A Motion Planning Algorithm in a Figure Eight Track

C. Jardon, B. Sheppard, and V. Zaveri

Abstract - We design a motion planning algorithm to coordinate the movements of two robots along a figure eight track, in such a way that no collisions occur. We use a topological approach to robot motion planning that relates instabilities in motion planning algorithms to topological features of configuration spaces. The topological complexity of a configuration space is an invariant that measures the complexity of motion planning algorithms. We show that the topological complexity of our problem is 3 and construct an explicit algorithm with three continuous instructions.

Keywords : topological robotics; motion planning; topological complexity

Mathematics Subject Classification (2020) **:** 55P99; 05C85; 90C35, 05C90

1 Introduction

The motion planning problem in robotics relates to the movement of robots from one location to another in their physical space. When there are several robots moving in the same physical space without collisions we need to study the configuration space.

The configuration space gives a complete specification of the position of every robot in the physical space and a path in the configuration space is interpreted as a particular collision-free movement from the initial position of the robots to their final one.

When X is the configuration space, the motion planning problem consists in constructing an algorithm that takes as input a pair of initial and final configurations $(x_i, x_f) \in X \times X$, and produces a continuous path $\alpha : I \to X$ from the initial configuration $x_i = \alpha(0)$ to the final one $x_f = \alpha(1)$. Farber initiated a topological approach to this problem in [1]. Let PX be the space of all paths in X. The *motion planning algorithm* (MPA) is a map $s : X \times X \to PX$ defined by sending a pair of points to a path between them. This approach favors maps that depend continuously on the input since they will translate into more robust algorithms in real-life problems. However, Farber proved that there exists a continuous MPA in X if and only if X is contractible. Therefore, most MPAs in real life may have essential discontinuities, or instabilities, due to the topology of X.

Motivated by this, Farber introduced the *topological complexity* of a space. This is a numerical homotopy invariant which measures the minimal number of instabilities of any MPA on a space. Essentially, the topological complexity of a space X, $TC(X)$, is the smallest k such that $X \times X$ can be covered by k sets over each of which there is a continuous local section. We will call *instructions* each of these local sections. Then, the topological complexity of a space X is the minimal number of instructions in any MPA in X.

We present our approach to calculating the topological complexity of the motion planning problem for two robots moving in a figure eight graph, and construct an explicit algorithm with the minimal number of instructions given by the topological complexity.

Our constructive strategy is based on finding a *spine* $Z \subset X$ which is a deformation retract of X with lesser dimension and design the algorithm in Z. Then extend the algorithm to the whole configuration space X by using the traces of the homotopy deformation. We also give a translation of the algorithm for the physical space that shows explicitly how to move each of the two robots from any initial position to any final position.

This paper will be organized as follows. Section 2 introduces basic terminology used throughout the paper along with the basic definitions of MPA and configuration spaces. Section 3 explains the construction of the configuration space and its flat representation. In section 4 we discuss the topological complexity and its basic properties. Section 5 gives a detailed description of the homotopy deformation into the spine. We also show in this section that the spine is homotopy equivalent to a wedge of seven circles, which allows us to calculate the topological complexity of the configuration space. In section 6 we present our main algorithm in the spine, and extend it to the whole configuration space. We translate the previous algorithm in the configuration space to the physical space in section 7.

2 Definitions and Terminology

2.1 Motion Planning Problem

One of the goals of robotics is to create autonomous robots. A set of robots is a mechanical system that accepts descriptions of tasks and executes them without further human intervention. The input description specifies what should be done and the robots decide how to do it and perform the task. The *motion planning problem* consists of designing a general algorithm for the robots to move from an initial position to a final one. We will use a topological approach to the robot motion planning problem initiated by Farber in [1].

2.2 Basic Notions

Since continuity is a key feature of our algorithm, we will start by recalling its definition for topological spaces. Let X and Y be topological spaces.

We say that $f : X \to Y$ is *continuous at* $a \in X$ if whenever V is an open neighborhood of $f(a)$ in Y, there is an open neighborhood U of a in X such that $f(U) \subset V$. If f is continuous at every point $a \in X$ then we say f is *continuous* on X.

A *path* from a point A to a point B in a topological space X is a continuous function α from the unit interval $I = [0, 1]$ to X with $\alpha(0) = A$ and $\alpha(1) = B$. The *path space*,

PX, is the space of all paths α in X, i.e. $PX = \{\alpha \mid \alpha : [0,1] \to X\}$. A space X is path-connected if for all points $x, y \in X$ there exists a path from x to y.

The *product space* of X and Y will be the Cartesian product $X \times Y = \{(x,y) \mid x \in X, y \in Y\}$ with the product topology.

The map $e : PX \to X \times X$ which takes a path α as its input and returns the pair $(\alpha(0), \alpha(1))$ is called the *evaluation map*. Here $\alpha(0)$ is the initial point of the path and $\alpha(1)$ is the final one.

If X is a path-connected space, let the function $s : X \times X \to PX$ be a section of the evaluation such that $e \circ s = id_{X \times X}$. This section takes as input two points and produces a path α between them as output:

$$s : (A, B) \mapsto \alpha \text{ and } \alpha(0) = A, \alpha(1) = B$$

where $\alpha(0) = A$ is the initial point and $\alpha(1) = B$ is the final point of the path.

Let $f : X \to Y$ be a bijection. If both the function f and the inverse function $f^{-1} : Y \to X$ are continuous, then f is called a homeomorphism. If such a function exists, we say X is *homeomorphic* to Y using the notation $X \approx Y$.

Let h and h' be continuous maps from X to Y. We say that h is *homotopic* to h', denoted by $h \simeq h'$, if there exists a continuous map

$$F : X \times I \to Y$$

such that

$$F(x, 0) = h(x) \text{ and } F(x, 1) = h'(x)$$

for each $x \in X$.

A map $f : X \to Y$ is a *homotopy equivalence* if there exists a map $g : Y \to X$ where $f \circ g \simeq id_Y$ and $g \circ f \simeq id_X$. If such maps exist, we say X and Y are *homotopy equivalent* and write $X \simeq Y$. Note that homeomorphic spaces are always homotopy equivalent.

A topological space X is *contractible* if it is homotopy equivalent to a point.

2.3 Configuration Space

The variety of all the possible states that any mechanical system can take is called the *configuration space*. Each point in the configuration space represents a state of the system. The space Γ where the robots are able to move will be called the *physical space*. The configuration space of n robots moving in a space Γ without collisions, denoted $C^n(\Gamma)$, is defined as:

$$C^n(\Gamma) = (\Gamma \times \Gamma \times \cdots \times \Gamma) - D.$$

Here D represents the pairwise diagonal:

$$D = \{(x_1, x_2, \ldots, x_n) \in \Gamma^n \mid x_i = x_j \text{ for some } i \neq j\}$$

given by the states in which at least two robots occupy the same place.

3 Configuration Space of two Robots on a Figure Eight Track

Let Γ denote the *figure eight* space, which is a wedge sum of two circles $S^1 \vee S^1$. We will consider the motion planning problem of two robots moving in the track $\Gamma = S^1 \vee S^1$. See figure 1.

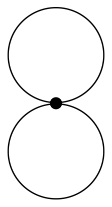

Figure 1: Physical space Γ.

