biomimetiks
innovative solutions

Desarrollo de un algoritmo biomimético para el proyecto OPEN MOVE ("Plataforma software integradora de datos de transporte para la planificación de movilidad multimodal")

Marzo 2013

Contenido

1. RESUMEN 3

2. ANÁLISIS DEL PROBLEMA 4

3. ALTERNATIVAS BIOMIMÉTICAS 5

3.1 Hormigas (transmitir información con mínimas conexiones) 5

3.2 Moho mucilaginoso (encontrar la ruta más corta) 5

3.3 Desarrollo embrionario en vertebrados (coordinar redes distribuidas) 6

3.4 El sistema nervioso de la mosca de la fruta (coordinar redes distribuidas) 7

4. EVALUACIÓN Y DISCUSIÓN 9

5. IMPLEMENTACIÓN 10

5.1 Posibles alternativas en la implementación 12

5.1.1 Problema en la unión directa del servidor central al MIS 12

5.1.2 D variable 13

5.1.3 Definir el primer servidor a preguntar del MIS 13

1. RESUMEN

El proyecto “Open Move” consiste en desarrollar una plataforma software integradora de datos de transporte para la planificación de movilidad multimodal.

El objetivo es ofrecer información de movilidad multimodal al ciudadano. Para ello, se utilizará una plataforma que permitirá disponer de toda la información necesaria sobre los medios de transporte públicos que se deben utilizar para realizar un recorrido determinado adaptado al usuario.

Las alternativas existentes para el cálculo de rutas son escasas en cuanto a datos sobre el transporte público además de estar limitadas geográficamente, ya que los datos se encuentran cerrados en diferentes servidores (o en un único servidor central) y es muy difícil el acceso de terceros.

En este proyecto se pretende tener un conjunto de servidores descentralizado, en el que cada uno de los servidores tenga unos datos concretos y puedan ser consultados y recogidos con facilidad por un tercero y así calcular el recorrido óptimo con la participación de todos los servidores necesarios.

El trabajo de Biomimetiks en este proyecto consiste en identificar soluciones biomiméticas adaptables a los requisitos de diseño de la plataforma “Open Move”, que permita coordinar los servidores y tener un flujo de información dinámico entre ellos.

2. ANÁLISIS DEL PROBLEMA

El objetivo del proyecto, como hemos definido anteriormente, es obtener un itinerario de transporte público entre dos puntos geográficos definidos por el usuario.

Inicialmente, disponemos de un conjunto de servidores que deberemos coordinar para poder llegar a dicho objetivo. Cada uno de estos servidores tiene una serie de datos diferentes a los de los otros servidores, existiendo también la posibilidad de que estos servidores compartan algún que otro dato, es decir, que un dato pueda llegar a estar en más de un servidor. De este modo, se puede decir que los servidores están conectados mediante esos puntos comunes. Estos datos que hemos mencionado, son información sobre diferentes puntos.

Si los puntos origen y destino del recorrido corresponden a un mismo servidor no tenemos problemas (el servidor sabe calcular la ruta óptima porque el mismo tiene toda la información). El problema es, si estos puntos no corresponden al mismo servidor. Entonces, ¿cómo sabemos qué servidores tienen que formar parte para calcular la ruta óptima?

Aunque cada servidor sabe de forma individual si puede ayudar a calcular la ruta óptima (es decir, si tiene que formar parte del conjunto de servidores que calculan dicha ruta), es necesario que el servidor que inicia la búsqueda tenga acceso a esa información que tiene cada uno de ellos. El objetivo es conocer cuáles son los que tienen que participar para calcular la ruta óptima sin tener que preguntar a cada uno de ellos.

Por tanto, lo que queremos, es obtener la máxima cantidad de información con el mínimo número de preguntas.

$$\beta = \frac{\uparrow \text{Información}}{\downarrow \text{nº de preguntas}}$$

Para ello, hemos buscado estrategias biomiméticas sobre cómo obtener la mayor información minimizando el número de preguntas a agentes informantes (aquellos que disponen de dicha información).

