Rose-Hulman Undergraduate Mathematics Journal

Volume 24
Issue 1

Article 6

Motion Planning Algorithm in a Y-Graph

David Baldi
Wright College, City Colleges, Chicago, davidbaldi@gmail.com

Motion Planning Algorithm in a Y-Graph

Cover Page Footnote
The author wishes to thank Dr. Hellen Colman for her sponsorship.

ℝose-Hulman Undergraduate Mathematics Journal
Volume 24, Issue 1, 2023

Collision-Free Motion Planning Algorithm in a Y-Graph

By *David Baldi*

Abstract. We present an explicit algorithm for two robots to move autonomously and without collisions on a track shaped like the letter Y. Configuration spaces are of practical relevance in designing safe control schemes for automated guided vehicles. The topological complexity of a configuration space is the minimal number of continuous instructions required to move robots between any initial configuration to any final one without collisions. Using techniques from topological robotics, we calculate the topological complexity of two robots moving on a Y-track and exhibit an optimal algorithm realizing this exact number of instructions given by the topological complexity.

1 Introduction

The AGV, or automated guided vehicle, has recently exploded in popularity and utility, with uses ranging from moving parcels to their destinations along factory floors to transporting people across busy cities to providing enjoyment to amusement park riders. In a nutshell, AGVs are mobile robots that operate without an onboard driver and which move along a specified path using lasers, radio waves, or magnetic tape, among other means of navigation.

Not all AGVs are created equal. Some play irreplaceable roles in their respective industries, while some are somehow less necessary. Vancouver, Canada, hosts the SkyTrain, the world's second-largest automated guideway transit (AGT) systems. It runs both underground and on elevated rails and serves about a half million people daily. In a similar but more restricted fashion, O'Hare Airport in Chicago, Illinois, employs automated people movers (APMs), shuttling individuals between different terminals. APMs are a subtype of AGVs that function as low-density modes of transit, within an airport, as mentioned, or within a rail-free urban setting. One such example is a driverless taxi system operating in downtown Las Vegas, Nevada. Amusement parks make up a significantly smaller subsection of the AGV industry but a subsection nonetheless, as is the case with the Antarctica: Empire of the Penguin ride in Orlando, Florida, the 2018 REVO-GT from Oceaneering International, Inc., and the 2019 Dark Ride from Simworx. Finally, among the applications of AGVs that do not transport humans, one finds the Roomba automated vacuum for residential use as well as the vastly more complex

Mathematics Subject Classification. 55R80
Keywords. configuration spaces, algebraic topology, motion planning

Amazon warehouse *Pegasus* robots. Pegasus drive units are two-foot-tall machines that meet employees at specified locations to receive a package before moving along a guided path, carrying the package, and dispensing it to the appropriate conveyor belt. Other mobile robots employed by Amazon [8] include the Xanthus unit and Kiva unit, the latter of which positions itself underneath a tall stack of storage bins and moves it along a warehouse floor. Between 2014 and 2020 Amazon increased its total use of mobile robots from 14,000 units to 200,000 units and as of 2022, Amazon employs over 500,000 robotic drive units [9]. The rate of implementation of AGVs has increased rapidly and seems to continue to increase. AGVs are not going anywhere.

Our algorithm exhibits the minimal number of instructions to move two robots on a track shaped as a letter Y. These robots may be carts moving in corridors that can accommodate one cart at a time, or vehicles moving on a network of tracks that join three separate locations to a central one.

We use techniques from topological robotics to find the minimal number of instructions. We present a new concrete algorithm for the case of two robots on a Y-track. This algorithm is optimal in the sense that the motion planning is performed with the minimal number of instabilities given by the topological complexity.

We first build the configuration space associated with the problem of moving two robots on a Y-track. This is the space where we will construct our algorithm. Configuration spaces in mathematics were introduced in the sixties by Fadell and Neuwirth [4] and first used in robotics in the nineties [11]. More recently, configuration spaces of robots moving on graphs have been studied by Farber [5] and Ghrist [6], amongst others. In particular, the study of algorithms in a graph shaped as the letter Y was initiated by Ghrist and Koditschek [7] and pursued further by Lütgehetmann [12], Abrams [1], Salgado and Velazquez [13] and others.

The configuration space of two robots moving on a Y-track is the space of all possible positions of the robots without collisions. The motion planning algorithm we built assigns a path between an initial and a final configuration in a continuous way. In most real world situations these continuous global algorithms do not exist. Farber proved that only situations with contractible configuration spaces will admit a global algorithm and he introduced the notion of *topological complexity* of configuration spaces to measure the discontinuities for all other cases.

Farber also proved that the topological complexity of a space is invariant under homotopy. This is the key result that will allow us to construct our algorithm not in the original configuration space, but in a simpler, homotopically-equivalent space.

We calculate the topological complexity and deduce the instructions in this simpler space. Next, we use the homotopy deformation to describe the instructions in the whole configuration space. Lastly, we translate our instructions back to the physical space where the robots actually move.

The organization of the paper is as follows. In Section 2 we introduce some basic topological notions. Section 3 introduces the configuration spaces. Section 4 explains the idea of continuous motion planning. Section 5 introduces the topological complexity and shows the concrete homotopy deformation of our configuration space. In section 6 we give a detailed construction of our algorithm in the different parts of the configuration space. The final section compiles the previous instructions and presents the complete algorithm in both the configuration and the physical space.

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

2 Preliminaries

We will briefly introduce a few basic topological concepts upon which we will then build our discussion of the *motion planning problem* and then the elucidation of our proposed *motion planning algorithm*.

2.1 Continuity

The concept of the continuity of a function is ubiquitous in mathematics and is of particular importance in the field of topology within which AGV motion planning problems are studied. In a continuous function, a small variation in the argument induces a small variation of the value of the function. We may draw on a modern street navigation system to illustrate the point: a navigation function, such as that of Google Maps, taking as inputs the starting address and desired destination, gives as the output the path between the inputs. The function is continuous if a slight change in the starting or desired address does not result in a drastically different path output. If one were traveling to New York City from Chicago, for example, one would expect the path output to be similar regardless of whether one were traveling from north Chicago or south Chicago (which would be a very small distance relative to the distance to New York City).

