Geometría Computacional para redes de sensores

por

Belén Palop de Río, Universidad de Valladolid

1. Redes de sensores

Como tantos otros avances tecnológicos, las Redes de Sensores comenzaron a ser investigadas desde ámbitos militares. Alrededor de 1980 comenzó a extenderse el interés por ellas desde ámbitos no militares, coincidiendo en el tiempo con la aparición de la Geometría Computacional. Cada vez se publican más artículos en el contexto de la investigación en Redes de Sensores con un alto contenido de Geometría Computacional. Por una parte, la Geometría Computacional proporciona soluciones; por la otra, las Redes de Sensores proporcionan nuevos retos geométricos. No parece estar recibiendo en este momento en España suficiente atención por parte de ambas comunidades esta atractiva y fructífera relación, por lo que nos proponemos en este breve artículo discutir algunas de estas colaboraciones.

Según el diccionario de la Lengua Española de la RAE, un sensor es un *Dispositivo que detecta una determinada acción externa, temperatura, presión, etc., y la transmite adecuadamente.* Para realizar su misión, un sensor generalmente consiste en una CPU, una memoria, un transmisor, una batería, y un elemento *sensible* (en el sentido de *que siente*). Actualmente se utilizan en muchísimos contextos los sensores, abarcando desde los sensores de lluvia o velocidad hasta los termómetros marinos o los sensores de concentración de sal. Dependiendo de los datos que se quieran recoger, se buscará un mayor poder de CPU, de transmisión, o de almacenamiento.

El 13 de Febrero de 2009, la revista Nature publicó el artículo titulado *Tracking Long-Distance Songbird Migration by Using Geolocators* [6]. El estudio se llevó a

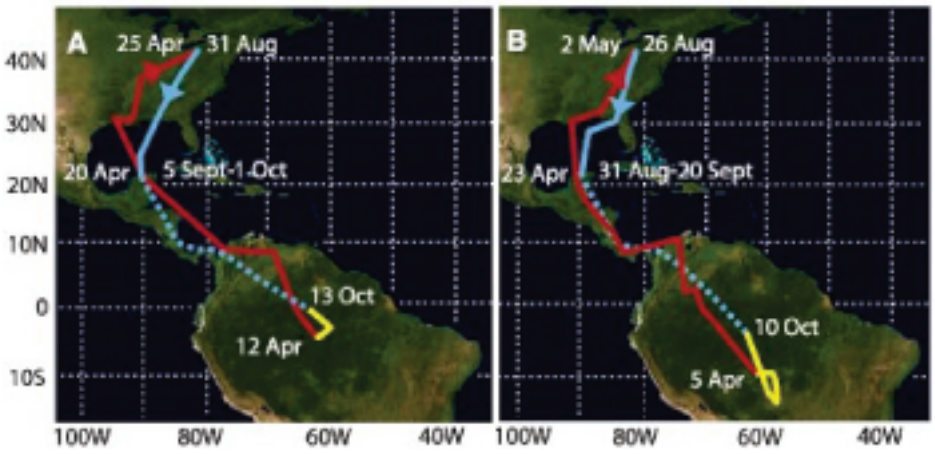

Figura 1: Recorridos migratorios recuperados de los sensores en el estudio [6].

cabo montando a la espalda de 34 pájaros unas *mochilas* con sensores de geolocalización. Estos sensores de forma cilíndrica miden unos 2cm de largo y tienen un diámetro de menos de 5mm. El peso, incluyendo la batería de 1,5 años de duración, es de 1,2 gramos. Cada minuto, el sensor mide la luz ambiental y, cada 10 minutos, almacena el valor máximo obtenido. De esta manera, se estima cuál es la ubicación geográfica del sensor con cierta precisión. El tamaño de la memoria es de 128Kb para programas y 512Kb para datos. Era la primera vez que se había conseguido seguir tan de cerca el vuelo de pájaros pequeños durante una migración. Hasta la elaboración de este estudio, se suponía que estos pequeños pájaros eran capaces de volar unos 150 km por día. En este estudio vieron que algunos pájaros habían recorrido 2.500 km en tan sólo 5 días y que una hembra había realizado un promedio de 833 km por día de vuelo durante 13 días consecutivos (de los que 9 dedicó a volar y 4 a descansar). Lamentablemente, de las 34 mochilas sólo fueron recuperadas 7. Este hecho, añade una dimensión más al problema de encontrar el sensor más adecuado para cada contexto: su precio.