Hemos obtenido cuatro posibles alternativas, basadas en diferentes sistemas naturales: las hormigas, el moho mucilaginoso, el desarrollo embrionario en vertebrados y el sistema nervioso de la mosca de la fruta. Esta última solución ha sido la considerada más óptima para resolver nuestro problema.

3. ALTERNATIVAS BIOMIMÉTICAS

3.1 Hormigas (transmitir información con mínimas conexiones)

A la hora de transmitir la información, las hormigas rojas optan por un sistema descentralizado en el que existen una serie de *hormigas hub*, que se encargan de manejar la mayor parte de la información. Esto permite una transmisión rápida y eficiente de la información.

La zona que une la cámara de entrada con el interior del hormiguero en sí mismo, es un punto caliente de intercambio de información. La localización de las hormigas, así como el comportamiento espacial de dichas hormigas (trayectoria que siguen) es lo que determina que hormigas se convierten en *hormigas hub*. Cuanto más tiempo se encuentran unas hormigas en un punto caliente de intercambio de información, más probabilidades hay de que se conviertan en *hormigas hub*. El otro factor determinante será su comportamiento espacial, cuanto más tortuosa sea la trayectoria de una hormiga, menor será su "conectividad".

La existencia de *hormigas hub*, facilita el flujo rápido de información sin necesidad de incrementar el número de interacciones. La existencia de individuos altamente conectados, reduce el número de interacciones necesarias para conectar dos nodos cualesquiera.

Lo que se planteaba, era que existieran unos nodos/servidores que actuaran igual que las *hormigas hub*. Es decir, que aquellos nodos/servidores que posean un mayor número de conexiones, fueran los responsables del manejo y la transmisión de la información. Esto, permitiría acelerar el flujo de información, y reducir al mismo tiempo, el tiempo de computación.

En el caso de las hormigas, son la localización y su comportamiento espacial las que determinan qué hormigas se autopostulan como hubs de información. Por ello, para completar la analogía con el hormiguero, sería necesario definir unas pautas o reglas que regulen la conversión de nodos/servidores normales a nodos/servidores hub en función de la búsqueda realizada por el usuario de Open Move.

3.2 Moho mucilaginoso (encontrar la ruta más corta)

A la hora de tejer su red de transporte de nutrientes, los hongos y los mohos deben encontrar un equilibrio entre el gasto de recursos necesario para acceder a una fuente de alimento, y la energía que dicha fuente le aporta. Los investigadores que han estudiado el proceso de búsqueda de alimento lo han dividido en 2 fases:

1. Fase exploratoria (Sobreproducción de nodos y enlaces)
2. Fase de consolidación:
    2.1 Selección y reforzamiento positivo.
    2.2 Reciclado de vías de transporte y reestructuración de la red.

Un buen ejemplo de esto es el profundamente estudiado moho mucilaginoso. El moho mucilaginoso diseña caminos de manera plenamente distribuida y sin necesidad de coordinación central. Tejiendo redes capaces de transportar la máxima cantidad de nutrientes con el mínimo gasto en infraestructura.

Algunos investigadores han logrado modelar el crecimiento del hongo a través de programas informáticos, y haciendo una analogía con el comportamiento de los circuitos eléctricos, han programado un algoritmo que permite dar con la ruta más corta entre dos puntos de manera distribuida. Lo que se propone, es basarse en estos algoritmos ya desarrollados para diseñar el algoritmo encargado de buscar la ruta más rápida en Open Move.

Esta alternativa se ha descartado, ya que su objetivo es la definición de la ruta más rápida. Esta necesidad se resuelve en la plataforma “Open Move” utilizando el algoritmo del Open Trip Planner.

3.3 Desarrollo embrionario en vertebrados (coordinar redes distribuidas)

En este caso, lo que se propone es inspirarse en los mecanismos de señalización celular encargados de controlar la diferenciación celular a lo largo del desarrollo embrionario de los vertebrados, con el objetivo de lograr que de nuestra red distribuida emerja una columna vertebral de nodos/servidores que nos asista en la búsqueda del camino más rápido entre dos puntos. De este modo, cada vez que se solicita una búsqueda, evitamos consultar a todos los nodos/servidores de la red, consultando únicamente a aquellos que conectan de manera efectiva nuestro punto origen y destino.

