La Columna de Matemática Computacional

Sección a cargo de

Tomás Recio

El objetivo de esta columna es presentar de manera sucinta, en cada uno de los números de La Gaceta, alguna cuestión matemática en la que los cálculos, en un sentido muy amplio, tengan un papel destacado. Para cumplir este objetivo el editor de la columna (sin otros méritos que su interés y sin otros recursos que su mejor voluntad) quisiera contar con la colaboración de los lectores, a los que anima a remitirle (a la dirección que se indica al pie de página[1]*) los trabajos y sugerencias que consideren oportunos.*

En este número...

En este número de *La Gaceta*, presentamos... un artículo del Catedrático de Geometría y Topología de la Universidad de Málaga, Dr. Aniceto Murillo. Versa sobre la intersección de dos ámbitos de la Ciencia y la Tecnología aparentemente lejanos, la Topología y la Robótica. ¿Qué puede tener que ver la Topología con esos androides cuya imagen ha sido popularizada por tantas obras literarias y cinematográficas?

Si pensamos en una versión más modesta (pero más desarrollada en la práctica) de los robots, en los útiles brazos robóticos que aparecen en la industria automovilística, en la carrera espacial o en la cirugía, etc., es posible imaginar, por ejemplo, un papel importante en su diseño y manipulación para ramas de la matemática tales como la Teoría de Control, pero ¿qué puede tener que ver la Topología con la programación de estos brazos articulados? Responder a esta pregunta, mostrando las múltiples y profundas relaciones entre Topología y Robótica, es el objetivo del trabajo que aparece en esta edición de *La Columna...*

Este editor, que se interesó, hace veinte años, por diversos problemas de Álgebra y Geometría Computacional en Robótica, recuerda que, indefectiblemente, surgían problemas topológicos tanto en el análisis del espacio de configuraciones de un robot como en la planificación de sus movimientos. Y recuerda, también, como un antecedente remoto del trabajo que ahora se presenta en esta *La Columna...*, el interesante artículo de divulgación de Daniel Baker titulado *Some topological problems in robotics*, publicado en 1990 en la revista *The Mathematical Intelligencer*, vol. 12, n.º 1.

[1]Tomás Recio, Departamento de Matemáticas, Facultad de Ciencias, Universidad de Cantabria, 39071-Santander; tomas.recio@unican.es

Desde entonces, las aplicaciones de la Topología a la Robótica han ido consolidándose como un campo activo de investigación, con conferencias internacionales monográficas como las celebradas en el ETH de Zurich en 2003 y 2006, con secciones en el ICIAM (el congreso internacional de referencia sobre matemática industrial y aplicada) de 2007 o en el Congreso Europeo de Matemáticas de 2008. La relevancia e interés de este campo de investigación ha merecido incluso una reseña en la prestigiosa revista *Science* (en su edición del 8 de agosto del 2003: vol. 301, n.º 5634, p. 756).

Por todo ello consideramos de especial interés la contribución del profesor Murillo, con el fin de divulgar entre los lectores de *La Gaceta* algunas nociones y problemas fundamentales de las aplicaciones de la Topología a la Robótica.

Para finalizar, queremos que quede constancia de nuestra invitación a presentar, cuando Murillo considere oportuno, un nuevo artículo en el que nos hable, en un futuro, sobre las aplicaciones, en sentido contrario, de la Robótica a la Topología, a las que hace referencia al final de este trabajo.

Robótica topológica: complejidad topológica de planificadores de movimientos[†]

por

Aniceto Murillo

Introducción

Poco podía sospechar Isaac Asimov[1] que, cuando acuñó el término robótica en uno de sus cuentos de ciencia ficción de la colección *I robot*, estaba de hecho dando origen al nombre de un importante campo de la ciencia y la tecnología actual. A decir verdad, el término *robot*, con la acepción con la que hoy lo conocemos, fue usado por primera vez en 1920 por el también autor de ciencia ficción Karel Capek[2] en su obra de teatro *R.U.R. Rossum's Universal Robots*. De forma tal vez excesivamente general, con el único objeto de situar en contexto el presente artículo, podemos decir que la robótica tiene por objeto el diseño y construcción de *robots*, entendiendo por tales los agentes mecánicos y/o electrónicos con autonomía parcial o total para detectar propiedades de su entorno e interactuar con él, así como los algoritmos preconcebidos y debidamente implementados que hagan realizar a estos robots, de forma inteligente y eficiente, los fines para los que han sido construidos.

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 algoritmos 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 escondida

[†]Este trabajo puede considerarse en cierta forma una versión ampliada y, aunque también en clave divulgativa, quizás más rigurosa, de la conferencia «Robótica Topológica» impartida en la Universidad del País Vasco en abril de 2008, durante el ciclo «Un paseo por la geometría» (2007/2008) organizado por los profesores Raúl Ibáñez y Marta Macho, a quienes aprovecho para agradecer el ímprobo esfuerzo que hacen en la ardua labor de divulgar las matemáticas.

[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]Karel Capek (1890–1938) ha sido uno de los escritores en lengua checa más importantes del siglo XX.

habitación sorteando todos los posibles obstáculos es un planificador de este movimiento. Cómo hacer que un automóvil aparque de forma autónoma sin impactar 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.

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* [10], 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.

Veamos con algunos ejemplos cómo conceptos geométricos y topológicos básicos 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 ilustración adjunta (figura 1).

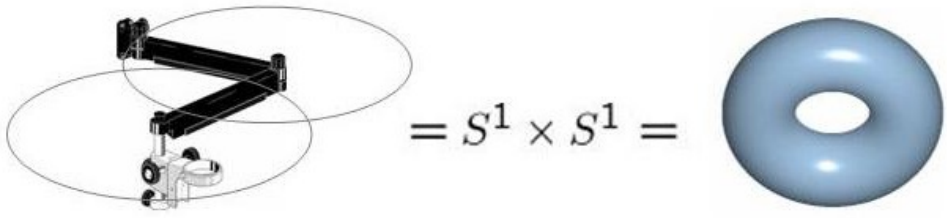


Figura 1.

