COLLISION FREE MOTION PLANNING ON A WEDGE OF CIRCLES

ELIF SENSOY

Abstract. We exhibit an algorithm with continuous instructions for two robots moving without collisions on a track shaped as a wedge of three circles. We show that the topological complexity of the configuration space associated with this problem is 3. The topological complexity is a homotopy invariant that can be thought of as the minimum number of continuous instructions required to describe the movement of the robots between any initial configuration to any final one without collisions. The algorithm presented is optimal in the sense that it requires exactly 3 continuous instructions.

(Elif Sensoy, Author) Wright College, 4300 N. Narragansett Avenue, Chicago, IL 60634 USA

(Hellen Colman, Advisor) Department of Mathematics, Wright College, 4300 N. Narragansett Avenue, Chicago, IL 60634 USA

E-mail address: esensoy@student.ccc.edu, hcolman@ccc.edu.

1. Introduction

It is an important problem in robotics to coordinate the movements of vehicles along a system of tracks without collisions. This problem was originally studied by Ghrist in [7] by constructing the configuration space $C^n(G)$ of n particles on a graph G, given by

$$C^n(G) = \{(x_1, \dots, x_n) \in G^n \mid x_i \neq x_j \text{ for } i \neq j\}.$$

Every collision-free movement between two configurations of n points on the graph G corresponds to a path in the configuration space $C^n(G)$. The *motion planning problem* consists of finding a function that assigns to any pair of points in the configuration space, a path between them.

Mathematically, we can formalize this idea with the aid of the evaluation fibration. For any path-connected space X, let $P(X)$ denote the space of continuous paths in X. The *evaluation fibration* $\mathrm{ev} : P(X) \to X \times X$ sends a path α to its endpoints: $\mathrm{ev}(\alpha) = (\alpha(0), \alpha(1))$. A section s of this fibration takes a pair of points as input and gives a path between those points as output. This section can be viewed as a rule

$$s : X \times X \to P(X)$$

which assigns to each pair of initial and final positions, a path between them in X. If s is continuous, we have that any small perturbation in the initial and final positions will lead to only small variations between the assigned paths. Farber [4] proved that it is impossible to find a section that is continuous over the whole domain $X \times X$ if space X is not contractible. Motivated by this, Farber introduced the topological complexity of a space, see [3] for a more recent survey on the subject. The topological complexity of X, denoted by $TC(X)$, is a measure of this inability to find a continuous section over the whole domain. The topological complexity of a space is invariant under homotopy.

We provide an explicit algorithm for two robots moving in a wedge of three circles Γ, which is optimal in the sense that the motion planning is performed with the minimal number of instabilities given by the topological complexity, $TC(C^2(\Gamma))$.

The organization of the paper is as follows. In the second section, we describe the statement of the motion planning problem in our case and give a thourough description of the configuration space. In the third section, we introduce the topological complexity.

Our approach consists of presenting a comprehensive construction of the configuration space for the case of two robots moving on a wedge of circles and then building a deformation retract of this space into a simpler space that we call the *network*. We dedicate section 4 to this construction. We also describe here some homeomorphic spaces to the network that will make possible the calculation of the topological complexity in the next section.

In section 5, we use the homotopy invariance to calculate the topological complexity of the configuration space, which provides us with the minimal number of instructions for our algorithm.

We exhibit an algorithm with a minimal number of instructions in section 6. This is the main section of this article. We give a detailed description of the instructions in the configuration space as well as in the physical space. We show how these instructions work in a sample case of two robots moving from a particular initial configuration to a final one.

This paper is the result of an undergraduate research project at Wilbur Wright College supervised by Professor Hellen Colman.

2. Motion Planning Problem

We consider two robots confined to move on a graph given by three loops joined at a point, where a collision between robots is not allowed. This restricted movement of the robots in tracks is quite common in applications of motion planning and can be viewed as an instance of the problem of simultaneous control of two objects avoiding collisions with each other.

While there is a lot of literature about motion planners in graphs, there are relatively few explicit examples of efficient motion planning algorithms carried out in detail in particular graphs. For a different approach to construct an algorithm in a lollilop graph, see [1].

Our problem involves constructing a program that allows two robots to move from their initial positions to their final ones without collisions. We want our algorithm to be robust, in the sense that small variations in the measuring of initial and final positions should lead to just small variations in those paths.

We will now define some terms that we will use throughout the paper. A *robot* is a mechanical system capable of moving autonomously. The *physical space* is the physical environment in which the robots move.

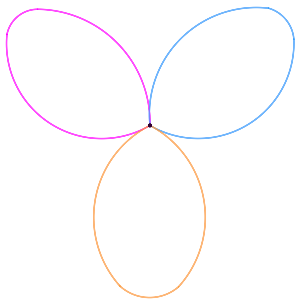
In our setting, the robots are two points (A, B) and the physical space Γ is a wedge of three circles, $\Gamma = \bigvee_3 S^1$.

2.1. **Configuration Space.** Although the motion planning problem is defined in the physical space where the robots move, we will study another auxiliary space where it will be easier to design the motion: the configuration space. A *state* of the system is a specification of the positions of all robots, and the *configuration space* is the space of all possible states.

Motion planning problems involve navigation of the system through the configuration space in order to achieve a certain objective. The problem of finding individual paths for each robot in the physical space from initial to final positions translates into finding a single path in the configuration space from initial to final states.

Let (A, B) be two robots moving in a graph Γ that consists of three circles joined at a point. The physical space is a wedge of circles as shown in figure 1.

FIGURE 1. Physical space $\Gamma = \bigvee_3 S^1$

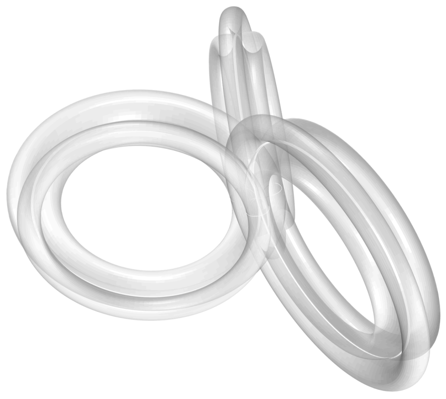


In order to construct the configuration space X of two robots moving on a track Γ, $X = C^2(\Gamma)$, first we find the Cartesian product of $\Gamma \times \Gamma$ of all states and then exclude the diagonal Δ of collision states. Formally, the configuration space is

$$X = C^2(\Gamma) = \Gamma \times \Gamma - \Delta = \Big(\bigvee_3 S^1 \times \bigvee_3 S^1\Big) - \Delta$$

The space $\Gamma \times \Gamma$ consists of three groups of three tori joined by a wedge of three circles. The circles in this wedge are meridians of the three tori in each group, and each group is a union of three tori with a common parallel circle. See figure 2.

Figure 2. Cartesian product $(\bigvee_3 S^1) \times (\bigvee_3 S^1)$

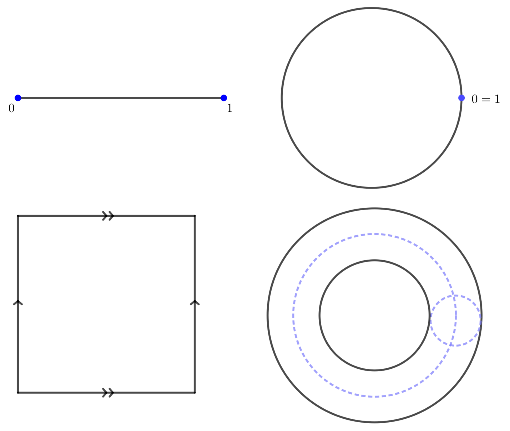


The product $\Gamma \times \Gamma$ consists of nine tori in total and is considerably complicated to visualize since even though the dimension of the space is two, it is not possible to embed it in R^3.