2.2 Homeomorphism and Homotopy

A *homeomorphism* is a bijective, continuous function whose inverse function is also continuous. If two topological spaces X and Y admit a homeomorphism between them, we say that they are homeomorphic and we write $X \approx Y$. We illustrate the point with letters of the English alphabet. For instance, the letters O and D are homeomorphic to one another as shown in figure 1. Similarly, T and Y are homeomorphic to one another but neither of them to O or D. See figure 2.

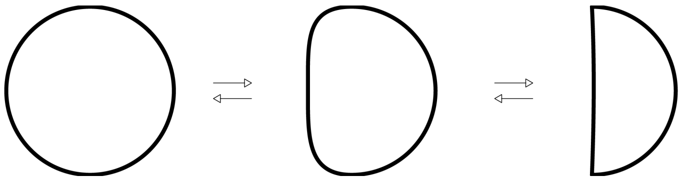

Figure 1: O-to-D homeomorphism.

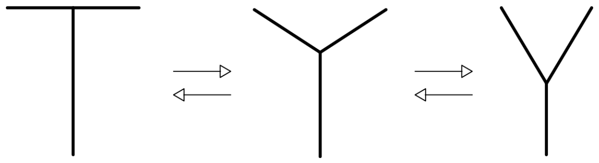

Figure 2: T-to-Y homeomorphism.

We may formally define a *homotopy* as follows: Given two continuous maps f and g from X to Y, a homotopy from f to g is a map

$$\mathrm{H} : \mathrm{X} \times \mathrm{I} \longrightarrow \mathrm{Y}$$

such that

$$\mathrm{H}(x, 0) = f(x)$$

and

$$\mathrm{H}(x, 1) = g(x).$$

A homotopy equivalence between X and Y is a pair of continuous maps $f : \mathrm{X} \to \mathrm{Y}$ and $g : \mathrm{Y} \to \mathrm{X}$, such that gf is homotopic to the identity map id_X and fg is homotopic to the identity map id_Y. In this case, X and Y are said to be homotopy equivalent, or of the same homotopy type. We write $\mathrm{X} \simeq \mathrm{Y}$.

All homeomorphic spaces are spaces with the same type of homotopy, but the converse does not hold. A space can be deformed via a homotopy into a space of a different dimension. On the other hand, dimension in a homeomorphism is an invariant, thus those deformations permissible with homotopic spaces are prohibited with homeomorphisms. An interval I and any subinterval I_s are homeomorphic and of dimension 1. However, if one were to collapse one of the spaces, say I_s, to a point, the dimension of I_s would fall to zero and I_s would thus no longer be homeomorphic to I. An interval I is homeomorphic to another interval J, but neither are homeomorphic to a point P. However, I, J, and P have the same type of homotopy.

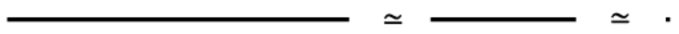

Figure 3: Interval-point homotopy.

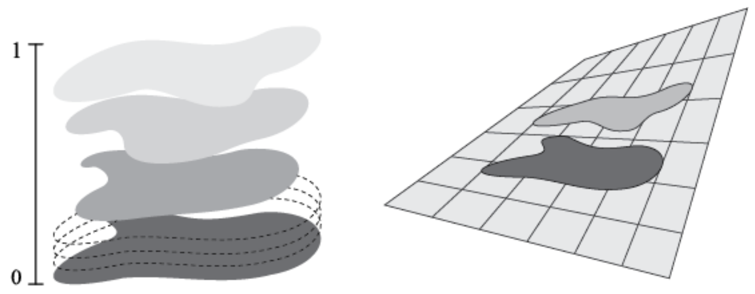

Figure 4: A homotopy from f to g.

The letter D has the same homotopy type as the letter A but they are not homeomorphic.

2.3 Cartesian Product and Path Space

We will define the *Cartesian product* $A \times B$ of two sets A and B as the set of ordered pairs (a, b) where $a \in A$ and $b \in B$.

A *path* α in a topological space X is a function given by

$$\alpha : I \longrightarrow X.$$

A path α from a to b is a path $\alpha : I \longrightarrow X$ such that

$$\alpha(0) = a, \ \alpha(1) = b.$$

We refer to the path as having an initial point $\alpha(0)$ and final point $\alpha(1)$. The topological space of such paths, or *path space*, is the set of all paths and is denoted

$$PX = \{\alpha | \alpha : I \to X\}.$$

A space X is said to be *path-connected* if for all $x, y \in X$ there exists a path from x to y. Finally, it will help to think of the image of a path α as a curve on the space X as in figure 5.

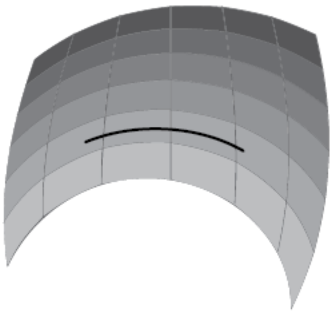

Figure 5: A path on a space X.

2.4 Section of the Evaluation Function

Given what we have seen regarding the Cartesian product and path space, we are now in a position to discuss the *evaluation function.* The domain of the evaluation function is the path space PX and its codomain is the Cartesian product of X with itself. We formally denote the evaluation function as

$$ev : \mathrm{PX} \to \mathrm{X} \times \mathrm{X},$$

that is, the evaluation function takes as input a path $\alpha \in \mathrm{PX}$ and associates with it an ordered pair $(a, b) \in \mathrm{X} \times \mathrm{X}$ corresponding to the initial and final points if the path. A *section* $s : \mathrm{X} \times \mathrm{X} \to \mathrm{PX}$ of the evaluation function satisfies $s(a, b) = \alpha$ with $\alpha(0) = a$ and $\alpha(1) = b$. In other words,

$$\begin{tikzcd} \mathrm{PX} \arrow[d, "ev"] \\ \mathrm{X} \times \mathrm{X} \arrow[u, bend left, "s"] \end{tikzcd}$$

such that $ev \circ s = \mathrm{id}$.