[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,...

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 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 de dos circunferencias $S^1 \times S^1$, que, como es bien sabido, tanto desde el punto de vista geométrico como topológico puede identificarse a un toro. De forma general, un brazo articulado con n brazos donde cada uno de ellos posee m_i «grados de libertad esféricos», $i = 1, \ldots, n$, puede identificarse al producto de n esferas $S^{m_1} \times \cdots \times S^{m_n}$. Aquí usamos la notación estándar, a saber: la esfera de dimensión m se define como los puntos del espacio euclídeo de dimensión $m+1$ de módulo 1,

$$S^m = \{(x_1, \ldots, x_m) \in \mathbb{R}^m,\ x_1^2 + \cdots + x_m^2 = 1\}.$$

Pongamos otro ejemplo: imaginemos que tres robots se desplazan sin colisionar en un determinado espacio ambiente al que llamaremos M. Simplificando en gran medida el problema (veremos después que de hecho, para nuestros objetivos, es lícito hacer tal simplificación), vamos a suponer 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 X queda representado por ternas de puntos de M distintos entre sí (figura 2). Esto es,

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

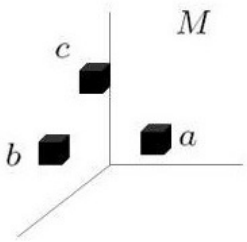


Figura 2.

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 [5]. Claro está, este ejemplo puede generalizarse de forma natural a n robots y en un espacio ambiente M con distintos obstáculos para dar lugar al espacio de configuraciones

$$F(M; n) = \{(x_1, \ldots, x_n) \in M^n,\ x_i \neq x_j,\ \text{si } i \neq j\}.$$

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

Consideremos un determinado espacio de configuraciones 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 A, B, del espacio de configuraciones X. El *output* del algoritmo debe proporcionar un camino sin interrupciones del estado A al estado B, esto es, una curva, una aplicación continua $f_{A,B}\colon [0,1] \to X$ que comience en A y termine en B,

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

La primera obstrucción para que esto sea posible es que, dados dos estados cualesquiera del espacio de configuraciones, siempre sea posible pasar de uno a otro por una curva, esto es, hemos de exigir y así lo haremos de aquí en adelante, que el espacio de configuraciones sea arco-conexo.

Con esta idea en mente, consideremos el conjunto de todas las curvas en X, esto es, el conjunto de todas las aplicaciones continuas 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:

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

Pues bien, 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 de γ, esto es, una sección de γ:

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

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.

1. Complejidad topológica de algoritmos de movimientos

Recordemos que el conjunto de todas las curvas $X^{[0,1]}$ puede ser dotado de una estructura topológica compatible con las que se tienen en X y en el intervalo $[0,1]$: dos aplicaciones están próximas una de la otra si las imágenes de un compacto del intervalo por estas funciones son conjuntos próximos de X. Esta idea da lugar a la llamada *topología compacto-abierta* cuyos abiertos son uniones de intersecciones finitas de «abiertos tipo» de la forma (K,U), donde K es un compacto del intervalo $[0,1]$, U es un abierto de X, y que consta de todas las curvas en X que llevan el compacto K dentro de U.

Como quiera que esta topología hace que nuestra aplicación natural γ sea continua, tiene sentido preguntarse cuándo una determinada sección de γ será también continua, esto es, cuándo un algoritmo planificador del movimiento dependerá continuamente de los estados inicial y final que se tomen como input. La respuesta, fácilmente demostrable, como veremos, con conocimientos 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 $\alpha\colon X \times X \to X^{[0,1]}$ continuo si y sólo si el espacio X es contráctil.*

Para recordar que significa qué un espacio sea contráctil necesitamos hacer un breve repaso a nociones básicas de *teoría de homotopía* (deformación de espacios). Una *homotopía* entre dos aplicaciones continuas $f, g\colon X \to Y$ entre espacios arbitrarios no es más que una aplicación continua

$$H\colon X \times [0,1] \to Y, \quad H(x,0) = f(x), \quad H(x,1) = g(x), \quad \text{para todo } x \in X.$$

En efecto, puede pensarse en H como una deformación de f a g a lo largo del tiempo marcado por el intervalo $[0,1]$. Deformar aplicaciones continuas lleva de forma inmediata a formalizar el concepto de deformación entre espacios topológicos. Decimos que X e Y tienen el mismo *tipo de homotopía* (X puede deformarse a Y y viceversa), y denotamos por $X \simeq Y$, si existen aplicaciones continuas $f\colon X \to Y$ y $g\colon Y \to X$ de forma que $f \circ g \simeq 1_Y$ y $g \circ f \simeq 1_X$.

Un espacio X es *contráctil* si tiene el tipo de homotopía de (puede ser deformado a) un punto. Así, el espacio euclídeo de cualquier dimensión, cualquier entorno convexo de este espacio, en particular, los discos, son espacios contráctiles. Por el contrario, las esferas, el toro, un disco agujereado son espacios que no pueden ser deformados a un punto de forma continua. Nótese que, de las definiciones que acabamos de hacer, se desprende de forma evidente que un espacio X es contráctil si, dado un punto fijo $x_0 \in X$, 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 $x \in X$.

Estamos ya en condiciones de demostrar nuestro teorema.

Demostración. Supongamos en primer lugar que existe una sección continua $\sigma\colon X \times X \to X^{[0,1]}$ de γ. Definimos entonces la aplicación

$$H\colon X \times [0,1] \longrightarrow X, \quad H(x,t) = \sigma(x, x_0)(t).$$

Esta aplicación es continua por serlo σ y, además, $H(x,0) = \sigma(x,x_0)(0) = x$, $H(x,1) = \sigma(x,x_0)(1) = x_0$, por lo que X es, en efecto, contráctil.