In this case, the configuration space is $C^2(\Gamma) = (\Gamma \times \Gamma) - D$. We will visualize this space $X = C^2(\Gamma)$ first when embedded in $\mathbb{R}^3$ and then we will explain its flat representation that will be useful for calculations.

3.1 Visualization of the Configuration Space X

The robots in Γ are represented by triangles and squares. The triangle robot will be also referred to as the *first* robot and the square robot as the *second* one. We shade each circle with a different color to distinguish the two separate circles in the configuration space. Each concrete position in the physical space, see for instance figure 2, will have a corresponding state in the configuration space.

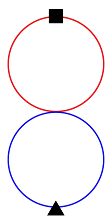

Figure 2: First and second robot in Γ

The Cartesian product $\Gamma \times \Gamma$ represents all positions of the two robots in Γ. The product of a figure eight by itself is given by four connected tori as in figure 3. For robots in the same circle, their states are represented by the red and blue tori whereas the purple tori represent states where the robots are in different circles.

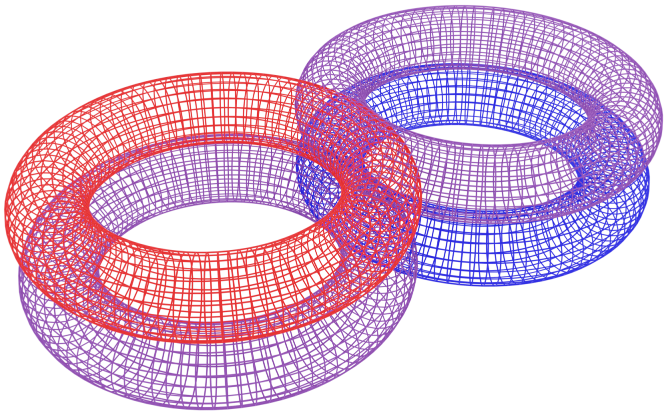

Figure 3: Cartesian product $(S^1 \vee S^1) \times (S^1 \vee S^1)$.

To obtain the configuration space, we need to remove the points in this Cartesian product that correspond to positions in which both robots are at the same place. We observe that the two tori representing positions of robots in the same circle have a diagonal removed from them, whereas the other two tori have just one point removed.

The diagonal $D = \{(x, y) \in (S^1 \vee S^1) \times (S^1 \vee S^1) |\ x = y\}$ determines a curve on the red and blue tori. The configuration space X is the Cartesian product $\Gamma \times \Gamma$ with this curve removed. See figure 4.

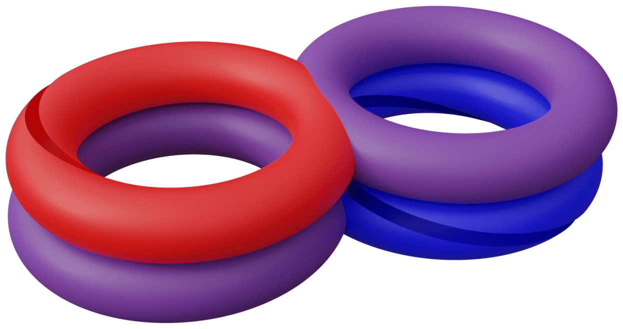

Figure 4: $X = (\Gamma \times \Gamma) - D$.

We will represent now the configuration space in the two-dimensional space by first considering the circle S^1 as a quotient of the unit interval where the points 0 and 1 are identified, i.e. $S^1 \approx I/\{0 \sim 1\}$. See figure 5.

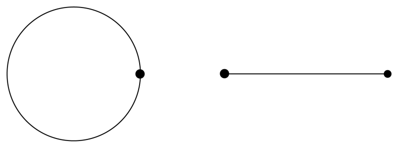

Figure 5: Flat representation of the circle.

The flat representation of the Cartesian product $\Gamma \times \Gamma$ is given by four connected squares with their sides identified as in figure 6.

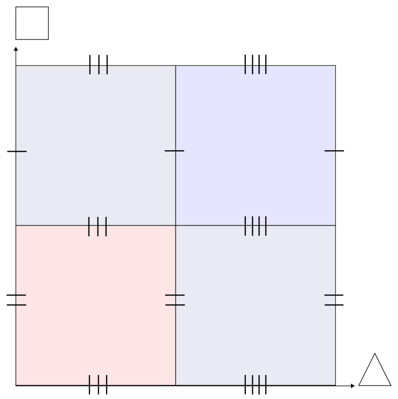

Figure 6: The product $\Gamma \times \Gamma$

The diagonal $D = \{(x, y) \in (S^1 \vee S^1) \times (S^1 \vee S^1) |\ x = y\}$ is given by the diagonals of the red and blue squares in the flat representation. The configuration space is the Cartesian product $\Gamma \times \Gamma$ with the diagonal D removed. See figure 7.

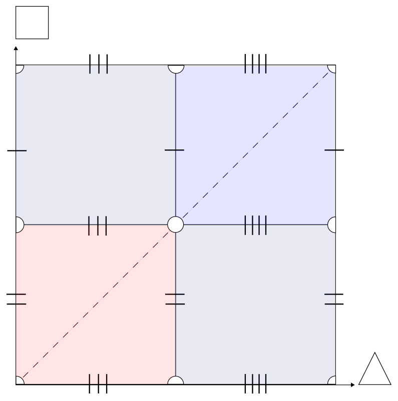

Figure 7: Flat representation of $X = (\Gamma \times \Gamma) - D$

Once that the diagonal is removed, we can visualize the flat representation of the configuration space as the following parallelogram with sides identified as shown in figures 8 and 9.

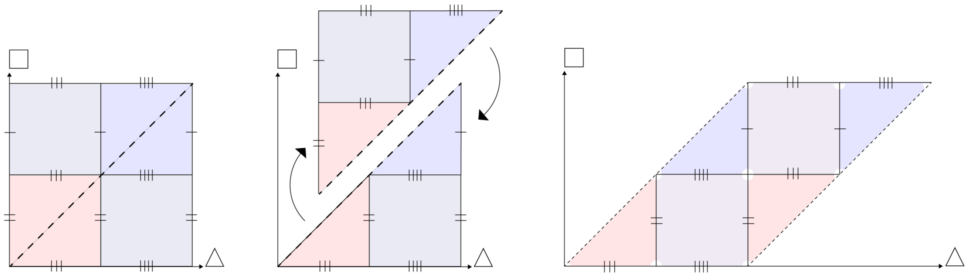
Figure 8: Transition from square to parallelogram representation.

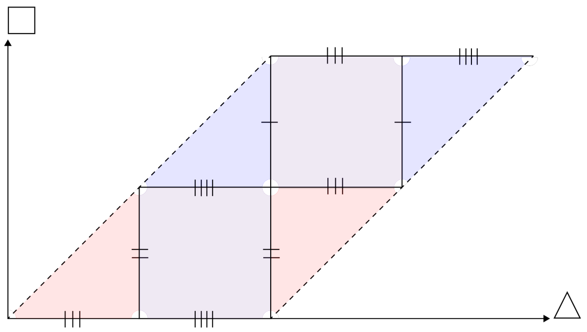
Figure 9: Flat representation of $X = (\Gamma \times \Gamma) - D$ as a parallelogram.