3 Configuration Space

The *configuration space* $\mathrm{X} = \mathrm{C}^n(\Gamma)$ of a topological space Γ is a mathematical abstraction that models the states corresponding to allowed positions of n robots moving in a physical space Γ.

The configuration space represents all possible states corresponding to the positions of n robots in the physical space. One such robot can move freely within some topological space, such as on a track or in a plane, if it is alone and there are no obstacles in the space. As we will see in more detail when we increase the number of robots in a system, in our case to two robots ($n = 2$), those two robots in the same physical space means that one robot's movement is constrained by the movement, or lack thereof, of the other robot. Two robots cannot occupy the same position simultaneously. The diagonal is the set of points in the product such that at least two robots coincide, and we formalize the notion of a diagonal as

$$\Delta = \{(x_1, x_2, \ldots, x_n) \in \Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_n | \; x_i = x_j \text{ for some } i \neq j\}.$$

Formally, the configuration space of Γ is

$$\mathrm{C}^n(\Gamma) = \Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_n - \Delta.$$

3.1 Configuration Spaces of an Interval

Figure 6 illustrates an example of one robot moving in a physical space $\Gamma = \mathrm{I} = [0,1]$. (Our convention going forward will be to use a hollow or solid figure to represent the initial or final position, respectively.) The configuration space of such a topological space is identical to the physical space since, because there is just one robot, its states are just its positions in I.

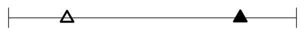

Figure 6: A single robot on a track I.

Figure 7, however, illustrates two robots on the same track. Each robot restricts the movement of the other. It is clear that both robots cannot occupy the same position simultaneously. This places restraints on the possible configurations of the robots.

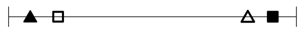

Figure 7: Two robots sharing a track.

The configuration space $\mathrm{C}^2(\mathrm{I})$ is a subset of the Cartesian product; it is equal to the product minus the diagonal, $\mathrm{I} \times \mathrm{I} - \Delta$. We may visualize it using a Cartesian plane as in figure 8. On this plane, the diagonal disconnects the Cartesian product and the configuration space is the disjoint union of two triangles.

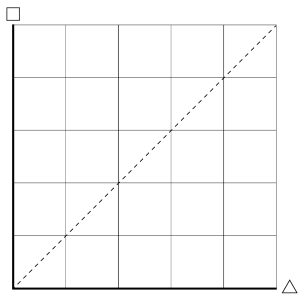

Figure 8: $C^2(I) = I \times I - \Delta$

The interval $I = [0, 1]$ as the x-axis will represent the positions of the *triangle robot* and the same interval as the y-axis will represent the *square robot*. We plot the triangle robot first and maintain this convention throughout this paper. Note the diagonal line, which we will discuss shortly.

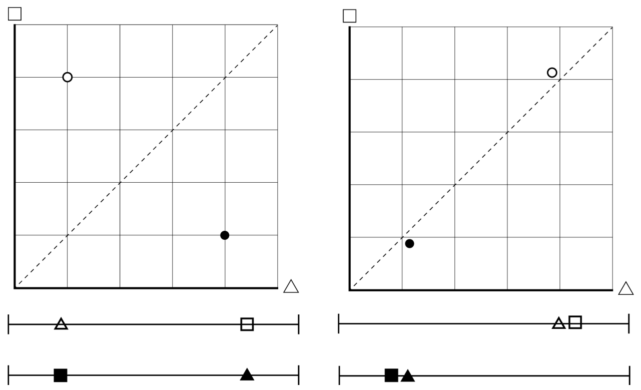

Figure 9: Four possible states in the configuration space $C^2(I)$.

In figure 9 we see represented four configuration states of the two-robot track when the robots are spread out (left) and nearer one another (right).

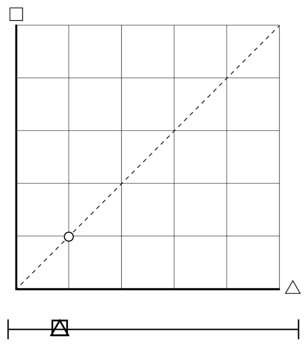
Figure 10: An illegal configuration.

In figure 10 we see represented an illegal configuration. The diagonal Δ corresponds to the segment of line $y = x$ and represents illegal configurations for our system of 2 robots–namely, a point representing a configuration will not reside on that line because the two robots cannot occupy the same position on the track. Considering these physical limitations, we see now why it is necessary to subtract Δ from the Cartesian product. Furthermore, because the configuration space in this case is a non-connected space, being disconnected by the diagonal, there can be no motion planning algorithm for the system of two robots that we have seen. This corresponds in the physical space to the fact that is impossible for robots to swap positions within the interval. See for instance initial and final positions in figure 9.

We will turn now to the physical space of a Y-graph and its configuration space with likewise two robots.

3.2 Configuration Space of a Y-Graph

We have seen that an increase in the number of robots results in a more complicated configuration space. We will see now that an increase in the complexity of a physical space corresponds also with an increase in the complexity of the configuration space. Our physical space now will be a graph on four vertices and three edges in the form of the letter Y.

Let us now elucidate the structure of a Y-graph with two robots on it.

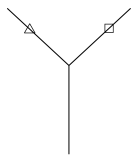
Figure 11: Two robots on a Y-graph.

Figure 11 depicts our two robots on a Y-graph. The product Y × Y of the Y-graph is obtained, as with the Cartesian product of the interval, by extrusion of the graph with itself.

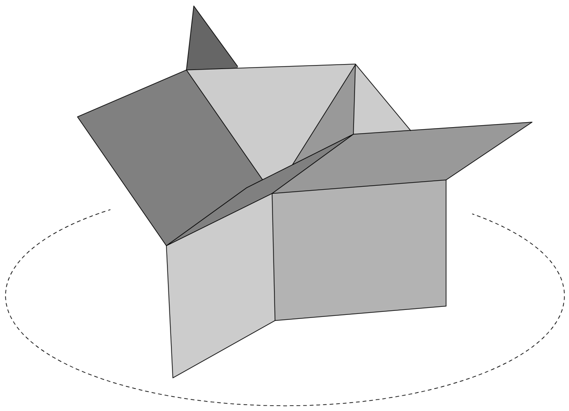

Figure 12: Immersion of the Cartesian product of the Y-graph in $\mathbb{R}^3$.