Supongamos ahora que nuestro espacio de configuraciones es contráctil y tomemos $H\colon X \times [0,1] \longrightarrow X$ una contracción. Definimos entonces la aplicación $\sigma\colon X \times X \to X^{[0,1]}$ mediante

$$\sigma(x,y)(t) = \begin{cases} H(x,2t) & \text{si} \quad t \leq 1/2, \\ H(y, 2-2t) & \text{si} \quad t \geq 1/2. \end{cases}$$

Es elemental comprobar que σ es continua y, además, es una retracción de γ, pues $\sigma(x,y)(0) = H(x,0) = x$ y $\sigma(x,y)(1) = H(y,0) = y$. □

Este hecho, puramente topológico, puede ser aplicado de forma inmediata al diseño de algoritmos planificadores de movimientos. En cuanto un espacio de configuraciones resulte ser contráctil, una contracción cualquiera, que podrá ser formulada por ecuaciones más o menos complicadas, resulta ser de hecho un tal algoritmo listo para ser implementado en el robot. Sin embargo, los espacios de configuraciones, tal y como hemos visto en ejemplos anteriores, no son contráctiles en general. Por tanto, un algoritmo planificador de un determinado movimiento puede ser más complicado y contener más de una orden dada, como en el caso anterior, por una única aplicación continua (la contracción).

¿Cómo de complicado? ¿Cuál es el menor número de órdenes que ha de contener un algoritmo? O dicho de otro modo: ¿Cuál es el menor número de decisiones que un robot (o conjunto de robots) ha de tomar para moverse dentro de su espacio de configuraciones? Ese número, la medida estándar que actualmente mide esta complejidad, es lo que se conoce como *complejidad topológica* de un espacio de configuraciones X. Este índice fue introducido en 2003 por Michael Farber[4] [6], aunque podría recordar en cierta medida a una refinada versión, adaptada a la robótica de planificadores de movimientos, del invariante del mismo nombre introducido por Smale [15] y asociado a cualquier algoritmo. Tratemos de formalizar esta idea:

Cada vez que el robot se mueva en un entorno del espacio de configuraciones X de forma que todos los posibles estados iniciales y finales de tal entorno constituyan un trozo contráctil de $X \times X$, o más generalmente, un trozo de $X \times X$ donde exista una sección continua de γ, esa sección, como hemos visto, nos da de hecho un algoritmo planificador del movimiento, siempre y cuando, claro está, los estados inicial y final queden dentro de ese trozo. El índice al que hacíamos referencia es por tanto el menor número de tales trozos con que podemos recubrir todo $X \times X$. Formalmente, y definido exclusivamente en términos topológicos:

Definición 1. *La* complejidad topológica *de un espacio de configuraciones X, $TC(X)$, es el menor número n para el que existe un recubrimiento por abiertos $\{U_1, \ldots, U_n\}$ de $X \times X$, $\cup_{i=1}^{n} U_i = X \times X$, y de forma que para cada U_i, $i = 1, \ldots, n$, existe una sección continua de γ,*

$$\sigma_i \colon U_i \to X^{[0,1]}, \quad \gamma \circ \sigma_i = 1_{U_i}.$$

Si no es posible encontrar un tal recubrimiento diremos que la complejidad topológica de X es infinita, $TC(X) = \infty$.

La primera sorprendente propiedad, que el lector con elementales conocimientos de topología puede probar por sí mismo, es que este índice no cambia si deformamos continuamente nuestro espacio de configuraciones, esto es, la complejidad topológica es un invariante del tipo de homotopía, o expresado en otros términos,

$$\text{si} \quad X \simeq Y \quad \text{entonces} \quad TC(X) = TC(Y).$$

Así, para calcular el menor número de instrucciones que un algoritmo planificador de movimientos ha de contener, podemos deformar a nuestro antojo y conveniencia

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

el espacio de configuraciones siempre y cuando, como es natural, esa deformación sea continua. De esta forma, tal y como hemos visto en el teorema anterior, la complejidad topológica de cualquier espacio de configuraciones contráctil es siempre 1.

Consideremos ahora un simple robot formado por un brazo articulado con uno de los extremos fijos y con un sólo grado de libertad, esto es, el brazo gira siempre en un mismo plano. Claramente, su espacio de configuraciones es la circunferencia S^1, un espacio no contráctil, por lo que su complejidad topológica ha de ser necesariamente mayor que 1. Veamos que $TC(S^1) = 2$.

En efecto consideremos, por una parte, el abierto $U \subset S^1 \times S^1$ formado por todos los pares de puntos no antipodales, esto es:

$$U = \{(A, B) \in S^1 \times S^1,\ A \neq -B\}.$$

En este caso, dado un par de puntos $(A, B) \in U$, el arco más corto de A a B (arco bien definido pues tales puntos nos son antípodas) recorrido con velocidad constante y parametrizado por el intervalo $[0, 1]$, camino al que denotamos por $\sigma_U(A, B)\colon [0, 1] \to S^1$, define de hecho una sección continua de γ,

$$\sigma_U\colon U \to {S^1}^{[0,1]}.$$

El segundo abierto que tomamos está formado sencillamente por pares de puntos distintos de S^1,

$$V = \{(A, B) \in S^1 \times S^1,\ A \neq B\}.$$

Para ir de un punto a otro de uno de estos pares de forma continua y que, además, este camino dependa continuamente de tales pares, procedemos de la siguiente forma. Dado $(A, B) \in V$, como quiera que A y $-B$ no son antípodas, unimos en primer lugar A con $-B$, como antes, por el arco más corto, esta vez parametrizado por el intervalo $[0, 1/2]$. En el «medio segundo» restante unimos $-B$ con B mediante el arco

$$\big(\operatorname{sen}\pi(2t-1) - \cos\pi(2t-1)\big)B, \qquad t \in [1/2, 1]. \tag{1}$$

Este proceso nos da una curva continua

$$\sigma_V(A, B)\colon [0, 1] \to S^1, \quad \sigma_V(A, B)(0) = A, \quad \sigma_V(A, B)(1) = B,$$

de forma que

$$\sigma_V\colon V \to {S^1}^{[0,1]}$$

es también una aplicación continua y sección de γ. Como quiera que $U \cup V = S^1 \times S^1$, concluimos que, en efecto, $TC(S^1) = 2$.

