a_i^\top d \leq b_i \\ & C \succeq 0 \end{aligned}$$

Tối đa hóa thể tích ellipsoid nội tiếp đa diện

Ràng buộc:
$\|a_i^\top C\|_2 + a_i^\top d \leq b_i$ đảm bảo ellipsoid nằm trong nửa không gian $a_i^\top x \leq b_i$

IRIS - Phân tích Độ phức tạp

Độ phức tạp mỗi vòng lặp:

SeparatingHyperplanes: $O(m \cdot n \cdot k)$

- m: số lượng obstacles
- n: số đỉnh trung bình mỗi obstacle
- k: số vòng lặp QP để tìm điểm gần nhất (CVXPY solve)

InscribedEllipsoid: $O(h^3)$

- SDP với h ràng buộc siêu phẳng
- Interior point method: $O(h^3)$ mỗi vòng lặp SDP

IRIS - Độ phức tạp Tổng thể

Công thức tổng quát

$$O(T \cdot (m \cdot n \cdot k + h^3))$$

Với T: số vòng lặp IRIS (thường 3-10 vòng)

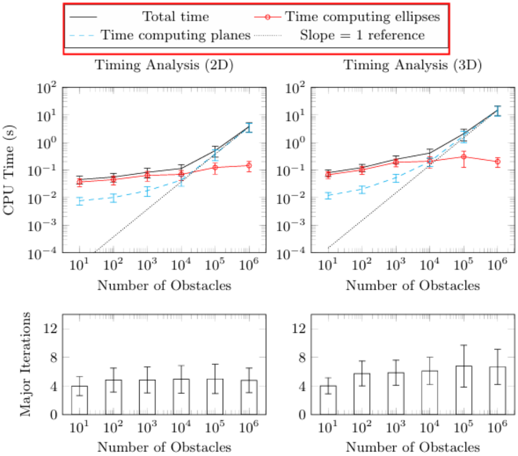
Giải thích các thành phần:

- T**:** Số vòng lặp IRIS - thường hội tụ nhanh (3-10)
- $m \cdot n \cdot k$**:** Chi phí tìm siêu phẳng phân cách
- h^3**:** Chi phí giải SDP cho ellipsoid nội tiếp cực đại

Nhận xét:

- Độ phức tạp tăng tuyến tính theo số obstacles (m)
- Phần SDP (h^3) thường là cố định do số lượng siêu phẳng không thay đổi nhiều
- Hội tụ nhanh (T nhỏ) làm thuật toán khả thi trong thực tế

IRIS - Độ phức tạp Thực nghiệm



Hình: Thời gian thực thi IRIS tăng tuyến tính với số lượng obstacles.

IRIS - Đặc tính

Thuộc tính chính:

- **Loại:** Phương pháp xấp xỉ
- **Hội tụ:** 4-8 vòng lặp

Xử lý obstacles không lồi:

- **Work space:** Bao lồi hoặc phân hoạch trước
- **Configuration space:** IRIS-NP, C-IRIS

VCC - Giải quyết Vấn đề Seeding

Hạn chế của IRIS:

- IRIS gia đình các obstacles là lồi, nên cần 1 bước tiền xử lý trước
- Cần chọn seed points tốt (người dùng tự chọn)

Giải pháp VCC:

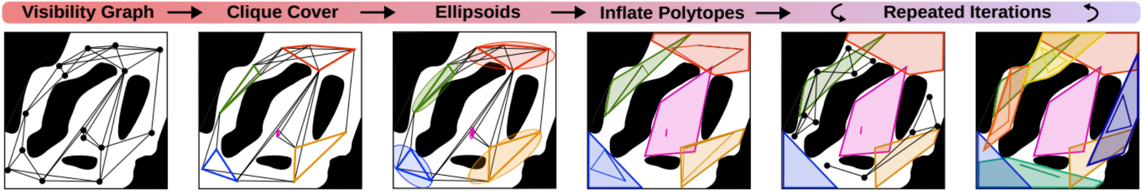
- Tự động hóa quy trình seeding
- Sử dụng cấu trúc toàn cục của không gian tự do