The Cartesian product $Y \times Y$ is a surface that can be immersed in $\mathbb{R}^3$. Figure 12 shows the immersion of our 2-dimensional space $Y \times Y$ in 3-dimensional space. This immersion is useful for visualizing many properties of the Cartesian space and exhibits self-intersections since $Y \times Y$ cannot be embedded in 3-dimensional space.

We now look at the space from different perspectives to give the reader a way to "see" the immersion. We begin by labeling figure 12 as below.

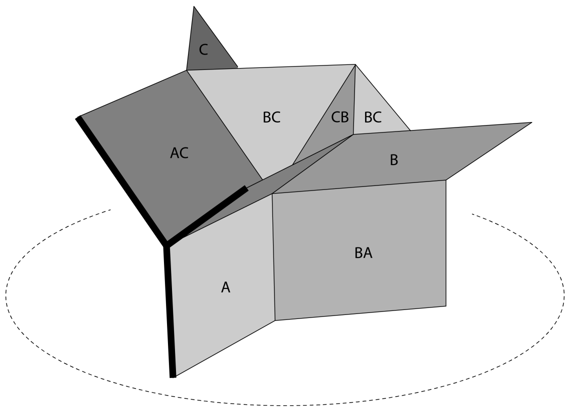

Figure 13: Immersion of the 2-dimensional configuration space in 3 dimensions.

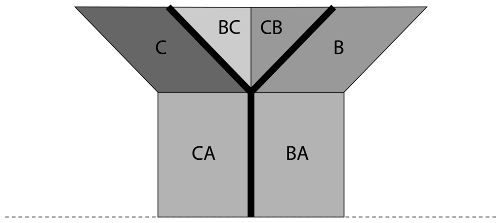

Figure 14: Frontal view of the 3-dimensional immersion of the space.

Figure 14 represents the space from a frontal perspective. Here, wing CA is visible. The reference circle on which the space rests is here depicted as a dashed line segment.

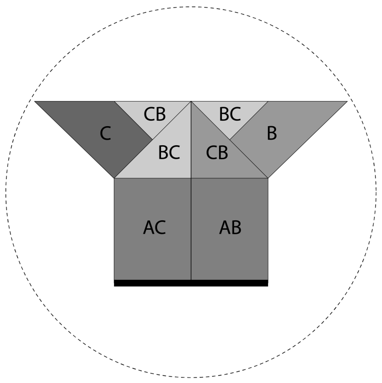

Figure 15: Top-down view of the 3-dimensional immersion of the space.

Figure 15 represents the space from a top-down perspective. The reference circle on which the space rests is here depicted unaltered. The thick-lined Y-shaped portion of the front of the space is depicted as a single solid line.

The figures that follow analyze the correspondence between states in the configuration space and robots' positions.

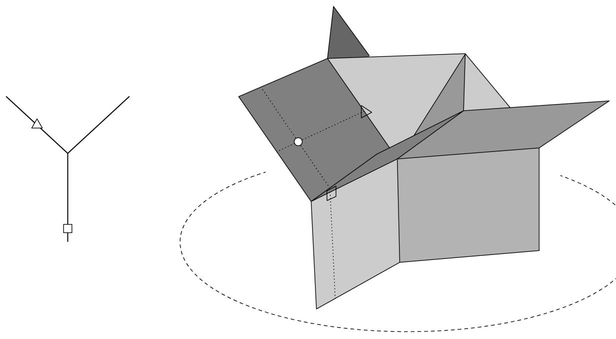

Figure 16: One possible set of positions on a Y-graph with their corresponding state on the Cartesian product.

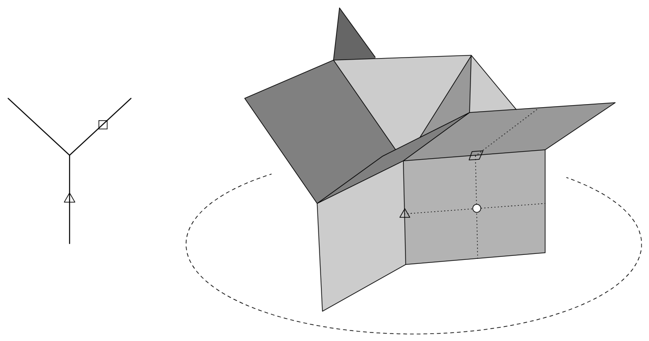

Figure 17: Another possible set of positions on a Y-graph with their corresponding state on the Cartesian product.

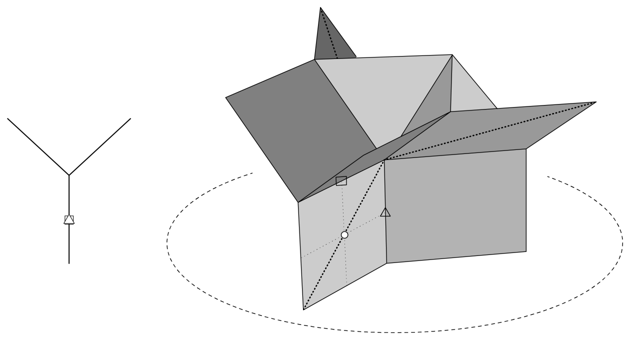

Figure 18: The Cartesian product Y × Y with an example of illegal state.

In Figure 18 we can see in the Y-graph on the left that the two robots are superimposed on the lower leg of the graph. This illegal position translates onto $Y \times Y$ into a point in the diagonal. We must eliminate these points from the Cartesian product to obtain the configuration space.

Let us continue, then, to remove the diagonal Δ from the represented space. The diagonal is the union of the diagonals of the squares A, B and C. Each of these squares is divided into two triangles. Once the diagonal is removed, the resulting space $Y \times Y - \Delta$ can be embedded in $\mathbb{R}^3$.