A lo largo del desarrollo embrionario, células que inicialmente son exactamente iguales se van diferenciando a título individual, para acabar dando lugar a los diferentes tejidos que podemos encontrar en un individuo adulto (tejido nervioso, tejido muscular, tejido óseo, tejido adiposo, etc.). Se trata de un proceso totalmente distribuido y carente de coordinación central, en el que en función de los estímulos que encuentra en su entorno, cada una de las células va tomando decisiones de manera plenamente independiente. A pesar de lo aparentemente caótico del proceso, el resultado es una estructura coherente y funcional, como podemos comprobar al observar cualquier ser vivo.

Este proceso está regulado principalmente por mecanismos de señalización celular en los que una célula determinada, se diferencia o no, en función de los estímulos químicos que encuentra en su entorno. Dicha reacción incluye frecuentemente un proceso de señalización intercelular, en el que una vez diferenciada, la célula libera

una sustancia química a su entorno, que induce o inhibe la diferenciación de las células vecinas.

Lo que se proponía en este caso, es que los nodos/servidores copiasen estos mecanismos de señalización celular e indujesen una respuesta en sus nodos/servidores vecinos. Al tratar a los nodos/servidores como agentes autónomos que actúan siguiendo una serie de normas sencillas para marcarnos el camino correcto, se reduce la cantidad de información a procesar de manera centralizada. Lo complicado en este caso, es determinar una vez más las pautas o reglas que deben seguir nuestros nodos/servidores.

Esta alternativa ha sido descartada, porque las normas a determinar varían en función de la búsqueda realizada.

3.4 El sistema nervioso de la mosca de la fruta (coordinar redes distribuidas)

En el sistema nervioso de la mosca de la fruta, unas pocas células se diferencian como centros de conexiones (Sensory Organ Precursor, *SOP*). Dichos *SOP* dan lugar finalmente a una serie de pelillos largos que ayudan a la mosca a detectar cambios en su entorno más próximo. A la hora de desarrollar estos órganos sensoriales, la mosca debe encontrar un equilibrio entre sentir el máximo posible, y reducir el gasto que conlleva el desarrollo de pelillos largos.

El conocimiento en detalle del mecanismo celular a través del cual las moscas seleccionan qué células madre se convierten en *SOP* y cuáles no, ha servido de inspiración para solucionar un problema con el que nos encontramos a la hora de coordinar los servidores que forman parte de una red de computación distribuida. Más concretamente para elegir una red local de líderes (maximal independent set, *MIS*). Un *MIS* es un conjunto de servidores *A*, de modo que cualquier servidor que forma parte de la red, se encuentra o bien dentro de *A*, o directamente conectado a un servidor que forma parte de *A*; no habiendo dos servidores dentro de *A* que se encuentren conectados de manera directa.

En la mosca de la fruta, la selección de *SOP's* se realiza durante la fase de larva y pupa. La transformación de una célula madre en *SOP* es un proceso totalmente estocástico. Puesto que a pesar de que todas las células están programadas para transformarse en *SOP*, no todas lo hacen. Esto es así porque una vez que una determinada célula se convierte en *SOP*, secreta unas sustancias que evitan que sus células vecinas asuman ese mismo rol. Como resultado, se forma una red de *SOP´s* en la que cualquier célula o bien es una célula *SOP*, o se encuentra conectada directamente a un *SOP*, justamente lo mismo que necesitamos en un *MIS*. Inspirarse en este proceso celular ha permitido, desarrollar un algoritmo capaz de calcular el MIS de una red distribuida de una manera rápida y fiable.