VCC - Giải quyết Vấn đề Seeding

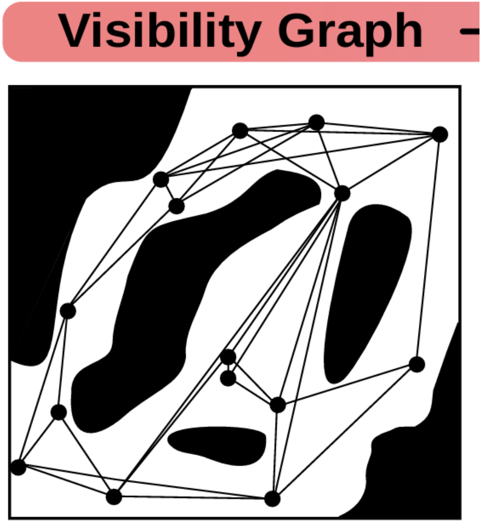
Quy trình 4 bước:

1. Lấy mẫu n điểm trong C_{free}
2. Xây dựng đồ thị khả kiến (visibility graph)
3. Tìm lớp phủ clique (MAXCLIQUE + ILP)
4. Khởi tạo ellipsoid và lấp đầy (IRIS-like)

Ý tưởng cốt lõi: Điểm "nhìn thấy" nhau $\Rightarrow$ có khả năng cùng vùng lồi



VCC - Bước 1: Lấy mẫu & Xây dựng Đồ thị Khả kiến

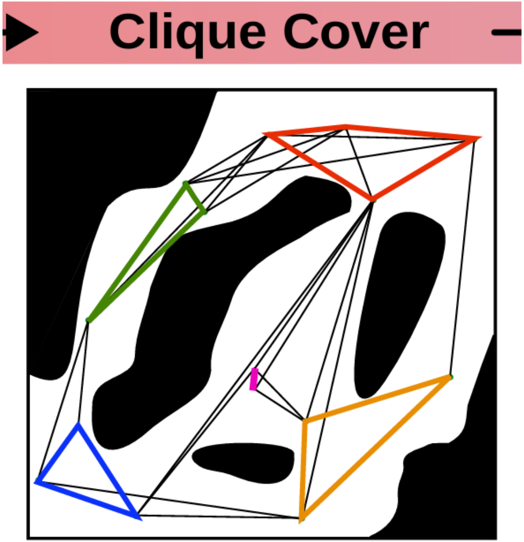


Visibility Graph Construction:

- Lấy mẫu n điểm ngẫu nhiên trong C_{free}
- Tạo cạnh giữa hai điểm nếu đoạn thẳng nối chúng không va chạm
- Xây dựng đồ thị khả kiến (visibility graph)

Định nghĩa

Visibility graph: Đồ thị có cạnh nối hai đỉnh khi đoạn thẳng nối chúng nằm hoàn toàn trong không gian tự do



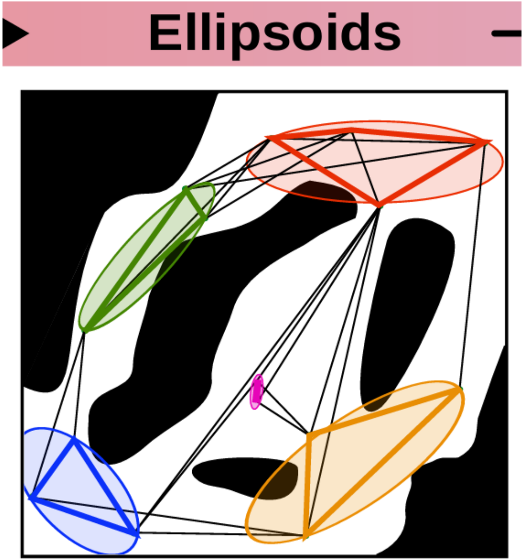
Clique Cover Problem:

- Tìm tập hợp các clique lớn (đồ thị con đầy đủ)
- Sử dụng MAXCLIQUE algorithm
- Giải bài toán Integer Linear Programming (ILP)

Ý tưởng cốt lõi:
Các điểm "nhìn thấy" nhau có khả năng nằm trong cùng một vùng lồi

Con đường thẳng = an toàn

VCC - Bước 3: Khởi tạo Ellipsoid



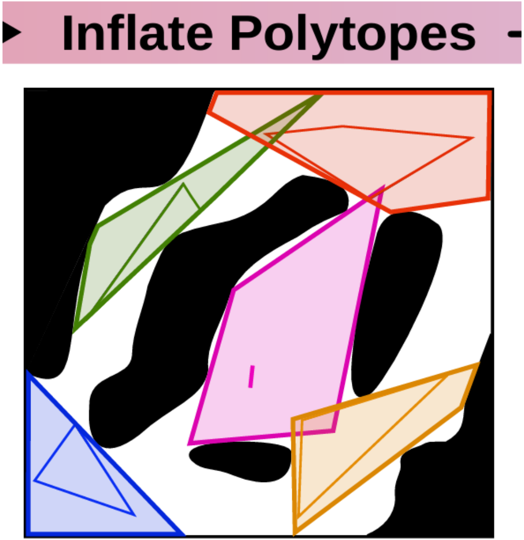
Ellipsoid Initialization:

- Tính ellipsoid có thể tích nhỏ nhất bao quanh tất cả điểm trong clique
- Cung cấp tâm (seed point) cho bước tiếp theo
- Xác định hình dạng ban đầu (các trục chính)

Lợi ích:

Ellipsoid đã được "thông báo" về hình học cục bộ

$\Rightarrow$ Hiệu quả hơn IRIS tiêu chuẩn

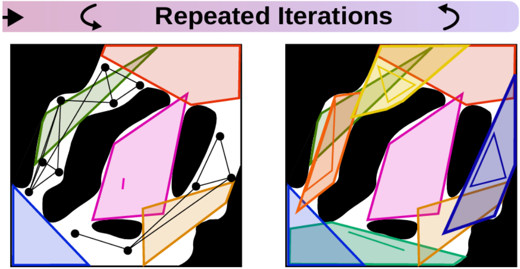


Inflation Process:

- Sử dụng thuật toán tương tự IRIS
- Một vòng lặp duy nhất (thay vì nhiều vòng)
- Tạo đa diện lồi cuối cùng từ ellipsoid

Hiệu quả cao hơn IRIS:
Ellipsoid ban đầu đã được “thông báo” về hình học cục bộ
$\Rightarrow$ Loại bỏ nhiều vòng lặp tốn kém

VCC - Bước 5: Lặp lại và Hội tụ



Iteration Process:

- Lấy mẫu mới từ không gian trống còn lại
- Lặp lại các bước 1-4 cho vùng chưa phủ
- Tiếp tục đến khi đạt độ phủ α đủ

Điều kiện dừng - Quá trình kết thúc khi:

- Đạt độ phủ mong muốn α
- Không còn không gian đáng kể để phủ
- Số vùng lồi đã đủ cho ứng dụng

VCC - Đặc tính và Ưu điểm

Đặc tính:

- **Loại:** Phương pháp xấp xỉ
- **Mục tiêu:** Tối thiểu hóa số vùng với độ phủ α
- **Phương pháp:** Lai ghép hình học và ILP

Ưu điểm:

- Tự động hóa hoàn toàn
- Không phụ thuộc hình dạng obstacles
- Sử dụng kiểm tra va chạm chung

Vai trò trong Graph of Convex Sets

Khuôn khổ GCS:
Yêu cầu phân hoạch lồi trước khi hoạch định chuyển động

Vai trò phân hoạch:

- Các vùng lồi = đỉnh trong đồ thị G
- Chất lượng phân hoạch $\Rightarrow$ hiệu suất GCS
- Phân hoạch kém $\Rightarrow$ đồ thị phức tạp

Chuỗi phụ thuộc

Thuật toán phân hoạch tiên tiến $\Rightarrow$ GCS khả thi $\Rightarrow$ Giải quyết bài toán DOF cao

Phương pháp theo Môi trường & Độ phức tạp

2D đơn giản:

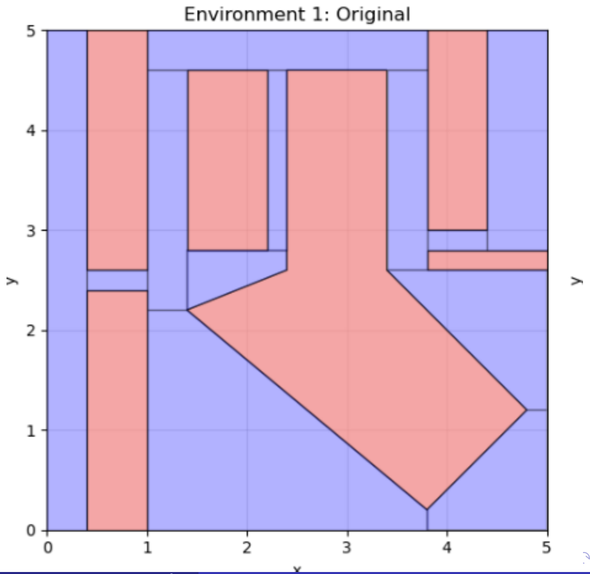
- Exact Decomposition
- Đa giác: 12 vùng lồi
- Maze: Grid 50×50