To remove the diagonal, we take into consideration that the collision points are the states where the two robots are in the same circle in Γ: only one of the tori in each group would contain the entire diagonal circle. Once the diagonal is removed, each of these three tori will be cut open across its diagonal circle and homeomorphic to a cylinder. The other two tori in each group will be missing one point after the diagonal is removed.

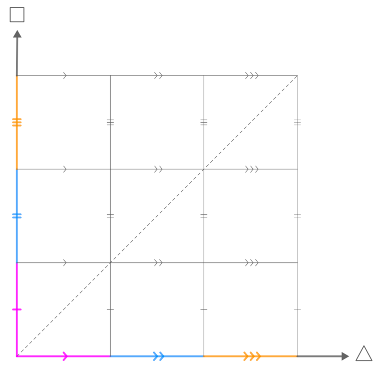
The configuration space is then composed of six tori and three cylinders all joined in an intricate way. Our approach will be to construct a flat representation of this space that will allow us to manipulate the space and understand it better. In this representation, each of the circles in the wedge are considered segments with the extreme points identified and each torus corresponds to a square with the edges identified as in the figure 3.

Figure 3. Interval to circle and square to torus



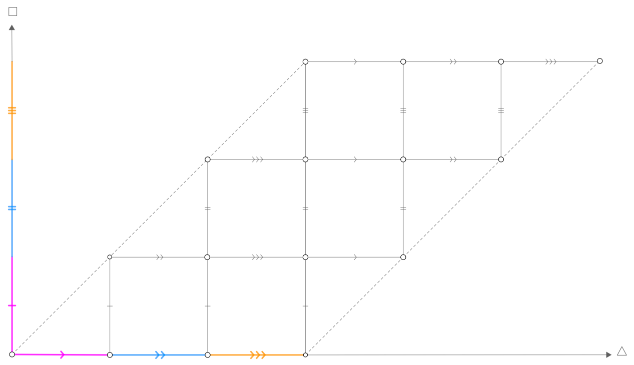
To track the robots in this representation, we will use the Cartesian system of coordinates, where the x-axis represents the positions of the first robot A and the y-axis represents the positions of the second robot B. Each state represents the combined positions of the two robots (A, B) in $\Gamma \times \Gamma$. Alternatively, we will also use a triangle shape $\triangle$ for the robot A and a square shape $\square$ for the robot B. Figure 4 depicts the flat representation of $\Gamma \times \Gamma$.

FIGURE 4. Flat Representation of $\Gamma \times \Gamma$



The next step is to subtract the diagonal Δ from the Cartesian product of $\Gamma \times \Gamma$ to obtain the configuration space $X = \Gamma \times \Gamma - \Delta$. Our configuration space X consists of six squares with vertices removed and with the edges identified; and three pairs of triangles, each pair with one edge identified. The squares correspond to those configurations where the two robots are on different circles in Γ, and the triangles correspond to the positions of robots in the same circle. Therefore, the configuration space X consists of six squares and six triangles with the identifications are shown in figure 5.

FIGURE 5. Flat representation of $\Gamma \times \Gamma - \Delta$



3. TOPOLOGICAL COMPLEXITY

The notion of topological complexity of the configuration space X, $TC(X)$, was introduced by Farber in [4]. If X is the configuration space of robots moving in a physical space Γ, then the topological complexity is a homotopy invariant that measures the navigational complexity of robots moving in Γ. This number can be defined as the minimum number of *continuous rules* required to describe the movement between any two points in X. A motion planning algorithm takes a pair of points as input, the initial and the final states, and returns a continuous path as output between them. In other words, a motion planning algorithm is a section $s : X \times X \to PX$ of the evaluation map $\text{ev} : PX \to X \times X$. That is, the section s receives as input the initial and final positions of the robots (A_i, B_i) and (A_f, B_f) and returns as output a path $\alpha : I \to X$ such that $\alpha(0) = (A_i, B_i)$ and $\alpha(1) = (A_f, B_f)$.

As we stated before, we would like a *robust* motion planning algorithm. Unfortunately, most navigation plans are discontinuous. Farber showed that a continuous navigation plan exists if and only if X is contractible and for non-contractible spaces considered navigation plans that are continuous only when restricted to subsets of the whole Cartesian product $X \times X$. The *topological complexity* of

X is the minimal k for which $X \times X$ can be covered by k open subsets over each of which the section is continuous. These open sets are called *domains of continuity*.

In the same paper, Farber proved that the topological complexity is invariant under homotopy.

Theorem 3.1. [4, theorem 3] If X has the same type of homotopy as Y, then $TC(X) = TC(Y)$.

This result will allow us to find the topological complexity of our configuration space by deforming X into a simpler space with the same type of homotopy and then calculating its topological complexity.

4. Homotopy of the Configuration Space

As previously mentioned, topological complexity is the same across spaces with the same type of homotopy. Finding another space N that is homotopically equivalent to X will reduce the complexity of the configuration space and allow us to work on a relatively simpler space, which will be of great use not only to calculate the topological complexity but also to describe the final algorithm.

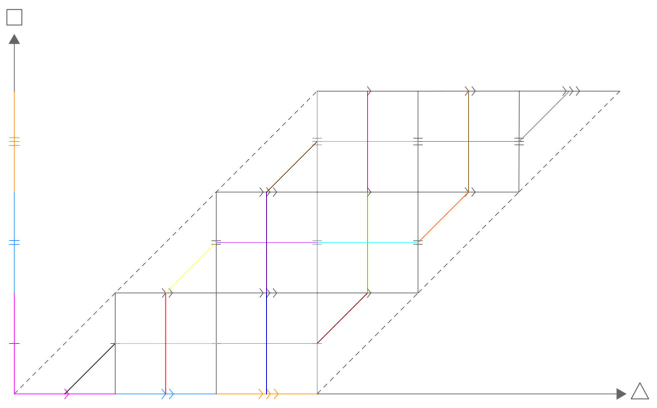
4.1. **The network N.** We construct a homotopy

$$H : X \times I \to X$$

that deforms the whole space X into a graph, which we call the network N. We denote $H(x,t) = H_t(x)$. We have that $H_0(X) = X$ and $H_1(X) = N$, where N is a graph.

The network $N \subset X$ consists of horizontal, vertical and diagonal segments with the identifications showed in figure 6.

Figure 6. The network N



We refer to the network parts in squares as *cross segments* and in triangles as *diagonal segments*.

The deformation of squares into cross segments is illustrated in figure 7 and the triangles into diagonal segments in figure 8.

Figure 7. Cross segments

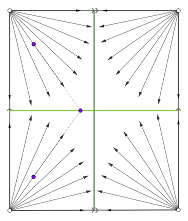
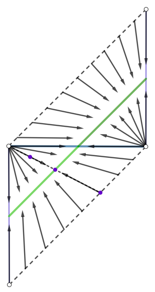
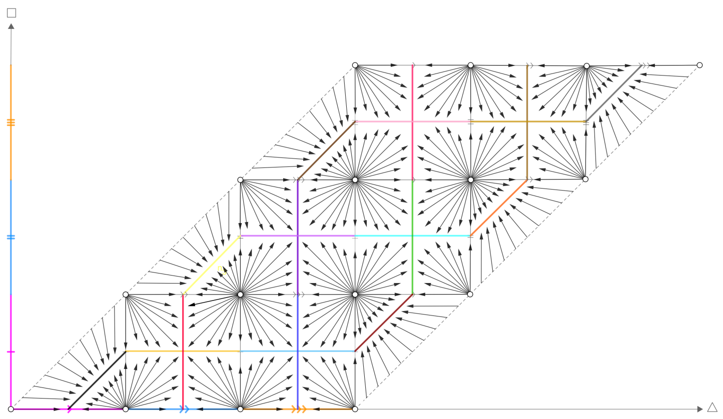


FIGURE 8. Diagonal segments



We show in figure 9 a diagram representing the traces $H_t(x)$ of the homotopy H for some $x \in X$. These traces will be useful to transit in and out of the network N in the configuration space X.

