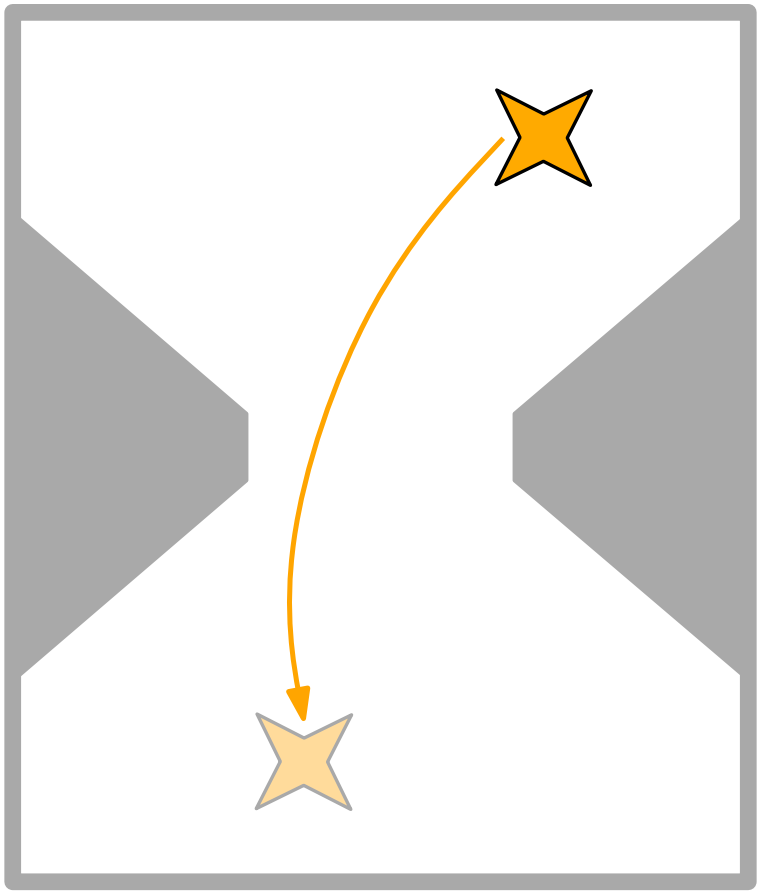
Multi-robot motion planning

Kiril Solovey

Tel-Aviv University, Israel

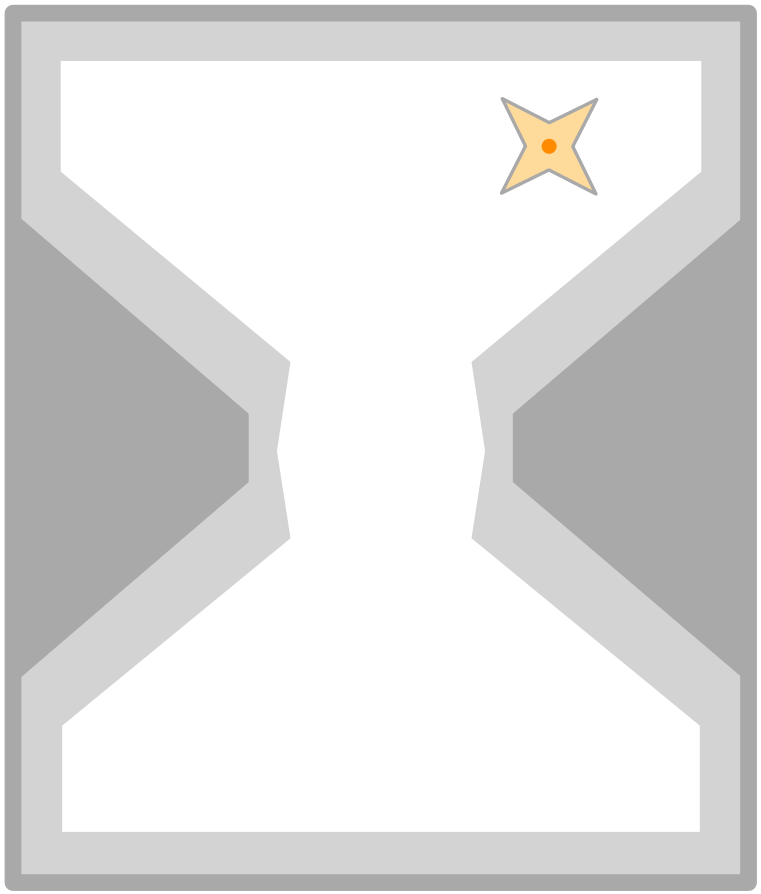
Computational Geometry

What is Motion Planning?



Motion Planning

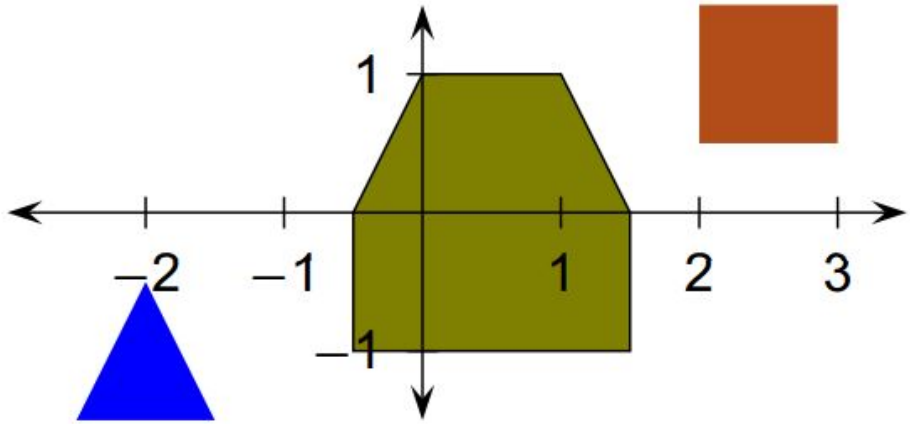
- **Workspace:** Description of the world consisting of robot and obstacles (2D/3D)
- **Degrees of Freedom (dofs):** Minimal number of parameters to uniquely define the state (position) of robot
- **Configuration Space (C-Space):** Space of parameters that define the robot's configuration (dD)
- **Free Space:** Set of collision-free configurations



Planning for a translating polygon using Minkowski sums

Definition 1 (Minkowski sum)

Let P and Q be two point sets in $\mathbb{R}^d$. The *Minkowski sum* of P and Q, denoted by $P \bigoplus Q$, is the point set $\{p + q | p \in P, q \in Q\}$.

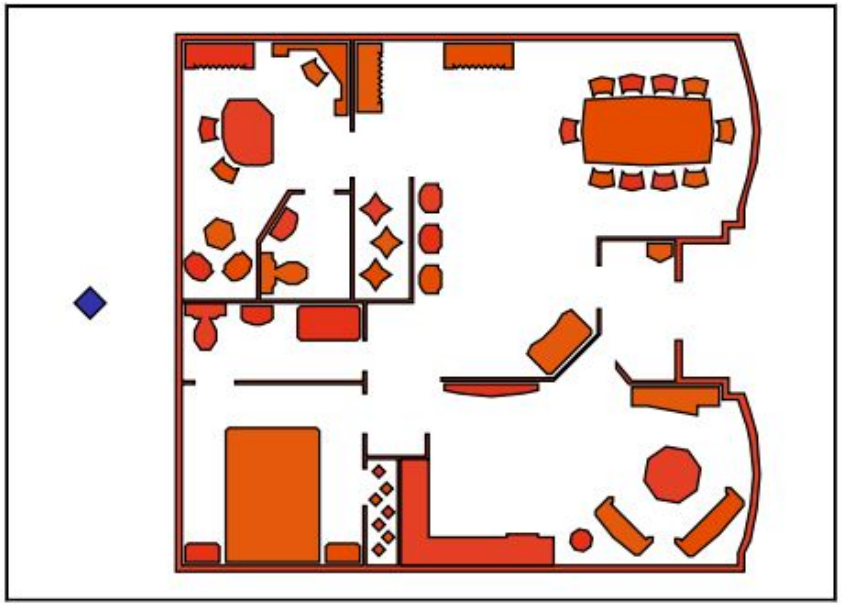


P, Q — polytopes of m and n complexity.

Space	Type	Complexity
$\mathbb{R}^2$	convex, convex	$\Theta(m+n)$
	convex, arbitrary	$\Theta(mn)$
	arbitrary, arbitrary	$\Theta(m^2n^2)$

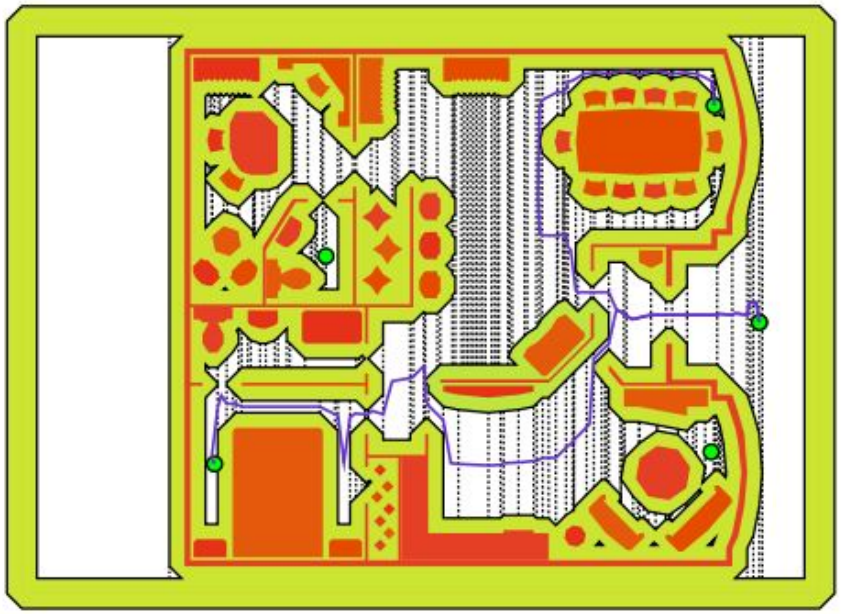
Figures borrowed from http://acg.cs.tau.ac.il/courses/algorithmic-robotics/spring-2011/slides/ms_handouts.pdf.

Planning for a translating polygon using Minkowski sums



The workspace:

- a diamond-shaped robot translating in a house amidst polygonal obstacles



The configuration space:

- the forbidden-configuration space,
- the free-configuration space decomposed into trapezoidal faces
- the queries, and the resulting paths, if exist.

Figures borrowed from http://acg.cs.tau.ac.il/courses/algorithmic-robotics/spring-2011/slides/ms_handouts.pdf.

Planning for a translating and rotating polygon

A configuration of a robot is defined by $(x, y) \in \mathbb{R}^2$ and an angle $\theta \in [0, 1]$.

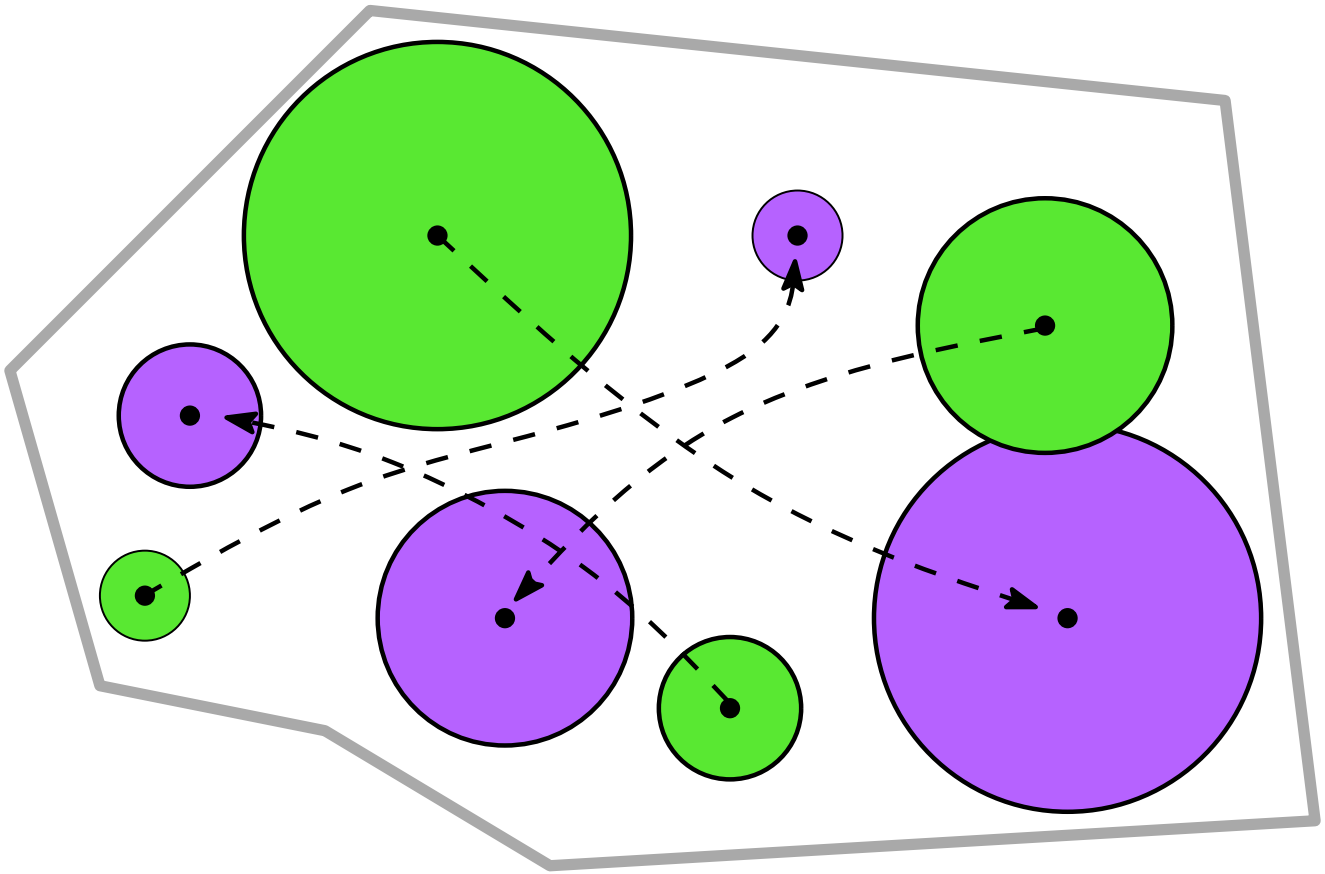
An algorithm for a ladder robot running in $O(n^5 \log n)$ exists [Schwart and Sharir, 83].