3D Quadrotor:

- Axis-aligned boxes
- Thu nhỏ theo robot radius

High-DOF (7/14-DOF):

- IRIS Approximate
- Seed qua Inverse Kinematics
- Dual arms: Tích Descartes



Ví dụ 2D - Phân hoạch Chính xác

```python
# vertices of the safe regions
vertices = [
    np.array([
        [ 0.4,  0.],
        [ 0.4,  5.],
        [ 0. ,  5.],
        [ 0. ,  0.]
    ]),
    np.array([
        [0.4, 2.4],
        [1. , 2.4],
        [1. , 2.6],
        [0.4, 2.6]
    ]),
    np.array([
        [1.4, 2.2],
        [1.4, 4.6],
        [1. , 4.6],
        [1. , 2.2]
    ]),
```

Two-Dimensional Example
12 vùng lồi V-polytope

Chuyển đổi sang H-polytope:

$$\mathcal{R}_i = \{x \in \mathbb{R}^2 : A_i x \leq b_i\}$$

Sử dụng `ConvexHull` algorithm

Kết quả: Phù hợp hoàn hảo với GCS framework

Ví dụ Maze - Grid Decomposition

```python
regions = []
edges = []
for x in range(maze_size):
    for y in range(maze_size):
        regions.append(HPolyhedron.MakeBox([x, y], [x+1., y+1.]))
        C = y + x * maze.ny
        if not maze.map[x][y].walls['N']:
            edges.append((C, C + 1))
        if not maze.map[x][y].walls['S']:
            edges.append((C, C - 1))
        if not maze.map[x][y].walls['E']:
            edges.append((C, C + maze.ny))
        if not maze.map[x][y].walls['W']:
            edges.append((C, C - maze.ny))
```

Maze Planning
Lưới đều 50×50

Mỗi ô $=$ vùng lồi Q_i

Graph connectivity:

- Dựa trên cấu trúc tường
- Hai ô kề không tường $\Rightarrow$ có cạnh
- Đơn giản nhưng kém hiệu quả khi kích thước lưới lớn

Ví dụ 3D Quadrotor - Chi tiết Box Decomposition

Thuật toán chia nhỏ không gian:

- **Phòng (Indoor):** Hộp duy nhất cho mỗi ô

$$\text{size} = 2.5 - (\text{wall\_offset} + \text{quad\_radius})$$

- **Ngoài trời (Outdoor):**
  - *Không cây:* Hộp lớn bao phủ toàn bộ
  - *Có cây:* 4 hộp xung quanh (trên, dưới, trái, phải)
- **Khe hở tường:**
  - *Cửa:* Hộp hẹp cao
  - *Cửa sổ:* 1-2 hộp nhỏ
  - *Không tường:* Hộp lớn nối phòng

High-DOF: KUKA 7-DOF & Dual Arms 14-DOF

Giải pháp IRIS-based:

- Sử dụng `IrisInConfigurationSpace` (C-IRIS)
- Xử lý chướng ngại vật không lồi trong C-space
- Sinh đa diện lồi Q_i quanh seed points
- Song song hóa cho nhiều seed khác nhau

Dual Arms: Kết hợp không gian

Tích Descartes hoặc hợp lồi của các vùng từ từng cánh tay riêng lẻ

Kết luận

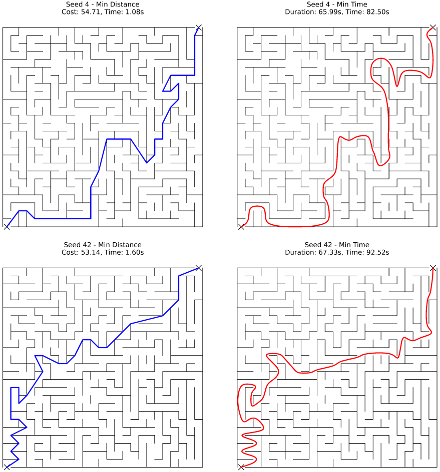
Ứng dụng theo môi trường:

- 2D: Exact decomposition
- 3D: Box/Grid decomposition
- High-DOF: IRIS-based approximation