De hecho este proceso puede generalizarse a cualquier S^n con n impar, considerado como el espacio de configuraciones de un brazo articulado con un extremo fijo que se mueve, esta vez, con n grados de libertad, esto es, en $\mathbb{R}^n$.

En efecto, basta seguir paso por paso el procedimiento anterior sustituyendo el arco de la fórmula (1) por este otro:

$$-\cos\pi(2t-1)B + \operatorname{sen}\pi(2t-1)Y(B), \qquad t \in [1/2, 1],$$

donde Y es un campo continuo de vectores tangentes de módulo constante igual a 1 sobre la esfera S^n. Recordemos que un campo de vectores tangentes sobre la esfera no es más que una aplicación continua

$$Y\colon S^n \to \mathbb{R}^{n+1}$$

de forma que para cada punto $p \in S^n$, los vectores p e $Y(p)$ sean perpendiculares, $Y(p) \perp p$. Un tal campo viene dado, por ejemplo, por esta sencilla fórmula (téngase en cuenta que n es impar):

$$Y(x_1, x_2, x_3, x_4, \ldots, x_n, x_{n+1}) = (-x_2, x_1, -x_4, x_3, \ldots, -x_{n+1}, x_n).$$

Concluimos así que, para cualquier n impar, $TC(S^n) = 2$.

Sin embargo, con herramientas propias de la topología algebraica, puede probarse que no existen campos vectoriales sobre las esferas de dimensión par que no se anulen en algún punto, por lo que el procedimiento anterior no puede usarse en este caso.

Por el contrario, usando también herramientas más o menos elementales de topología, se puede probar que no basta con dos instrucciones para diseñar un algoritmo planificador de movimiento para este sencillo brazo articulado si se mueve en $\mathbb{R}^{n+1}$ con n par, en particular en el espacio tridimensional. De hecho, en este caso se tiene que

$$TC(S^n) = 3 \quad \text{para} \quad n \quad \text{par}.$$

Sin que sirva de precedente, hagamos en este párrafo una breve concesión a los lectores que dispongan de estas herramientas topológicas y probemos esta aseveración: denotemos por $H^*(X)$ el álgebra de cohomología de un espacio X, y consideremos en $H^*(X) \otimes H^*(X)$ la subálgebra A generada por los elementos

$$\{\alpha \otimes 1 - 1 \otimes \alpha,\ \alpha \in H^*(X)\}.$$

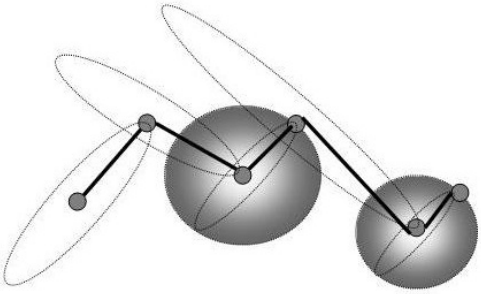
Se puede probar entonces que la complejidad topológica de X es mayor que el índice de nilpotencia de A, esto es, el mayor número de elementos de A cuyo producto resulta ser no nulo. Si nuestro espacio es S^n, con n par, la clase fundamental de volumen $\alpha \in H^n(S^n)$ proporciona un elemento $\alpha \otimes 1 - 1 \otimes \alpha \in A$ cuyo cuadrado

$$(\alpha \otimes 1 - 1 \otimes \alpha)^2 = -2(\alpha \otimes \alpha)$$

es no nulo y por tanto la complejidad topológica de S^n es al menos 3. Por otro lado, otra acotación, esta vez superior, que más adelante detallaremos en términos de otro invariante (véase la fórmula (2)), no nos deja margen y concluimos que, en efecto, $TC(S^n) = 3$ para n par.

Igualmente, métodos puramente topológicos nos permiten calcular la complejidad topológica de espacios de configuraciones que hemos estudiado en ejemplos anteriores [6, 7].

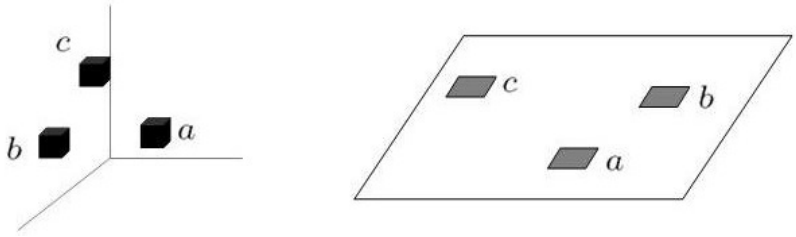
Consideremos en primer lugar un robot consistente en un brazo articulado de n elementos de los cuales n_1 brazos sólo tienen un grado de libertad, esto es, se mueven en un plano y los restantes n_2 brazos se mueven libremente en el espacio (figura 3). La complejidad topológica de este espacio de configuraciones es $n_1 + 2n_2 + 1$.



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

Figura 3.

Por otra parte, la complejidad topológica del espacio de configuraciones de n robots moviéndose sin colisionar en un plano (o más generalmente en cualquier espacio euclídeo de dimensión par) resulta ser $2n - 2$. Por contra, si estos n robots se mueven en un espacio euclídeo de dimensión impar, la complejidad topológica es $2n - 1$ (figura 4).