Seguramente, a los investigadores que realizaron este experimento, les habría gustado equipar un número mayor de pájaros con los sensores para hacer un muestreo más amplio. Además, habrían querido que la geolocalización hubiera sido a través de GPS (mucho más exacta) y, puestos a pedir, les habría gustado disponer de todos los datos en tiempo real y sin tener que descartar 9 posiciones de cada 10 captadas. Los motivos que les llevaron a no hacer el experimento tal y como ellos querrían son de varios tipos:

- Económicos: Sabemos que, hoy en día, el precio de un dispositivo de GPS ronda los 100 euros, que multiplicados por el número de pájaros y descontando aquellos que pudieran perderse, eleva los costes del experimento sustancial-

mente. (Aunque el GPS no se puede perder por definición, contemos con que un pájaro puede decidir quedarse a vivir en el Amazonas; o que es devorado por un águila; o que muere sobrevolando el océano; o que se le agota la batería...)

- Duración de la batería: Un dispositivo de GPS consume muchísima energía por las transmisiones constantes de información. Si tenemos en cuenta que comunicar 1 bit cuesta tanto como realizar 1.000 operaciones de CPU, es evidente que disminuir las comunicaciones repercute enormemente en la duración de la batería.

- Memoria: Un sensor estándar tiene unos 512K de memoria dedicada a datos. Si se almacenan en la memoria las mediciones realizadas por el sensor en cada minuto, la memoria se llenaría en aproximadamente 100 días de vuelo. Si el ciclo migratorio es de un año, es evidente que se necesita realizar algún tipo de selección de los datos que se van a almacenar y recuperar posteriormente.

- CPU: La manera en que se descartaban los datos en el experimento era muy sencilla, guardando el máximo de cada diez. Una mayor capacidad de cómputo de la CPU, podría permitir realizar una mejor localización. Por precio, duración de la batería y limitaciones de memoria, se procura que la CPU realice tareas elementales siempre que sea posible.

- Peso: Todos hemos tenido algún dispositivo de GPS en la mano. Y una pila. Y un dispositivo de memoria (por ejemplo, flash). Creo que no es necesario explicar por qué un canario no podría transportar todo este equipaje a su espalda durante un vuelo.

A partir de este ejemplo, empezamos a vislumbrar que una red de sensores no tiene mucho que ver con una red de potentes ordenadores que colaboran entre sí para realizar una tarea. Por el contrario, una red de sensores es mucho más parecido a un grupo de hormigas de escasa inteligencia y con un sistema muy elemental de comunicación. El Problema (*nótense las mayúsculas*) en redes de sensores será cómo hacer que esas pequeñas inteligencias que toman decisiones locales con datos locales, consigan alcanzar un objetivo global. Y este problema se planteará en todos los aspectos de las redes de sensores, desde la detección de la topología de la red, hasta el encaminamiento de los mensajes o la síntesis de información.

Para modelizar el comportamiento de los sensores se suele suponer que un sensor tiene un determinado radio de *sensibilidad*, es decir, que el lugar geométrico de los puntos de los que capta información un sensor se modeliza con un círculo de radio r_s. Por otra parte, e independiente del anterior, un sensor tiene un radio

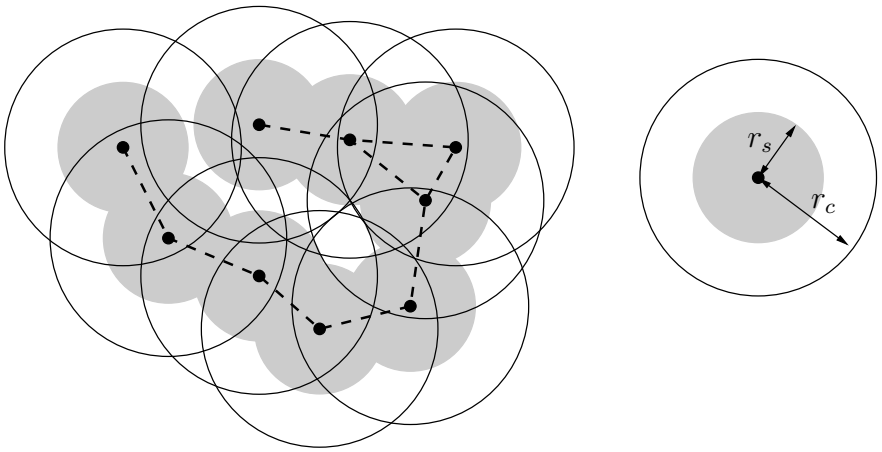

Figura 2: Ejemplo de una red de sensores en la que cada sensor se conecta (línea discontinua) con todos aquellos que están dentro de su radio de comunicación. La zona sombreada representa la cobertura de la red.