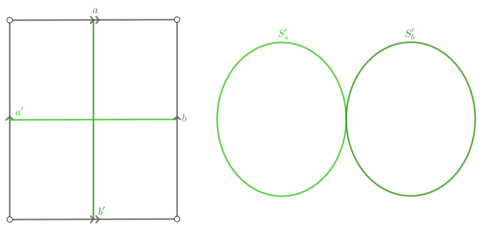
FIGURE 9. Homotopy traces in X



4.2. **The chain C.** Following the identification of sides in the flat representation of X, we can observe that the network N is homeomorphic to a chain of circles.

Observe that the points along the diagonal are removed from our product space, forming tori with a removed point. Each torus minus a point is deformed into the cross segments in the corresponding square, namely a figure eight. See figure 10.

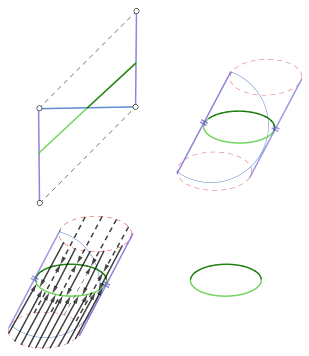
FIGURE 10. Torus minus a point into a figure eight



As we glue the two triangle parts along their identified edge, we obtain a parallelogram with two sides identified. Once that we glue these sides of the parallelogram together, we obtain a cylinder.

Each cylinder is deformed into the diagonal segments in the corresponding triangles, namely a circle. See figure 11.

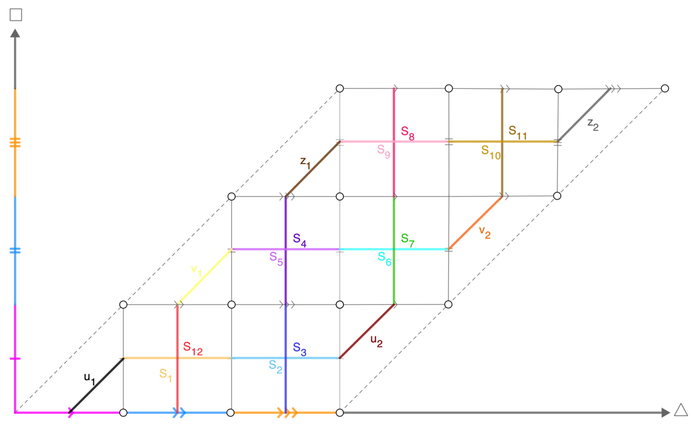
Figure 11. Cylinder into a circle



We obtained a figure eight from each of the six squares in X and a circle from each of the three parallelograms. In total, 12 circles + 3 circles = 15 circles. We will call this union of circles the chain C. We have that the network N is homeomorphic to the chain C.

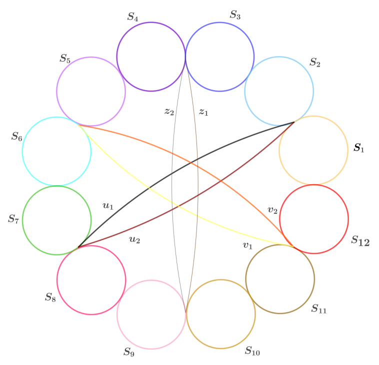
We use the following naming system for the cross and diagonal segments in N. The cross segments are $S_1, S_2, ...S_{12}$ and diagonal segments are $u_1, u_2, v_1, v_2, z_1, z_2$. Observe that the pairs of diagonal segments (u_1, u_2), (v_1, v_2) and (z_1, z_2) form a circle each. See figure 12.

Figure 12. Labelling segments in the network N



Then we draw the chain of 15 circles accordingly, as reflected in figure 13.

FIGURE 13. Chain of 15 circles C



We will refer to the outer circles $S_1, S_2, ...S_{12}$ in the chain C as *border circles* and inner circles as *connecting circles*. The outermost semicircles forming the border circles will be called *exterior semicircles* and the innermost ones will be the *interior semicircles*. In the construction of our motion planning algorithm, these exterior semicircles will have a distinguished role. The exterior semicircles in C and in N are shown in figures 14 and 15.

FIGURE 14. Exterior semicircles in the chain C

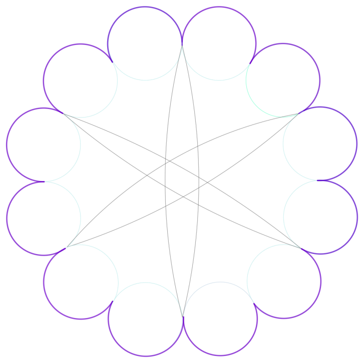
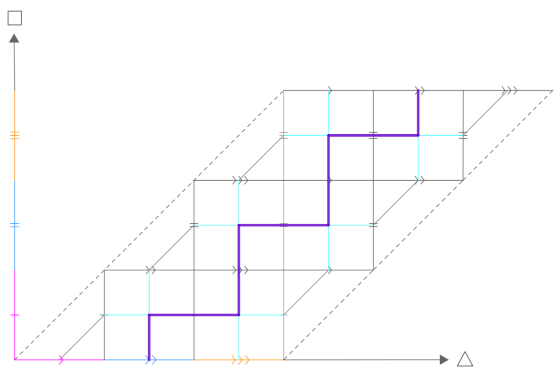
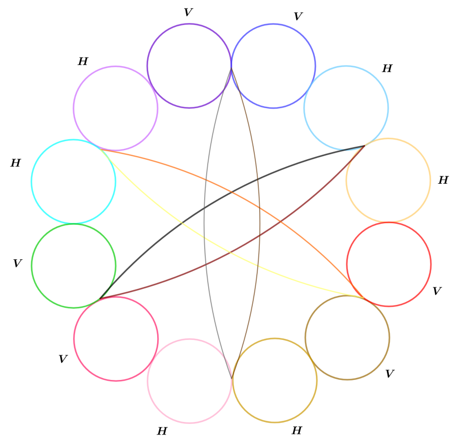


FIGURE 15. Exterior semicircles in the network N



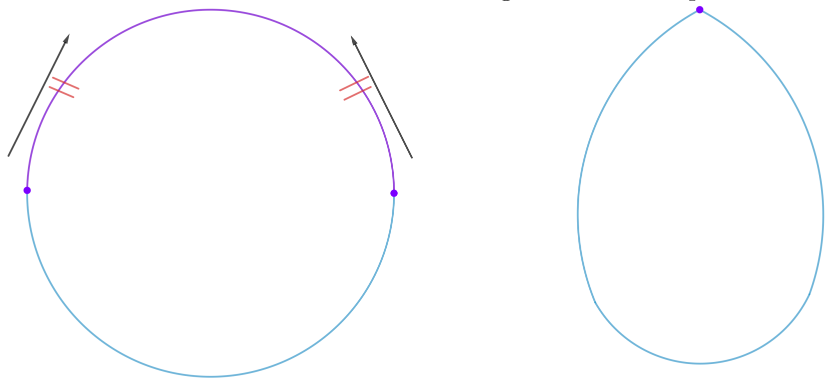
The chain circles in C that correspond to vertical cross segments in the network N are called *vertical circles*, and the circles that correspond to horizontal cross segments in N are *horizontal circles*. Vertical circles correspond to positions in the physical space at which robot A is at a pole (see definition 6.1) while robot B is at any place other than a pole in a different circle. Similarly, horizontal circles are the positions in Γ at which robot B is at a pole while robot A is at a position other than a pole in a different circle. Vertical and horizontal circles in C are shown in figure 16.