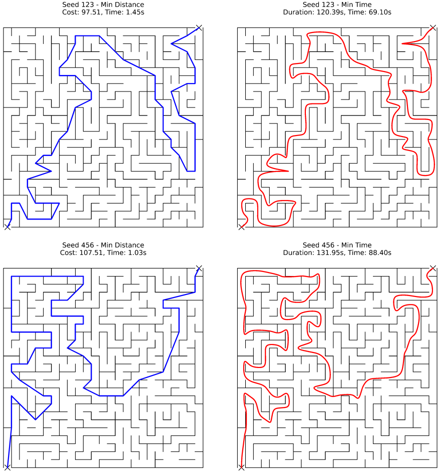
Kết nối với GCS

GCS framework phụ thuộc hoàn toàn vào chất lượng phân hoạch

Demo 1: Maze

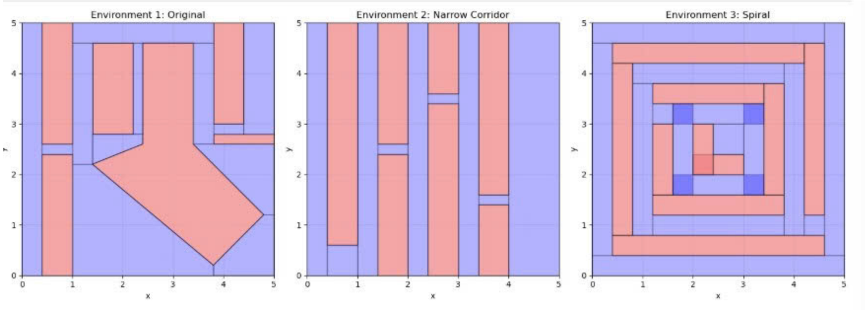


Hình:



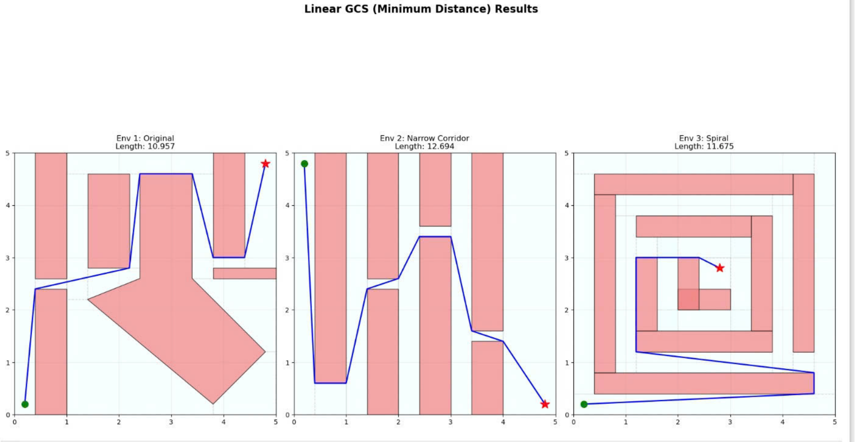
Hình:

Demo 2: 2D Env



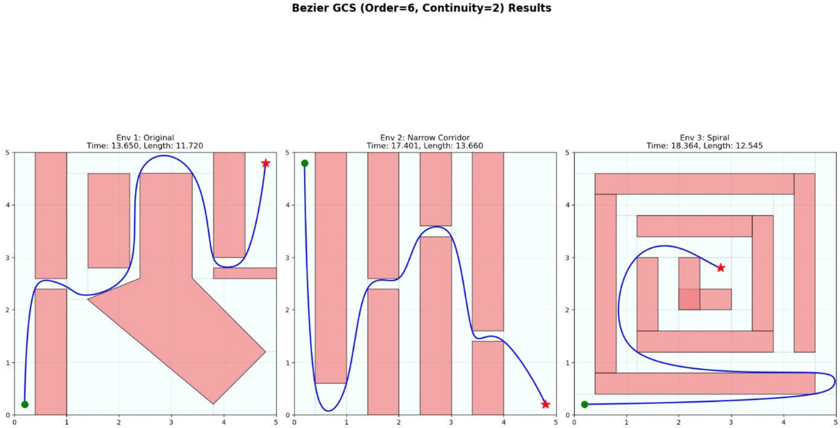
Hình:

Demo 2: 2D Env



Hình:

Demo 2: 2D Env



Hình:

Demo 3: Drone

Video demo.

Demo 4: Manipulation robot

Video demo.