Robótica Topológica

por
Aniceto Murillo, Universidad de Málaga

> *1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.*
>
> *2. A robot must obey orders given to it by human beings, except where such orders would conflict with the First Law.*
>
> *3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law..*
>
> **Isaac Asimov**, *I, robot.*

1. Introducción

A decir verdad, el término *robótica* no fue acuñado en importantes reuniones científicas por eruditos ingenieros, informáticos o matemáticos. Fue el prolífico escritor estadounidense Isaac Asimov[1] quien la utilizó por primera vez en 1941 en uno de sus cuentos cortos que conformaron más tarde, el volumen publicado en 1950 *"I, robot"*. A su vez, el término *robot* tal y como hoy lo conocemos fue usado por primera vez en 1920 por el también autor de ciencia ficción Karel Capeck[2] en su obra de teatro *"R.U.R. Rossum's Universal Robots"*. No diría yo que, cuando Asimov usó este vocablo, fuese del todo consciente que estaba acuñando el nombre de un importante campo

[1]Isaac Asimov (1920-1992) nació en Rusia y emigró a muy corta edad a los Estados Unidos. Fue bioquímico y un prolífico escritor de todo tipo de obras, no sólo de ciencia ficción.

[2]Karol Capek (1890-1938), ha sido uno de los escritores en lengua checa más importantes del siglo XX.

de la ciencia y la tecnología actual. Muy a grosso modo, la robótica tiene por objeto el diseño y construcción de *robots*, agentes mecánicos y/o electrónicos con autonomía parcial o total para detectar propiedades de su entorno e interactuar con él, así como algoritmos preconcebidos que hagan realizar a estos robots, de forma inteligente y eficiente, los fines para los que han sido construídos.

Una de las principales ramas de la Robótica donde en la actualidad se investiga con fruición es la llamada *planificación de movimientos*. También a grandes rasgos, el objetivo de esta crucial parte de la robótica donde confluyen la informática, la ingeniería y por supuesto las matemáticas, es la creación, diseño e implementación de algortimos en un robot o conjunto de robots que los permitan moverse adecuadamente por el entorno en el que "viven". El ejemplo típico de planificación de movimientos que mejor ilustra esta somera descripción, es el del *transportista de pianos*: imaginemos que un tal transportista (robot) se encuentra con un piano a la entrada de una casa y debe dejar la pesada carga en una precisa y (por desgracia para el que lo transporta) recóndita habitación. Un algoritmo que haga que tanto el transportista como su carga lleguen sanos y salvos al preciso rincón de la recóndita habitación, sorteando todos los posibles obstáculos, es un planificador de este movimiento. Cómo hacer que un automóvil aparque de forma autónoma sin colisionar con otros vehículos, cómo hacer que varios robots se desplacen sin colisionar en un determinado entorno, o simplemente, cómo hacer que un brazo articulado se mueva de una posición a otra, responden también al diseño de algoritmos planificadores de movimientos.

De forma general la planificación de un movimiento tiene dos importantes tareas que llevar a cabo. La primera de ellas diseñar, modelar el robot o conjunto de robots que intervienen en el movimiento así como el entorno donde éstos viven (obstáculos a sortear, capacidad de movimiento de los robots, posibles posiciones físicas de todos y cada uno de los robots,...). A toda esta amalgama de robots, mundo en el que viven y posibles situaciones de los mismos se le denomina *espacio de configuraciones*. Un estado de un espacio de configuraciones es por tanto una determinada posición del robot o conjunto de robots en el entorno donde se mueven. Así, el espacio de configuraciones del transportista de piano estaría formado por el transportista en sí, el piano, junto con la casa y todos los obstáculos anexos a ésta.

La segunda tarea, no menos importante, una vez conocido y diseñado el espacio de configuraciones, consiste en la creación e implementación de algoritmos que permitan al robot o conjunto de robots pasar de forma autónoma de un estado a otro del espacio de configuraciones.

2. Robótica Topológica

En la robótica moderna, métodos propios de la geometría y la topología pueden ser usados de forma eficiente para llevar a cabo estas dos tareas esenciales que conforman la planificación de un movimiento. El volumen *"Robot Motion Planning"* escrito por Jean Claude Latombe[3], es un excelente y enciclopédico trabajo donde pueden encontrarse las primeras aplicaciones de métodos topológicos y geométricos para el estudio de planificadores de movimientos.