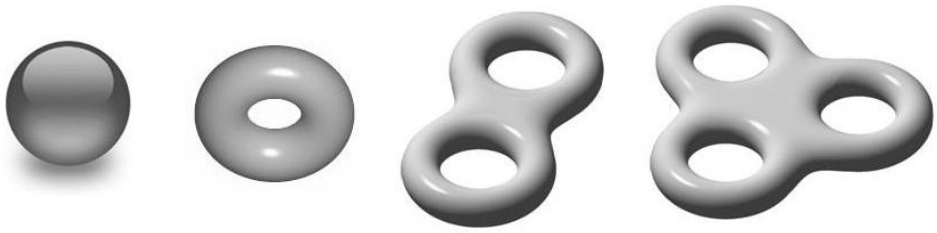


$TC = 2n - 1$ $\qquad$ $TC = 2n - 2$

Figura 4.

Terminemos esta colección de ejemplos indicando que si el espacio de configuraciones resulta ser (deformable a, o del tipo de homotopía de) una superficie orientable compacta de género 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 (figura 5).

Convencidos como estamos de que la complejidad topológica determina el número mínimo de instrucciones de un algoritmo planificador del movimiento en cuestión, cabe preguntarse no ya lo complejo que puede resultar implementar cada una de esas instrucciones en el robot o conjunto de robot que forman nuestro espacio de configuraciones, sino, yendo un paso más atrás, lo complejo que resulta de hecho determinar ese número mínimo de instrucciones. En otras palabras, ¿cuán difícil es calcular la complejidad topológica? Veamos un clarificador ejemplo que corrobora que tal pregunta no es en modo alguno banal.



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

Figura 5.

Consideremos un sencillo espacio de configuraciones dado por una barra que rota libremente alrededor de su centro con $n+1$ grados de libertad, esto es, en el espacio euclídeo $\mathbb{R}^{n+1}$. Como el lector puede fácilmente adivinar, tal espacio de configuraciones no es más que el espacio proyectivo de dimensión n al que denotaremos $\mathbb{R}P^n$. Pues bien, M. Farber, S. Tabachnickov y S. Yuzvinsky [8] han probado que la complejidad topológica de estos espacios, $TC(\mathbb{R}P^n)$, coincide con el más pequeño entero k para el que existe una aplicación *no singular*

$$f\colon \mathbb{R}^{n+1} \times \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^k.$$

Recordemos que una tal aplicación no es más que una aplicación continua que verifica dos sencillas propiedades:

(i) $f(\lambda x, \mu y) = \lambda\mu f(x, y)$, para cualesquiera $x, y \in \mathbb{R}^{n+1}$, $\lambda, \mu \in \mathbb{R}$.

(ii) Si $f(x, y) = 0$, entonces necesariamente $x = 0$ ó $y = 0$.

Por otra parte, hace ya más de 30 años, los distinguidos topólogos J. Adem, S. Gitler e I. James [2] demostraron que una tal aplicación no singular $f\colon \mathbb{R}^{n+1} \times \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^k$ existe si y sólo si el espacio proyectivo $\mathbb{R}P^n$ puede «sumergirse» en $\mathbb{R}^{k-1}$. Debemos ser cuidadosos y aclarar que por sumergir entendemos encontrar una *inmersión* $\mathbb{R}P^n \to \mathbb{R}^{k-1}$, esto es, una aplicación diferenciable y cuya diferencial en cada punto sea inyectiva (lo que la hace localmente inyectiva también). Nótese la sutil diferencia entre «sumergir» y ser *subvariedad regular*, es decir, exigirle a la inmersión que de hecho sea un encaje, un homeomorfismo sobre su imagen.

Para aclarar suficientemente estos conceptos pongamos un clarificador ejemplo: el plano proyectivo $\mathbb{R}P^2$ puede ser sumergido en el espacio euclídeo $\mathbb{R}^3$. La conocida *superficie de Boy*[5] es la imagen de una tal inmersión (figura 6).

Sin embargo $\mathbb{R}^4$ es el espacio euclídeo de menor dimensión donde el plano proyectivo se encaja como subvariedad regular. De hecho la aplicación

$$h\colon \mathbb{R}^3 \to \mathbb{R}^4, \quad h(x, y, z) = (xy, xz, y^2 - z^2, 2yz)$$

[5]Werner Boy (1879–1914) fue un matemático alemán, discípulo de D. Hilbert, quien le planteó el problema de la inmersión del plano proyectivo en el espacio euclídeo. En 1901, Boy describió la superficie que lleva su nombre a pesar de que la primera parametrización fue dada mucho después, en 1978 por Bernard Morin.

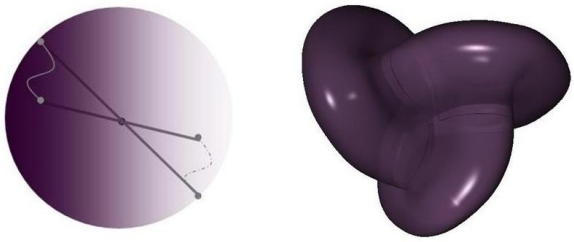


Figura 6. Superficie de Boy como espacio de configuraciones de un brazo articulado girando alrededor de su centro.

induce, restringiendo primero a S^2 y posteriormente haciendo el cociente identificando puntos antípodas, un «encaje» $\overline{h}\colon \mathbb{R}P^2 \hookrightarrow \mathbb{R}^4$.