El algoritmo desarrollado a partir del sistema nervioso de la mosca de la fruta es el siguiente:

1. Algorithm: MIS (n, D) at node u
2. For i = 0: log D
    3. For j = 0: M log n // M is constant derived below
        4. * exchange 1*
        5. v = 0
        6. With probability $\frac{1}{2^{\log D-i}}$ broadcast ***B*** to neighbors and set v = 1 // ***B*** is one bit
        7. If received message from neighbor, then v = 0
        8. * exchange 2 *
        9. If v = 1 then
            10. Broadcast B; join MIS; exit the algorithm
        11. Else
            12. If received message B in this exchange, then mark node u inactive; exit the algorithm
        13. End
    14. End
15. End

Donde cada nodo (servidor) recibe un input *n* (el número de servidores que participan) y de *D* (el mayor número de vecinos que cualquier servidor puede tener).

El algoritmo se ejecuta sincronizadamente en todos los nodos. El algoritmo se ejecuta en $log\, n$ fases donde cada fase consiste en $M\, log\, n$ pasos, siendo *M* una constante (definida en el artículo de referencia).

Inicialmente todos los nodos están activos. Cada paso de cada fase *i* consiste en dos intercambios:

1) Cada nodo activo envía un mensaje a sus vecinos con una probabilidad p_i, ($p_i = \frac{1}{2^{\log D-i}}$).Donde la probabilidad p_i cambia a medida que aumenta *i,* es decir varía con el número de rondas.
2) El nodo que ha enviado la señal en el primer intercambio se une al *MIS* si ninguno de sus vecinos ha enviado un mensaje en el primer intercambio. Además cada nodo unido al *MIS* envía otra vez una señal a sus vecinos diciéndoles que se inhiban.

El algoritmo termina cuando $i = log\, D,$ y por ello todos los nodos que no están unidos al *MIS*, con una alta probabilidad, no están conectados a ningún nodo activo y por consiguiente se pueden unir al *MIS*.

Todos los mensajes enviados en el algoritmo son de un bit y el tiempo requerido para ejecutar el algoritmo es de $log\, n\, log\, D,$ o en el peor de los casos, $(log\, n)^2$ .Por otro lado el número de mensajes enviados por los nodos activos en nuestro algoritmo es linear con el número de nodos del algoritmo.

4. EVALUACIÓN Y DISCUSIÓN

¿Por qué nos hemos decidido por la solución basada en el sistema nervioso de la mosca? ¿Qué ventaja nos proporciona tener el *MIS* de una red distribuida?

En nuestro caso, precisamente, nos ayuda a tener una red de servidores donde sólo haya un único paso entre un servidor no perteneciente al *MIS* y un servidor si perteneciente al *MIS*. De esta manera, somos capaces de tener acceso a la información del servidor no perteneciente al *MIS* sin tener que preguntarle.

Así, podremos acceder a toda la información de todos los servidores únicamente preguntando a los servidores que pertenecen al *MIS*. Es decir, hemos minimizado el número de servidores a los que tenemos que preguntar si son los necesarios para calcular la ruta óptima pudiendo disponer de toda la información.

A modo de conclusión podemos decir que preguntando únicamente a los servidores del MIS podremos saber cuáles son los servidores que tienen la información necesaria para calcular la ruta óptima.

5. IMPLEMENTACIÓN

Para poder aplicar el algoritmo anteriormente descrito en nuestro problema debemos hacer algunas modificaciones.

La principal modificación introducida es que en el conjunto de servidores que tenemos, definiremos un subconjunto de servidores donde nuestro objetivo será calcular un *MIS* específico para cada uno de ellos. Es decir, realizamos una división del conjunto de servidores en diferentes grupos. Cada grupo tendrá un servidor central donde el *MIS* que se halle será para ese servidor. Por tanto, para cada servidor tendremos un grupo y así mismo, un *MIS*. Por ejemplo, si tenemos en total 20 servidores, tendremos 20 subconjuntos y por tanto, 20 *MIS.*

Una vez calculado el *MIS* para cada uno de ellos, en el servidor central se guardará un listado con los servidores de su *MIS* y las coordenadas de éstos. De este modo, sabrá a cual preguntar primero (el más cercano) cuando vaya a calcular la ruta.