FIGURE 16. Horizontal and vertical circles in C



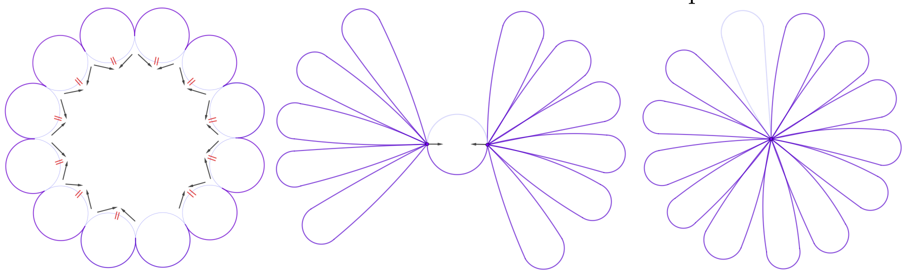
4.3. **The bouquet of circles B.** Now, we will find a homotopy equivalent space to the chain C. We see that each circle in the chain has two distinguished points that connect it to other circles. When we compress an interior semicircle into these points through homotopy, the interior semicircle contracts to a point, and the exterior semicircle forms a teardrop shape, as shown in the figure 17.

FIGURE 17. A Circle deforming into a Teardrop



As we apply the operation to the entire chain C, all interior semicircles except one will collapse into a point. The process will follow as these 11 interior semicircles collapse one by one forming eleven teardrops until they reach the last circle. At the final step, the two distinguished points will collapse into one forming two more teardrops. Hence, the total is 13 teardrops. We can observe this occurring in the figure 18.

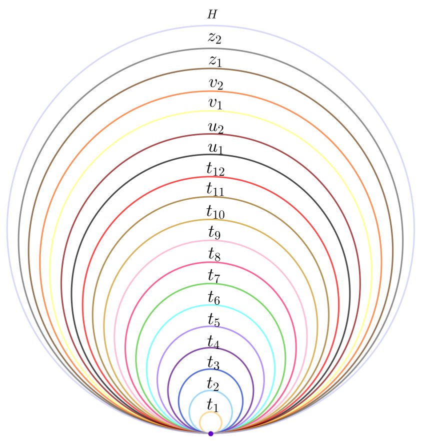
Figure 18. Border circles into 13 teardrops



As stated before, the connecting circles in the chain C consist of two different semicircles joined by their endpoints. When we identified these endpoints, each connecting circle forms two teardrops.

Therefore, we have 6 teardrops from connecting circles and 13 from border circles, forming a total of 19 teardrops joined at one point, namely, a bouquet of 19 circles, $B = \bigvee_{19} S^1$. See figure 19. Observe that the connecting semicircles in C lie on the outer rim of B.

Figure 19. Bouquet B of 19 circles



We have that the configuration space $X = C^2(\Gamma)$ is homotopic to a bouquet of 19 circles:

$$X \simeq N \cong C \simeq B$$

where $\simeq$ means homotopy equivalence and $\cong$ means homeomorphism.

5. Topological Complexity of X

In order to find the number of instructions for our motion planning algorithm, we consider the topological complexity of the configuration space, $TC(X)$.

Farber calculated the topological complexity of a graph G based on its first Betti number. The first Betti number of a graph G with n vertices, m edges and k connected components equals $b_1(G) = m - n + k$.

Theorem 5.1. [5] Let G be a connected graph, then

$$TC(G) = \begin{cases} 1 & \text{if } b_1(G) = 0 \\ 2 & \text{if } b_1(G) = 1 \\ 3 & \text{if } b_1(G) \geq 2 \end{cases} \tag{1}$$

Here $b_1(G)$ denotes the first Betti number of G.

We have shown that the configuration space X is homotopy equivalent to N. Then we showed that the network N is homeomorphic to a chain of circles C, which is homotopic to a wedge of 19 circles B. We know that the topological complexity of X will be equal to that of B by theorem 3.1. Since the first Betti number of a bouquet of nineteen circles is $b_1 = 19 \geq 2$, we have that $TC(B) = 3$ and therefore, $TC(X) = 3$.

Theorem 5.2. The topological complexity of the configuration space of two robots moving on a wedge of three circles, is three.

We conclude that any motion planning algorithm will require at least three continuous instructions in this scenario.

6. Motion Planning Algorithm

Our goal is to give an explicit description of three continuous instructions in the physical space where the robot motion takes place.

We will start by building the algorithm in the configuration space and then translate these instructions to the physical space by using all the constructions we have shown previously. Recall that any position in the physical space Γ corresponds to a state in the configuration space X. Our first step will be to move any state in X to the corresponding state in the network N following the traces of the homotopy shown in section 4.1.

The following definitions of specific positions and states in Γ and X will help to describe our algorithm.

Definition 6.1. The *vertex* is the center point of the physical space at which the three circles intersect and the *poles* are the antipodal points to the vertex in each circle in the physical space Γ.

The states in the network N correspond to 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.

Definition 6.2. The *nodes* in the chain C are the intersection points of the circles. The nodes where three circles intersect will be called *j-points*.

Definition 6.3. When a state is located in the interior of a cross segment in X, we will say it is a *cross state*. In the physical space Γ, cross states translate as the position of at least one robot being at the pole and the other at any of the other two circles but not at the vertex. A cross state in the chain C will correspond to a point in a border circle that is not a j-point. See figures 20 and 21.

Figure 20. Cross states in N and C

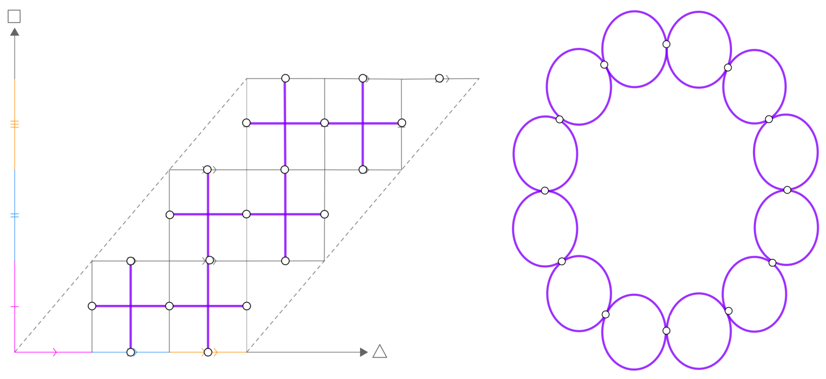
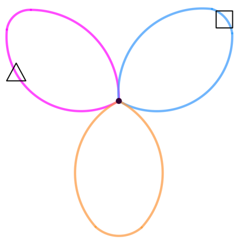


FIGURE 21. An example of cross state in Γ



Definition 6.4. A *cross center* is a cross state where two robots are at the intersection of horizontal and vertical network segments. A cross center corresponds to the positions in which both robots are at poles in different circles in Γ. We can observe this state in the chain C at the nodes that are not j-points. See figure 23.

FIGURE 22. Cross center states in N and C

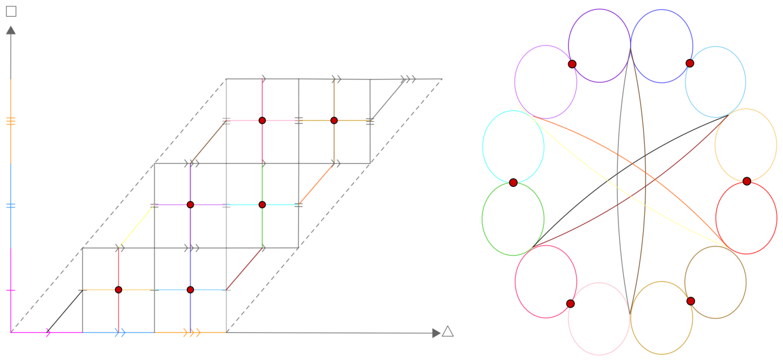
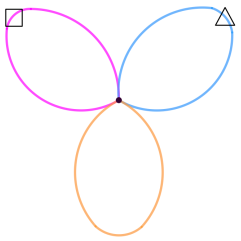


FIGURE 23. An example of cross center state in Γ



Definition 6.5. When the state is in the interior of a diagonal segment in the network N, we will say that the state is a *diagonal state*. In Γ, this state will correspond to antipodal positions in any circle, but neither is at the vertex. Diagonal states correspond to points in connecting circles with the exception of j-points. See figures 24 and 25.