Illustration of the configuration space [Atariah and Rote, 12]
https://www.youtube.com/watch?v=SBFwgR4K1Gk

Motion planning for robots with many dofs

- On the "piano movers" problem: doubly-exponential in dofs [Schwartz and Sharir, 83]
- A new algebraic method for robot motion planning and real geometry, exponential in dofs [Canny, 87]
- Motion planning is PSPACE-hard [Reif, 79]

Multi-robot motion planning



Motion planning for a small number of robots

- Two discs in $O(n^3)$ [Schwartz and Sharir, 83]
  - Three discs in $O(n^{13})$
- Two discs in $O(n^2)$ [Yap, 83]
  - Three discs in $O(n^3)$
- Two robots of certain types in $O(n^2)$ [Sharir and Sifrony, 91]

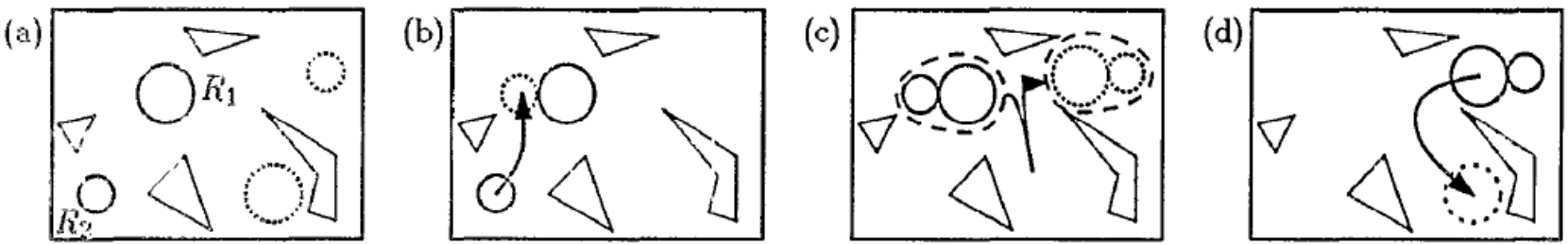
Motion planning for multiple robots

- Let $r_1, \ldots, r_m$ be m translating robots in the plane
- The collection of robots can be viewed as one large robot R
  - A configuration for the m robots is a point in $\mathbb{R}^{2m}$
  - with $2m$ degrees of freedom
- Thus, an exponential algorithm (in m) exists [Canny, 87]
- Can we do better?

Reducing the effective number of degrees of freedom

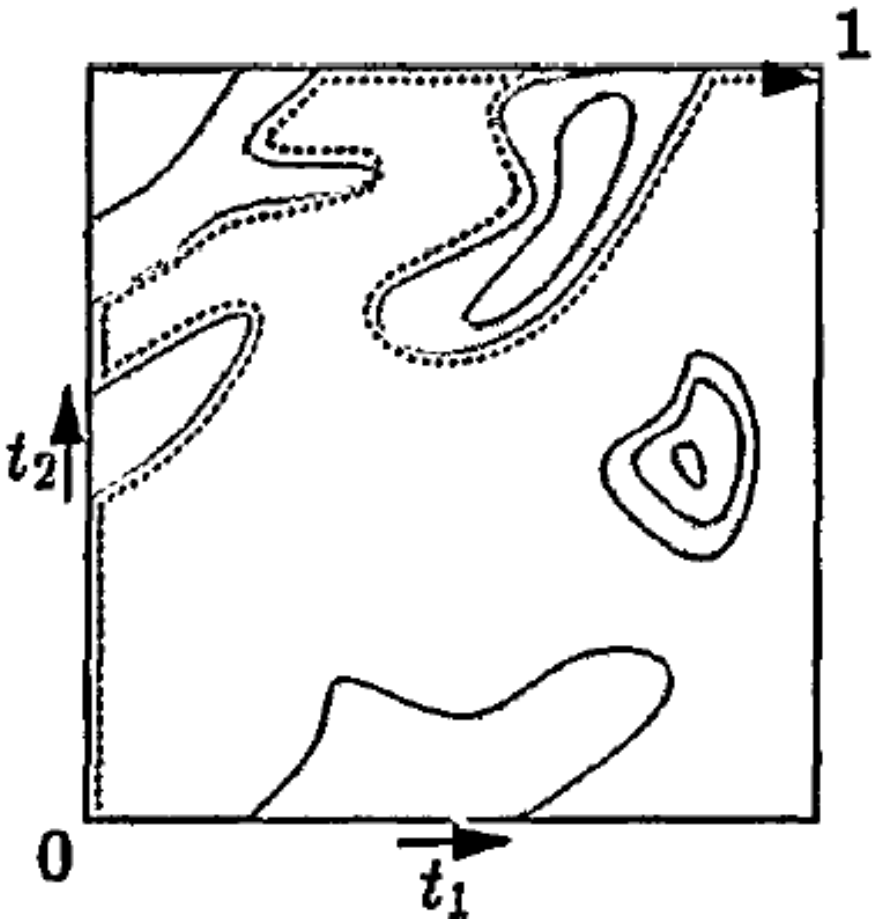
The number of dofs can be slightly reduced [Aronov et al., 98]

- If there exists a solution, then also there is a solution where the robots move attached
- This may reduce the number of DOFS
  - Two robots: $d_1 + d_2 - 1$
  - Three robots: $d_1 + d_2 + d_3 - 2$
  - m robots: $\sum_{i=1}^{m} d_i - m + 1$ (no proof)



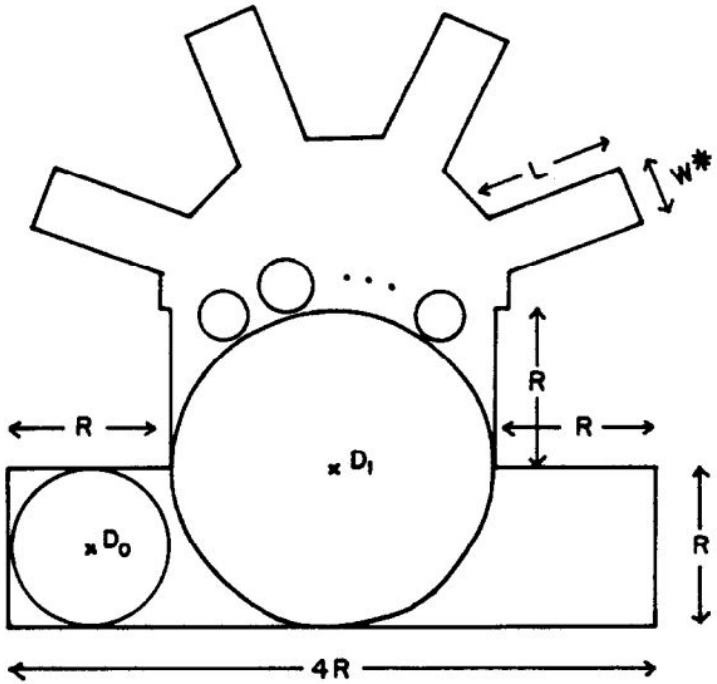
Reducing the effective number of degrees of freedom

"Proof" for the case of two robots



Hardness of multi-robot motion planning

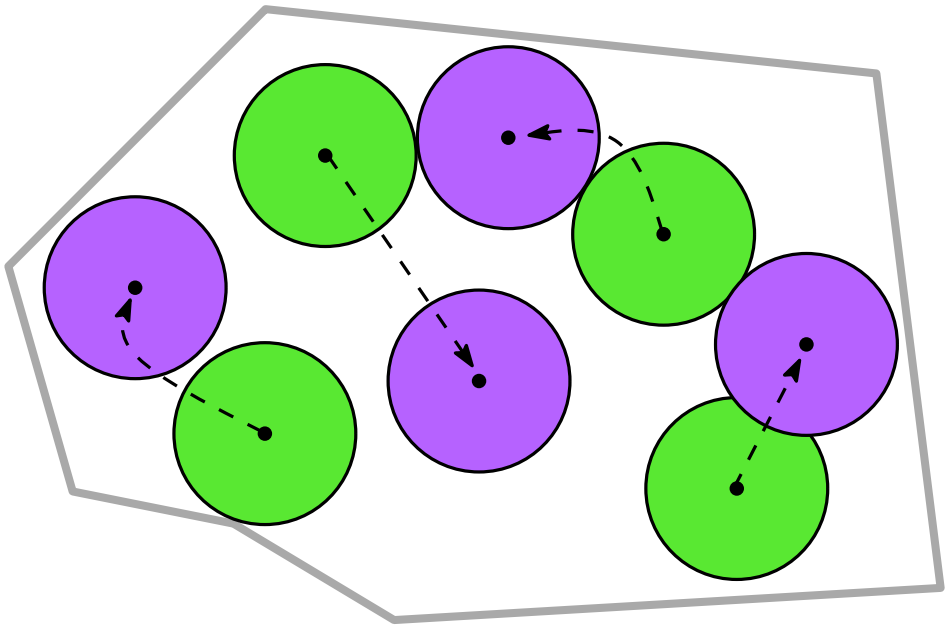
- PSPACE-hardness of translating rectangles
  - Different rectangles [Hopcroft et al., 84]
  - $2 \times 1, 1 \times 2$ rectangles [Hearn and Demaine, 05]
- NP-hardness of discs [Spirakis and Yap, 84]
  - Simple-polygon workspace



An illustration of the reduction from 3-partition where given a collection of $3n$ integers one needs to subdivide them into n sets with equal weights.

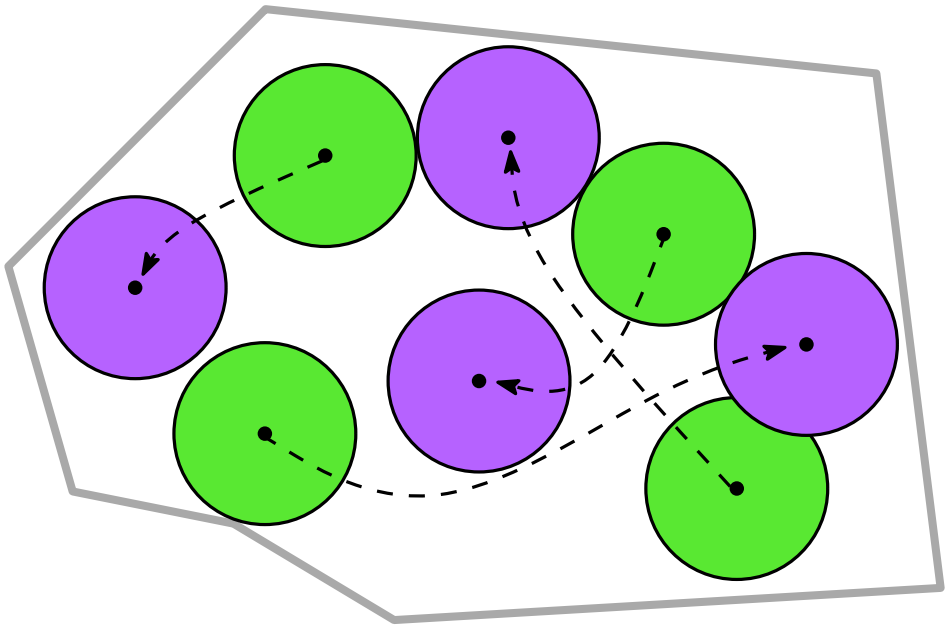
Unlabeled multi-robot motion planning

A variant of the multi-robot problem where the robots are identical and indistinguishable [Kloder and Hutchinson, 06], [S. and Halperin, 12], [Turpin et al., 12]



Unlabeled multi-robot motion planning

A variant of the multi-robot problem where the robots are identical and indistinguishable [Kloder and Hutchinson, 06], [S. and Halperin, 12], [Turpin et al., 12]