Otra de las condiciones necesarias para que la solución propuesta sea eficiente es que cada servidor debe guardar con quién tiene él puntos compartidos y con quién tienen nodos compartidos sus vecinos.

Pasos a seguir por el algoritmo:

A) Situación:

Tenemos un servidor al que llamaremos central que será donde se guarde qué servidores pertenecen a ese *MIS* y sus coordenadas geográficas. Podemos decir que es el “cabecilla” de este *MIS*. Es, desde dónde se enviarán las órdenes a ejecutar por los demás servidores.

Este servidor central sin realizar ningún intercambio se une directamente al *MIS* y sus vecinos próximos se inhiben directamente. Una vez realizado lo anterior, tanto el servidor central como sus vecinos directos ya no formarán parte del proceso de selección del *MIS.*

B) Pasos:

1) Se define la primera ronda como *i=0* y a todos los nodos se les etiqueta como *v=0.*

2) Cada servidor sabe cuántos vecinos tiene y por ello, cada uno calculará su probabilidad según esta ecuación donde *D* es el número de vecinos de cada servidor e *i* va variando con cada ronda. El servidor central será el encargado de enviar esta orden a todos los demás servidores.

$$p_i = \frac{1}{2^{\log D - i}}$$

3) Cada servidor envía su probabilidad a sus vecinos cercanos. De manera que cada servidor recibirá la probabilidad de los servidores vecinos. Esta orden es enviada por el servidor central para que sea ejecutada en los demás servidores. En el servidor central no se ejecuta.

4) Cada servidor compara su probabilidad con la de sus vecinos y si es mayor su probabilidad, los intercambios empezaran por ese servidor. Para ello, los servidores de mayor probabilidad (comparada con la de sus vecinos cercanos), enviarán un mensaje al servidor central y entonces el servidor central les enviara únicamente una orden a ellos diciéndoles los siguientes pasos. Por ellos los servidores de mayor probabilidad empezarán a la vez con los intercambios. Al igual que la orden anterior, esta es enviada por el servidor central

5) Una vez definidos los servidores que empiezan, comienzan los dos intercambios:

   i. Los servidores de mayor probabilidad envían un mensaje de un bit a sus vecinos cercanos. Este servidor que ha enviado la señal se pone como v=1 y sus vecinos se quedan como v=0.

   ii. Los servidores diferenciados como v=1 envían otro mensaje a sus vecinos diciéndoles que se inhiban y este servidor v=1 se une al *MIS.* Es decir, el servidor que se debe unir al *MIS* envía un mensaje al servidor central para que guarde su nombre y sus coordenadas en el listado de servidores del *MIS*. Este paso únicamente se ejecuta si ninguno de sus vecinos en el primer intercambio ha enviado un mensaje, en cuyo caso el servidor no se une al *MIS* ni sus vecinos se inhiben.

a y b son probabilidades tal que a>b. i. Esquema del primer intercambio donde B es el mensaje enviado a los vecinos. ii. Esquema del segundo intercambio donde I es el mensaje enviado a sus vecinos para que se inhiban.

6) Tanto el servidor unido al *MIS* como sus vecinos inhibidos salen del algoritmo, es decir ya no participan en las siguientes rondas.

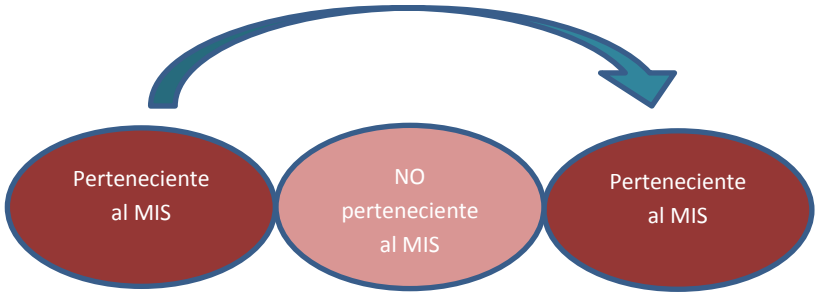
7) Una vez terminada esta ronda, se vuelve a definir *i*, en este caso como *i=1*, y volvemos a comenzar todo el proceso anteriormente descrito.