FIGURE 24. Diagonal states in N and C

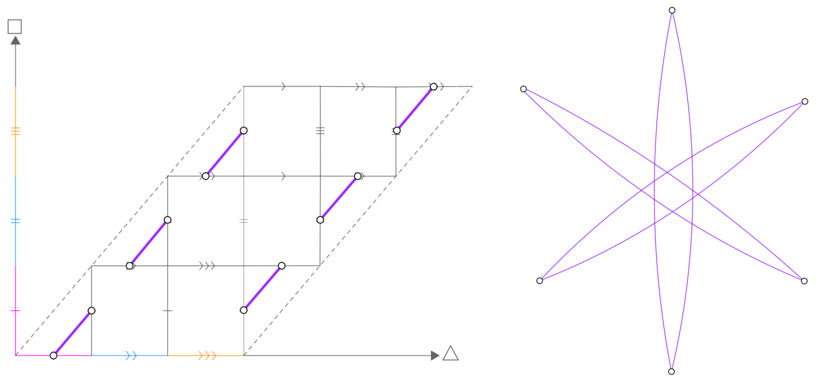
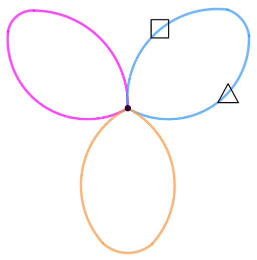


FIGURE 25. An example of diagonal state in Γ



We observe that j-states are states in N that are neither diagonal nor cross states. There are two types of j-states: *vertical* and *horizontal*. Vertical j-states are those located between a diagonal segment and a vertical cross segment, whereas horizontal j-states are found between a diagonal segment and a horizontal cross segment.

A j-state occurs when one robot in the physical space Γ is at a center while the other is at a pole. In a vertical j-state, robot B is at the vertex, whereas in a horizontal j-state, robot A is at the vertex while the other robot is at a pole. In the chain C, they correspond to the intersection points of border and connecting circles. Vertical j-point is between a connecting circle and two vertical circles, and a horizontal j-point is between a connecting circle and two horizontal circles. See figure 26 and 27.

FIGURE 26. j-states in N and Γ

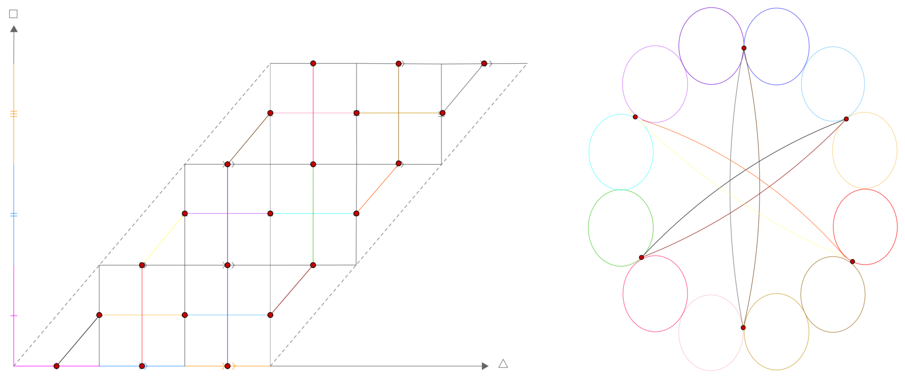
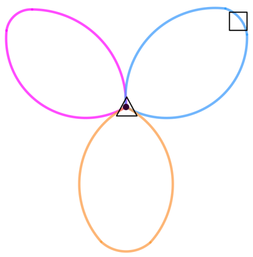


FIGURE 27. An example of j-state in Γ



In the network N the counterclockwise orientation is given by the orientation in X: along the x-axis from left to right and along the y-axis upwards. This orientation in the physical space and the chain C is illustrated in figures 28 and 29.

FIGURE 28. Counterclockwise direction in N and C

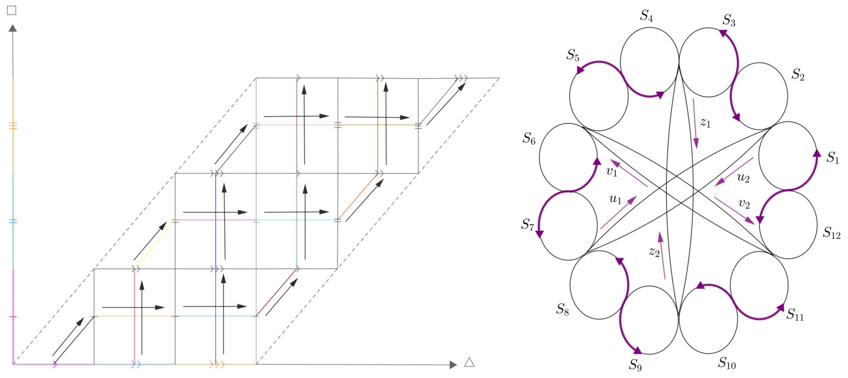
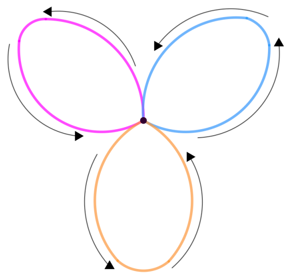


FIGURE 29. Counterclockwise direction in Γ



6.1. **Algorithm in the configuration space.** In this section, we will describe the instructions needed to move an initial state P_i in X to the final state P_f.

Recall from section 4 that the first step of the algorithm consists of projecting the initial and final states P_i and P_f into the network N following the traces of the homotopy $H : X \times I \to X$ as in figure 9. We will denote the projections $P_i^* = H_1(P_i)$ and $P_f^* = H_1(P_f)$.

Next, we will construct the algorithm in the network N to move from P_i^* to P_f^*. The following definitions will help to describe the algorithm in the network N and chain C.

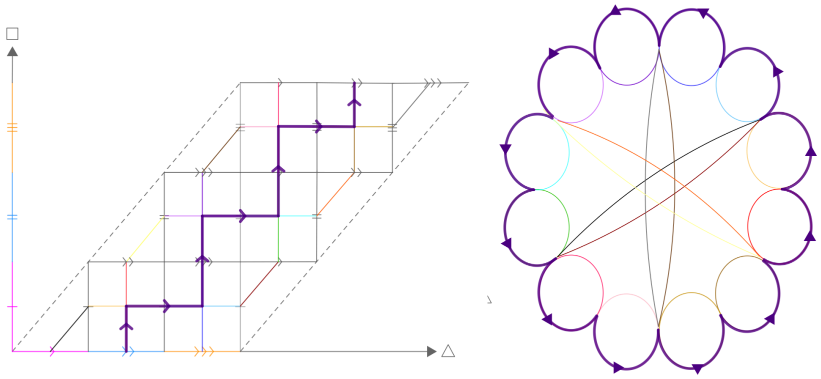
Definition 6.6. The *initial node* CN_i is the node in the chain C corresponding to the cross center or j-state in N at which the robots are initially located in N. If P_i^* is a cross state, then the initial chain node CN_i is the cross center of that cross state. If P_i^* is a diagonal state, then the initial chain node CN_i is the closest j-point in the counterclockwise direction. If P_i^* is a node itself, then $CN_i = P_i^*$

Similarly, the *final node* CN_f is the node in the chain C corresponding to the final state P_f^*.

Definition 6.7. The *zigzag circle* is the union of the exterior semicircles in the chain C. See figure 14 and 15.

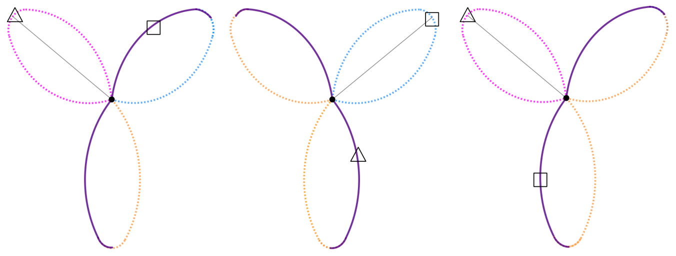
In figure 30 we show the zigzag circle in both network N and chain C with the counterclockwise orientation.