4 Topological Complexity

In order to program robots to move autonomously from any initial state to any desired state we will need a motion planning algorithm that takes as input the pair (A, B) where A is the initial state and B is the final state and outputs a continuous motion of the system starting at A and ending at B.

A *Motion Planning Algorithm* (MPA) is a section $s : X \times X \to PX$ of the evaluation map. It is a function that takes as input a pair of initial and final states, and outputs a path between them.

A motion planning algorithm $s : X \times X \to PX$ is a map that is not necessarily continuous. When it is continuous, if the inputs are slightly modified then the path between the inputs is only slightly modified too. The discontinuities of the MPA emerge as instabilities of the robot motion. We want to minimize such instabilities to produce instructions which are robust to noise: if there are small errors in the initial and final positions measurement, we want the paths associated to the exact positions and the one given by the measurement to be nearby.

Farber proved that the only case where a continuous motion planning algorithm exists is when the space is contractible.

Theorem 4.1 *[1] A continuous motion planning s: $X \times X \to PX$ exists if and only if the configuration space X is contractible.*

For spaces that are not contractible, all MPAs will be discontinuous. In real-life cases, we would like to minimize these discontinuities to produce optimally stable MPAs.

Farber invented a number known as *Topological Complexity*, $TC(X)$, that roughly can be described as the minimal number of continuous instructions which are needed to describe any motion planning algorithm in X.

This number measures how complex it is for the robots to navigate the space. For example, if the topological complexity of a space is two, then robots need a minimal number of two continuous instructions to move. The higher the TC is, the harder it is for the robots to navigate the space.

Definition 4.2 *[1][4] Let X be a path-connected topological space. The* topological complexity *of X, $TC(X)$, is the smallest number k, such that $X \times X$ can be covered by k sets $U_1, U_2, U_3, \ldots, U_k$ where on each of them there is a continuous local section $s_i : U_i \to PX$ for each $i = 1, 2, \ldots, k$.*

Each of these local sections will be our continuous instructions in the motion planning algorithm. For instance, if the space X is contractible we need only one section and $TC(X) = 1$. In the case of a circle, we need two sets to cover $S^1 \times S^1$. The first set is $U_1 = \{(x, y)|\ x \text{ is antipodal to } y\}$ and the second set is $U_2 = \{(x, y)|\ x \text{ is not antipodal to } y\}$. The first instruction over the set U_1 is *"move counterclockwise"* while the second instruction over U_2 is *"move following the shortest path"*. Both of these instructions are continuous and $TC(S^1) = 2$. Note that none of these instructions are continuous over $S^1 \times S^1$.

Farber also proved that this number is a homotopy invariant. If a space is homotopy equivalent to another, then their topological complexities are the same on their domains.

Theorem 4.3 *[1] If X is homotopy equivalent to Y, then $TC(X) = TC(Y)$.*

5 Homotopy Type of X

Our aim will be to find a simpler space that is homotopy equivalent to the space X and then construct our algorithm in the simpler space. Finding an algorithm and controlling the discontinuities directly in X is not trivial. Theorem 4.3 is crucial to reduce the complexity of the task by using the homotopy equivalence not only to calculate the topological complexity of a simpler space, but also to construct the actual algorithm in the simpler space and extend it to X afterward using the homotopy traces.

Ghrist proved that the configuration space of robots moving in a graph is homotopy equivalent to a space of lesser dimension.

Theorem 5.1 *[2] Given a graph Γ, the configuration space of N distinct points on Γ can be deformation retracted to a spine whose dimension is bounded above by the number of vertices of Γ of valency greater than two.*

Our figure-eight graph has only *one* vertex with valency greater than two and therefore the configuration space retracts to a spine of dimension 1. We know then that it is possible to shrink the space X to a one-dimensional space. We will show next the explicit construction of this spine for our graph.

5.1 Construction of the Spine

The *spine* Z of the configuration space is a one-dimensional space that carries all of the topology of the configuration space through reducing the full space by deformation retraction to a simpler space. The *center* of the physical space Γ is the point at which the circles intersect and the *poles* are the antipodal points to the center in each circle.

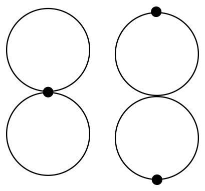

Figure 10: The center and the poles, respectively.

The spine Z is a set of segments in the flat representation with their extremes identified that represents the positions of the robots in Γ where at least one robot is at a pole and the other at a different circle or both robots are at antipodal positions in the same circle.

We construct a retraction r of the whole space X into the spine Z. The homotopy deformation H will follow the traces $H(\_,\mathrm{t})$ shown in figure 11.

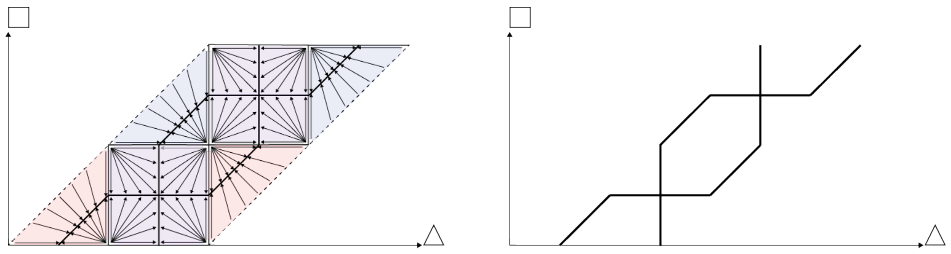

Figure 11: Homotopy traces of the retraction of X into Z.

The map r is a deformation retraction since it is a retraction and the composition with the inclusion is homotopic to the identity map in X.We observe the following identifications of the segment extremes in Z shown in figure 12.

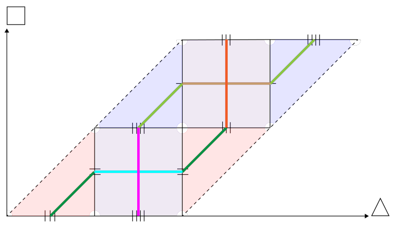

Figure 12: Spine Z.

These identifications make the spine Z homeomorphic to a chain of circles C. We color each segment to describe this homeomorphism.

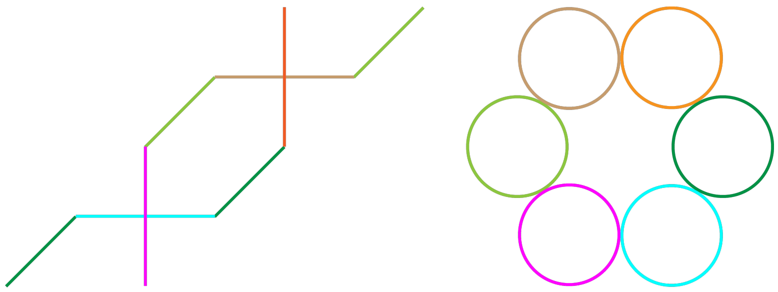

Figure 13: The spine Z is homeomorphic to the chain C.

We can also visualize the chain of circles C directly in the representation of X embedded in $\mathbb{R}^3$. By following the traces of the homotopy, the two cylinders deform into circles and each torus minus a point deforms into a figure-eight. See figure 14. Note that in figures 14 and 15, the dashed lines are the diagonal D.