de *comunicación* que determina el círculo de radio r_c de los puntos con los que puede comunicarse. Si bien es obvio que tanto la sensibilidad como la comunicación se pueden ver afectadas por agentes externos (obstáculos, interferencias, etc), es aceptable tomar el peor caso para el radio y modelizar las redes de esta forma (ver Figura 2).

Veremos a continuación algunos de los problemas y las soluciones que se plantean en el contexto de las redes de sensores en las que aparece de una manera u otra la Geometría Computacional.

2. Localización

En adelante, cuando hablemos de redes de sensores, no nos referiremos a los 10 o 20 sensores dispuestos en ubicaciones preestablecidas como podrían ser los radares de velocidad a lo largo de una red de carreteras. Los problemas que realmente nos interesa resolver incluirán centenares o miles de sensores que se *siembran* en una extensión de terreno como puede ser el interior de un volcán, el fondo del mar, o las ruinas de una ciudad tras un terremoto. Como hemos visto en la introducción, es razonable asumir que no disponen de GPS, que deben reducir al máximo las comunicaciones para ahorrar batería y que, además, algunos podrán desaparecer de la red cuando se les acabe la batería, o se rompan, o queden demasiado alejados del resto.

Uno de los primeros problemas que se deben resolver en la mayoría de las redes

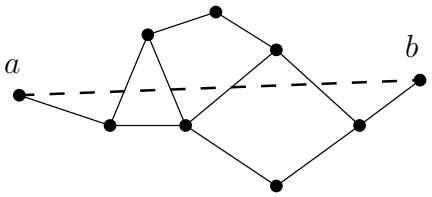

Figura 3: El conteo de saltos entre los sensores a y b es de 5. La distancia euclídea entre ellos (la longitud del segmento discontinuo) es necesariamente menor que 5.

de sensores es saber cuál es la posición de los sensores, ya sea relativa respecto al resto de sensores vecinos, ya sea absoluta. Esta información será necesaria tanto para los propios sensores, que deben por ejemplo saber a qué vecino enviar un mensaje cuyo destino conocen, como para el gestor de la red, a quien no le bastará con saber que la temperatura captada por un sensor es de 48 grados, sino dónde se ha captado esa información.

Las dos medidas de distancia más habituales entre sensores son el conteo de saltos (*hop-count*) y la distancia euclídea. El conteo de saltos indica el menor número de conexiones que hay en la ruta que menos saltos da entre dos sensores. Interpretando la red como un grafo, lo que estamos contando es el número de aristas en el camino más corto entre dos vértices. El conteo de saltos también sirve como cota superior de la distancia euclídea cuando no se necesita demasiada precisión. Obsérvese que, si todos los nodos se conectan con aquellos que están a distancia, a lo más 1, un conteo de k saltos indica que esos nodos están, a lo más, a distancia euclídea k (ver Figura 3). Veremos más adelante que en muchas ocasiones se utiliza esta medida como parámetro en algoritmos de encaminamiento, incluso cuando las localizaciones de los sensores son absolutas. Por ejemplo, podríamos encontrar un algoritmo en el que un sensor envía un mensaje a todos sus vecinos a distancia de 3 saltos.

La otra medida usual, la distancia euclídea, puede arrojar posiciones de los sensores relativas o absolutas dependiendo de si se dispone o no de sensores dotados de un dispositivo de GPS. Generalmente, no todos los sensores disponen de un dispositivo de GPS, pero podríamos pensar en la posibilidad de que una proporción de ellos lo tuviera. Existe un algoritmo iterativo (conocido como *iterative multilateration*) que asigna a cada sensor su ubicación absoluta partiendo de las ubicaciones de los sensores que disponen de GPS. Llamaremos anclas (*anchors*) a aquellos sensores que son capaces de proporcionar su ubicación de manera exacta. El resto de los sensores podrá hacer mediciones de ángulos, de retardos en la señal, de atenuación de la misma, etc. de cara a medir su distancia relativa a, al menos,