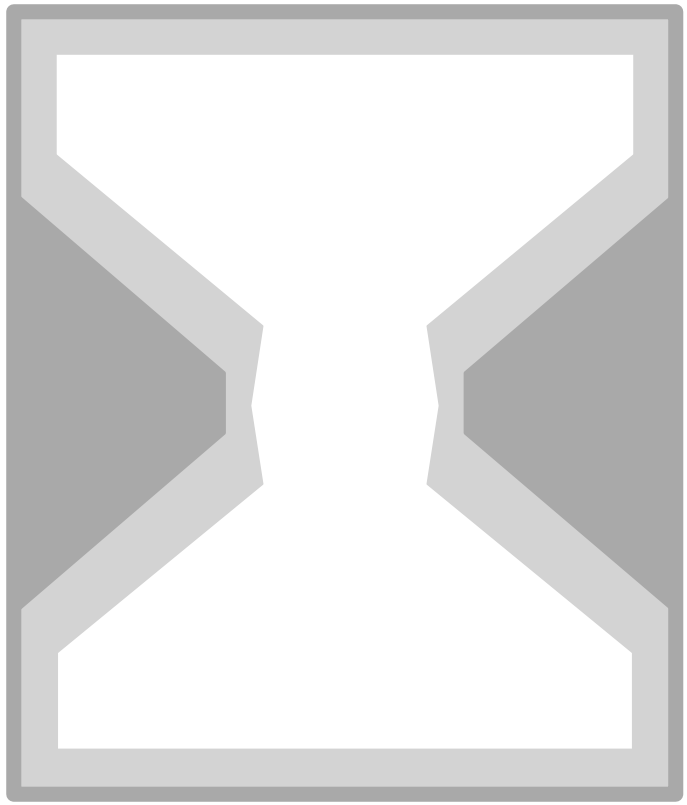
Sampling-based planning for a single robot

Capture connectivity of free space by random samples:

- PRM [Kavraki et al., 96], RRT [Kuffner and LaValle, 00], PRM* and RRT* [Karaman and Frazzoli], FMT* [Janson and Pavone, 13], LBT-RRT [Salzman and Halperin, 13], MPLB [Salzman and Halperin, 14]
- Sampling-based techniques are easy to implement, and can cope with challenging problems involving many dofs
- Most of these techniques are probabilistically complete
  - Solution is found given a sufficient amount of samples
- Some are even asymptotically optimal
  - Solution converges to the optimal solution

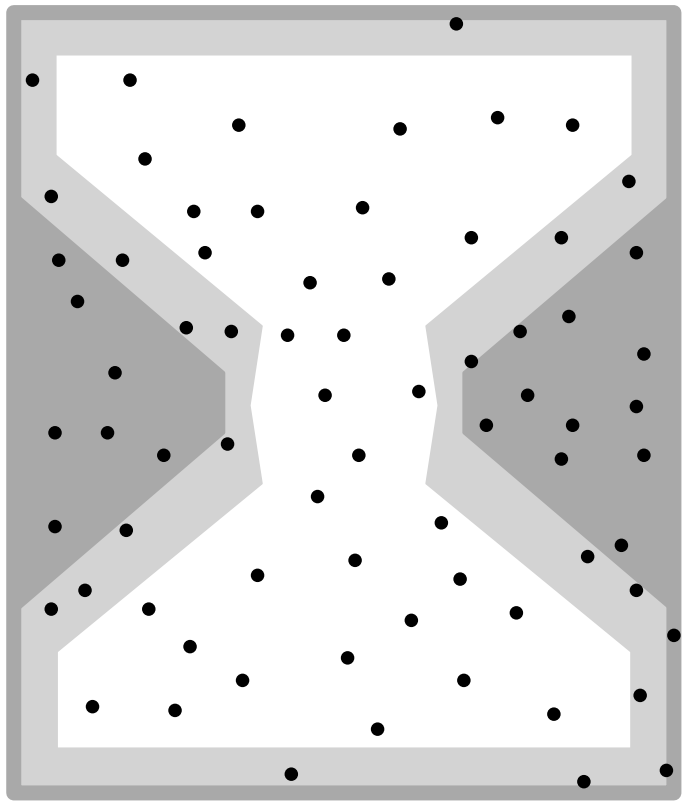
Sampling-based planning: PRM

- Sample a collection of configurations



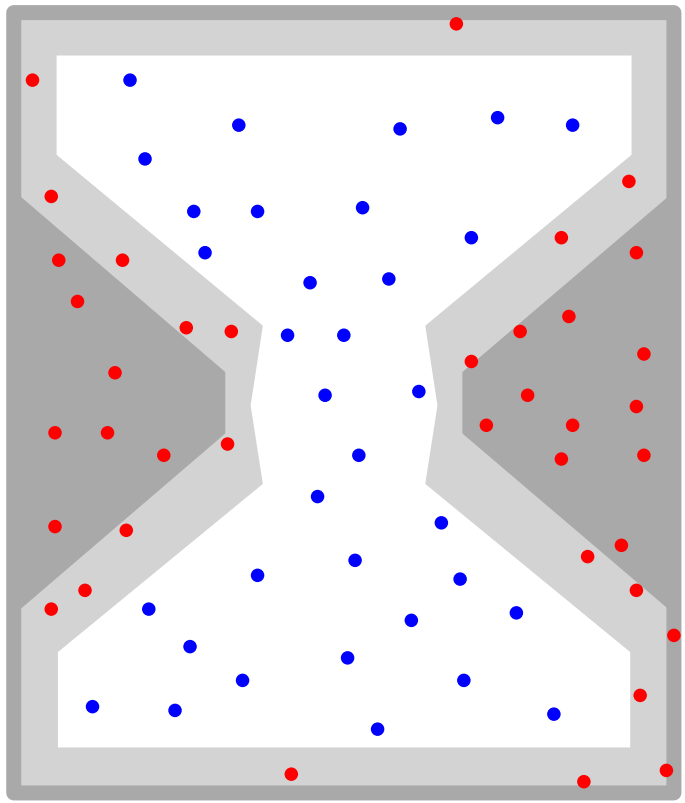
Sampling-based planning: PRM

- Sample a collection of configurations



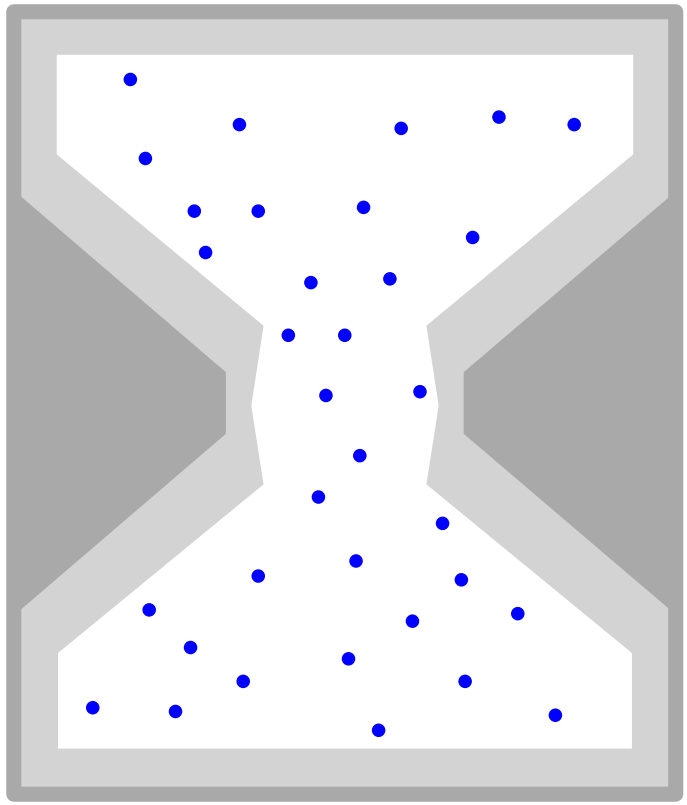
Sampling-based planning: PRM

- Sample a collection of configurations
- Discard invalid configurations
  - *Collision detector*



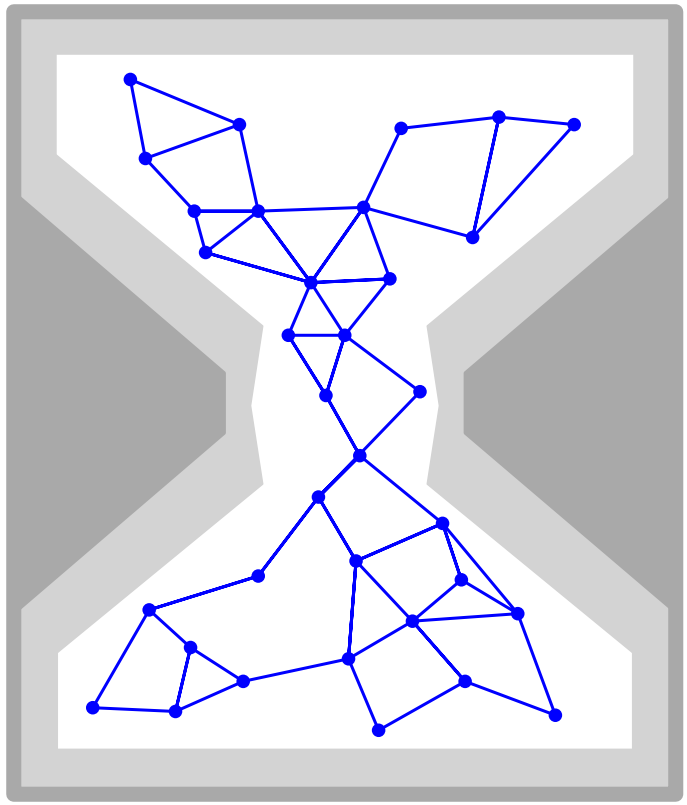
Sampling-based planning: PRM

- Sample a collection of configurations
- Discard invalid configurations
  - *Collision detector*



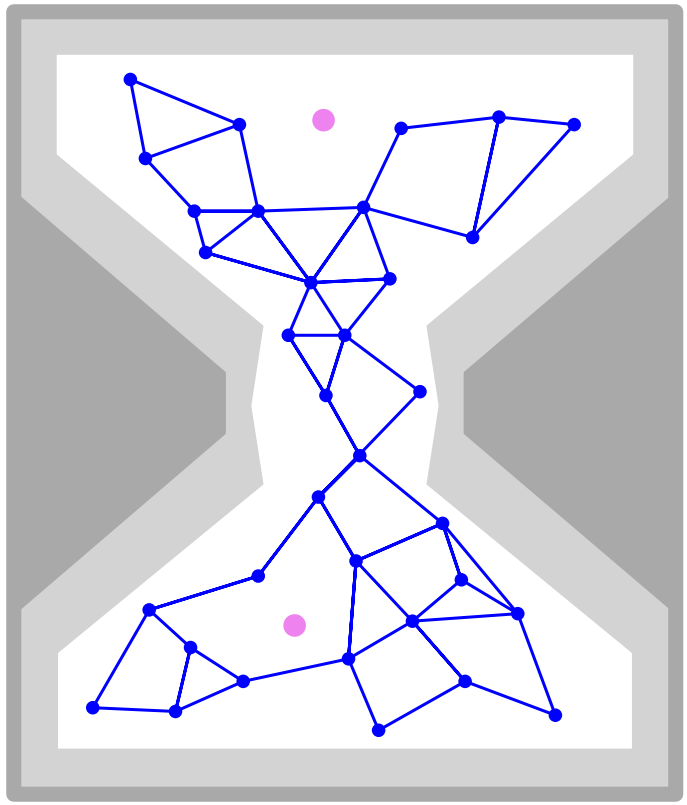
Sampling-based planning: PRM

- Sample a collection of configurations
- Discard invalid configurations
  - *Collision detector*
- Try to connect nearby samples
  - Validate connection with *local planner*



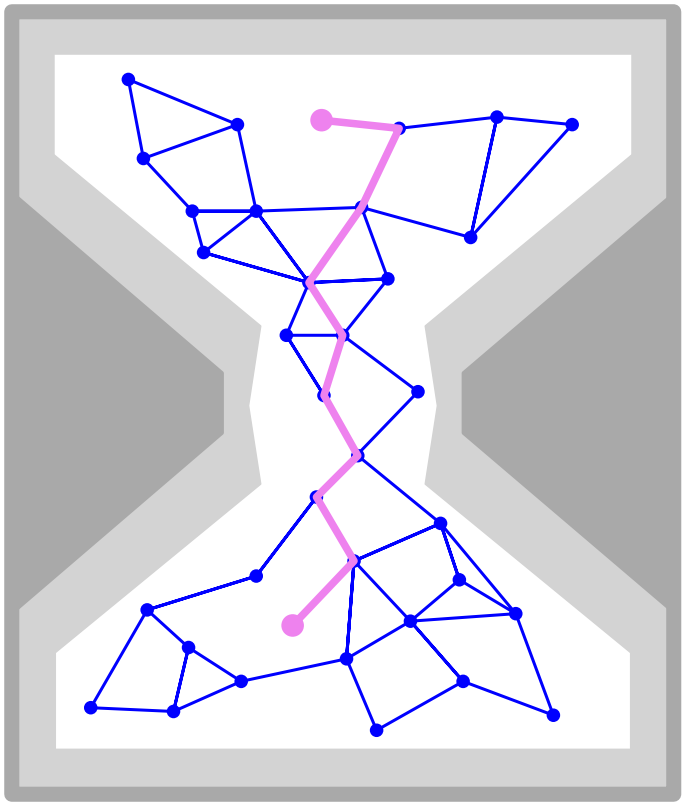
Sampling-based planning: PRM

- Sample a collection of configurations
- Discard invalid configurations
  - *Collision detector*
- Try to connect nearby samples
  - Validate connection with *local planner*
- **Query:**
  - Connect to nearby configurations
  - Retrieve path