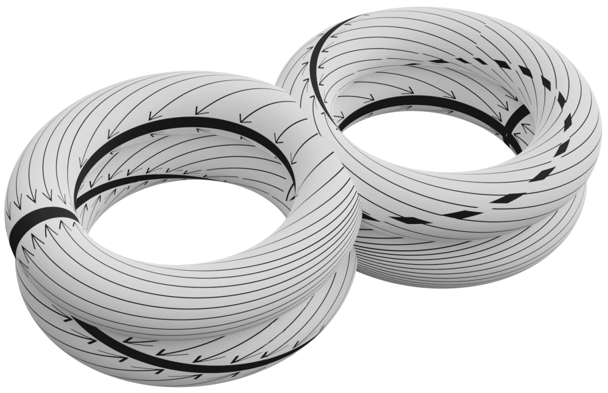

Figure 14: Traces of the homotopy in X.

We can visualize in the following figure the chain of circles C embedded in the three-dimensional space.

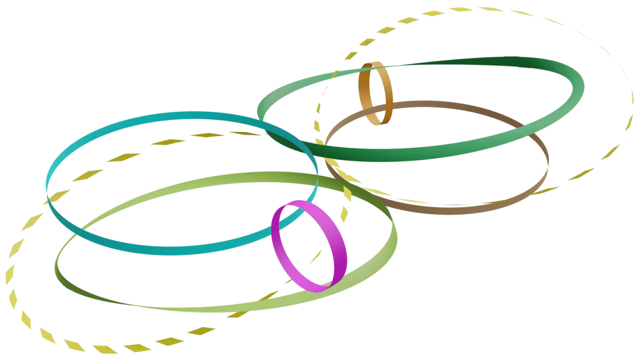

Figure 15: The chain C.

5.2 The Wedge of Seven Circles

We will show in this section that the chain C is homotopy equivalent to a wedge of seven circles Y. We perform a series of contractions in each circle as shown in figure 16.

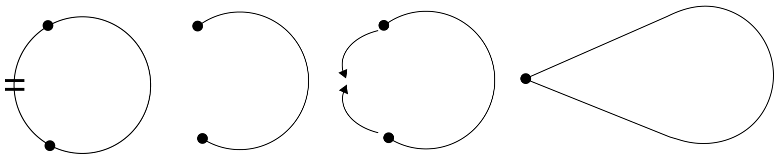

Figure 16: Contraction of a segment in a circle.

We repeat this deformation in each of the circles. See figures 17 and 18.

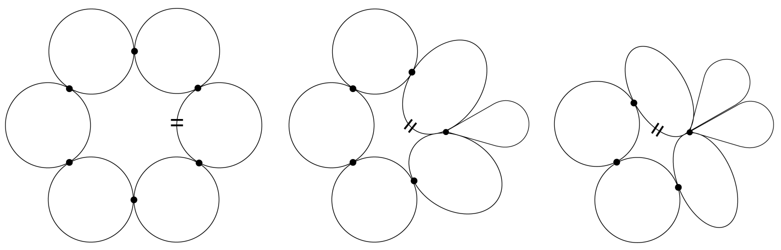

Figure 17: First stages of the deformation.

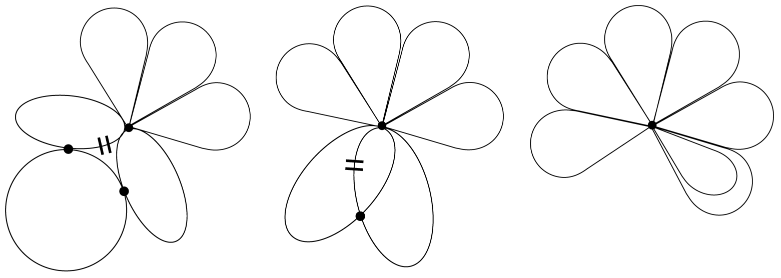
Figure 18: Last stages of the deformation.

The final space is a wedge of seven circles as shown in figure 19.

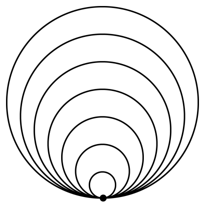
Figure 19: Wedge of seven circles Y.

5.3 Calculation of the Topological Complexity of the Configuration Space

The configuration space X is homotopy equivalent to the spine Z and Z is homeomorphic to the chain of circles C, which is homotopy equivalent to the wedge Y, i.e.

$$X \simeq Z \approx C \simeq Y.$$

Then, by Farber's theorem 4.3 we have that

$$TC(X) = TC(Z) = TC(C) = TC(Y). \tag{1}$$

Farber also studied and calculated the topological complexity of wedges of spheres:

Lemma 5.2 *[3] Let W denote the wedge of n spheres S^m, $W = S^m \vee S^m \vee \cdots \vee S^m$. Then* $TC(W) = \begin{cases} 2 & \textit{if } n = 1 \textit{ and } m \textit{ is odd,} \\ 3 & \textit{if either } n > 1, \textit{ or } m \textit{ is even.} \end{cases}$

For our space Y we have that $m = 1$ and $n = 7$. By lemma 5.2, we know that the topological complexity of a wedge of seven circles is 3, then $TC(X) = 3$ following equation 1. We know now that our algorithm will have to have at least 3 continuous instructions.

6 Algorithm in the Configuration Space

We will construct first our algorithm with three instructions in the spine Z that is homotopy equivalent to X. Then we will extend the algorithm to the whole configuration space X following the traces of the homotopy.

Recall that the spine Z in the flat representation is a network consisting of two crosses and four sub-diagonals.

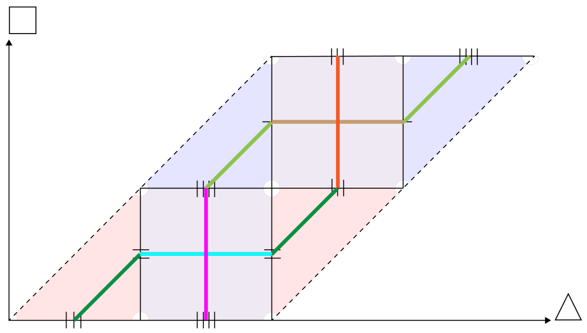

Figure 20: Space Z

When the state is in a diagonal segment in Z, we will say that the state is a *sub-diagonal state*. In the physical space Γ, this state will correspond to antipodal positions in the same circle.

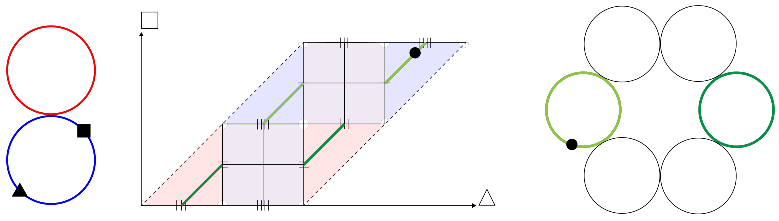

Figure 21: A sub-diagonal state.

When a state is located in a cross-segment in Z, we will say that it is a *cross-state*. In the physical space Γ, cross-states translate as the position of at least one robot being at a pole and the other anywhere on the opposite circle.