Nuestro algoritmo acabará de ejecutarse cuando todos los servidores hayan salido del algoritmo, es decir ya no participen en este. Esto ocurre cuando todos los servidores pertenezcan al *MIS* o estén directamente unidos a éste.

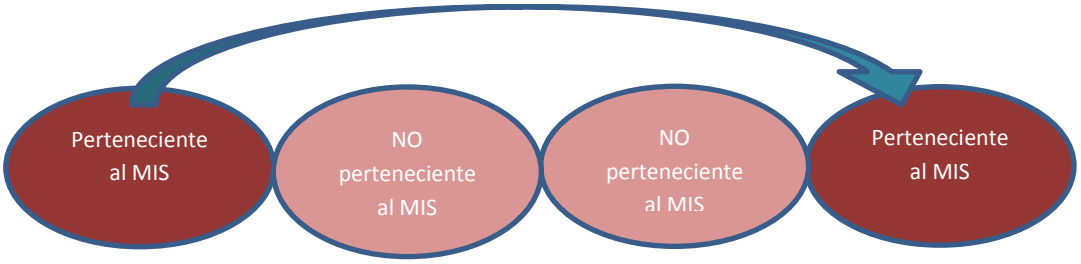
Una vez calculado el *MIS,* el servidor central en la búsqueda de la ruta preguntará a aquellos servidores de su lista de *MIS* más cercanos a él. La pregunta realizada al otro servidor será si éste o alguno de sus vecinos comparten algún nodo con su vecino no perteneciente al *MIS*. Por ello, como hemos mencionado antes, cada servidor debe de guardar con quién comparte él nodos y con quien comparten sus vecinos. A continuación se explica por qué debe de guardar estos datos.

Tenemos dos casos posibles dentro del *MIS*:

1)



2)



En el primer caso si pregunta únicamente a su vecino de *MIS* si comparte un nodo con el vecino de no *MIS*, este le responderá afirmativamente independientemente de que tenga o no tenga guardada la información de con quién comparten nodos su vecinos. Pero, en el segundo caso, si realizamos esa pregunta sin tener esos datos guardados la respuesta será negativa. Y entonces deberíamos introducir nuevas preguntas en cuyo caso nuestro objetivo de minimizar las preguntas no se vería satisfecho. En cambio sí se guardasen dichos datos preguntando si él o alguno de sus vecinos comparte los nodos, sería suficiente.

5.1 Posibles alternativas en la implementación

5.1.1 Problema en la unión directa del servidor central al MIS

En el caso en el que el hecho de que el servidor central se una al MIS directamente y que sus vecinos también se inhiban directamente, nos pueda causar algún problema,

podríamos establecer la misma probabilidad en la primera ronda en el servidor central que tienen los servidores de mayor probabilidad. Así, comenzarían a ejecutarse los servidores de mayor probabilidad entre los que estaría incluido el central. Si se diese el caso en el que el algoritmo no se puede ejecutar debido a lo expuesto en el segundo intercambio, en la siguiente ronda volveríamos a establecer al servidor central la misma probabilidad que los de mayor probabilidad.

5.1.2 D variable

Hemos definido *D* como el número de vecinos que tiene cada servidor, es decir un número constante para cada servidor. Como posibles modificaciones futuras, podríamos definir *D* como el número de vecinos activos que tienen los servidores, de manera que *D* fuese variando con cada ronda.

5.1.3 Definir el primer servidor a preguntar del MIS

Cabe mencionar que en un futuro, para nuestro problema concreto de cálculo de la ruta, deberíamos introducir una modificación basándonos en el comportamiento espacial de las hormigas “hub” tal que el primer algoritmo de su *MIS* al que pregunte sea el que está en línea recta hacia el servidor objetivo.