Sampling-based planning: PRM

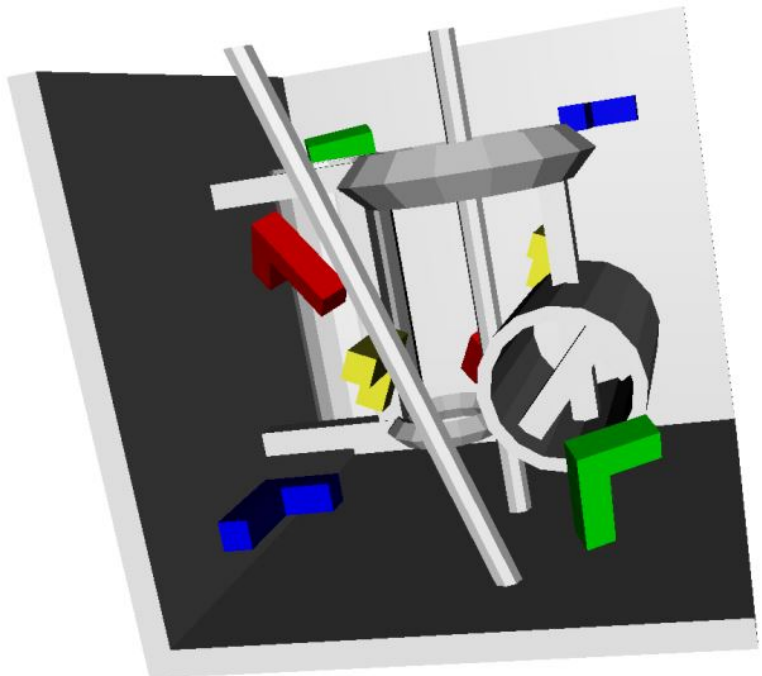
- Sample a collection of configurations
- Discard invalid configurations
  - *Collision detector*
- Try to connect nearby samples
  - Validate connection with *local planner*
- **Query:**
  - Connect to nearby configurations
  - Retrieve path



Sampling-based planning for multi-robot

Several techniques tailored for multi-robot exist:

- [Švestka and Overmars, 98], [Hirsch and Halperin, 02], [Kloder and Hutchinson, 06], [Salzman et al., 12], [S. and Halperin, 12], [Wagner and Choset, 13], [S., Salzman, and Halperin, 14]



* Test scenario with 48 dofs [SSH14].

Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey:

Efficient motion planning for unlabeled disc robots in simple polygons

to appear in *WAFR, 2014*

Contribution

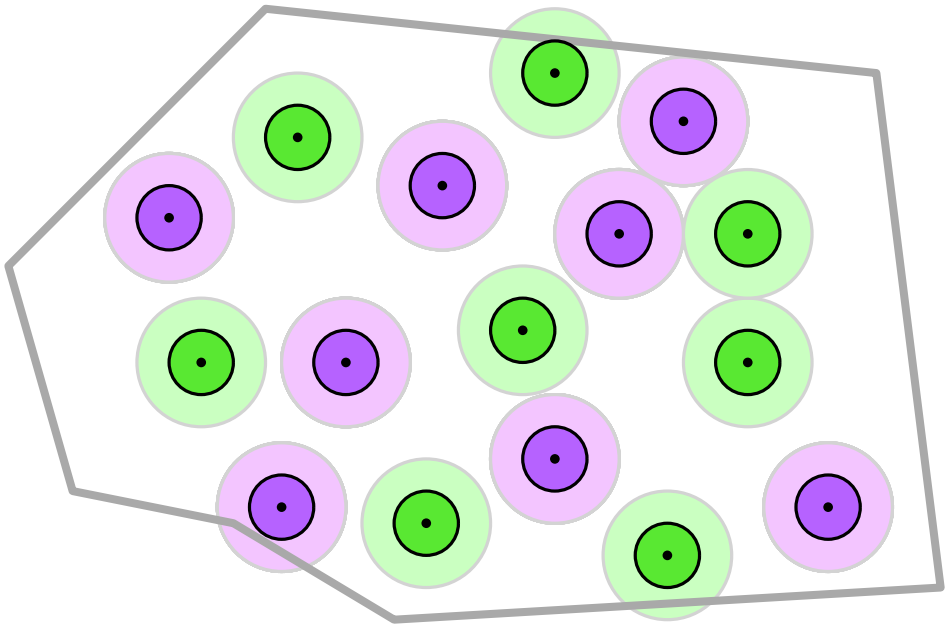
We present an algorithm for motion planning of m unlabeled discs in a simple polygon.

The input consists of m unit discs and two sets S, T of start and target positions, where $|S| = |T| = m$.

The complexity of our algorithm is $O(n \log n + mn + m^2)$, where n is the complexity of the polygon.

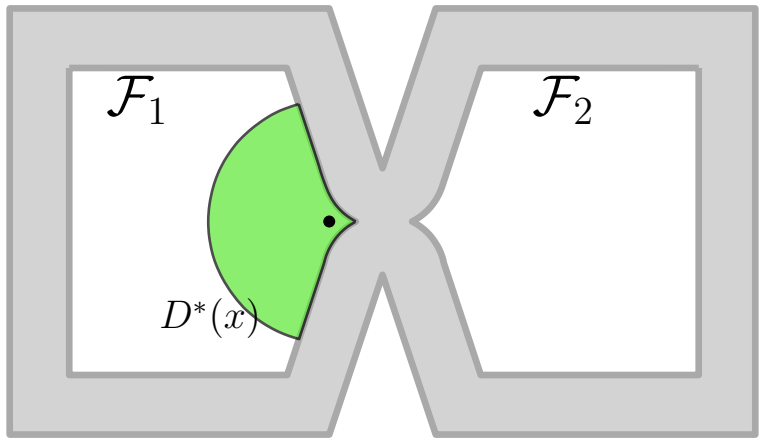
Separation condition

We assume that for any $x, y \in S \cup T$ we have $\|x - y\| \geq 4$.



Some definitions

- Workspace $\mathcal{W} \subset \mathbb{R}^2$ that consists of a simple polygon
- Obstacle space $\mathcal{O} = \mathbb{R}^2 \setminus \mathcal{W}$
- Free space $\mathcal{F} \subset \mathcal{W}$ represents all the collision-free positions
  - Denote by $\mathcal{F}_1, \ldots, \mathcal{F}_\ell$ the connected components of $\mathcal{F}$
  - Define $S_i := S \cap \mathcal{F}_i$, $T_i := T \cap \mathcal{F}_i$
- For $x \in \mathcal{W}$ define $\text{obs}(x) := \{y \in \mathcal{O} : \|x - y\| \leq 1\}$
- For $x \in \mathcal{F}$ denote by $\mathcal{D}_2(x)$ a disc of radius 2 centered in x
- For $x \in \mathcal{F}_i$ define $D^*(x) := \mathcal{D}_2(x) \cap \mathcal{F}_i$



Section 1

Basic properties of a single connected component

Single component of $\mathcal{F}$

We study the basic properties of the problem in the case of a single maximal connected component F_i.

Lemma 2

$\mathbb{R}^2 \setminus F_i$ is connected, i.e., F_i does not have holes.

Proof.

- Suppose that $\mathbb{R}^2 \setminus F_i$ is not connected
- Denote by X the hole in F_i and let $x \in X$
- Let $y \in \text{obs}(x)$ (note that $\text{obs}(x) \neq \emptyset$)
- We know that $\overline{xy} \cap \mathcal{F} = \emptyset$
- Hence $y \in X$ as well
- But $\mathcal{O}$ is connected and unbounded □

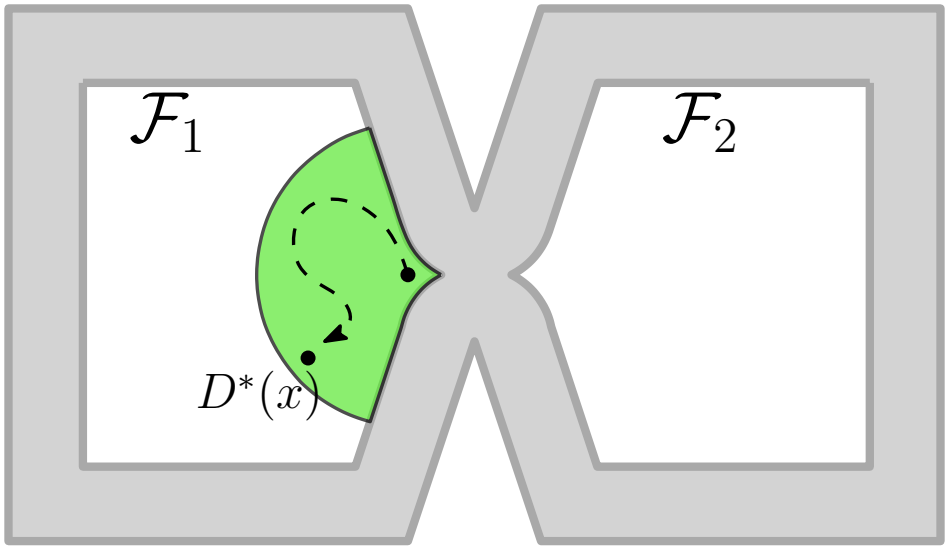
Property of $D^*(x)$

Lemma 3

For any $x \in \mathcal{F}$, $D^*(x)$ consists of a single connected component.

Thus, by this Lemma and by separation it follows that

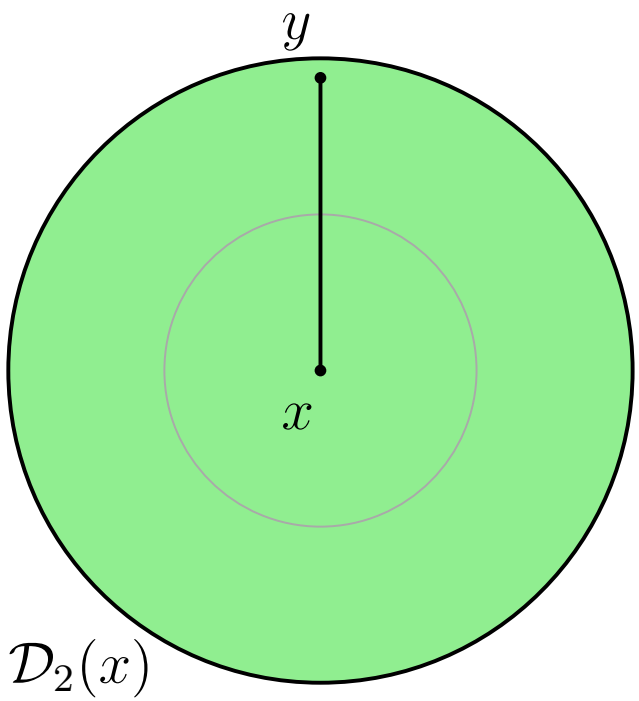
- A robot placed in $x \in S \cup T$ can move to any $x' \in D^*(x)$ without colliding with other robots
- Assuming that every other robot is located in $y \in S \cup T$ where $y \neq x$



Proof of Lemma 2

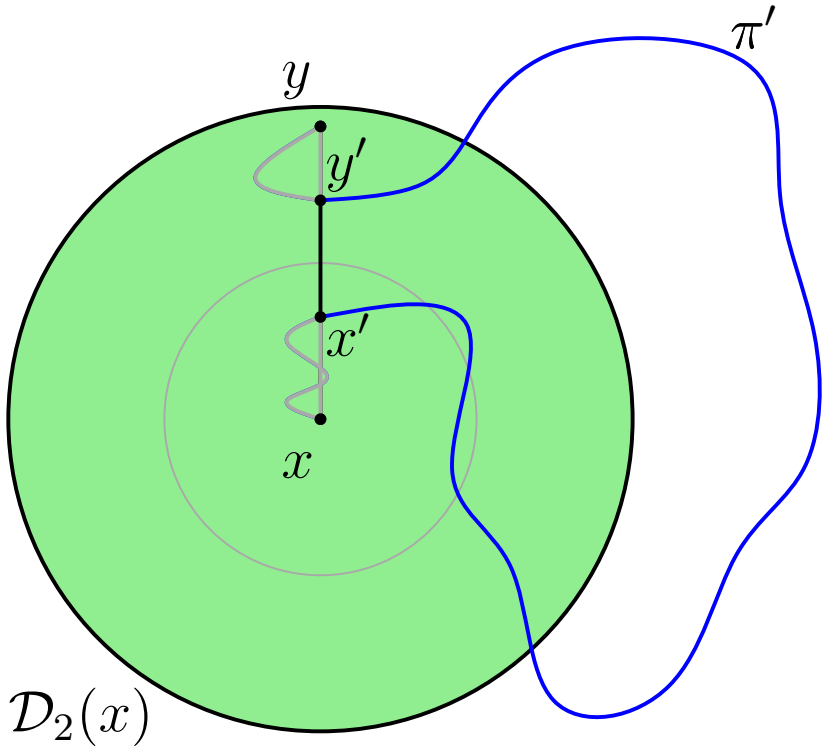
Assume by contradiction that y is in a different component of $D^*(x)$.