FIGURE 30. Zigzag circle in N and C



Observe that the zigzag circle positions that are not nodes, correspond in Γ to configurations in which the robot that is not at a pole or vertex is positioned at one of the two innermost semicircles opposite to the robot at the pole. We show sample positions of robots in the zigzag circle in figure 31.

FIGURE 31. Example of zigzag circle positions in Γ



Once the states are nodes, the idea of our algorithm will be to move robots from a node to another following the zigzag circle in the counterclockwise direction.

First we move the initial and final states P_i^* and P_f^* in N to initial and final nodes CN_i and CN_f following the shortest path or counterclockwise shortest path as described before.

As a result of this step, the states are nodes in the network N.

Next, we move counterclockwise in the zigzag circle from CN_i to CN_f, which is the fundamental step of our algorithm.

Finally, we reverse the motion from CN_f to P_f^*.

Algorithm 6.8. Therefore, the algorithm in the network N is as follows.

(1) **First Step** If P_i^* is a cross state, move by a shortest path to the initial chain node CN_i. If it is a diagonal state, move counterclockwise to CN_i. If P_i^* is a node, stay motionless. Do the same with the final state to obtain CN_f.
(2) **Second Step** Move from CN_i to CN_f following the zigzag circle counterclockwise. If $CN_i = CN_f$, stay motionless.
(3) **Third Step** Reverse the movement in first step to obtain P_f^*.

Algorithm 6.9. When transitioning to the chain C, we obtain the following motion planning algorithm:

(1) **First Step** If P_i^* is on a border circle, move shortest path to initial chain node CN_i. Otherwise, move counterclockwise to CN_i. Do the same with the final state to obtain CN_f.
(2) **Second Step** Move the robots from CN_i to CN_f counterclockwise through exterior circles in the chain C. If $CN_i = CN_f$, stay motionless.
(3) **Third Step** Reverse the movement in first step to move back from CN_f to P_f^* .

We can extend the range of the algorithm in the network and chain to all sets of points in the configuration space X by adding at the beginning and end, the steps given by the traces of the homotopy.

Algorithm 6.10. Hence, the algorithm in the configuration space X is:

(1) **Preliminary Step** Move from P_i to P_i^* following the traces of the homotopy. Do the same to move from P_f to P_f^*.
(2) **Main Step** Move from P_i^* to P_f^* following the above algorithm 6.8 described in N.
(3) **Final Step** Reverse the movement in preliminary step to move from P_f^* to P_f.

6.2. **Algorithm in the physical space.** We deduce now the instructions in the physical space Γ. The goal will be first to move the robots such that they will be in a network position, then to translate the algorithm in N to the physical space.

The movement determined by the traces of the homotopy is translated to the physical space as follows: if the robots are in the same circle, move them away from each other until they reach an antipodal position and if they are in different circles, move them until one reaches a pole position. In both cases ratio of the robots' speeds are given by the slopes of the homotopy traces shown in figure 9.

Once the initial and final positions are retracted to the network, we describe the movement within the network. The first step is to move any point in the network to a node. Then the movement will proceed between nodes.

The fundamental observation here is that the counterclockwise movement following the zigzag circle in the chain C restricts the movement in the physical space as shown in figure 31.

Recall that states in the network can be described as 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. If the robots are not already at a node, move the robot that is not at a pole to the pole following the shortest path if they are at different circles and move the robots simultaneously counterclockwise until one arrives at the vertex and the other at a pole if they are in the same circle.

Definition 6.11. We will say that the two robots are in a *pole-pole* position in Γ if they are both at poles. Observe that these positions correspond to cross center states in the network N.

Definition 6.12. We will say that the two robots are in a *pole-vertex* position in Γ if one is at a pole and another is at the vertex. Observe that these positions correspond to j-states in the network N.

Therefore, all nodes in the network can be described in the physical space as pole-pole or pole-vertex positions.

Now we will describe the movement in the physical space between these positions following the counterclockwise direction in the zigzag circle. We will say that are *allowed semicircles* the two innermost semicircles opposite to the stationary robot as shown in figure 31.

The following movements m_{VP} and m_{PV} describe the way to exit a pole-vertex or a pole-pole position.

m_{VP} From pole-vertex to contiguous pole-pole: Move the robot that is at the vertex counterclockwise in the allowed semicircles until it reaches the pole.

m_{PV} From pole-pole to contiguous pole-vertex: Move the only robot that can move counterclockwise in the allowed semicircles until it reaches the vertex.

Definition 6.13. A *zigzag movement from a pole-vertex to a pole-pole position* is a series of concatenated movements $m_{VP}m_{PV}m_{VP}\cdots m_{VP}$ as before where the first movement originates in the given pole-vertex position and the last movement finishes at the given pole-pole position. Similarly, a *zigzag movement from a pole-pole to a pole-vertex position* is a series of concatenated movements $m_{PV}m_{VP}\cdots m_{PV}$.

Algorithm 6.14. The algorithm for network positions is as follows.

(1) **First Step** If the two robots are at a pole-pole or pole-vertex position, stay motionless. Otherwise, if the robots are at different circles, move the robot that is not at a pole to the pole following the shortest path. If they are in the same circle, move the robots simultaneously counterclockwise until one arrives at the vertex and the other at a pole. Do the same for the final positions. Let us call *initial pole position* and *final pole position* the output of this step.
(2) **Second Step** Move from the initial pole position to the final one following the zigzag movement.
(3) **Third Step** Reverse the movement in the first step to go from the final pole position to the final position in the network.

Algorithm 6.15. To extend the previous algorithm to all the possible positions in the physical space, we add the preliminary and final steps given by the traces of the homotopy.

(1) **Preliminary Step** If the initial position of the robots are in the same circle, move them away from each other until they reach an antipodal position and if they are in different circles, move them until one reaches a pole position. Do the same with the final position. Let us call *initial network position* and *final network position* the output of this step
(2) **Main Step** Move from the initial to the final network position as in algorithm 6.14.
(3) **Final Step** Reverse the movement in the preliminary step to move from the final network position to the final position.

The above algorithm is discontinuous: if the initial or final configuration is a node, then a small perturbation of P may lead to a different motion. Note that if, for instance, we restrict the algorithm to the states P_i and P_f that are not nodes the algorithm is continuous here.

We previously proved that $TC(X)$ is 3. Our objective then will be to exhibit the three domains of continuity.

We will restrict the algorithm to the pairs in the following three regions U', V', and W' of the Cartesian product $X \times X$. The first region U' is the set of points P_i and P_f, such that neither of them is a node,

$$U' = \{(P_i, P_f)/P_i \notin CN, P_f \notin CN\}.$$

The second region V' consists of the pair of points, initial and final, such that one of them is a node and the other is not,

$$V' = \{(P_i, P_f)/(P_i \notin CN, P_f \in CN) \text{ or } (P_i \in CN, P_f \notin CN)\}.$$

The third region W' covers the rest of the Cartesian product,

$$W' = \{(P_i, P_f)/P_i \in CN, P_f \in CN\}$$

where both initial and final points are nodes.

Let U, V, and W be small open neighborhoods of U', V', and W' respectively such that U, V, and W deform into U', V', and W'. These are our domains of continuity.

Remark. Alternatively, we can also consider our sets U', V', and W' as the ENR's required in the original definition of tame motion planner by Farber [6] or as the sets required in the generalized version by Calcines [2] since we are dealing with ANR's.

6.2.1. *Sample case.* Now, we will show how our motion planning algorithm works in a sample case. Observe that the initial and final configuration does not have a corresponding motion in chain C.