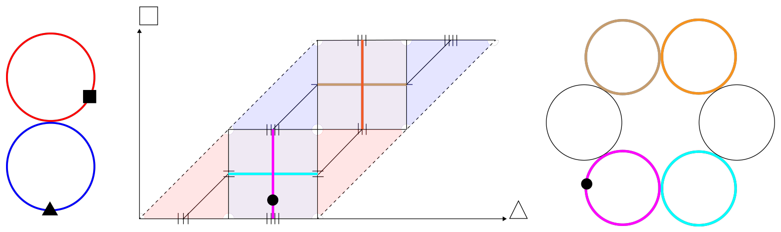

Figure 22: A cross-state.

A *cross center* is a cross-state at the intersection of horizontal and vertical cross-segments. A cross center corresponds to the positions in which both robots are at poles in different circles in Γ.

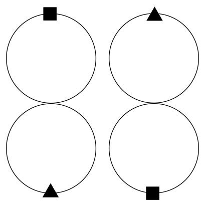

Figure 23: Cross centers in the physical space Γ.

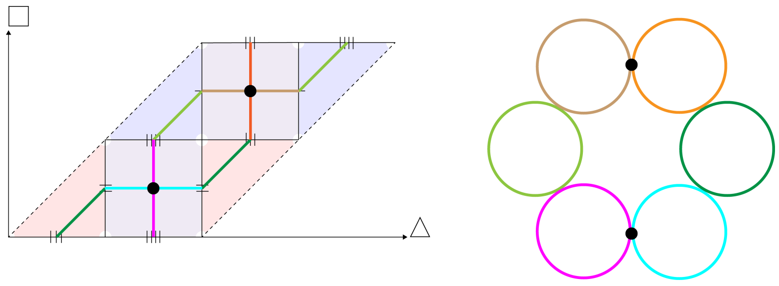

Figure 24: Cross centers in Z and C.

An *intersection point* is a state in Z at the intersection between a cross-segment and a sub-diagonal segment. In Γ, this state corresponds to positions in which one robot is at the center and the other at a pole. There are two types of intersection points: *H-points* located between a sub-diagonal segment and a horizontal cross-segment, and *V-points* located between a sub-diagonal segment and a vertical cross-segment. An H-point corresponds in the physical space to a position in which the first robot is at the center and the second robot is at a pole. Conversely for V-points.

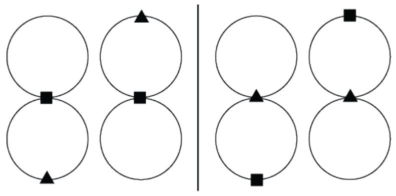

Figure 25: Intersection V-points(left) and H-points(right) in the physical space Γ.

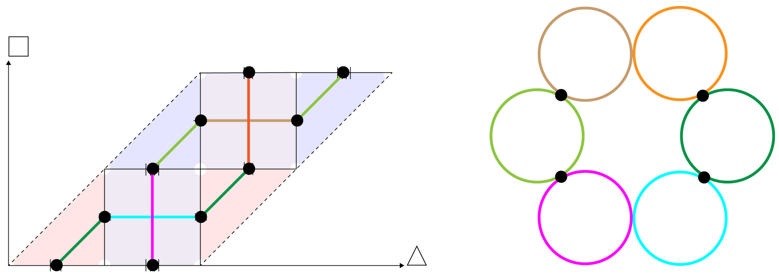

Figure 26: Intersection points in Z and C.

The cross and sub-diagonal segments in Z will constitute the circles in the chain C whereas the intersection points and cross centers will be the vertices in C.

6.1 Algorithm in the Chain C

The initial, final, and current states are the states corresponding to the initial, final, and current positions respectively.

If the current state is at a vertex in C, we will say that the *current circle* is the next immediate circle counterclockwise. Otherwise, the current circle is the circle where the current state is.

Let $(I, F) = ((x_i, y_i), (x_f, y_f))$ be a pair of initial and final states in $C \times C$. We define the following subsets of $C \times C$:

- $A = \{(I, F) \in C \times C \mid I$ is antipodal to F in the same circle$\}$
- $V = \{(I, F) \in C \times C \mid I$ or F is a vertex$\}$
- $V' = \{(I, F) \in C \times C \mid I$ and F are vertices$\}$

Algorithm 6.1 Set current state to initial state I

Instruction 1 For $(I, F) \in C \times C - (A \cup V)$:

```
1. Check if final state is in current circle.
    • If true, take shortest path to final state.
    • If false, move counterclockwise in current circle to next vertex and set
      current state to that vertex. Repeat 1.
```

Instruction 2 For $(I, F) \in (A \cup V) - V'$:

```
2. Check if final state is in current circle.
    • If true, check if final state is antipodal to current state.
        – If true, go counterclockwise to final state.
        – If false, take shortest path to final state.
    • If false, check if current state is a vertex.
        – If true, move counterclockwise in current circle to next vertex and set
          current state to that vertex. Repeat 2.
        – If false, take shortest path to next counterclockwise vertex and set
          current state to that vertex. Repeat 2.
```

Instruction 3 For $(I, F) \in V'$:

```
3. Check if final state is next counterclockwise vertex.
    • If true, go counterclockwise in the current circle to final state.
    • If false, go counterclockwise in the current circle to next vertex and set
      current state to that vertex. Repeat 3.
```

Each of these instructions is continuous in its domain.

6.2 Algorithm in Z

By following the homeomorphism between C and Z we will translate now each of these instructions to the spine Z.

We define a distinguished direction of movement in Z that will correspond to moving from vertices counterclockwise in the current circle in C. The *positive direction in* Z is given by:

- from a cross center, move to the right in the horizontal segment;
- from a H-point, move up in the diagonal segment;
- from a V-point, move up in the vertical segment;
- from any other point, move to the right in horizontal or diagonal segments and up in vertical segments.

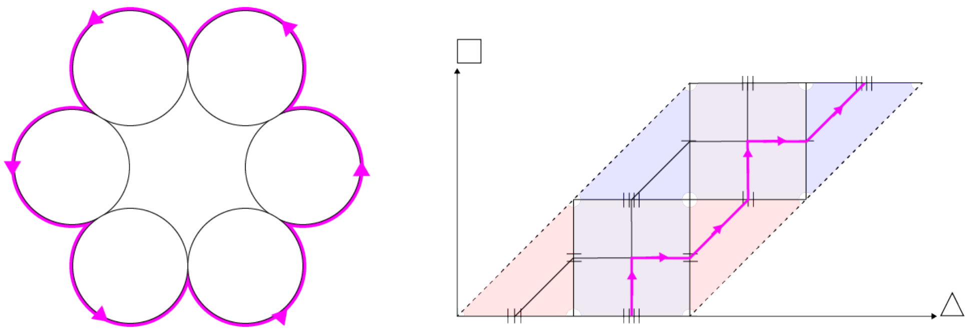
Figure 27: Positive direction in C and Z.

If the current state is at an intersection point or cross center in Z, we will say that the *current segment* is the next immediate segment in the positive direction. Otherwise, the current segment is where the current state is.

We consider Z to be a metric graph where each edge has length one and the distance between two points is given by the infimum of the lengths of paths joining them.