En efecto, por una parte, la geometría y la topología nos permiten describir de forma esencial el espacio de configuraciones de un determinado movimiento: imaginemos un robot consistente en un sencillo brazo articulado como el que muestra la siguiente ilustración:

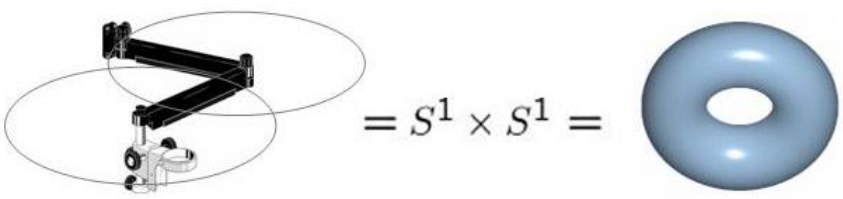


Figura 1

Nótese que cada uno de los brazos tiene un sólo "grado de libertad", esto es, puede rotar en un solo plano y por tanto su extremo describe una circunferencia (a la que los matemáticos nos gusta llamar S^1). Así pues, el conjunto de todos los posibles estados del brazo articulado, esto es, el espacio de configuraciones de este sencillo robot, puede ser interpretado de forma geométrica como el producto cartesiano de dos circunferencias $S^1 \times S^1$. Es sabido también que el producto de dos circunferencias desde el punto de vista geométrico o topológico no es más que un toro. Hemos pues identificado el espacio de configuraciones de este sencillo robot con un clásico objeto geométrico. De igual forma un brazo articulado con un número n de brazos (y cada uno de ellos con distinto grado de libertad) se identifica al producto de n esferas cada una de ellas de la dimensión correspondiente al grado de libertad del brazo.

Pongamos otro ejemplo: imaginemos que tres robots se desplazan sin colisionar en un determinado espacio ambiente al que llamaremos M. Para descri-

[3]Jean Claude Latombe, es profesor en el Departamento de Ciencias de la Computación de la Universidad de Stanford y lidera grupos de investigación punteros en distintos ámbitos de la robótica relacionados con la planificación de movimientos: cirugía asistida por robots, movimiento molecular, sensores de movimientos,...

bir topológicamente el correspondiente espacio de configuraciones recordemos que la topología estudia las propiedades cualitativas (espacio, forma,...) que no cuantitativas de los espacios geométricos. Podemos suponer por tanto que cada uno de los robots no son más que puntos que se mueven en el espacio ambiente M. Como quiera que los robots no pueden colisionar, cada estado del espacio de configuraciones al que llamaremos X queda representado por ternas de puntos de M distintos entre sí. Esto es,

$$X = \{(a, b, c) \in M \times M \times M, \quad a \neq b,\, b \neq c,\, c \neq a\}.$$

El/la lector/a iniciado/a reconocerá en esta descripción topológica al denominado *espacio de configuraciones de tres partículas* por el que se conoce, esta vez en ambientes físicos y matemáticos, a este espacio cuyo estudio ha dado lugar a importantes resultados en el lugar donde confluyen la topología algebraica, la geometría diferencial y la física matemática. Claro está, este ejemplo puede generalizarse de forma natural a un número mayor de robots y en un espacio ambiente M con distintos obstáculos.

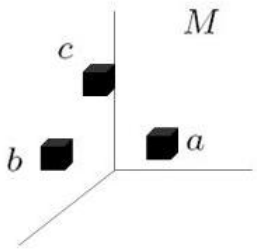


Figura 2

Una vez descrita de forma geométrica y/o topológica el espacio de configuraciones, la topología también permite formular la descripción de algoritmos planificadores de movimientos. Veamos cómo:

Consideremos un determinado espacio de configuraciones al que llamaremos X e intentemos abstraernos de todas las dificultades de índole mecánica, electrónica inherentes al mismo. Así las cosas, un algoritmo planificador del movimiento ha de tener como *input* dos estados cualesquiera (llamémosles A y B) del espacio de configuraciones X. El *output* del algoritmo debe proporcionar un camino sin interrupciones del estado A al estado B. Un tal camino puede ser descrito de forma topológica por una aplicación continua $f_{A,B}$ (sin cortes, sin interrupciones) del intervalo cerrado $[0, 1]$ en el espacio de configuraciones X de forma que comience en A, esto es $f_{A,B}(0) = A$, y termine en B, esto es $f_{A,B}(1) = B$,

$$f_{A,B}\colon [0, 1] \longrightarrow X, \quad f_{A,B}(0) = A, \quad f_{A,B}(1) = B.$$

Con esta idea en mente, consideremos el conjunto de todos los posibles caminos en X, esto es, el conjunto de todas las aplicaciones continuas (curvas) del intervalo $[0, 1]$ en X y denotémoslo por $X^{[0,1]}$. Tomemos a continuación la aplicación $\gamma\colon X^{[0,1]} \to X \times X$ que asocia a cada curva en X sus extremos:

$$\gamma\colon X^{[0,1]} \longrightarrow X \times X, \quad \gamma(f) = \big(f(0), f(1)\big).$$

Así pues, tal y como hemos acordado, observamos que un algoritmo planificador del movimiento de nuestro espacio de configuraciones no es más que una aplicación inversa por la izquierda (también llamada sección) de γ. Esto es, una aplicación

$$\alpha\colon X \times X \longrightarrow X^{[0,1]}, \quad \gamma \circ \alpha = 1_{X\times X}.$$

Hemos por tanto reducido, de forma topológica, el problema de encontrar algoritmos planificadores de movimientos a encontrar secciones de la aplicación que envía cada curva a sus extremos.

3. Complejidad topológica de un espacio de configuraciones

Como quiera que el conjunto $X^{[0,1]}$ de todas las curvas de X puede ser dotado de una estructura topológica (podemos decir cuando dos curvas están cerca una de otra) que hace que la aplicación γ sea continua, tiene sentido preguntarse cuando una determinada sección de γ, esto es, un algoritmo planificador del movimiento, es una aplicación también continua. La respuesta, aunque fácilmente demostrable con conocimiento elementales de topología, no es menos sorprendente y debe ser enunciada como teorema:

Teorema 1. *Para un espacio de configuraciones X existe un algoritmo planificador de movimientos $\alpha\colon X \times X \to X^{[0,1]}$ que es además una aplicación continua si y sólo si el espacio X es contráctil.*

Recordemos que un espacio X es contráctil si puede ser *deformado* a un punto. Si imaginamos que nuestro espacio X está hecho de plastilina muy moldeable, para ser contráctil debemos ser capaces de deformarlo a un punto, sin cortarlo ni pegar ninguna de sus partes. Así, una bola maciza, un disco, un plano, el espacio euclídeo son espacios contráctiles. La circunferencia, la esfera (hueca), el toro, un disco agujereado son por el contrario espacios que no pueden ser deformados a un punto.

Formalmente, el carácter contráctil puede ser formulado fácilmente en los siguientes términos: un espacio X es contráctil si existe una aplicación continua (deformación)

$$H\colon X \times [0,1] \longrightarrow X$$

de forma que $H(x,0) = x$ y $H(x,1) = x_0$ para cualquier $x \in X$. Por x_0 entendemos un punto fijo del espacio X.

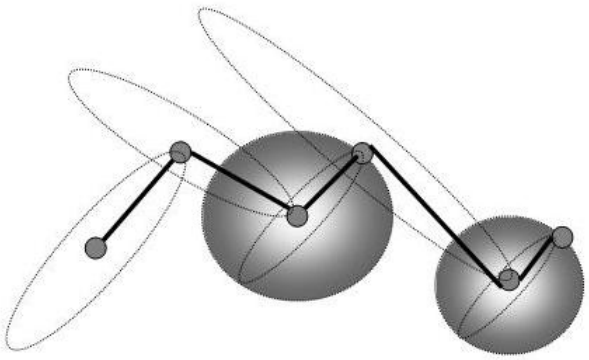
Pues bien, el resultado anterior nos dice que si un espacio de configuraciones resulta ser contráctil, una tal contracción produce una sección continua de γ y por tanto, un algoritmo planificador del movimiento.