- Distance between x and y is at most 2
- Any point on $\overline{xy}$ is within distance of 1 from x, y
- Since $x, y \in F_i$ it follows that $\overline{xy} \in \mathcal{W}$
- If $\overline{xy} \subset F_i$ we're done; assume otherwise



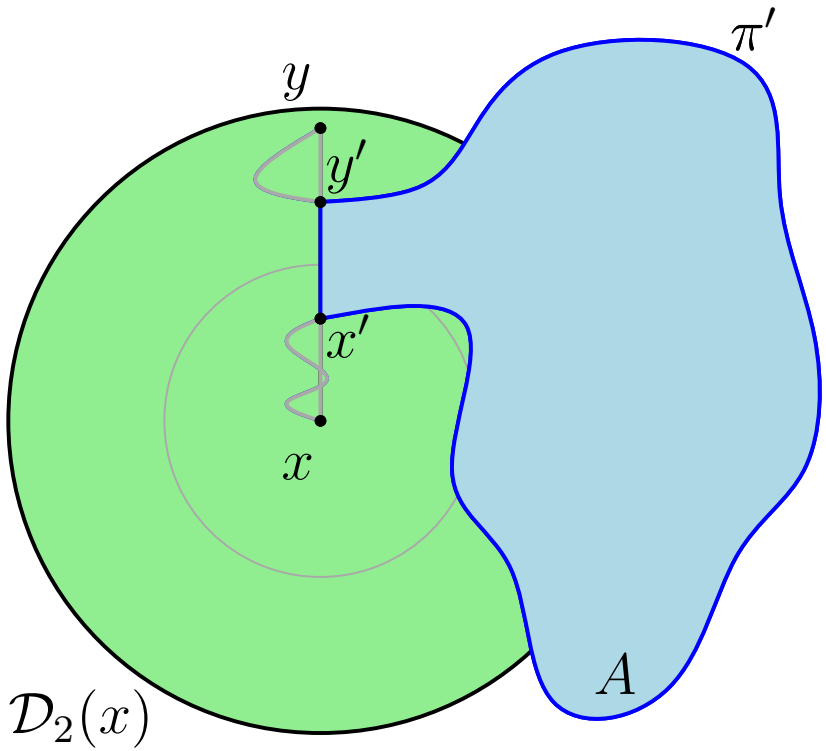
Proof of Lemma 2

- Since $x, y \in F_i$ there exists some simple path $x \subset F_i$ from x to y
- Define $x', y' \in \pi \cap \overline{xy}$ such that
  - x', y' are in two different component of $D^*(x)$
  - $\|x' - y'\|$ is minimized
- Define π' to be a subpath of π connecting x', y'



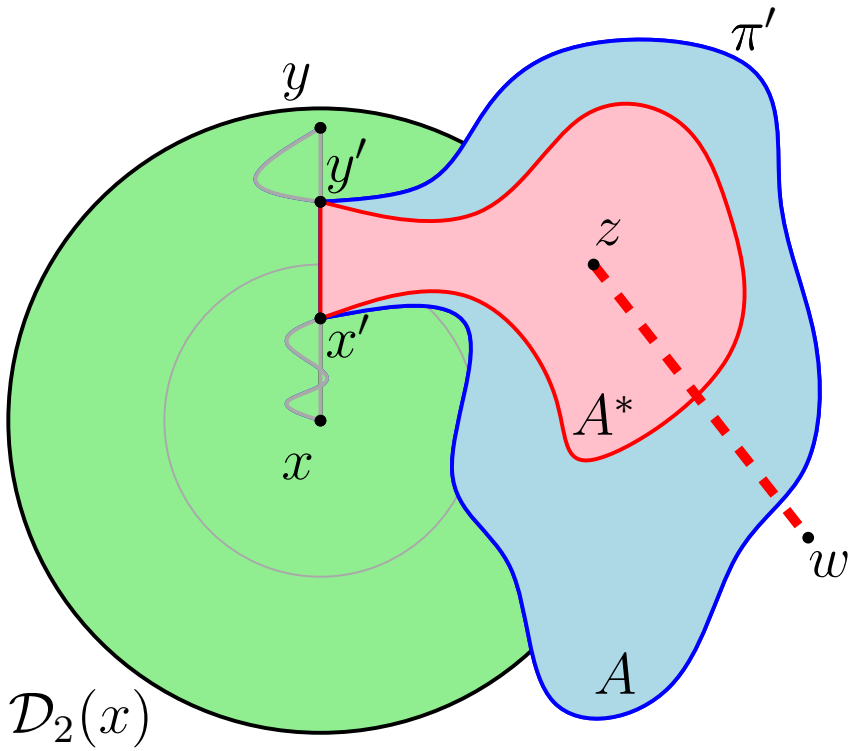
Proof of Lemma 2

- $\lambda := \pi \cup \overline{x'y'}$ is a simple closed curve
- Denote by A the area enclosed by λ
- Note that $\lambda \subset \mathcal{W}$ and consequently $A \subset \mathcal{W}$



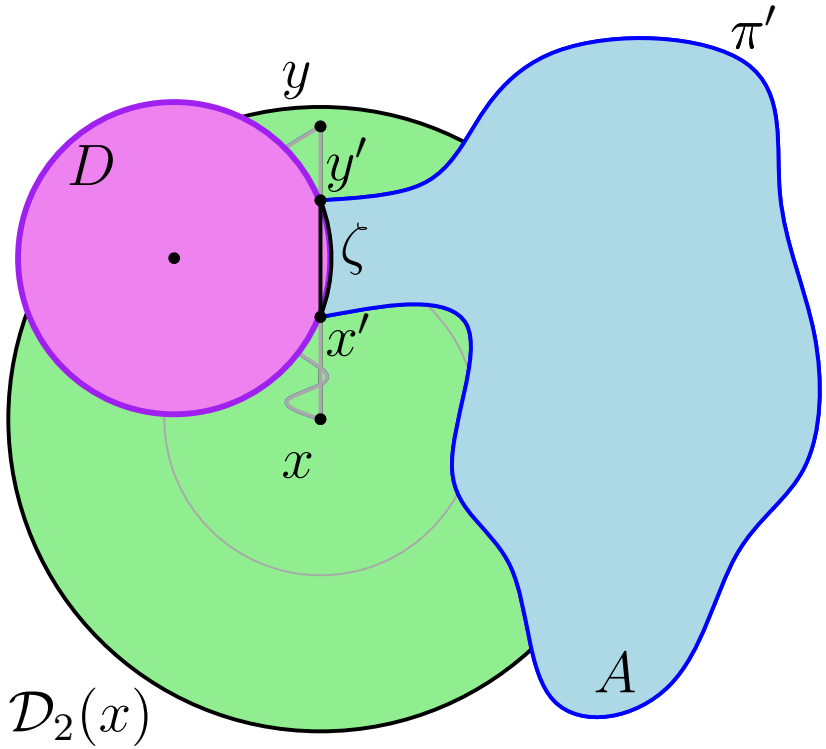
Proof of Lemma 2

- Define $A^* := A \setminus \mathcal{F}$
- For any $z \in A^*$ and $w \in \mathsf{obs}(z)$ it holds that $\overline{zw} \cap \pi' = \emptyset$
- Furthermore, for any $v \in \pi', o \in \mathcal{O}$, $\|v - o\| \geq 1$
- This also applies to x', y'



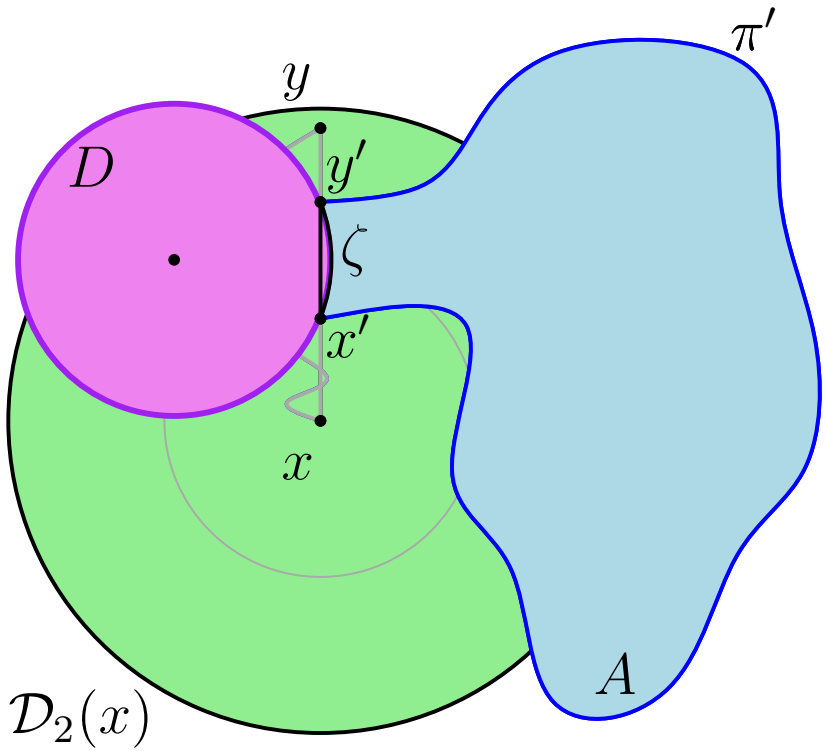
Proof of Lemma 2

- Let D be a circle of radius 1 such that
  - it passes through x', y'
  - its center is on the side of $\overline{x'y'}$ not in A
- Let ζ be the arc of D that is in A
- $A*$ is contained in the area enclosed by $\overline{x'y'}, \zeta$



Proof of Lemma 2

- ζ cannot penetrate $\partial(\mathcal{D}_2(x))$
- Thus $A^* \subset \mathsf{Int}(A) \cap \mathsf{Int}(\mathcal{D}_2(x))$
- There exists a path $\pi'' \subset F_i$ between x', y' along $\partial(A^*)$ such that $\pi'' \subset \mathsf{Int}(\mathcal{D}_2(x))$
- A contradiction □



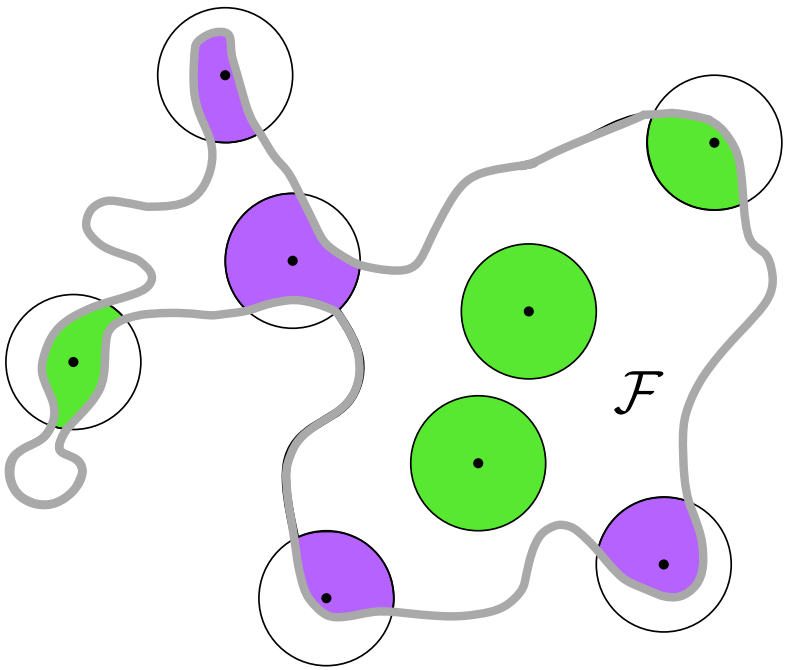
Section 2

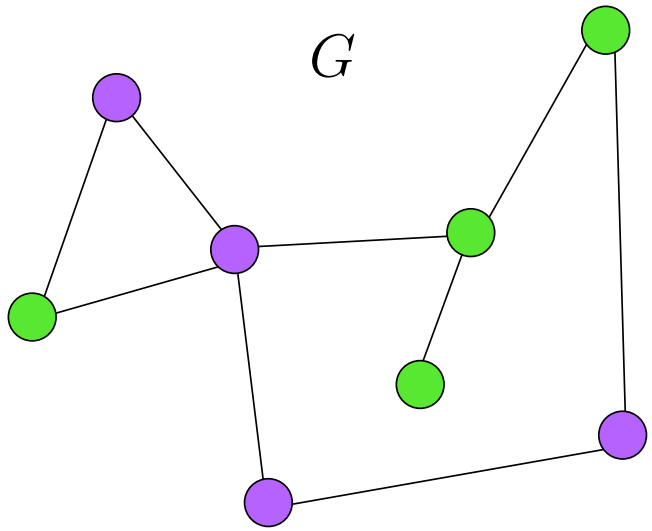
Algorithm for a single connected component