Let $(I, F) = ((x_i, y_i), (x_f, y_f))$ be a pair of initial and final states in $Z \times Z$. We define the following subsets of $Z \times Z$:

- $H = \{(I, F) \in Z \times Z \mid I$ and F are on the same segment and $d(I, F) = \frac{1}{2}\}$
- $J = \{(I, F) \in Z \times Z \mid I$ or F are at a cross center or intersection point$\}$
- $J' = \{(I, F) \in Z \times Z \mid I$ and F are both at cross centers or intersection points$\}$

Algorithm 6.2 Set current state to initial state I

Instruction 1 For $(I, F) \in Z \times Z - (H \cup J)$:

1. Check if final state is in current segment.
   - If true, take shortest path to final state.
   - If false, move in the positive direction in current segment to next point of intersection and set current state to that point of intersection. Repeat 1.

Instruction 2 For $(I, F) \in (H \cup J) - J'$:

2. Check if final state is in current segment.
   - If true, check if the distance from current state to final state is $\frac{1}{2}$.
     - If true, move in the positive direction to final state.
     - If false, take shortest path to final state.
   - If false, check if current state is a point of intersection.
     - If true, move in the positive direction along current segment to next point of intersection and set current state to that point of intersection. Repeat 2.
     - If false, take shortest path, in the positive direction, to next point of intersection and set current state to that point of intersection. Repeat 2.

Instruction 3 For $(I, F) \in J'$:

3. Check if final state is the next point of intersection in the positive direction.
   - If true, move in the positive direction along the current segment to final state.
   - If false, move in the positive direction along the current segment to next point of intersection and set current state to that point of intersection. Repeat 3.

6.3 Algorithm in X

We will extend now the algorithm 6.2 to the whole configuration space.

Given an initial state I, let I_s be the state corresponding to the point of intersection between the trace of the homotopy passing through I and the spine Z. Analogously, F_s is the corresponding state to the final one in the spine.

Algorithm 6.3 Let I and F be the initial and final states.

1. Find (I_s, F_s).

2. Move state I to I_s.

3. Execute algorithm 6.2 for (I_s, F_s).

4. Move state F_s to F.

Given the initial and final states shown in figure 28, the following figures illustrate algorithm 6.3 for this case.

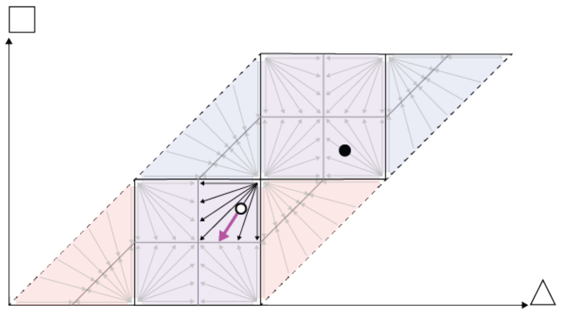

Figure 28: Step 2.

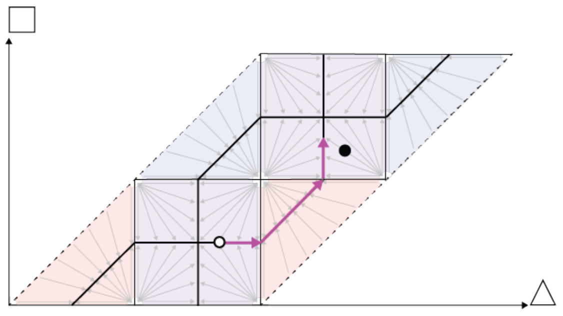

Figure 29: Step 3.

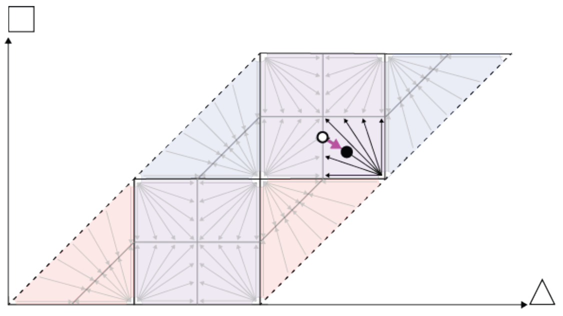

Figure 30: Step 4.

7 Algorithm in the Physical Space

We will translate now our algorithm 6.3 to the physical space Γ.

Observe that states in the spine Z correspond to positions in which at least one robot is at a pole if the robots are in different circles and antipodal positions if the robots are in the same circle.

7.1 States (I_s, F_s) in the Physical Space Γ

To translate the first step of the algorithm, we need to find the positions of the robots in Γ corresponding to the intersection of the traces with the spine.

If the robots are in the same circle, move the robots away from each other until they are in antipodal positions. See figure 31. If they are in different circles, move them away from the center until at least one of them reaches a pole position. See figure 32. If one robot is at the center, move the other robot until it reaches a pole position. See figure 33.

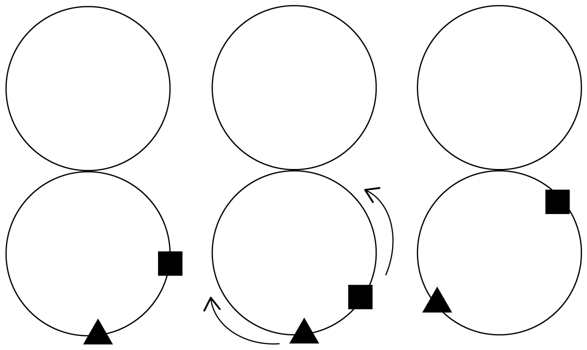

Figure 31: Moving to antipodal positions

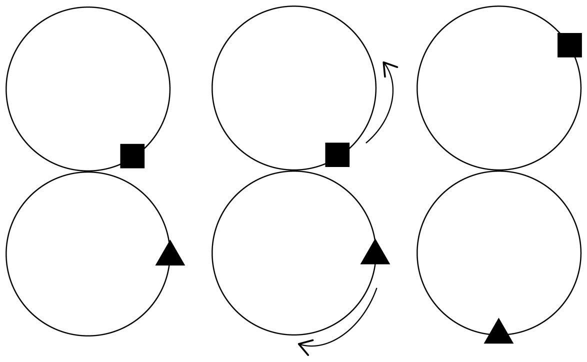

Figure 32: Moving until one robot reaches the pole.

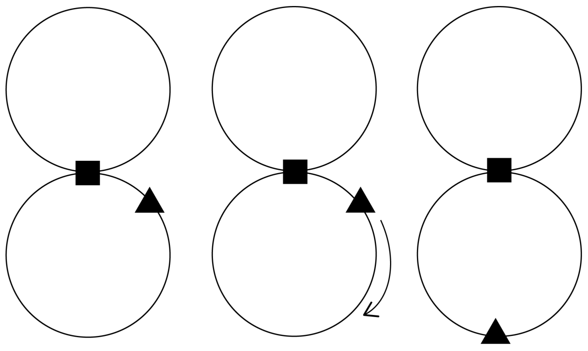

Figure 33: Robot in center scenario.