FIGURE 32. Preliminary Step in X: $P_i \to P_i^*$, $P_f \to P_f^*$

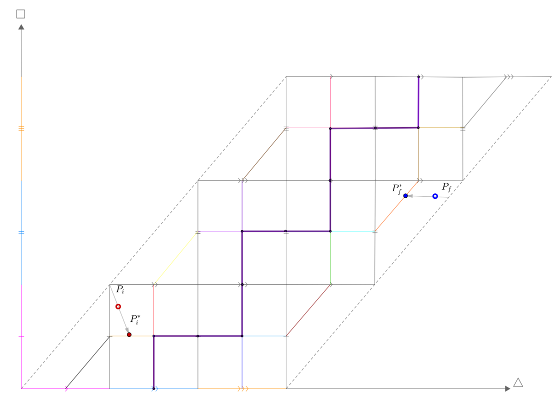


FIGURE 33. Preliminary Step in Γ

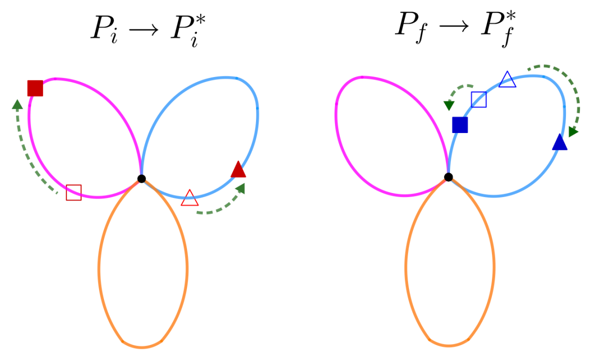


Figure 34. First Step in N and C: $P_i^* \to CN_i$, $P_f^* \to CN_f$

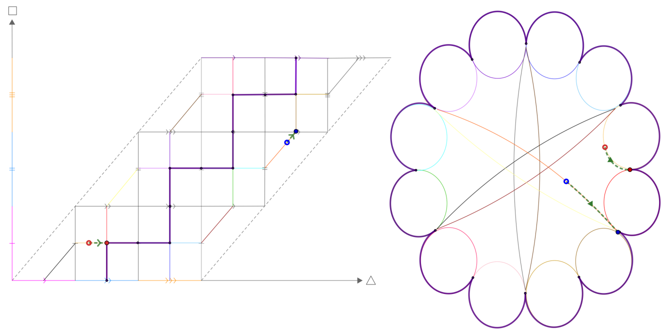


Figure 35. First Step in Γ

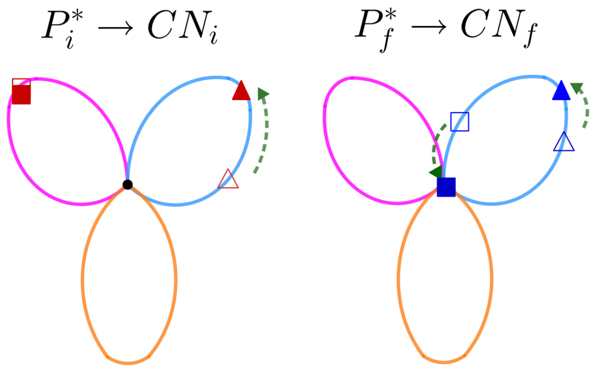
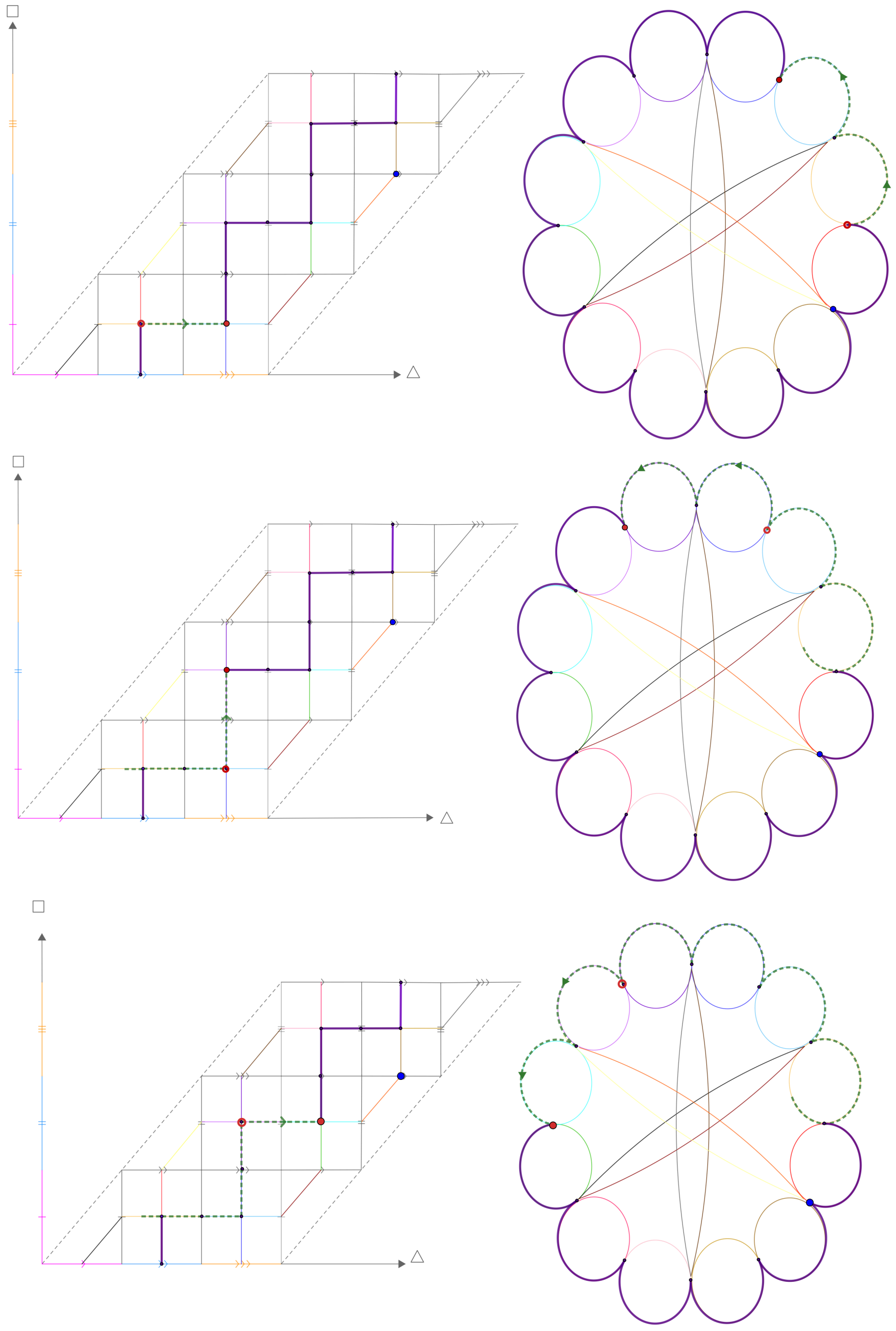
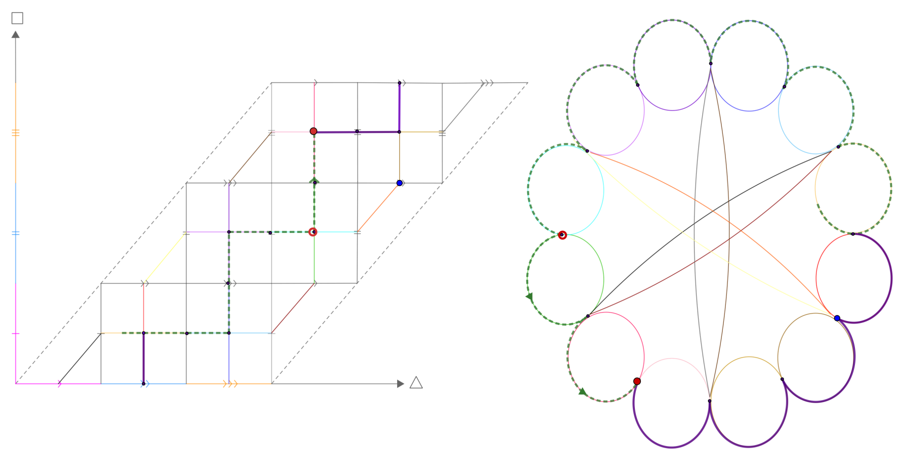
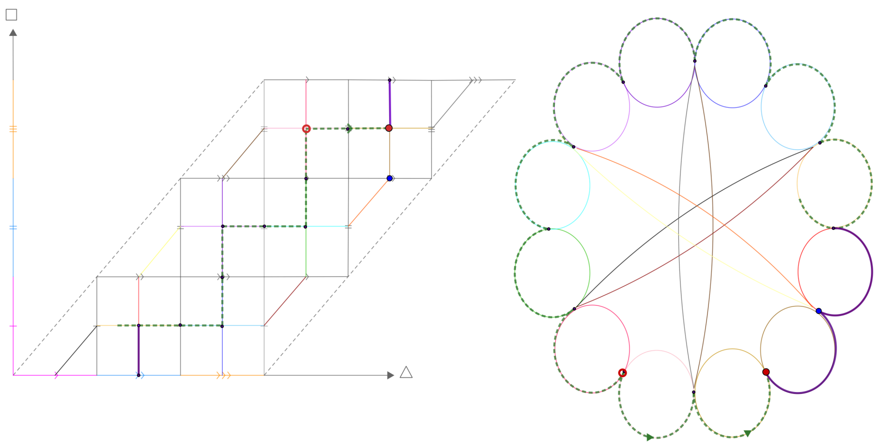