Algorithm for a single connected component

We focus on a specific component of $\mathcal{F}$

- Decompose the component of $\mathcal{F}$ into regions induced by $\mathcal{D}^*(x)$ for $x \in S \cup T$
- Construct a graph G that encodes the unlabeled problem
- Solve a discrete motion problem on G
- Transform graph solution to a collision-free path





Partition of F

Define

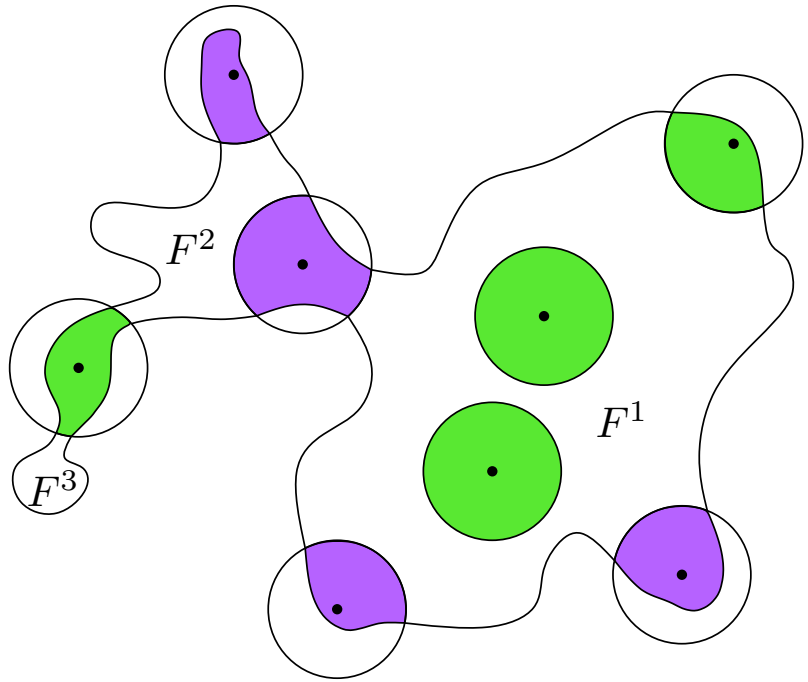
$$F^* := F \setminus \bigcup_{x \in S \cup T} D^*(x),$$

and let $\{F^1, \ldots, F^k\}$ be the partition of F^* into maximal connected components.

Due to *separation* and Lemma 2,

$$\mathcal{P} := \{D^*(x)\}_{x \in S \cup T} \cup \{F^1, \ldots, F^k\}$$

is a partition of F into connected subsets.



The adjacency graph $G = (V, E)$

Vertices:

- $v \in V$ corresponds to an element of $\{D^*(x)\}_{x \in S \cup T}$ (and consequently, an element of $S \cup T$)
- Denote by $P(v)$ the respective $D^*(x)$

Edges:

- $(u, v) \in E$ if there exists $F^i \in F^*$ that is contiguous with $P(u), P(v)$
- Denote by $P(u, v)$ such F^i

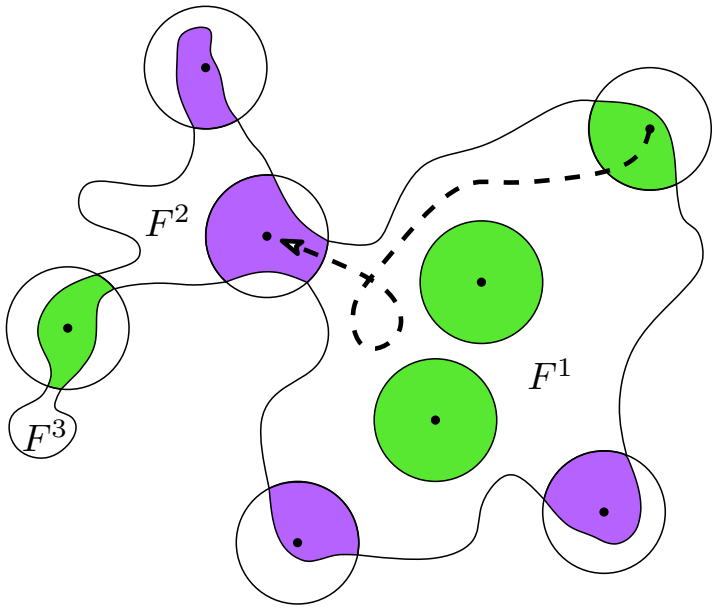
Let the set of *start* and *target* vertices of G be defined respectively as

$$V_S := \{v \in V : P(v) = D^*(s) \text{ for some } s \in S\},$$

$$V_T := \{v \in V : P(v) = D^*(t) \text{ for some } t \in T\}.$$

Observation

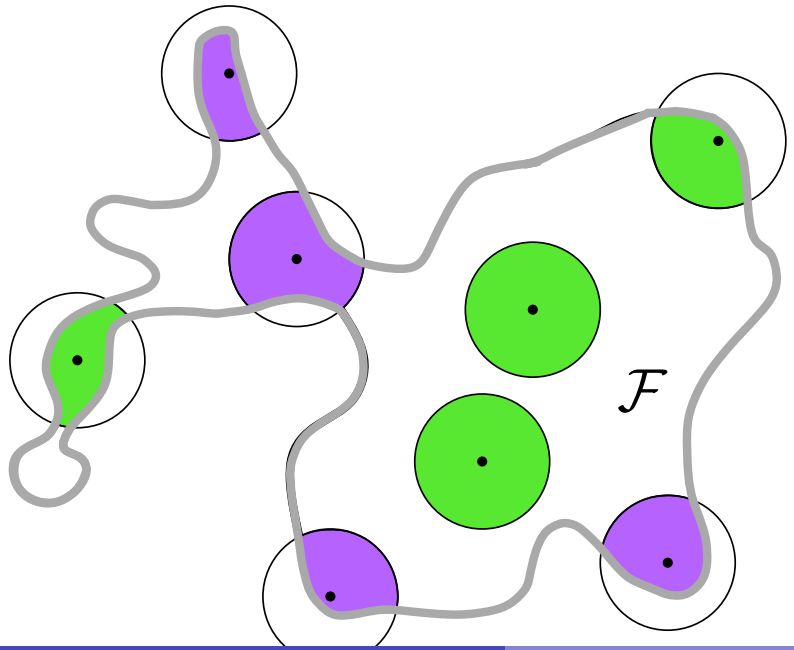
Suppose that $(u, v) \in E_i$ and let $x, x' \in S_i \cup T_i$ such that $D^*(x) = P_i(u), D^*(x') = P_i(u)$. Additionally assume that a robot r is placed in x and no robots are placed anywhere in $P_i(u)$ or $P_i(u, v)$. By Lemma 2 there exists a path for r from x to x' that passes only through $P_i(u), P_i(u, v), P_i(v)$.

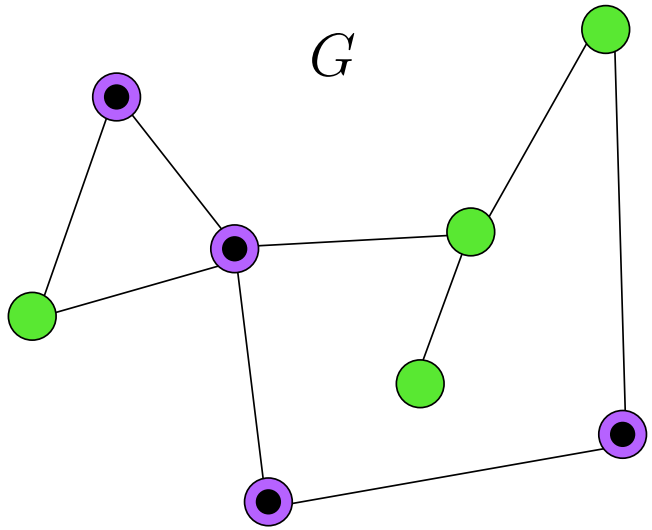


The unlabeled pebble motion problem on G

We define a discrete motion problem on G that represents the unlabeled continuous problem. Every robot in F is represented by a pebble in G

- Pebbles reside on vertices of G
  - Pebble that rests on a vertex v represents a robot occupying the corresponding start or target position
- Pebbles can move using edges of G
  - pebble moving along an edge (u, v) represents a movement of a robot the corresponding start/target positions that does not stray outside the pieces $P(u), P(u, v), P(v)$





The unlabeled pebble motion problem on G

The *unlabeled pebble motion problem* on G consists of moving the pebbles from V_S to V_T while imposing the following restrictions:

1. no two pebbles may occupy the same vertex
2. at most one pebble may be in transit on an edge at any given time

Lemma 4

Any move sequence for the pebbles on G following the above rules can be translated into a valid collision-free motion plan for the robots in F.

Unlabeled pebble motion

Lemma 5

Given an adjacency graph G there exists a solution for the pebble problem from V^S to V^T. Moreover, a solution can be found in $O(m^2)$ time.

Note that $O(m^2)$ are required in some case (G consists of a path, and all the start vertices are placed on one end, while the targets on the other).

Algorithm for the pebble problem

Denote by G' a spanning tree of G. The algorithm performs $O(m)$ *phases*, where in every phase a leaf node is removed from the tree.

1. Select some leaf vertex v of G'; In the end of the phase v will be removed from the tree (along with the pebble that it accommodates)
2. If $v \in V_T$
   - If v is already occupied do nothing
   - Otherwise, find the nearest pebble and move it to v
3. Otherwise $v \in V_S$
   - If v is empty do nothing
   - Otherwise, find the closest unoccupied vertex u
   - Push all pebbles along the path from v to u

Observation

In each phase every edge is traversed at most once.

Section 3

Implementation details and complexity analysis

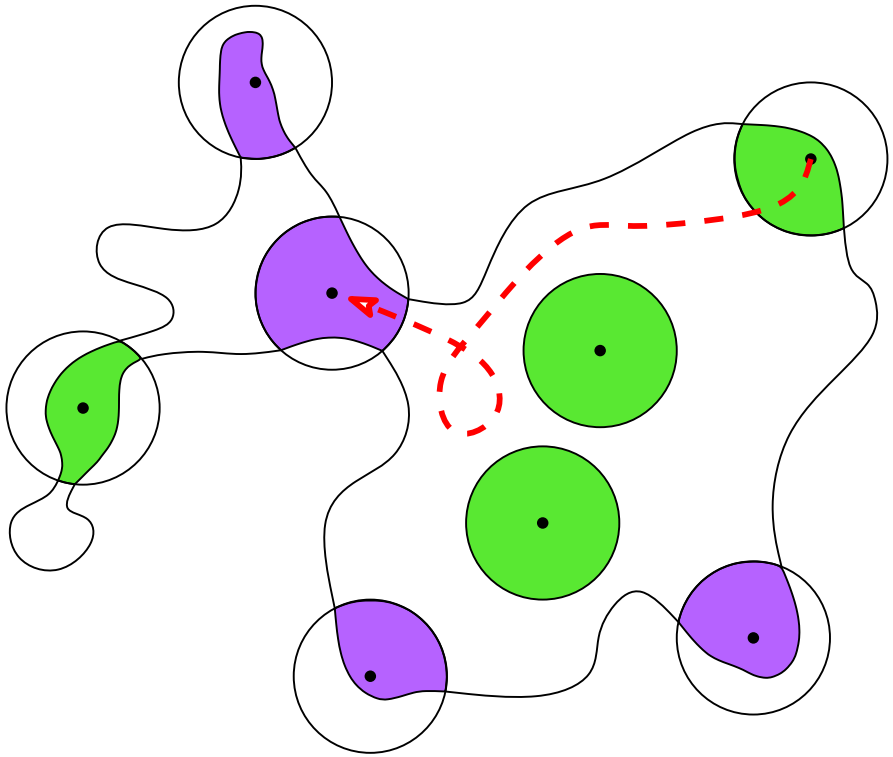
Implementation details

Need to address the following issues:

1. Partitioning F
2. Generating G
3. Generating paths

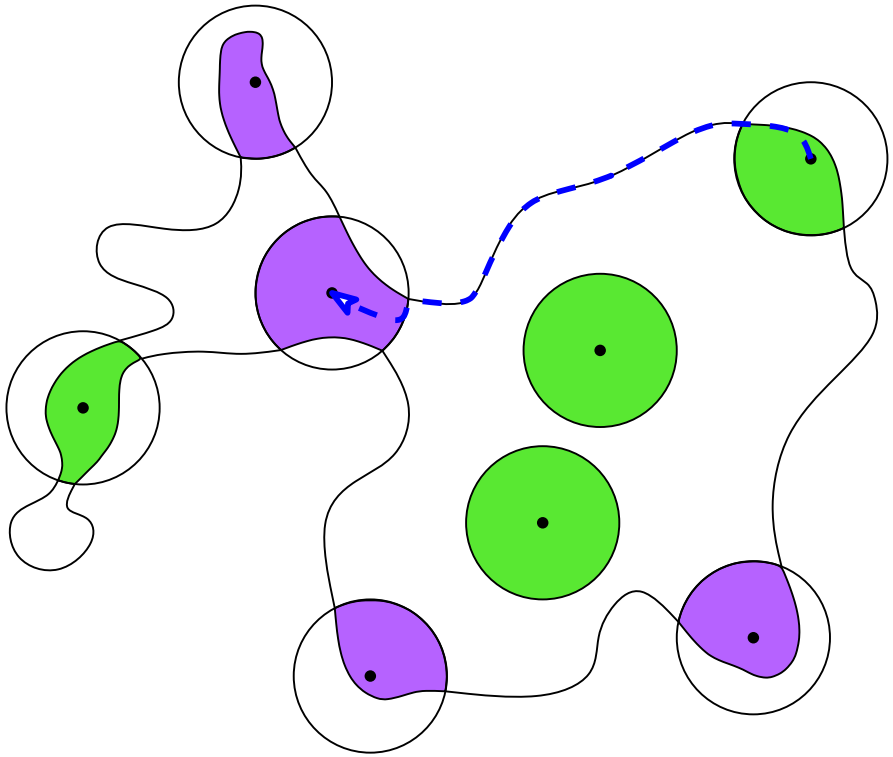
Path generation

Need to avoid generating complex paths that increase the running time of the algorithm.