Sin embargo, los espacios de configuraciones no son contráctiles en general, tal y como hemos visto en ejemplos anteriores, por lo que encontrar un algoritmo planificador puede ser complicado. ¿Cómo de complicado? La medida estándar hoy día que mide esta complejidad es la llamada *complejidad topológica* de un espacio de configuraciones X. Este índice fue introducido en 2003 por Michael Farber[4] y puede ser definido como el menor número de trozos (abiertos) con que podemos recubrir el producto $X \times X$ y donde en cada uno de estos trozos existe una sección continua de γ.

Este índice se corresponde en cierta medida con el menor número de órdenes que debe contener cualquier algoritmo que rija un determinado planificador de movimiento de un espacio de configuraciones dado.

Con ayuda de resultados que pueden probarse utilizando una batería más o menos elemental de teoría de homotopía, puede calcularse la complejidad topológica de espacios de configuraciones que hemos estudiado en ejemplos anteriores:

Consideremos en primer lugar un robot consistente en un brazo articulado de n elementos de los cuales n_1 brazos sólo tienen un grado de libertad, esto es, se mueven en un plano y los restantes n_2 brazos se mueven libremente en el espacio. La complejidad topológica de este espacio de configuraciones es $n_1 + 2n_2 + 1$.

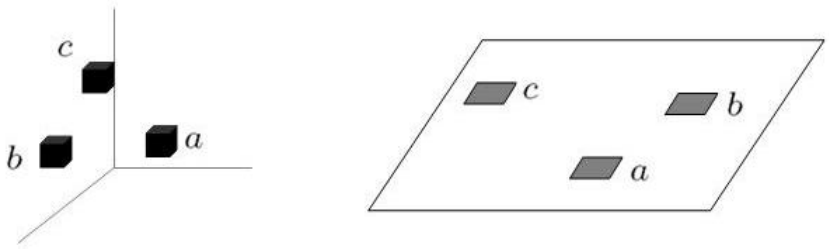


$n = 5, \quad n_1 = 3, \quad n_2 = 2, \quad TC = 8$

Figura 3

Por otra la complejidad topológica del espacio de configuraciones de n robots moviéndose sin colisionar en un plano resulta ser $2n - 2$. Por contra, si estos n robots se mueven en el espacio euclídeo la complejidad topológica es $2n - 1$.

[4]Michael Farber es profesor de la Universidad de Durham en el Reino Unido y experto en el uso de métodos topológicos en robótica.



$TC = 2n - 1$ $TC = 2n - 2$

Figura 4

Terminemos esta colección de ejemplos indicando que si el espacio de configuraciones resulta ser una superficie orientable compacta de genero g, esto es, un toro de g asas, su complejidad topológica es 3 si $g \leq 1$, es decir si se trata de una esfera o un toro estándar, o 5 si el número de asas es mayor o igual que 2.

$TC = 3$ $TC = 3$ $TC = 5$ $TC = 5$

Figura 5

Convencidos como estamos de que la complejidad topológica determina el número mínimo de instrucciones de un algoritmo planificador de movimiento, cabe preguntarse, ¿cuán complejo es calcular la complejidad topológica?, o en otras palabras, ¿cuán difícil es adelantar el mínimo número de instrucciones que un planificador de movimientos debe contener?

4. Complejidad algorítmica y complejidad topológica

Para contestar a esta pregunta, resumamos de forma breve en qué términos se mide la *complejidad algorítmica* (¡que no topológica!) de un problema dado.

Es habitual en ciencias de la computación entender un problema de decisión como una función $\Pi \to \{0, 1\}$ siendo $\Pi = \{I\}$ el conjunto de todas y cada una de las instancias del problema. La imagen de cada instancia I del problema por la función es 1 si la respuesta a esa instancia del problema es "Sí", y es 0 por contra, si la respuesta a tal instancia es "No". ¿Tiene esta

ecuación solución? ¿Es este número primo? ¿Es 26 la complejidad topológica de este espacio de configuraciones? Todos estos son ejemplos de problemas de decisión. En el primero de ellos cada ecuación es una instancia; en el segundo problema, una instancia del mismo es sencillamente un número natural; y en el tercero, cada instancia viene dada por un espacio de configuraciones.