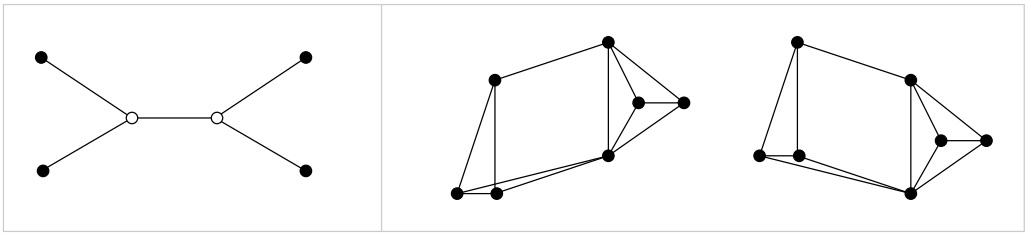

Figura 4:(Izquierda) Ninguno de los dos sensores blancos (no localizados) puede calcular su ubicación hasta que el otro lo haga. (Derecha) Las longitudes de las aristas en ambas redes son las mismas.

tres anclas para conseguir calcular su ubicación. Una vez un sensor conoce su posición, puede servir de ancla para otros. Este proceso iterativo se puede repetir hasta que todos los sensores conozcan su ubicación. Hay un trabajo muy interesante de Savvides y otros [5] en el que estudian cuál es el número de anclas necesarias para que este algoritmo termine dependiendo de la densidad de la red. Sólo para hacernos una idea de cómo de útil es este proceso, si distribuimos 200 sensores uniformemente en un cuadrado de 10×10 y el alcance de los sensores es de una unidad, resulta que necesitaríamos unas 50 o 60 anclas para asegurarnos (aunque sólo al 95 %) de que alguno de los 200 nodos ve a tres anclas y, por lo tanto, puede localizarse. Es obvio que si aumenta el alcance de los nodos o la densidad de la red, el porcentaje de nodos que necesitan GPS disminuye. Aunque salvásemos esta dificultad, aún nos quedaría el problema de que las mediciones de GPS no son totalmente exactas, que la triangulación a partir de tres anclas también introduce errores, y que este proceso iterativo está propagando estos errores al convertir en anclas sensores con datos poco fiables. Añadamos a los dos problemas anteriores uno bastante grave: el algoritmo se puede bloquear en situaciones más o menos simples si no garantizamos que el 100 % de los nodos ve (o verá según el proceso avance) tres anclas, como se puede ver en la Figura 4 (izquierda).

Por último, veamos cómo la acumulación de errores en las ubicaciones de los nodos puede llevar a realizaciones muy diferentes de una red. En la Figura 4 (derecha) podemos ver cómo los 7 nodos que aparecen tienen todos, al menos, tres vecinos. Supongamos que todos ellos han calculado su ubicación a partir de otros y que hay un cierto error acumulado en las mediciones del que cada nodo es *consciente*, por lo que sabe que su ubicación es imprecisa. Vemos en la figura que, aún respetando las distancias que cada nodo ha calculado respecto a sus vecinos, podemos obtener dos redes diferentes. Permítame el lector dejar aquí sólamente un puntero a la relación entre la localización de sensores y la teoría de la rigidez (*Rigidity Theory*), que permite plantearse problemas de ambigüedad en

la localización en redes poco densas (véase en [3] el primer artículo en el que se estudió esta relación).

¿Implican todos estos problemas que la localización no pueda ser correctamente resuelta? El número de redes de sensores que no disponen de GPS que se encuentran en funcionamiento en la actualidad es suficientemente alto como para que ya intuyamos que la respuesta es negativa. En la práctica, las redes son muy densas y los casos en los que estamos viendo en los que la localización puede fallar comparten un mismo perfil: la red no es suficientemente densa como para que entre todos los vecinos de un nodo haya tres anclas o para que no se generen redes rígidas. La gran mayoría de los estudios experimentales se realizan sobre redes simuladas, con una densidad alta (digamos alrededor de 6-10 vecinos por nodo) y con unas condiciones que podríamos considerar favorables.