Figure 36. Second Step in N and C: Move from CN_i to CN_f







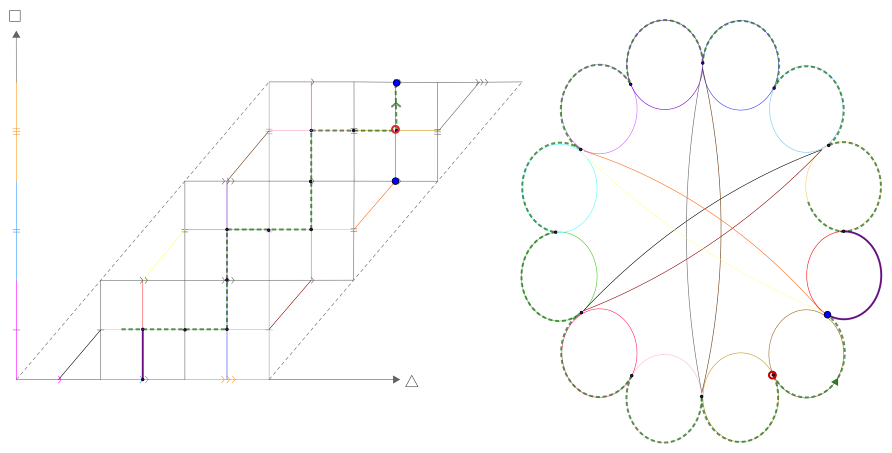


Figure 37. Second Step in Γ

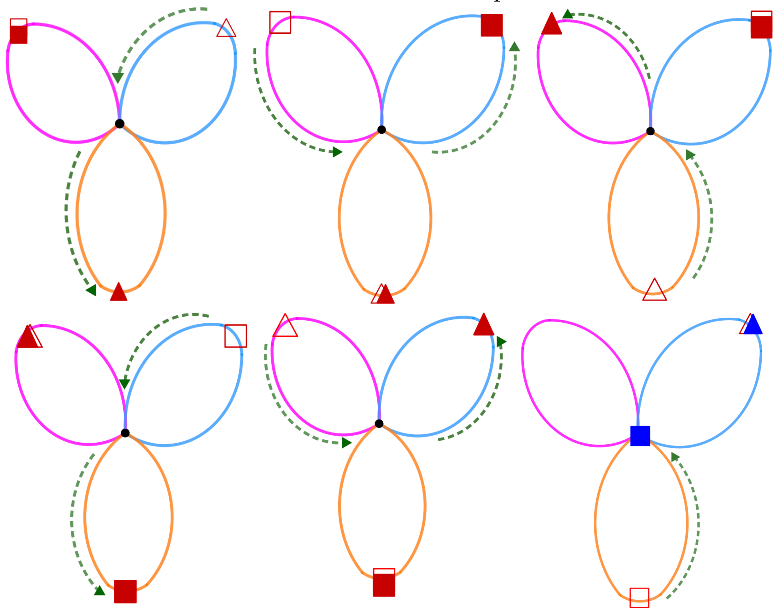


Figure 38. Third Step in N and C: $CN_f \to P_f^*$

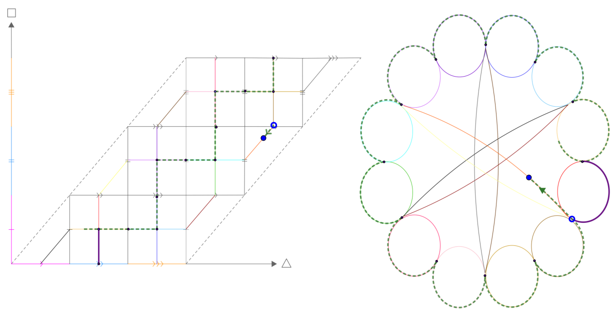


FIGURE 39. Third Step in Γ

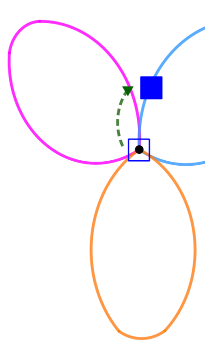


FIGURE 40. Final Step in N and C: $CN_f \to P_f^*$

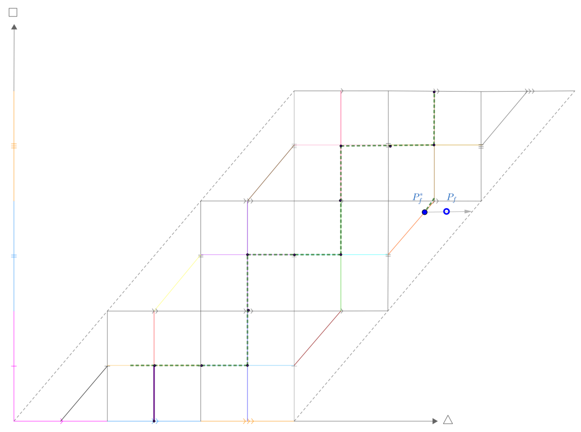
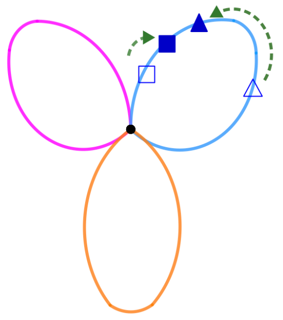


FIGURE 41. Final Step in Γ

$$P_f \longrightarrow P_f^*$$



References

[1] A. Boughrira, H. Colman *A Motion Planning Algorithm in a Lollipop Graph*, To appear: Minnesota Journal of Undergraduate Mathematics, 2020.

[2] J.M. Garca-Calcines *A note on covers defining relative and sectional categories*, Topology and its Applications, **265** (2019), 106810.

[3] M. Grant, G. Lupton and L. Vandembroucq *Topological Complexity and Related Topics*, Contemp. Math., Amer. Math. Soc., vol. 702, Providence, RI, 2018.

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

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

[6] M. Farber, *Topology of robot motion planning*, Morse theoretic methods in nonlinear analysis and in symplectic topology, NATO Sci. Ser. II Math. Phys. Chem., **217** (2006), 185–230.

[7] R. Ghrist, *Configuration spaces and braid groups on graphs in robotics*, in: *Knots, braids, and mapping class groups–papers dedicated to Joan S. Birman (New York, 1998)*, pp. 29–40, AMS/IP Stud. Adv. Math., vol. 24, Amer. Math. Soc., Providence, 2001.