Para que un algoritmo pueda procesar un problema es necesario codificar cada una de las instancias I del mismo como una secuencia finita de números naturales, o más sencillamente de ceros y unos de una cierta longitud. A ésta se le llama longitud de la instancia I.

Los problemas de decisión más sencillos de resolver son los problemas *polinomiales*. Un problema de decisión decimos que pertenece a la clase P (*polynomial*) si existe un algoritmo $\mathcal{A}$ que resuelve cada instancia del problema en tiempo polinomial respecto a la longitud de la instancia, esto es, existe un polinomio p de forma que para cada instancia I del problema (el número primo, la ecuación, el espacio de configuraciones,...) el algoritmo $\mathcal{A}$ resuelve el problema para la instancia I en tiempo $p(l)$ siendo l la longitud (o el tiempo de ordenador si queremos pensar en esos términos) requerida para codificar la instancia I.

Por otra parte un problema de decisión pertenece a la clase NP (*nondeterministic polynomial*) si existe un algoritmo $\mathcal{A}$ que tiene por entrada pares (I, C) formados por una instancia del problema más un input añadido C que denominamos certificado (candidato a solución) y que opera de la siguiente forma:

- Para cada instancia I del problema para la que la respuesta es "Sí" existe un certificado $C(I)$ de forma que tomando $(I, C(I))$ como entrada para el algoritmo $\mathcal{A}$, éste reconoce en tiempo polinomial que en efecto la respuesta a la instancia I es "Sí".

- Para cada instancia I del problema para la que la respuesta es "No", cualquier entrada para el algoritmo $\mathcal{A}$ de la forma (I, C) hace que éste concluya en tiempo polinomial que en efecto la respuesta para I es "No".

En otras palabras, la clase P está formada por aquellos problemas para los que es fácil (polinomial) encontrar una solución, mientras que la clase NP está formada por aquellos problemas para los que es fácil decidir si un candidato a solución en efecto lo es. Decidir si una ecuación polinómica tiene una solución de una determinada característica puede no ser fácil. Sin embargo sí que lo es decidir si un determinado número de esa característica es en efecto solución de la ecuación. Con este otro ejemplo seguro que los/as lectores/as se sentirán inmediatamente identificados/as: Demostrar un determinado teorema en general no es fácil pero sí que lo es verificar si una demostración dada es en efecto correcta y constituye una prueba del teorema en cuestión.

Obviamente todo problema de la clase P también está en la clase NP

puesto que el algoritmo que lo resuelve en tiempo polinomial no necesita de certificado. Así pues $P \subset NP$. Pero, ¿es esta inclusión estricta? En otras palabras, ¿será cierto que cada problema de decisión para el que sea fácil validar un candidato a solución, posee de hecho un algoritmo polinomial que lo resuelva? Esta cuestión, propuesta independientemente por Stephen Cook y Leonid Levin[5] continúa abierta y constituye uno de los principales problemas en teoría de la complejidad algorítmica y por ende, en ciencias de la computación. Hemos de decir que la mayoría de los expertos consideran que $P \neq NP$, esto es, que la inclusión $P \subset NP$ es estricta y que hay por tanto problemas NP que no pueden resolverse en tiempo polinomial.

En orden creciente de dificultad se encuentra la clase de problemas NP-completos. Un problema es NP-completo si pertenece a la clase NP y cualquier otro problema de NP puede reducirse polinomialmente a él. En otras palabras si supiéramos resolver un problema NP-completo podríamos también resolver (en el mismo tiempo) cualquier otro problema NP.

El principal ejemplo de problema NP-completo es el del *coloreado de grafos*. Expliquémoslo de forma breve: un grafo G puede ser entendido como un conjunto finito $V(G)$ cuyos elementos son los *vértices del grafo* junto con una colección de pares (no ordenados) de vértices $A(G)$ denominados *aristas*.

Vamos a suponer que todo vértice del grafo está conectado a otro vértice por alguna arista y que ninguna arista conecta a un vértice consigo mismo.