3. Encaminamiento

La programación de las redes de sensores exige un tratamiento no centralizado de la información. Los algoritmos habituales de manejo de grafos resultan inútiles en las redes de sensores en la mayoría de las ocasiones porque no cumplen este requisito. En una red de sensores no habrá un nodo de mayor capacidad de cómputo, memoria o transmisión que resuelva los problemas. Por el contrario, cada nodo dispondrá de sus (pocos) recursos e, idealmente, un mismo programa en ejecución. Resultan en estas condiciones especialmente atractivas las técnicas de los algoritmos voraces o avariciosos, que buscan la solución a un problema realizando exploraciones locales y tomando la mejor opción a partir de esa información parcial. Es obvio que no todos los problemas se pueden resolver así y que tomar decisiones basándose siempre en el *corto plazo*, no lleva necesariamente al objetivo buscado en el *largo plazo*. Veremos en esta sección cómo las técnicas avariciosas permiten atacar el problema del encaminamiento de mensajes en una red de sensores con cierto éxito y cómo podemos delimitar en qué circunstancias ese éxito estará garantizado.

Una vez establecida la ubicación de cada nodo, es necesario para que la red funcione que los nodos puedan comunicarse entre sí. Es sencillo hacer que cada cual hable con sus vecinos (a distancia de un salto). La dificultad radica en saber con cuál de los vecinos hablar para que un determinado mensaje llegue a un destinatario del que únicamente conocemos su ubicación.

Los protocolos *proactivos* mantienen tablas de ruta en cada nodo. Por una parte, debemos recordar que entre otras restricciones, los sensores tienen poca capacidad de memoria, por lo que estos protocolos limitan el tamaño de la red. Además, cuando las redes son muy dinámicas o los sensores son móviles, es imposible mantener las tablas de ruta en cada uno de los nodos.

Por otra parte, los protocolos *reactivos* construyen las rutas bajo demanda.

Por ejemplo, el DSR (Dynamic Source Routing), se basa en almacenar algunas rutas ya calculadas en los nodos intermedios a medida que se van generando. Cada envío de un mensaje deja un rastro en los nodos intermedios. Cada nuevo envío intenta reutilizar esa información a la vez que actualiza las tablas. Si bien el comportamiento es bueno en redes estables, en redes móviles o dinámicas es menos eficiente, así como cuando los caminos recorridos por los mensajes son muy largos (los nodos intermedios reciben una carga que depende de la distancia en saltos entre el origen y el destino).

Otro ejemplo de protocolo reactivo que ha recibido mucha atención es el encaminamiento geográfico (*Geographical Routing*). Las hipótesis en que se basa son:

- Cada nodo conoce su propia ubicación.
- El emisor conoce la ubicación del destinatario.
- Los nodos conocen las ubicaciones de sus *vecinos* (a distancia de saltos 1).
- Cada paquete puede guardar una cantidad constante de información sobre la ruta (dicho de otro modo, un paquete no puede guardar información sobre toda la ruta recorrida, por lo que no puede saber si un determinado nodo ya fue visitado).
- El grafo subyacente es el grafo de discos unidad, es decir, que cada nodo se conecta con todos aquellos que están a su alrededor dentro de un disco de radio uno.

A partir de estas premisas, se establece el siguiente algoritmo avaricioso en todos los sensores de la red:

> Cuando llegue un paquete del que no eres el destinatario, envíalo a tu vecino a distancia de saltos 1 que más lo haga progresar al destino.

Al medir la progresión, tenemos numerosas opciones (ver Figura 5, izquierda), de las cuales las más extendidas son reenviar al vecino que más cerca esté del destino según la distancia euclídea *distance routing*, o al que menos se desvíe angularmente de la dirección marcada (conocido como *compass routing*).

Como se puede ver, el algoritmo avaricioso es sencillo de programar, de ejecutar para cada nodo, y sólo realiza una transmisión por arista. O al menos eso parece. Los algoritmos avariciosos tienen siempre esta pequeña trampa: es fácil programarlos y son muy intuitivos. La dificultad está en probar que realmente funcionan, o qué condiciones tienen que darse para que funcionen. En particular, la estrategia del *compass routing* puede entrar en un ciclo en el que el mensaje es continuamente reenviado y nunca llega a su destino (si se construye el grafo convenientemente)