Después de todo lo anterior, concluimos, por tanto, que la complejidad topológica de $\mathbb{R}P^n$ coincide con el menor entero k tal que $\mathbb{R}P^n$ puede «sumergirse» en $\mathbb{R}^{k-1}$. Pues bien, encontrar este número en general, en función de n, es aún un problema abierto en el que numerosos geómetras y topólogos han trabajado desde su formulación a principio de los años sesenta [4]. En la siguiente tabla [8] recogemos $TC(\mathbb{R}P^n)$, para $n \leq 23$, cuyo cálculo ha empleado (¡sorpresa!) profundos resultados en teoría de homotopía entre los que podemos destacar, por su importancia y aplicación en muy distintos contextos, la solución de Adams [1] al *problema de existencia de aplicaciones de invariante de Hopf* 1.

n	$TC(\mathbb{R}P^n)$	n	$TC(\mathbb{R}P^n)$	n	$TC(\mathbb{R}P^n)$
1	2	9	16	17	32
2	4	10	17	18	33
3	4	11	17	19	33
4	8	12	19	20	35
5	8	13	23	21	39
6	8	14	23	22	39
7	8	15	23	23	39
8	16	16	32		

Así pues, calcular la complejidad topológica de un sistema de configuraciones no parece una tarea sencilla en general, por lo que, antes de estudiar la dificultad de implementar en nuestro espacio de configuraciones un determinado algoritmo, debiéramos conocer lo complejo que resulta crear un tal algoritmo, conocer al menos, cuántas instrucciones ha de contener.

2. Complejidad Algorítmica y Complejidad Topológica

Para contestar a esta cuestión, debemos contar con nociones muy básicas de la rama de ciencias de la computación que versa sobre la *complejidad algorítmica*

(¡que no ha de confundirse con la topológica!). Para nuestro propósito nos basta con conocer de forma sucinta en qué términos se mide esta complejidad asociada a un problema dado. Para ello, al objeto de ser breve y con el deseo de ser mejor entendido por aquellos lectores que se acercan a estos temas por primera vez, y menos aburrido para los expertos, permítanme cierta vaguedad en lo que a continuación reseño.

Es habitual en Ciencias de la Computación entender un problema de decisión como una función $\Pi \to \{0, 1\}$ donde $\Pi = \{I\}$ es 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 éste 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. Se denomina *lenguaje* del problema al conjunto de todas las instancias con respuesta «Sí».

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 de 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 si I pertenece en efecto al lenguaje del problema, es decir si la respuesta para tal instancia 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 I no pertenece al lenguaje de nuestro problema.

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 determinadas características puede no ser fácil. Sin embargo, sí que lo es decidir si un determinado número de tales características es en efecto solución de la ecuación. Con este otro

ejemplo seguro que la mayoría de 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[6] [3, 13] 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 podrían no resolverse en tiempo polinomial.

En orden creciente de dificultad se encuentra la clase de problemas NP-completos. Para introducirlos deberíamos explicar brevemente el concepto de reducibilidad entre problemas. Un problema Π es *reducible polinomialmente* a otro Π' si existe una transformación $T\colon \Pi \to \Pi'$ y un polinomio p de forma que:

- Para cada instancia $I \in \Pi$ de longitud l, $T(I)$ es una instancia de Π' de longitud acotada por $p(l)$.
- Una instancia I pertenece al lenguaje de Π si y sólo si $T(I)$ pertenece al lenguaje de Π'.

Diremos entonces que un problema Π es NP-completo si pertenece a la clase NP y cualquier otro problema NP puede reducirse polinomialmente a él. Así, si supiéramos resolver un problema NP-completo podríamos también resolver, en el mismo rango de tiempo, cualquier otro problema NP.

Podría parecer, dada su condición de «problema insignia», capaz de resolver todo problema NP, que es complicado encontrar problemas NP-completos. Al contrario, hay cientos de ellos [9]. Quizás, uno de los ejemplos típicos de problema NP-completo es el del *coloreado de grafos*: 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 el grafo es finito, conexo (cualquier par de vértices está conectado a través de una sucesión de aristas), simple (no hay más de una arista conectando los dos mismos vértices) y sin lazos (ninguna arista conecta a un vértice consigo mismo); mostramos uno con estas características en la figura 7. Diremos que dos vértices $v, w \in V(G)$ son adyacentes si están conectados por una arista: $(v, w) \in A(G)$. Con esta notación, 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. Un k-coloreado de G es pues una aplicación

$$f\colon V(G) \longrightarrow \{1, 2, \ldots, k\} \quad \text{tal que} \quad f(v) \neq f(w) \quad \text{si} \quad (v, w) \in A(G).$$

[6] Stephen Cook, 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 *Siete Problemas del Milenio*, denominados así por el prestigioso *Clay Institute*.

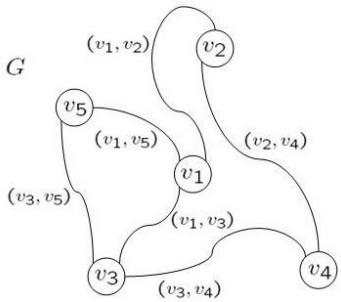


Figura 7.

Pues bien, si fijamos un entero $k \geq 3$, decidir si un grafo dado es k-coloreable es un problema *NP*-completo [3]. Es conveniente hacer constar que 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 $V(G) = \{v_1, \ldots, v_n\}$. Una codificación de este grafo viene dada por la llamada *matriz de adyacencia*, matriz cuadrada de orden n cuya entrada en el lugar (i, j) es 1 si $(v_i, v_j) \in A(G)$, y 0 en caso contrario.