The ratio between the speeds at which the robots are moving is determined by the slope of the traces of the homotopy discussed in section 5.1. If a trace of the homotopy is given by the curve $(x(t), y(t))$, then the slope of the trace is given by the ratio between the speeds $y'(t)$ and $x'(t)$.

7.2 Algorithm in the Physical Space for the Positions in Z

We will translate here the movement in the positive direction in Z to the physical space. Recall that a cross-center state corresponds to both robots being at poles, and intersection points in Z correspond to one robot being at a pole and the other at the center. In the physical space, we will call these positions *vertex positions.*

The *positive direction in* Γ is given by:

- if both robots are at poles, move robot 1 counterclockwise;
- if the first robot is at the center and the other at a pole, move both counterclockwise;
- if the first robot is at a pole and the other at the center, move the second robot counterclockwise.

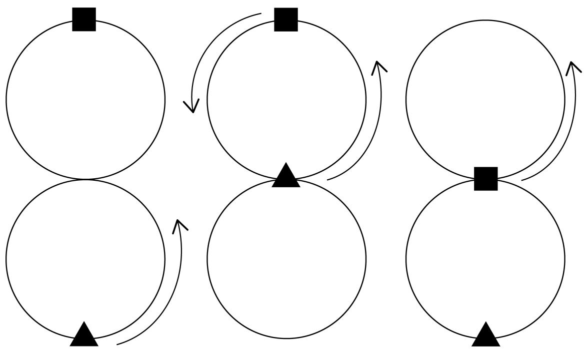

Figure 34: Positive direction in Γ.

If the current position is a vertex position, we will say that its *neighborhood* is the set of positions determined by moving the current position in the positive direction until it circles back to itself. Otherwise, the neighborhood is given by moving counterclockwise the robot that is not at a pole all around its circle if the robots are at different circles or moving counterclockwise both robots together if they are antipodal in the same circle.

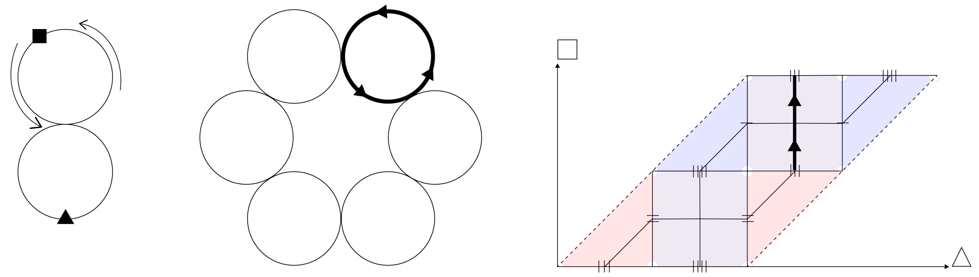

Figure 35: A neighborhood of a non-vertex position.

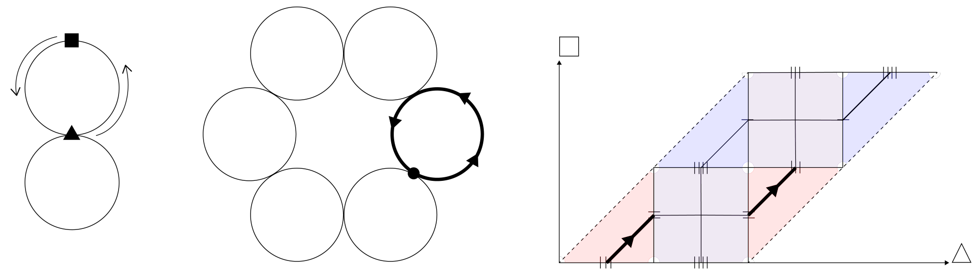

Figure 36: A neighborhood of a vertex position.

Let $(I, F) = ((x_i, y_i), (x_f, y_f))$ be a pair of initial and final positions in $\Gamma \times \Gamma$. We define the following subsets of $\Gamma \times \Gamma$:

- $K = \{((x_i, y_i), (x_f, y_f)) \mid x_i$ and x_f are antipodal or y_i and y_f are antipodal$\}$
- $L = \{((x_i, y_i), (x_f, y_f)) \mid (x_i, y_i)$ or (x_f, y_f) is a vertex position$\}$
- $L' = \{((x_i, y_i), (x_f, y_f)) \mid (x_i, y_i)$ and (x_f, y_f) are vertex positions$\}$

Algorithm 7.1 Set current position to initial position I.

Instruction 1 For $(I, F) \in \Gamma \times \Gamma - (K \cup L)$:

1. Check if final position is in the neighborhood of the current position.
    - If true, move each robot simultaneously to their final position taking the shortest path.
    - If false, move in the positive direction in current neighborhood until next vertex position. Set current position to this one. Repeat 1.

Instruction 2 For $(I, F) \in (K \cup L) - L'$:

2. Check if final position is in the current neighborhood.
    - If true, check if at least one of the robots have its initial and final positions antipodal.
        - If true, move in the positive direction to final position.
        - If false, move each robot simultaneously to their final position taking the shortest path.
    - If false, check if current position is a vertex position.
        - If true, move in the positive direction in current neighborhood to the next vertex. Set current position to this one. Repeat 2.
        - If false, take shortest path to next vertex position in the positive direction and set current position to this one. Repeat 2.

Instruction 3 For $(I, F) \in L'$:

3. Check if final position is the next vertex position in the positive direction.
    - If true, move in the positive direction in the current neighborhood to final position.
    - If false, move in the positive direction in the current neighborhood to next vertex position and set current state to that position. Repeat 3.

7.3 Complete Algorithm in the Physical Space Γ

We will write now the complete algorithm for all positions in Γ.

Algorithm 7.2 Let I and F be the initial and final positions.

1. If the robots are in the same circle, move the robots away from each other until they are in antipodal positions.

2. If they are in different circles, move them away from the center until at least one of them reaches a pole position.

3. Repeat steps 1 and 2 with the final position. Let us call initial Z-position and final Z-position the output of this step.

4. Execute algorithm 7.1 for the initial Z-position and final Z-position.

5. Move back from the final Z-position to the final position reversing the movement done in the first step.

In the following case scenarios, we describe the movements in the physical space as well as in the configuration space.

7.3.1 Case Scenario 1

Let us consider the initial and final positions shown in figure 37.

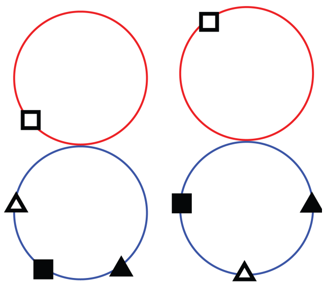

Figure 37: Initial and final positions and their corresponding spine positions.

The path to move from initial state to final state in the configuration space is shown in figure 38.

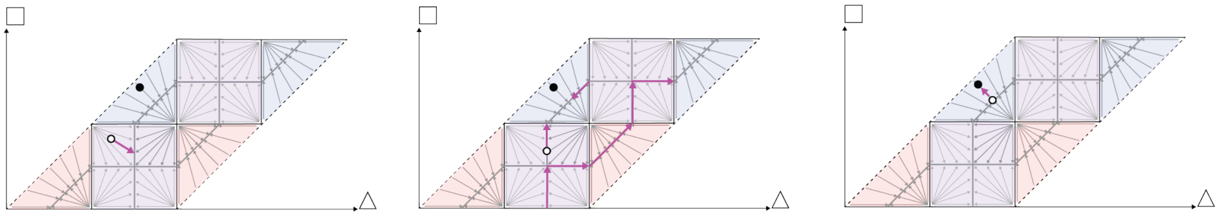

Figure 38: Path in X for case 1.