Path generation

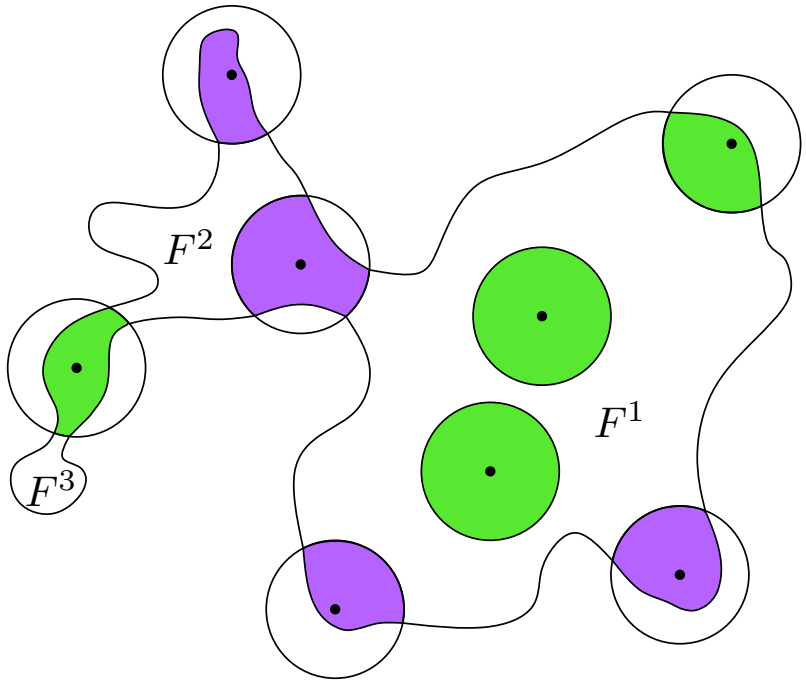
We will restrict the robots to move on the boundaries of the decomposition, because we can bound the combinatorial complexities of the boundaries.



Partitioning $\mathcal{F}$

We analyze the combinatorial complexity of

- F
- $F^* := F \setminus \bigcup_{x \in S \cup T} D^*(x)$
- $\mathbb{D} := \bigcup_{x \in S \cup T} D^*(x)$



Main ingredient in the analysis:

Theorem 6 ([Kedem,Livne,Pach and Sharir, 83])

Let $A_1, \ldots, A_k, B$ be convex polygon where $A_1 \ldots, A_k$ have n corners and B has a constant number of corners. Additionally assume that $A_1, \ldots, A_k$ have disjoint interiors, and let $K_i := A_i \bigoplus B$. Then the combinatorial complexity of $K := \bigcup_{i=1}^{k} K_i$ is $O(n)$.

Complexity of F and F^*

Lemma 7

The combinatorial complexity of F is $O(n)$.

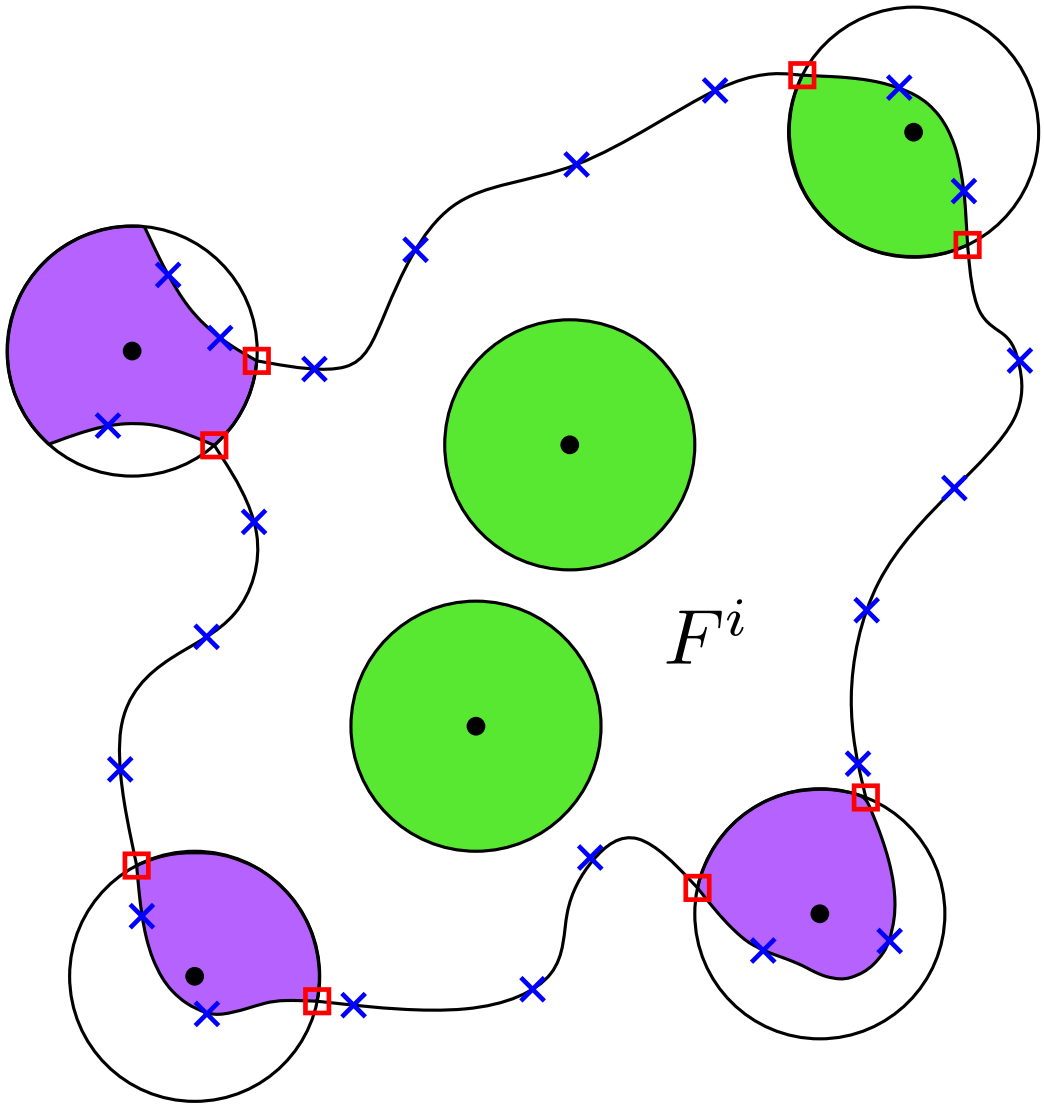
Proof.

- Decompose the complement of the workspace polygon into $O(n)$ trapezoids
- Take the Minkowski sum of every trapezoid with a disc of radius 1
- The complement of the union of sums equals F
- By KLPS the complexity of the union is $O(n)$ □

Corollary 8

The combinatorial complexity of F^* is $O(m+n)$.

Complexity of $\mathbb{D} := \bigcup_{x \in S \cup T} D^*(x)$



Complexity of $\mathbb{D} := \bigcup_{x \in S \cup T} D^*(x)$

Lemma 9

The combinatorial complexity of $\mathbb{D}$ is $O(m+n)$.

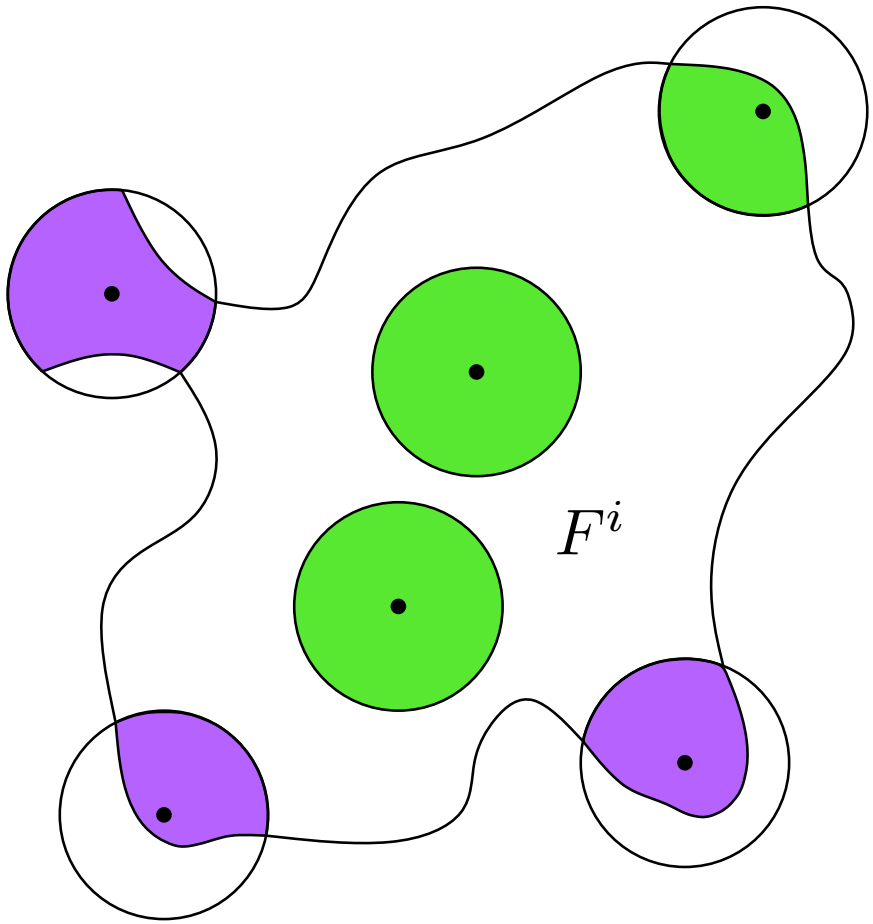
Proof. Denote by $d := \{d_1, d_2, \ldots\}, f := \{f_1, f_2, \ldots\}, f^* := \{f_1^*, f_2^*, \ldots\}$ the segments that define $\partial(\mathbb{D}), \partial(F), \partial(F^*)$, respectively.

Note that d consists of elements of f, f^* and additional segments that are subsegments of the elements of f, denoted by $f' := \{f_1', f_2', \ldots\}$.

Note that the complexity of $d \cap (f \cup f^*)$ is bounded by $O(m+n)$. Additionally, for each element of f' its endpoints are vertices of $\partial(F), \partial(F^*)$ and this implies that the complexity of d does not exceed $O(m+n)$. □

Generating the adjacency graph

Let F^i be some the maximal connected components of F^*. We construct a subset of G that we denote by G_i.