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

Llegado a este punto, el lector puede, con toda razón, argumentar que el problema que nos ocupa, a saber, el cálculo de la complejidad topológica, no es un problema de decisión, sino, más bien, un problema de *optimización*, ¿cuál es el menor número de instrucciones que hemos de implementar en nuestro sistema de configuraciones?, de la misma forma que lo son estos otros: ¿Cuál es el número más pequeño de colores que necesitamos para colorear un grafo? o ¿cuál es el camino más corto para hacer un recorrido turístico entre determinadas ciudades? Pues bien, para aclarar este punto hemos de indicar que, dado un algoritmo que resuelva un determinado problema de decisión, podemos encontrar otro que resuelva, en el mismo rango de tiempo, el problema de optimización asociado. Pongamos un ejemplo: supongamos que tenemos un algoritmo $\mathcal{A}$ que resuelve el problema de decisión del coloreado de grafos cuyo input es un entero $k \geq 2$ y un grafo G, y cuyo output es 1 si G puede ser k-coloreado y 0 en caso contrario. Veamos que en la misma escala de tiempo podemos resolver el problema de optimización asociado cuyo input consiste en un grafo G y cuyo output es el llamado *número cromático*, el menor entero k para el que G puede ser k-coloreado. En efecto, tomemos un grafo G de n vértices, digamos $n = 80$. Preguntemos entonces al algoritmo $\mathcal{A}$ si tal grafo puede ser coloreado con 40 colores. Si es así, el número cromático de G está entre 1 y 40. En caso contrario tal número vive entre 40 y 80. En el primer supuesto preguntemos de nuevo a nuestro algoritmo $\mathcal{A}$ si G puede ser 20-coloreado... Continuemos de esta forma usando $\mathcal{A}$ en cada paso tomando como input la mitad del intervalo que, por el paso anterior,

sabemos que contiene el número cromático. Es evidente entonces que, al cabo de $\log_2 n$ pasos, habremos de hecho encontrado el número cromático.

Hemos encontrado pues un nuevo algoritmo $\mathcal{A}'$ cuyo input es el mismo (salvo el entero k, claro) que necesita el que ya teníamos, y que resuelve el problema de optimización asociado en tiempo $T \cdot \log_2 n$, siendo T el tiempo que necesita nuestro algoritmo inicial para resolver el problema en cada paso.

Como quiera que el factor $\log_2 n$ es siempre menor que n^2, un polinomio en n, número de vértices del grafo, el carácter polinomial o no de este nuevo algoritmo dependerá únicamente de que lo sea el algoritmo inicial.

Dicho de otro modo, si hay un algoritmo «rápido» que resuelva el problema de decisión, también existe un algoritmo igual de rápido que resuelve el problema de optimización asociado. Podemos por tanto y así lo haremos a partir de ahora, restringirnos a problemas de decisión.

Una vez realizada esta brevísima incursión por el mundo de la complejidad algorítmica podemos volver a la tarea que nos ocupa y preguntarnos, ahora de forma más precisa: ¿Cómo de complejo desde el punto de vista algorítmico es el cálculo de 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[7] [14]. Dado un espacio topológico X, la *categoría de Lusternik-Schnirelmann* o *LS-categoría* de X, a la que denotaremos por $\mathrm{cat}(X)$, es el menor entero n para el que existe un recubrimiento de X formado por n-abiertos y cada uno de ellos contráctiles en X. Es necesario hacer notar aquí la sutil diferencia entre ser A un subespacio contráctil de X (concepto que ya hemos definido anteriormente) y ser A *subespacio contráctil en* X. Esta última noción, algo más general que la contractibilidad «a secas», se define requiriendo la existencia de una contracción $H\colon A \times I \to X$ que, como siempre, satisfaga $H(a,0) = a$ y $H(a,1) = x_0$ con x_0 punto de X. Esto es, el punto al que se deforma A (o la imagen de la deformación en un determinado instante) puede estar fuera del propio A.

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. Este invariante de tan sencilla definición ha resultado ser de valiosa utilidad en distintas ramas de la 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, dado un determinado espacio de configuraciones X, la complejidad topológica de X y la LS-categoría de X quedan relacionadas por la siguiente fórmula [6]:

$$\mathrm{cat}(X) \leq TC(X) \leq 2\,\mathrm{cat}(X) - 1. \tag{2}$$

[7] Lazar Lusternik (1899–1981), matemático ruso, desarrolló su trabajo 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, encaminados fundamentalmente a demostrar la conjetura de Goldbach.

Esta desigualdad, junto con otras acotaciones de tipo cohomológico como la reseñada cuando obteníamos la complejidad topológica de las esferas de dimensión par, es de hecho herramienta clave para el cálculo de la complejidad topológica de numerosos espacios de configuraciones. Además, a partir de esta fórmula es inmediato deducir que es tan difícil, desde el punto de vista algorítmico, calcular la complejidad topológica de un espacio de configuraciones como su LS-categoría.

Por otra parte, dado un grafo G de vértices $V(G) = \{v_1, \ldots, v_n\}$ y aristas $A(G) = \{(v_r, v_s)\}_{(r,s)\in J}$ y un entero $k \geq 3$, es posible construir un espacio topológico $X_{G,k}$ en cuyos grupos de homotopía puede leerse la estructura del grafo, sus vértices y sus aristas [11].

Recordemos que, dado un espacio topológico X, sus grupos de homotopía, $\pi_n(X)$, uno por cada entero $n \geq 1$, son grupos, conmutativos si $n \geq 2$, cuya definición refleja gran parte de la geometría del espacio: cada $\pi_n(X)$ está formado por clases de homotopía de aplicaciones continuas $S^n \to X$ de la esfera n-dimensional en nuestro espacio. La multiplicación es asimismo muy «geométrica»: el producto de dos elementos de $\pi_n(X)$ representados por $f, g\colon S^n \to X$ es el representado por la aplicación $f \cdot g\colon S^n \to X$ obtenida colapsando, en primer lugar, el ecuador de la esfera a un punto (con lo que se obtiene a su vez dos esferas de la misma dimensión unidas por este punto) y aplicando f y g en cada una de las nuevas esferas (figura 8).

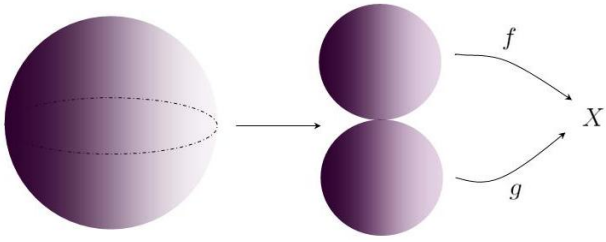


Figura 8. El producto en los grupos de homotopía.