The steps of the algorithm to move from initial to final positions in the physical space are shown in figure 39.

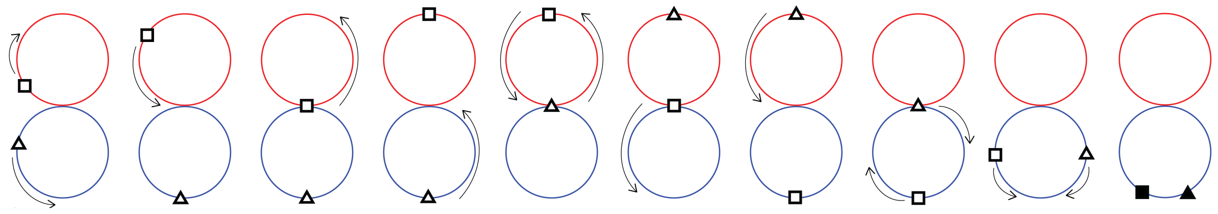

Figure 39: Steps of the algorithm in Γ for case 1.

7.3.2 Case Scenario 2

Let us consider the initial and final positions shown in figure 40.

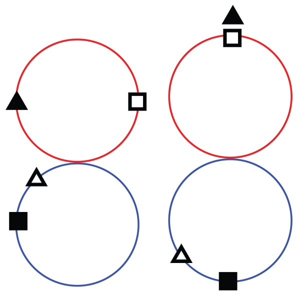

Figure 40: Initial and final positions and their corresponding skeleton positions.

The path to move from initial state to final state in the configuration space is shown in figure 41.

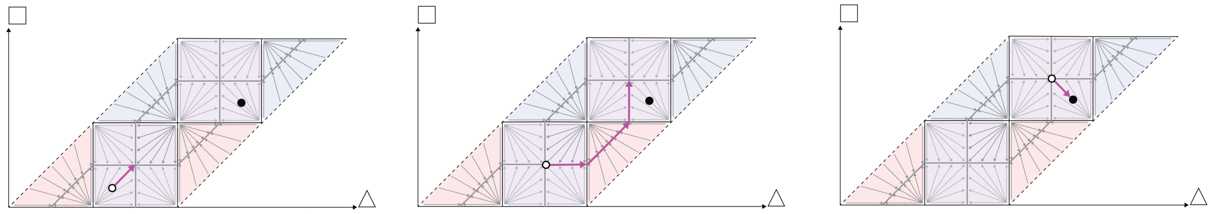

Figure 41: Path in X for case 2.

The steps of the algorithm to move from initial to final positions in the physical space are shown in figure 42.

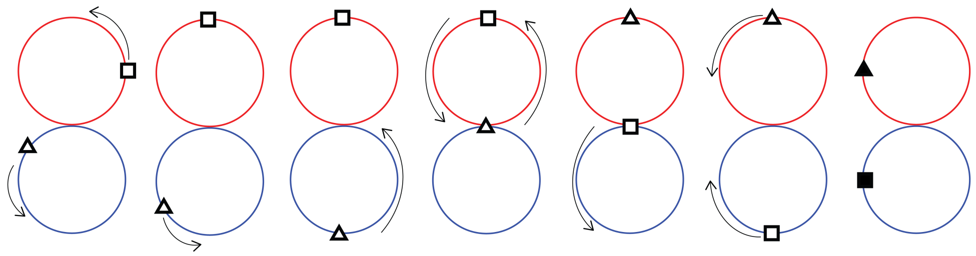

Figure 42: Steps of the algorithm in Γ for case 2.

7.3.3 Case Scenario 3

Let us consider the initial and final positions shown in figure 43.

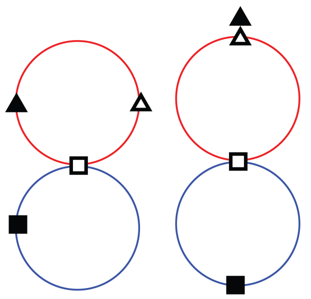

Figure 43: Initial and final positions and their corresponding skeleton positions.

The path to move from initial state to final state in the configuration space is shown in figure 44.

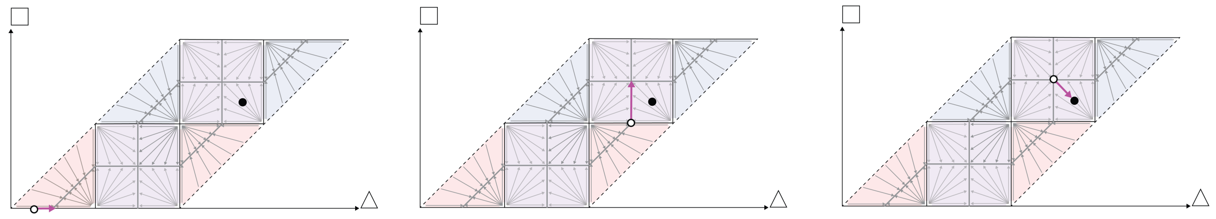

Figure 44: Path in X for case 3.

The steps of the algorithm to move from initial to final positions in the physical space are shown in figure 45.

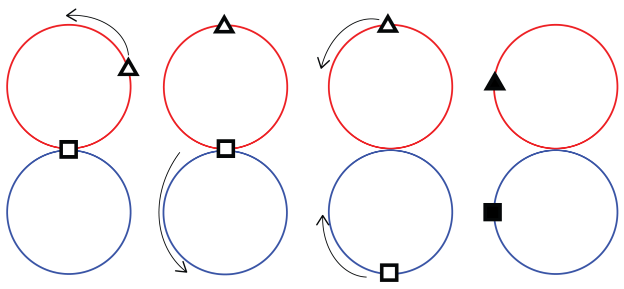

Figure 45: Steps of the algorithm in Γ for case 3.

References

[1] M. Farber, Topological complexity of motion planning, *Discrete Comput. Geom.*, **29** (2003), 211–221.

[2] R. Ghrist, D. Koditschek, Safe Cooperative Robot Dynamics on Graphs, *SIAM J. Control Optim.*, **40** (2002), 1556–1575.

[3] M. Farber, Instabilities of robot motion, *Topology Appl.*, **140** (2004), 245–266.

[4] J.M. García-Calcines, A note on covers defining relative and sectional categories, *Topology Appl.*, **265** (2019), 106810.

Cristian Jardon
Wilbur Wright College
4300 N Narragansett Ave
Chicago, IL 60634
E-mail: `cjardonccc@gmail.com`

Brian Sheppard
Harold Washington College
30 E. Lake Street
Chicago, IL 60601
E-mail: `bshepp@gmail.com`

Veet Zaveri
Wilbur Wright College
4300 N Narragansett Ave
Chicago, IL 60634
E-mail: `veet.zaveri@gmail.com`

Received: November 9, 2022 **Accepted**: May 26, 2023
Communicated by Mona Mocanasu