Hole and boundary positions

Denote by $B_i \subseteq S \cup T$ the start/target positions share a segment with the outer boundary of F^i, namely

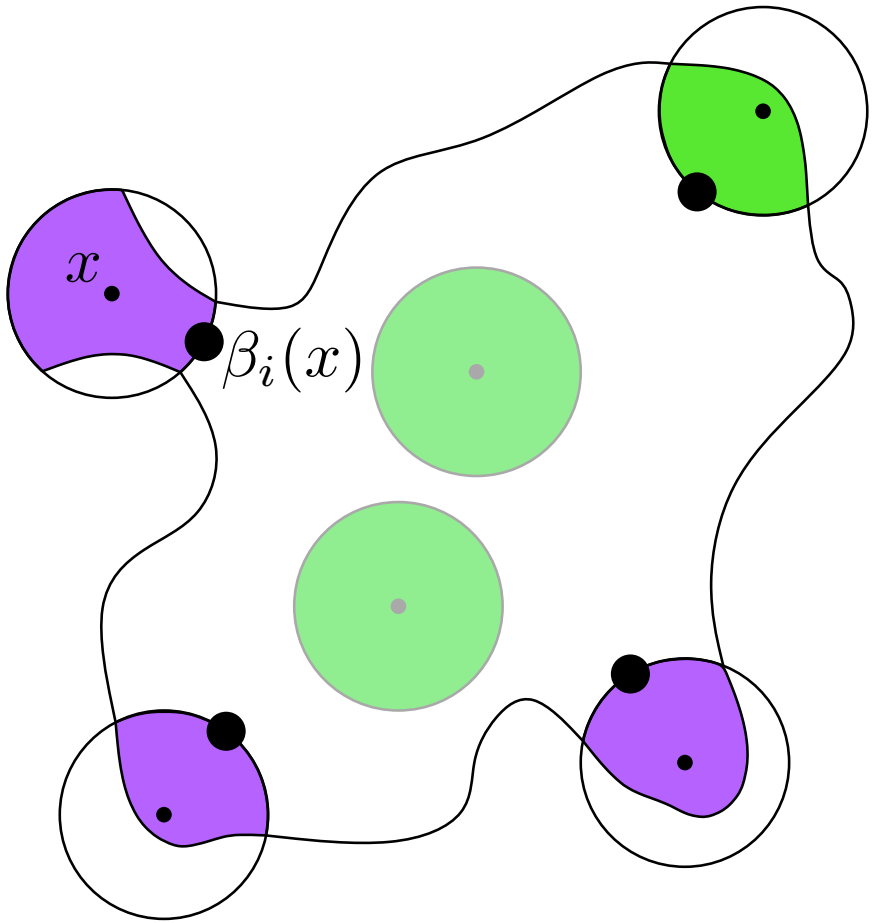
$$B_i := \{x : \partial(\mathcal{D}_2(x)) \cap \partial(F) \neq \emptyset \wedge \partial(\mathcal{D}_2(x)) \cap \partial(F^i) \neq \emptyset\}.$$

Similarly, $H_i \subseteq S \cup T$ denotes positions that produce holes in F^i, namely

$$H_i := \{x : \partial(\mathcal{D}_2(x)) \cap \partial(F) = \emptyset \wedge \partial(\mathcal{D}_2(x)) \cap \partial(F^i) \neq \emptyset\}.$$

Dealing with boundary positions

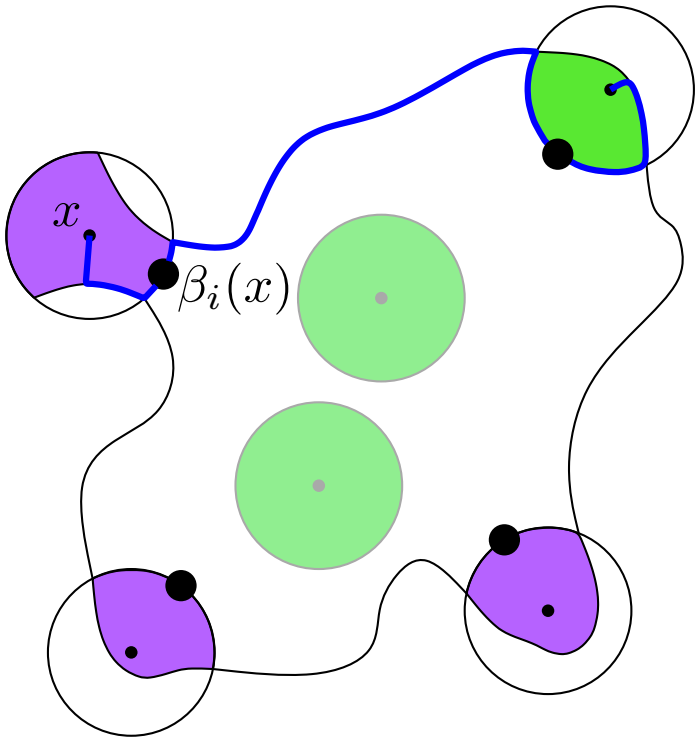
We put the elements of B_i in a circular list Γ_i, according to the order along $\partial(F^i)$. For each $x \in B_i$ we assign a point on $\partial(F^i)$ and denote it by $\beta_i(x)$. For such x we add a vertex to G_i.



Dealing with boundary positions

Suppose that $x, x' \in S \cup T$ are two consecutive elements of Γ_i. We add an edge between between the respective vertices in G_i. The actual path between x, x' consist the following parts:

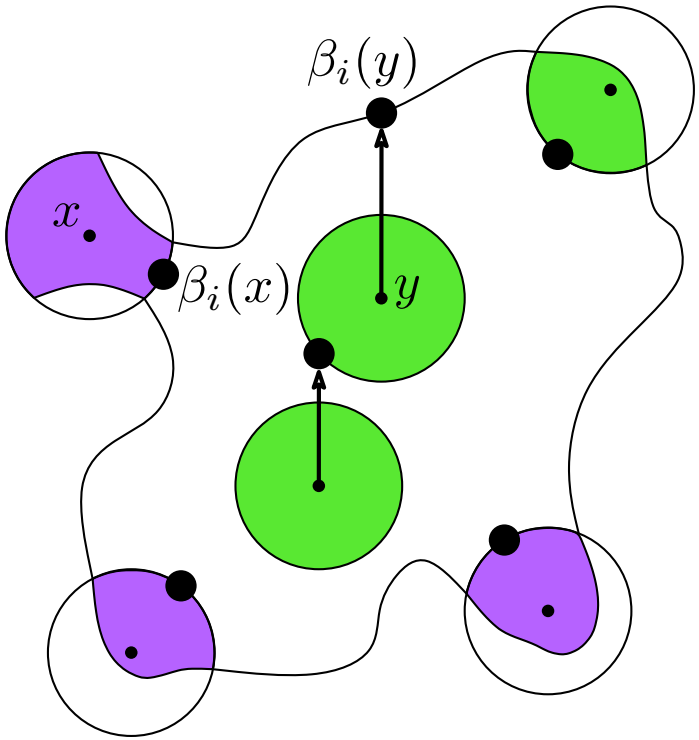
- A path within $D^*(x)$ from x to $\beta_i(x)$
- A path along $\partial(F^i)$ from $\beta_i(x)$ to $\beta_i(x')$
- A path within $D^*(x)$ from $\beta_i(x')$ to x'



Dealing with hole positions

For every $x \in H_i$ we shoot a vertical ray until it hits $\partial(F^i)$. Denote by c the intersection point.

- If $c \in \partial(\mathcal{D}_2(x'))$ for some $x' \in H_i, x' \neq x$ add an edge (x, x') to G_i
- Otherwise, x is added to Γ_i, according to the intersection point c



Merging components of G

Two subgraphs G_i, G_j are connected via $x \in S \cup T$ such that $x \in B_i, x \in B_j$.

Complexity analysis

The pebble solver produces $O(m^2)$ moves. Thus, a naive analysis may lead to a running time of $O\left(m^2(m+n)\right)$. Instead, we consider the complexity induced by each phase of the pebble solver.

Theorem 10

The algorithm runs in $O(n \log n + mn + m^2)$ time.

Proof. Recall that the pebble solver operates in $O(m)$ phases. In every phase it does one of the following:

1. Moves a pebble to an unoccupied target leaf
2. Pushes a pebble from an occupied start leaf

Every phase transforms into a path (or multiple paths) of complexity $O(m+n)$.

Complexity of type (1) phase[1]

Suppose that a pebble is moved from u to v along the path $u = w_1, w_2, \ldots, w_l = v$.

Denote by $x(w_i)$ a start/target position that represents each vertex and w_i. Thus, the configuration-space path passes through $D^*(x(w_i))$ at most twice (leaving and entering at most once).

Denote by $\beta(w_i, w_{i+1})$ the portion of $\partial(F^*)$ used by the robot to move from $x(w_i)$ to $x(w_{i+1})$. Every two such portions $\beta(w_i, w_{i+1}), \beta(w_j, w_{j+1})$ are disjoint.

As the complexities of $\partial(\mathbb{D}), F^*$ is $O(m+n)$ we may conclude that the complexity of every phase is $O(m+n)$ and the overall complexity is $O(n \log n + m^2 + mn)$, where the $n \log n$ factor represents the complexity of generating the various partitions. □

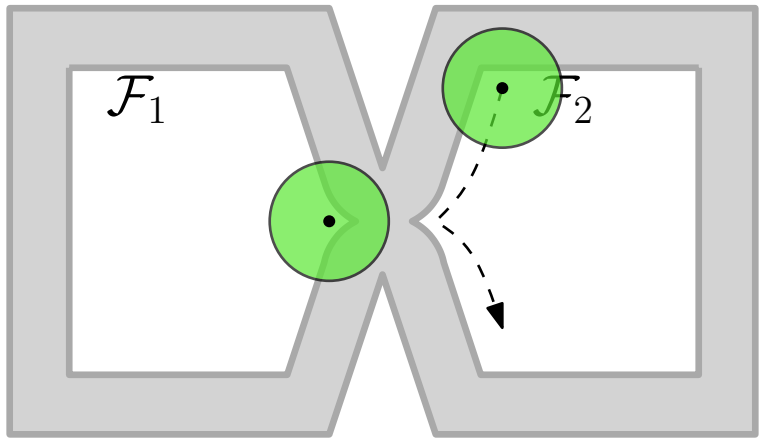
[1] Analysis for the second type of phases is similar.

Section 4

Combining single-component plans

Combining single-component plans

$\mathcal{F}_1, \ldots, \mathcal{F}_\ell$ are the connected components of $\mathcal{F}$.
It may happen that a robot from one connected component interferes with a path from another connected component.

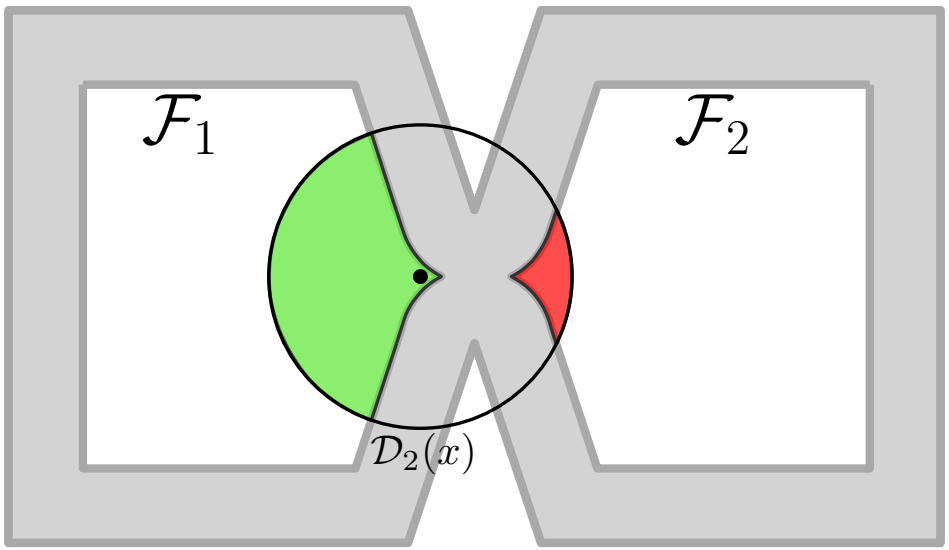


Interference between different components of $\mathcal{F}$

Definition 11 (Interference)

A position $x \in \mathcal{F}_i$ *interferes* with $\mathcal{F}_j$ (and denote $x \in I_{(i,j)}$) if

$$\mathcal{D}_2(x) \cap \mathcal{F}_j \neq \emptyset.$$



Interference between different components of $\mathcal{F}$

Lemma 12

If $x_i \in \mathcal{F}_i$ interferes with $\mathcal{F}_j$, and $x_j \in \mathcal{F}_j$ interferes with F_i then

$$\mathcal{D}_2(x_i) \cap \mathcal{D}_2(x_j) \neq \emptyset.$$

- By Lemma 3 and separation it follows that
  - $x_i \in S_i \cup T_i, x_j \in S_j \cup T_j$ cannot cause interference simultaneously
- Interference induces ordering on $\mathcal{F}_i, \mathcal{F}_j$

* We also prove a cyclic version of this Lemma.

Combining single-component plans

We show that there exists an ordering

$$\sigma : \{1, 2, ..., \ell\} \to \{1, 2, ..., \ell\}$$

of the connected components such that we may execute the single-component motion plans, so long as we do so in the order

$$F_{\sigma(1)}, F_{\sigma(2)}, ..., F_{\sigma(\ell)}.$$

Combining single-component plans

We construct a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ called the *directed-interference forest*. Every $F_i \in \mathcal{F}$ is represented by a node $v_i \in \mathcal{V}$.

A directed edge (v_i, v_j) is added to $\mathcal{E}$ if one of the following holds:

- there exists a start position $s \in S$ such that $s \in I_{(i,j)}$
- there exists a target position $t \in T$ such that $t \in I_{(j,i)}$

Lemma 13

$\mathcal{G}$ is a directed forest.

Thus, a topological sorting of $\mathcal{G}$ induces an ordering σ

Open problems

- Efficient solution when workspace is no longer a simple polygon
- Similar technique for the standard multi-robot problem
- Varying separation s value
  - For $s \geq 4$ the problem is polynomial and always has a solution
  - For $s = 2$ the problem is PSPACE-hard
  - What happens when $s = 2 + \varepsilon$?