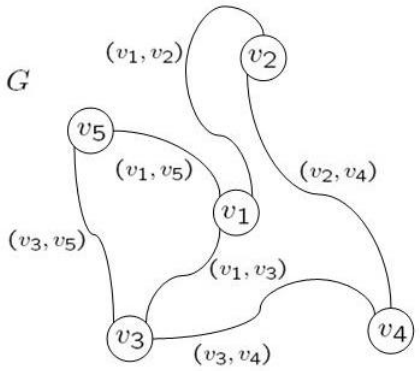


Figura 6

Pues bien, colorear un grafo G con k colores, $k \geq 2$, consiste en asignar a cada vértice un color de forma que vértices adyacentes tengan siempre colores distintos. Dos vértices $v, w \in V(G)$ son adyacentes si hay una arista que los une, esto es, si $(v, w) \in A(G)$. Formalmente un k-coloreado de G es pues una aplicación

$$f\colon V(G) \longrightarrow \{1, 2, ..., k\}$$

de forma que si $(v, w) \in A(G)$ es una arista, entonces $f(v) \neq f(w)$.

Pues bien, si fijamos un entero $k \geq 3$, decidir si un grafo dado es k-coloreable es un problema NP-completo. Es conveniente hacer constar que

[5]Stephen Cooke, profesor de la Universidad de Toronto, y Leonid Levin, profesor de la Universidad de Boston, formularon independientemente la famosa cuestión ¿$P = NP$? en 1971. Aún sin resolver, es éste uno de los *7 Problemas del Milenio* denominados así por el prestigioso *Clay Institute* y cuya resolución está premiada con un millón de dólares.

hay distintas formas de codificar un grafo (instancia del problema) pero todas ellas tienen una longitud acotada polinomialmente por el número de vértices. Tomemos por ejemplo un grafo G con vértices $V(G) = \{v_1, ..., v_n\}$. Una codificación de este grafo viene dada por la llamada *matriz de adyacencia*, que no es más que una matriz cuadrada de orden n cuya entrada en el lugar de fila i y columna j es 1 si los vértices v_i y v_j son adyacentes, esto es, si $(v_i, v_j) \in A(G)$, y 0 en caso contrario. Como se observa de forma elemental esta codificación tiene longitud acotada por un polinomio en n, el número de vértices.

Por último, también en escala creciente de dificultad, un problema diremos que es NP-difícil si cualquier otro problema en NP puede reducirse en tiempo polinomial a él. El hecho de no suponer que este problema sea NP es la única sutil diferencia que lo distingue de un problema NP-completo.

Una vez realizada esta brevísima incursión por el mundo de la complejidad algorítmica podemos volver a la tarea que nos ocupa. Sabemos que la complejidad topológica de un determinado espacio de configuraciones nos da una idea de la cantidad de órdenes (y por ende de la dificultad) que debe contener un algoritmo planificador del movimiento del espacio de configuraciones en cuestión. Sabemos además calcularlos en algunos casos. Pero, desde el punto de vista algorítmico, ¿cómo de complejo es calcular la complejidad topológica?

Es necesario en primer lugar relacionar un invariante tan actual como la complejidad topológica con otro invariante mucho más antiguo pero no menos importante, introducido y desarrollado a finales de los años veinte del siglo pasado por L. Lusternik y L. Schnirelmann[6]. Dado un espacio topológico cualquiera X la *categoría de Lusternik-Schnirelmann* o *LS-categoría* de X, a la que denotaremos por $cat(X)$, es el menor entero n para el que existe un recubrimiento de X formado por n-trozos abiertos y cada uno de ellos contráctiles X. Así por ejemplo la categoría de un espacio contráctil es por tanto 1 y la de una esfera es 2 para lo que basta con tomar cada uno de los dos hemisferios que trivialmente son contráctiles. Este invariante de tan sencilla definición ha resultado ser de valiosa utilidad en distintas ramas de las matemáticas, desde la geometría diferencial, a la topología algebraica, pasando por los sistemas dinámicos y como vemos ahora, también en la robótica topológica.

En efecto, resulta que dado un determinado espacio de configuraciones X,

[6]Lazar Lusternik (1899-1981) fue un matemático ruso, famoso por sus trabajos en topología y geometría diferencial.