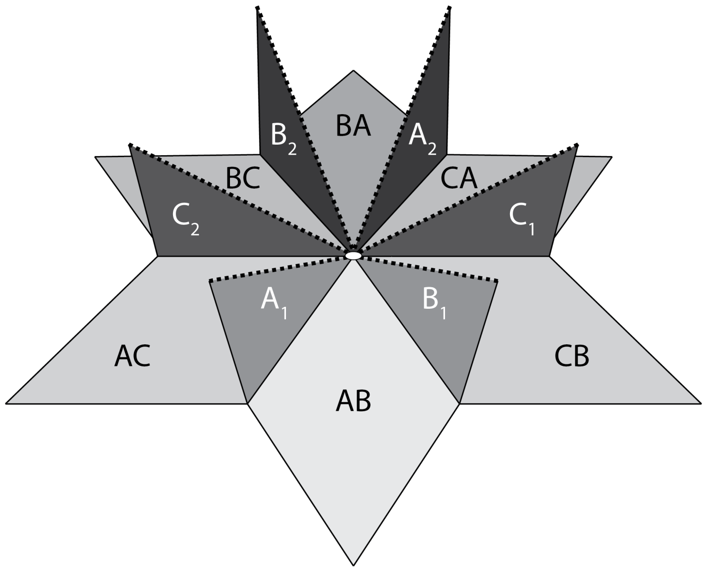

Figure 19: Graph of $Y \times Y$ with diagonals physically cut.

In figure 19 we see a simple space representing the configuration space of $Y \times Y - \Delta$. Indices denote those spaces now separated by the removal of the diagonals. The deleted central vertex is exaggerated for clarity. Let us call X this representation of the configuration space. We have

$$X \approx Y \times Y - \Delta.$$

4 Motion Planning

Let us look at the issue of navigating a topological space by a robot or robots (the motion planning problem) and the resolution of the problem (the motion planning algorithm).

4.1 Motion Planning Problem

The motion planning problem is the problem of finding for a robot a possible path between two points: given an initial and a final position, is there a path from the initial position to the final position? To this problem the solution is the motion planning algorithm.

4.2 Motion Planning Algorithm

A motion planning algorithm is, at its core and as alluded to with our navigation function example above, a function that, given as inputs an initial and a final destination, outputs a path between the points. What we want, then, is what is referred to as the *section*, denoted s, of the evaluation function that returns a path as just described. Our motion planning algorithm is the section itself of the evaluation function defined in section 2.4. Formally, given $(a,\ b) \in X \times X$, we want a path $\alpha \in P(X)$ such that $\alpha(0) = a$ and $\alpha(1) = b$.

While motion planning algorithms can be used to find the "most scenic route" on a road trip, they are most commonly used to locate the shortest or cheapest path between points. We are interested in the *existence* of a path chosen in a continuous manner.

4.2.1 One Robot On a Circular Track. Let us now look at an example of a motion planning algorithm executed on a circle S^1 and containing a single robot. The configuration space in this case is still just a circle. Figure 20 details this.

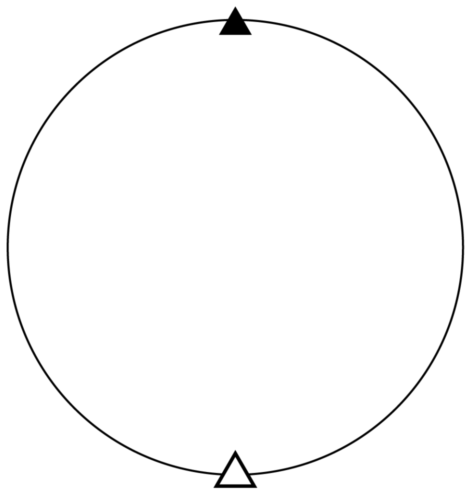

Figure 20: An initial position and final position on a circle in an antipodal arrangement.

Let us assume that the robot is at a position $x_{initial}$ some distance from its destination x_{final} where x_{final} is not antipodal to the initial position, i.e. the initial and final positions are not directly across from one another on the circle. Then a motion planning algorithm is straightforward. It would essentially tell the robot: "Determine the shortest path from $x_{initial}$ to x_{final}. Then, travel that path."

But what of the case in which the initial and final positions *are* antipodal? We would need to resolve this issue of ambiguity. For instance, we could pass an instruction to the robot to have it move counterclockwise in this case.

However, there is a problem: the function is not continuous. If we were to move the robot's initial position slightly in the counterclockwise direction on the circle, the output path would differ from the original path slightly. But if we were to move the initial position the same amount in the opposite direction the output path would change disproportionately to the small change in the initial position of the robot.

Thus there are two cases in which the initial and final positions of a robot on a circle track can be: either in an antipodal arrangement or not. We have an instruction for each case: If initial and final positions are antipodal, follow the counterclockwise path to the final position; otherwise, follow the shortest path. Each motion planning instruction is continuous but neither can serve as a continuous global motion planning algorithm. As we will see in the next section, it is impossible to find a continuous global motion planning algorithm for a circle.

5 Topological Complexity

Farber proved that most spaces do not have a continuous motion planning algorithm.

Theorem 5.1. [5] A globally-defined continuous motion planning algorithm in X exists if and only if X is contractible.

As Farber's theorem stipulates, and given that most configuration spaces are not contractible, it is rarely the case that one may find such a single-instruction, global algorithm. Farber further proposes a new invariant that measures the navigational complexity on a space. The *topological complexity* of a space X, TC(X), is the minimum number of continuous instructions needed to move a robot from any initial to any final position.

Definition 5.1. [5, 3] The *topological complexity* TC(X) of a connected space X is defined as the *minimum* integer k such that the Cartesian product $X \times X$ can be covered by k subsets $U_1, U_2, \dots, U_k$, such that for any $i = 1, 2, \dots, k$ there exists a *continuous* local section $s_i : U_i \to PX$ with $ev \circ s_i = i_U : U_i \hookrightarrow PX$.