Lamentablemente, para describir explícitamente y con toda precisión el espacio $X_{G,k}$ necesitaríamos introducir nociones nada elementales de teoría de homotopía que se escapan al carácter divulgativo de este texto. Sin embargo, sí que podemos decir que el rango de sus grupos de homotopía coincide con el número de vértices y aristas del grafo. Este hecho es clave pues también nos permite poder explicar de forma sencilla cómo este espacio $X_{G,k}$ puede codificarse, «discretizarse» si me permiten, en función del grafo, para su posterior implementación en un algoritmo dado: su codificación viene dada por un álgebra de polinomios un tanto especial junto con una «diferencial» definida en la misma. Esta álgebra está generada, sobre los números racionales, por tantas indeterminadas como vértices y aristas tenga el grafo G, esto es

$$\mathbb{Q}[x_i, y_{r,s}], \quad i = 1, \ldots, n, \quad (r,s) \in J.$$

Para describir la diferencial d en esta álgebra de polinomios, basta que lo hagamos sobre las indeterminadas y extenderla por la regla de la derivación al resto. Pues

bien, definimos

$$d(x_i) = 0, \quad i = 1, \dots, n,$$
$$d(y_{r,s}) = \sum_{m=1}^{k} x_r^{k-m} x_s^{m-1}, \quad (r,s) \in J.$$

Es entonces un sencillo ejercicio comprobar que, para codificar esta álgebra con la correspondiente diferencial, se requiere una longitud acotada polinomialmente por el número de vértices del grafo G, que, como hemos recalcado con anterioridad, requiere por tanto la misma longitud que para codificar el propio grafo G.

Pues bien, en estas condiciones se puede demostrar la siguiente propiedad fundamental [11]:

Teorema 2. *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 abiertos contráctiles en el espacio.*

A la vista de este teorema, y dado que la codificación del espacio $X_{G,k}$ requiere una longitud acotada por un polinomio evaluado en la longitud de la codificación del propio grafo G (como hemos recalcado con anterioridad esta codificación depende polinomialmente del número de vértices), podemos concluir que el problema del k-coloreado de grafos se puede reducir polinomialmente al problema de determinar si la LS-categoría de un espacio es finita o no.

Como quiera que el k-coloreado de grafos es un problema NP-completo, deducimos que el cálculo de la LS-categoría es un problema NP-difícil.

Ahora bien, por la fórmula (2), la LS-categoría de $X_{G,k}$ será infinita si y sólo si lo es su complejidad topológica. Dicho de otro modo:

Teorema 3. *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 es 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 4. *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. Lo es también simplemente 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, por una parte refrenda la profundidad de los numerosos resultados que conciernen el diseño y complejidad de algoritmos planificadores de movimientos obtenidos por otros tantos y distinguidos expertos en robótica (John Canny, John Hopcroft, Daniel E. Koditschek, John Reif, Micha Sharir,...). Por otra

parte, ratifica el éxito del rápido progreso en la utilización de métodos topológicos en este campo.

Aunque, si me dejan serles sincero, lo que más apasionante me resulta de lo que les acabo de contar es la puerta que se abre en el camino inverso. En lo que concierne a, si me permiten el atrevimiento, la «Topología Robótica». Esto es, a la aplicación de métodos y resultados propios de la robótica y complejidad en la resolución de problemas de teoría de homotopía en los que estoy interesado. Sin embargo, como diría D. Javier Krahe, eso es otra canción.

Referencias

[1] Frank Adams, *On the non-existence of elements of Hopf invariant one*, Annals of Math., 72(2) (1960), 20–104.

[2] José Adem, Samuel Gitler y Ioan James, *On axial maps of a certain type*, Boletín de la Sociedad Matemática Mexicana, 17(2) (1972), 59–62.

[3] Stephen Cook, *The complexity of theorem proving procedures, Proc.*, Third Annual ACM Symposium on the Theory of Computing, ACM, New York, (1971), 151–158.

[4] Don Davis, *Immersions of projective spaces: a historical survey*, *Algebraic Topology, Oaxtepec 1991*, Contemporary Math., 146 (1993), 31–37.

[5] Edwuard Fadell y Sufian Husseini, *Geometry and Topology of Configuration Spaces*, Springer Monographs in Mathematics, Springer-Verlag, Berlin, 2001.

[6] Michael Farber, *Topological complexity of motion planning*, Discrete and Computational Geometry, 29(2) (2003), 211–221.

[7] Michael Farber y Mark Grant, *Topological complexity of configuration spaces*, preprint, 2008.

[8] Michael Farber, Sergey Tabachnikov y Sergey Yuzvinsky, *Topological robotics: motion planning in projective spaces*, International Math. Research Notices, 34 (2003), 1853–1870.

[9] Michael Garey y David Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Company, New York, 1979.

[10] Jean Claude Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[11] Luis Lechuga y Aniceto Murillo, *Complexity in rational homotopy*, Topology, 39(1) (2000), 89–95.

[12] Luis Lechuga y Aniceto Murillo, *Topological complexity of formal spaces*, *Topology and Robotics*, Contemporary Math., 438 (2008), 105–114.

[13] Leonid Levin, *Universal'nye perebornye zadachi*, Problemy Peredachi Informatsii, 9(3) (1973), 265–266.

[14] Lazar Lusternik y Lev Schnirelmann, *Méthodes Topologiques dans les Problèmes variationnels*, Hermann, Paris, 1934.

[15] Stephen Smale, *On the topology of algorithms, I*, Journal of Complexity, 3 (1987), 81–89.

[16] Herbert Wilf, *Algorithms and Complexity*, A K Peters, Wellesley, MA, 2002.

Aniceto Murillo, Departamento de Álgebra, Geometría y Topología, Universidad de Málaga, Ap. 59, 29080 Málaga

Correo electrónico: aniceto@uma.es

Página web: http://agt.cie.uma.es/~aniceto/