Lev Schnirelmann (1905-1938) fue también un matemático ruso conocido, además de por sus trabajos en colaboración con Lusternik, por los profundos resultados que probó en teoría de números, intentando demostrar la conjetura de Goldbach.

la complejidad topológica de X y la LS-categoría de X quedan relacionadas por la siguiente fórmula:

$$cat(X) \leq TC(X) \leq 2cat(X). \quad (1)$$

A partir de esta fórmula es sencillo deducir que es tan difícil calcular la complejidad topológica de un espacio de configuraciones como su LS-categoría.

Por otra parte, dados un grafo G y un entero $k \geq 3$, es posible construir un espacio topológico $X_{G,k}$ cuya codificación está acotada por un polinomio en el número de vértices del grafo G (y por tanto, como hemos recalcado con anterioridad, requiere la misma longitud que para codificar el grafo G) y para el que además se puede demostrar la siguiente propiedad:

G es k-coloreable si y solamente si la LS-categoría de $X_{G,k}$ es infinita, esto es, no es posible recubrir el espacio $X_{G,k}$ por un número finito de trozos abiertos y contráctiles en el espacio.

Lamentablemente, para describir explícitamente el espacio $X_{G,k}$ necesitaríamos introducir nociones nada elementales de teoría de homotopía que se escapan al carácter divulgativo de este texto. No obstante, lo dicho aquí es suficiente para concluir que, en términos algorítmicos, podemos reducir polinomialmente el problema del k-coloreado de grafos al problema de determinar si la LS-categoría de un espacio es finita o no.

Ahora bien, por la fórmula (1), la LS-categoría de $X_{G,k}$ será infinita si y sólo si lo es su complejidad topológica. Con esto hemos demostrado el siguiente teorema:

Teorema 2. *El problema del k-coloreado de grafos se puede reducir polinomialmente al problema de determinar si la complejidad topológica de un determinado espacio de configuraciones es finita.*

Como quiera que el problema del k-coloreado de grafos en NP-completo y lo hemos reducido polinomialmente a determinar la finitud de un espacio de configuraciones dado, este último problema es también NP-difícil. Esto es, hemos demostrado:

Teorema 3. *Determinar la complejidad topológica de un espacio de configuraciones es NP-difícil.*

Así pues, no sólo desarrollar un algoritmo en sí para implementarlo en un determinado robot es complicado. También lo es el cálculo de la cantidad mínima necesaria de instrucciones de un tal algoritmo o, en otras palabras, de las decisiones que el robot o conjunto de robots deben tomar. Este hecho, sin embargo no debe ser desalentador. Más al contrario, constituye un refrendo de

los numerosos y profundos progresos en la utilización de métodos geométricos y topológicos en este campo de la robótica.

Quisiera por último manifestar en estas últimas líneas mi más sincero agradecimiento a los organizadores de *Un paseo por la geometría 2007/2008*, Raúl Ibañez y Marta Macho. No sólo por permitirme impartir la charla cuyo resumen aquí reseño o por su amabilísima hospitalidad. Sobre todo, por la inestimable y difícil labor, a veces ingrata, de hacer llegar las matemáticas a rincones donde nunca pensé que fuese posible.

Bibliografía

[1] I. Asimov, *I, Robot*, Panther Science Fiction, 1968.
[2] M. Farber, *Topological complexity of motion planning*, Discrete and Computational Geometry 29(2), 211-221, 2003.
[3] M. Farber and M. Grant, *Topological complexity of configuration spaces*, preprint ArXiv: 0806.4111v1 [math.AT], 2008.
[4] J.C. Latombe, *Robot Motion Planning*, Kluwer, 1991.
[5] L. Lechuga y A. Murillo, *Complexity in rational homotopy*, Topology 39(1), 89-95, 2000.
[6] L. Lechuga y A. Murillo, *Topological complexity of formal spaces*, Topology and Robotics, Contemporary Math. 438, 105-114, 2008.

Aniceto Murillo
Universidad de Málaga
Departamento de Álgebra,
Geometría y Topología
Apartado 59
29080 Málaga
e-mail: ***aniceto@agt.cie.uma.es***
http://agt.cie.uma.es/∼aniceto