The topological complexity for a space can be difficult to calculate. Current efforts focus on estimating, rather than calculating, topological complexity of certain spaces.

Applying the previous theorem, then, we can see that for a single robot in a non-contractible space, such as a circle, there can be no global motion planning algorithm.

Farber also proved that TC is an invariant of homotopy type, that is, if one space can be continuously deformed into another, then their topological complexity is the same.

Theorem 5.2. [5] If $X \simeq Y$ then $TC(X) = TC(Y)$.

5.1 Topological Complexity of a Robot Moving on a Circle

According to theorem 5.1, given that a circle is not contractible, the topological complexity of a circle, $TC(S^1)$, is greater than 1. And, as we have seen in the previous section, we have found 2 instructions to define a global motion planning algorithm for our two robots. Therefore, $1 < TC(S^1) \leq 2$ and so $TC(S^1) = 2$.

5.2 Homotopy type of the configuration space $X = C^2(Y)$

We turn now to a continuous deformation of the configuration space X of two robots moving on a track Y. Namely, we will homotopically simplify X. We will execute our motion planning algorithm on a homotopically equivalent, simpler space.

We discussed in the previous section the relationship between topological complexity and homotopy types, that is, that TC does not change between spaces that share the same homotopy type. We are now in a position to use theorem 5.2 to calculate TC in a simpler space and then infer from this the topological complexity of the space X.

The space X consists of a base composed of six quadrangles and six triangles attached to the base. We will refer to the edges where a triangle and two quadrangles meet as *rays*. The outermost vertices of X are *peaks* and the indented vertices between said peaks are *valleys*; there are six of each. Finally, the set of edges connecting peaks and valleys to one another, including the peaks and valleys themselves, will be referred to as the *boundary*. The interior of these edges will be called *intermediate* states.

We begin the deformation of the space X by collapsing vertically downward the six triangular faces into the rays. We then deform the base to the boundary by compressing the six quadrangular faces outward by “expanding" the deleted central vertex. This downward-outward deformation, taken to its extreme as we are doing here, results in a six-pointed star that is homeomorphic to a circle. In sum, all faces of X are continuously deformed to a circle. The boundary remains fixed during the process.

The paths determined by the deformation are called the *traces of the homotopy*.

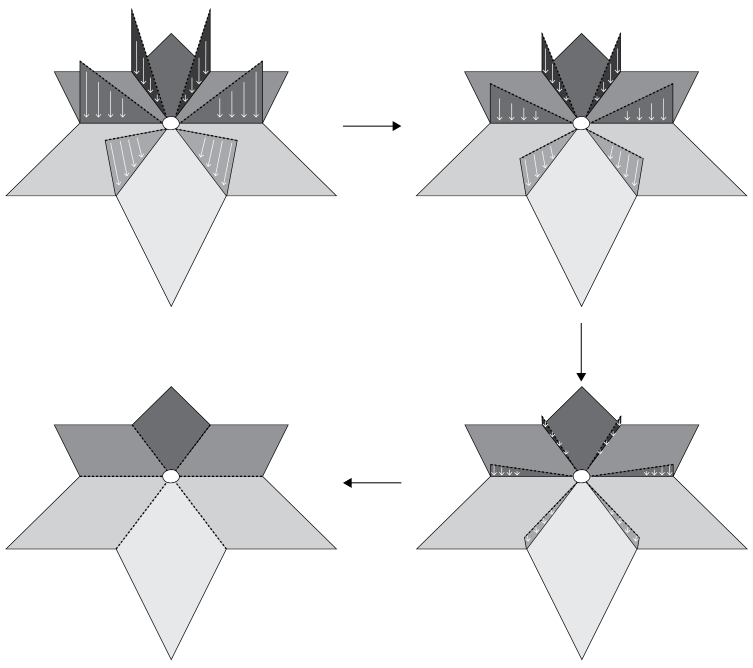

Figure 21: Stages of deformation of the triangles into the rays. Arrows indicate traces of the homotopy.

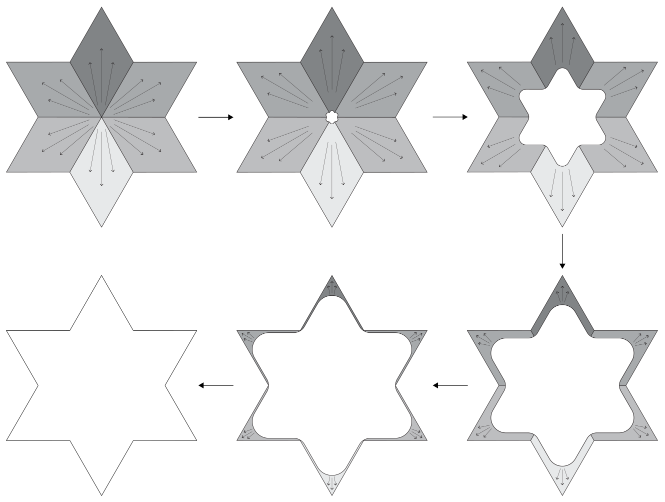

Figure 22: Stages of deformation of the base into the boundary. Arrows indicate traces of the homotopy

We have thus shown that X is homotopy equivalent to a circle, that is,

$$X \simeq S^1.$$

Therefore,

$$TC(X) = TC(S^1) = 2,$$

meaning that our motion planning algorithm will consist of no less than two instructions.

6 Navigation Via the Configuration Space

We will discuss in this section the various movements that together form the algorithm by which the robots move from an initial state to a final state in the configuration space.

First, we will set some notation and introduce the idea of orientation of the robots in the physical space. We will say of a robot in the physical space, positioned on the *endpoint* of a leg, that it is *facing* the *center*. We will also consider that robot to have two adjacent legs, one being on its *left* and the other on its *right*.

A robot at the center will be simply facing the other robot. It has two adjacent legs as well, one on its left and one on its right.

Physical space correspondence to configuration space Boundary configurations consist of those states in which at least one robot is at an endpoint, and the other on another leg or at the center.

Ray configurations consist of those states in which a robot is at the center of the physical space, with the other robot being anywhere else.