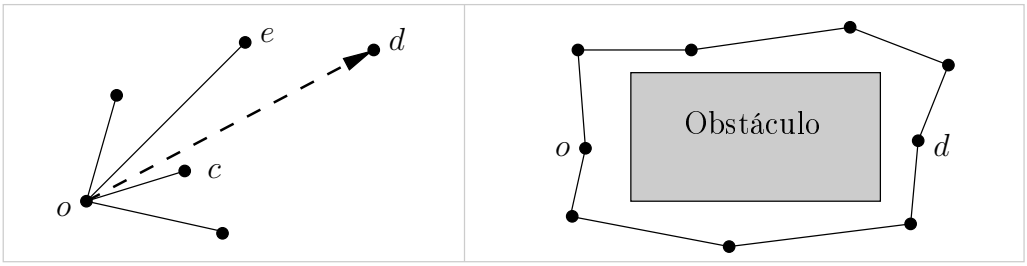

Figura 5: (Izquierda) Para enviar un mensaje de o a d, se optará por enviarlo al nodo e si se usara *distance routing*, mientras que con *compass routing* se enviaría a c. (Derecha) El *Geographical Routing* puede quedarse bloqueado en el nodo o usando *distance routing* si el destinatario es, por ejemplo, el nodo d.

como se demostró en [4]. En ese mismo artículo, se establece que, si el grafo subyacente es la triangulación de Delaunay [1] de los sensores, entoces siempre está garantizada la entrega del mensaje. En la Figura 5 (derecha) podemos ver que la estrategia del *distance routing* también falla y un mensaje puede quedar bloqueado en un nodo tal que ningún vecino permite progresar al mensaje si la densidad de la red no es suficientemente alta.

Quizás sea en este contexto donde la Geometría Computacional ha tenido uno de sus mayores logros en la interacción con las redes de sensores. En 2001, Bose, Morin, Stojmenovic y Urrutia [2] publicaron en la revista Wireless Networks un artículo titulado *Routing with guaranteed delivery in ad hoc wireless networks*. En Marzo de 2010, el número de citas a este artículo arrojado por *Google Scholar* es de 1071. Para poner en contexto esta cifra, podemos tomar como referencia que la revista científica de más impacto, *Nature*, tiene un promedio de 31 citas por artículo, mientras que la revista en la que fue publicado este trabajo tiene un promedio de 2 citas por artículo. Estamos pues hablando de un *disco de platino* en investigación sobre redes de sensores. ¿Y dónde radica el secreto de este éxito? Como siempre en estos casos, la pócima no tiene un único ingrediente ni es fácil saber cuál de ellos es el esencial. Yo lo resumiría en que estamos hablando de un trabajo entendible desde ambas disciplinas, con el formalismo necesario para poder demostrar teoremas y no sólo establecer teorías, teniendo en cuenta el modelo subyacente sobre el que se resuelve el problema geométrico, y con simulaciones del comportamiento de los algoritmos propuestos.

Bibliografía

[1] M. Abellanas, *Envolvente convexa, triangulación de Delaunay, diagrama de Voronoi: tres estructuras geométricas en una, con muchas aplicaciones*, Un paseo por la Geometría, UPV-EHU, 2007.
divulgamat.ehu.es/weborriak/TestuakOnLine/06-07/PG-06-07-Abellanas.pdf
[2] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, *Routing with guaranteed delivery in ad hoc wireless networks*, Wireless Networks 7 (6), 609-616, 2001.
[3] T. Eren, D. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, P. N. Belhumeur, *Rigidity, Computation, and Randomization in Network Localization*, INFOCOM, 2004.
[4] E. Kranakis, H. Singh, J. Urrutia, *Compass Routing on Geometric Networks*, Proc. 11 th Canadian Conference on Computational Geometry, 51-54, 1990.
[5] A. Savvides, C. C. Han, M. B. Srivastava, *Dynamic Fine Grained Localization in Ad-Hoc Sensor Networks*, Proceedings of the 5th International Conference on Mobile Computing and Networking (MobiCom'01, Rome), 166-179, 2001.
[6] B. J. M. Stutchbury, S. A. Tarof, T. Done, E. Gow, P. M. Kramer, J. Tautin, J. W. Fox, V. Afanasyev, *Tracking Long-Distance Songbird Migration by Using Geolocators*, Science 323 (5916), 2009.

Belén Palop del Río
Universidad de Valladolid
Escuela Universitaria Politécnica
Departamento de Informática
Francisco Mendizábal s/n, 47014 (Valladolid)
e-mail: ***b.palop@infor.uva.es***