Triangle configurations consist of those states in which both robots are on the same leg.

Finally, configurations in the interior of the quadrangle consist of those states in which the robots are not on the same leg but also neither at the center or at an endpoint.

Peak configuration As mentioned in section 5.2, peak states lie in the boundary of our configuration space. In a peak state of the configuration space, when both robots are positioned at an endpoint in the physical space, both will have an occupied leg on one side and an unoccupied leg on the other.

Valley configuration Like peak states, valley states also lie in the boundary. In a valley state of the configuration space, the robot on the endpoint in the physical space will have both legs to its left and right unoccupied. The center is occupied by the other robot. We also consider that center robot to have an unoccupied leg both to its right and to its left.

Intermediate configuration An *intermediate* configuration is one in which one robot is at an endpoint and the other is anywhere on another leg but not at that leg's endpoint.

6.1 Movement in the Boundary

We are now in a position to know how motion initiates if the initial state is a peak, a valley, or an intermediate configuration.

The robots are to first check their initial positions against the final positions. They will have a different set of instructions depending on the initial and final positions being antipodal or not.

6.1.1 Antipodal Movement in the Boundary..

If the final state is antipodal to the initial state, the robots will execute the appropriate instructions detailed below. In this antipodal arrangement, the robots are simply switching positions in the physical space. If the initial position of one robot is equal to the final position of the other robot and vice versa, we say that the robots are *to be switched*.

Instruction 1. Counterclockwise movement in the boundary.

To begin, we consider our current state to be the initial state.

1. If the current state is as specified below, follow the instructions enumerated.
   (a) From a peak configuration along the circular space, counterclockwise movement translates to moving the robot with an unoccupied leg on its right toward that leg's endpoint. If the final state is not reached in the process, the next state reached is another peak.
   (b) From a valley configuration along the circular space, counterclockwise movement translates to moving the center robot toward the endpoint of the unoccupied leg on its right. If the final state is not reached in the process, the next state reached is a peak.
   (c) From an intermediate configuration along the circular space, counterclockwise movement translates to moving the "free-roaming" robot toward the center if it is on the right-side leg of the static robot. Otherwise, if the free-roaming robot is on the static robot's left leg, it is moved toward the nearest endpoint. If the final state is not reached in the process, the next state reached is a valley or a peak.
2. Check the new current state. If the current state is the final state, the process terminates. Otherwise, repeat step 1.

6.1.2 Shortest Path Movement in the Boundary..

We have just seen how to navigate in the boundary of the configuration space when the initial and final states are antipodal to one another. We turn our attention now to all other cases.

Counterclockwise navigation in the boundary required a means of coordination that is not applicable to shortest-path movements. That is, while reaching an antipodal position necessitated distinguishing between the robots' left and right legs before initiating movement, movement to a final state that is not antipodal to the initial state must rely on the determination of distances in the physical space in order to succeed.

First, we will consider our graph to be a *metric graph* as defined by Bridson and Haefliger in [10]. With this metric, each edge has length one and the *distance* between two robots is given by the infimum of the lengths of all paths joining them.

Standard Traversal All movement in the boundary is a composition of a single, shorter movement. Movement in the boundary from one peak to another, for example, consists of one robot remaining stationary and the other moving from the initial position at the endpoint of its leg to the final position at the endpoint of the unoccupied leg, i.e. the robot travels the entire lengths of two legs in the physical space when the configuration state is moving from peak to peak. A complete revolution of the state of robots around the boundary consists of a single such movement, which we will call a *standard traversal*, being performed three times alternately by each robot for a total of six times.

A standard traversal, however, need not be executed *fully*, that is, from peak to peak. The state of robots can move from, say, an intermediate state to an adjacent valley. We will say of such standard traversals that they are executed *partially*.

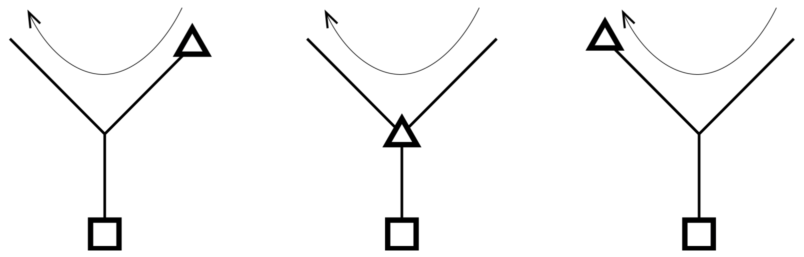
Figure 23: A full standard traversal.

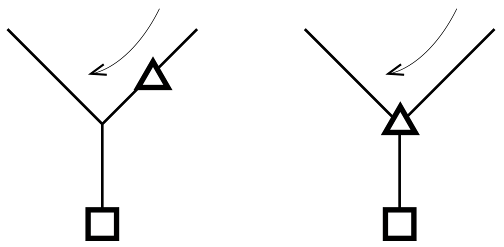
Figure 24: A partial standard traversal.

Shortest Path Determination When the initial and final states are antipodal, there are two equidistant paths between them. This is not the case with all other initial-final state configurations. Instead, the latter cases are distinguished by the fact that there is a shortest path.

A peak-to-peak standard traversal consists of one robot moving from its position at an endpoint to the unoccupied endpoint of another leg; the other robot is static. This standard traversal consists of the robot first moving to the center of the physical space (which is a valley of the configuration space) and then to the aforementioned endpoint (which is an adjacent peak). In total, the robot travels two units in the physical space, by the metric defined earlier.

Once it is determined that the initial and final states are not antipodal, the system must compute the distance to be travelled via each available path. Then, the shortest path is selected and followed to the final state. This distance correlates directly to the number of movements the robots will have to execute in the physical space: the smaller the value, the fewer the movements.

We can now describe the movement of a robot from an initial to a final position if the positions correspond to non-antipodal states in the configuration space. In this case, the robots will perform a sequence of standard traversals until the final configuration has been reached.

Instruction 2. Shortest path movement in the boundary. To begin, we consider our current state to be the initial state.

1. Determine the shortest path.
2. Perform a (partial) standard traversal in the direction of the shortest path.
3. Check the new current state. If the current state is the final state, the process terminates. Otherwise, repeat step 2.

6.2 Movement from Triangle to Ray

We will specify in this section how the deformation of the triangle into the ray by means of the homotopy described in section 5.2 translates to the physical space.

As the triangle is deformed downward to the ray, all possible states on the triangle are shifted to the ray. Moving the robot nearest the center to the center, while the robot nearest the endpoint remains static, translates to moving the configuration state from the triangle to the ray.

Instruction 3. Movement from triangle to ray.

1. Move to the center the robot whose current position is closest to the center while the other robot remains static.

6.3 Movement from Quadrangle to Boundary

In deforming the quadrangle we are shifting all states outward directly to the boundary following the traces of the homotopy.

Instruction 4. Movement from quadrangle to boundary.

1. Move both robots simultaneously toward their nearest respective endpoints until at least one is at an endpoint. The ratio between their speeds is given by the slope of the traces of the homotopy described in section 5.2.

In particular, if one robot is at the center, but the other is not at an endpoint, the trace of the homotopy is horizontal. The robot that is at the center will remain static while the other one moves to its nearest endpoint. This describes the movement from a ray to a peak in the configuration space.

7 Complete Motion Planning Algorithm

We have seen how movement in the physical space and the configuration space are related; how each state in the configuration space is deformed via the homotopy to a boundary state; and how movement from an initial state to a final state is executed in the boundary depending on whether or not the states are antipodal to one another. We turn our attention now to the complete motion planning algorithm that will be an iteration of the processes described above. To begin with, we will be moving the initial state to the boundary and will refer to that new state as the *boundary initial state*. Likewise, the final state shifted to the boundary will be the *boundary final state*.

7.1 Complete motion planning algorithm in the configuration space

Our system of robots will first check their initial state. If they find themselves in a triangle configuration, i.e. if they are on the same leg, the one nearest the center will move to the center. Otherwise, if the robots are in a quadrangle configuration (on separate legs) they will spread out by each moving to its nearest endpoint. This process will output the boundary initial state. Simultaneously, the system of robots will check the final state. If the final state is in a triangle configuration, the position of the final state nearest the center will be moved to the center. Otherwise, if the positions are in a quadrangle configuration, they are moved to their nearest respective endpoints. This process will output the boundary final state. Once in a boundary configuration, the actual robots will proceed counterclockwise around the configuration space if their boundary initial state is antipodal to the boundary final state; otherwise, they will follow the shortest path to the boundary final state. Once this is accomplished, they will move into the final state by following the inverse path to the one taken while moving to the boundary final state from the final state.

7.2 Complete motion planning algorithm in the physical space

We may now navigate via the physical space, following the steps below.

Instruction 5. Movement from any initial positions to any final positions in the physical space.

1. Check the initial positions. If both robots are on the same leg, follow instruction 3. Otherwise, follow instruction 4. Repeat until at least one robot is at an endpoint.
2. Check the final positions. If they are on the same leg, follow instruction 3. Otherwise, follow instruction 4. Repeat until at least one position is an endpoint.
3. If the robots are now to be switched, proceed with instruction 1. Otherwise, proceed with instruction 2.
4. Move the robots to their final position following the reverse movement to the one performed in step 2.

8 Generalizations

This algorithm could be potentially generalized in two directions: by augmenting the number of robots in the graph or by augmenting the complexity of the graph. In the first case, the increase in the number of robots results in a direct increase in the dimensions of the configuration space, which can make the visualizations more complicated. In the latter case, increasing the complexity of the graph and still obtaining concrete algorithms was explored, for instance, in the works of Allaoua Boughrira [2] and Elif Sensoy [14] for the cases in which the graph also contains loops.

References

[1] Abrams, A. *Configuration Spaces and Braid Groups of Graphs.* PhD thesis, UC Berkley, 2000.

[2] Boughrira, A. and Colman, H. *A Motion Planning Algorithm in a Lollipop Graph*, Minnesota Journal of Undergraduate Mathematics, Vol. 6 no. 1 (2021).

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

[4] Fadell, E. and Neuwirth, L. *Configuration Spaces*, MATHEMATICA SCANDINAVICA Vol 10 (1962), 111–118.

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

[6] Ghrist, R. *Configuration spaces of graphs and robotics,* in Braids, Links, and Mapping Class Groups: the Proceedings of Joan Birman's 70th Birthday, AMS/IP Studies in Mathematics, vol. 24, 29–40.

[7] Ghrist, R. and Koditschek, D. *Safe cooperative robot dynamics on graphs,* SIAM J. Control Optim. 40. (2002), 1556–1575.

[8] Girija, P., Mareena, J., Fenny, J., Swapna, K. and Kaewkhiaolueang, K. *Amazon Robotic Service (ARS)* (2021). Engineering and Technology Management Student Projects. https://archives.pdx.edu/ds/psu/36434

[9] Del Rey, Jason. (2022). Amazon's robots are getting closer to replacing human hands. https://www.vox.com/recode/2022/9/27/23373588/amazon-warehouse-robots-manipulation-picker-stower

[10] Bridson M.R., Haefliger A. *Basic Concepts. In: Metric Spaces of Non-Positive Curvature,* Grundlehren der mathematischen Wissenschaften (A Series of Comprehensive Studies in Mathematics), vol 319. (1999), Springer, Berlin, Heidelberg.

[11] Latombe, J. *Robot Motion Planning,* Kluwer Academic Publishers 1991.

[12] Lütgehetmann, D. *Configuration spaces of graphs.* Masters thesis, Freie Universität Berlin, 2014.

[13] Salgado, R. and Velazquez, E. *Topological Complexity for Driverless Vehicles.* SIAM Undergraduate Research Online (SIURO), Vol.6. (2016).

[14] Sensoy, E. *Collision free motion planning on a wedge of circles.* SIAM Undergraduate Research Online (SIURO), Vol.14. (2021).

David Baldi
Wright College, City Colleges of Chicago
davidbaldi@gmail.com