



# UNIVERSITAT POLITÈCNICA DE VALÈNCIA

**Escola Tècnica Superior d'Enginyeria Informàtica**

PROYECTO FIN DE CARRERA / PROJECTE FI DE CARRERA

**Desarrollo de una infraestructura para la resolución distribuida  
de algoritmos de colonias de hormigas basados en espacios de  
búsqueda semánticos dinámicos**

Para optar a la titulación de / per a optar a la titulació de

**Ingeniero Informático**

Presentado por / presentat per:

**Víctor Manuel Martínez Valero**

Dirigido / tutorizado por  
dirigit / tutoritzat per

**Dr. Francisco Javier Jaén Martínez**

**José Antonio Mocholí Agües**

València, **14 de diciembre de 2010.**



## **Agradecimientos**

A Javi por ayudarme a desarrollar este proyecto desde la primera idea hasta la última.

A José Antonio y Alejandro por su inestimable ayuda en el aspecto técnico y los buenos ratos pasados en el laboratorio.

A Mar y Juan porque me hicieron terminar el proyecto. A Elena y Álex porque me hicieron terminar la carrera.

A mis padres y mi hermano porque me han hecho.

# Índice

AGRADECIMIENTOS .....	2
ÍNDICE .....	3
ÍNDICE DE FIGURAS .....	5
<b>CAPÍTULO 1. ALGORITMOS EVOLUTIVOS BASADOS EN COLONIAS DE HORMIGAS .....</b>	<b>6</b>
1.1. DESCRIPCIÓN DEL PROBLEMA .....	6
1.2. OPTIMIZACIÓN BASADA EN COLONIAS DE HORMIGAS .....	7
1.3. PROBLEMAS RESOLUBLES MEDIANTE OCH .....	8
1.4. HORMIGAS ARTIFICIALES .....	9
1.5. MODELOS DE OCH .....	10
1.5.1. <i>El Sistema de Hormigas</i> .....	10
1.5.2. <i>El sistema de colonias de hormigas</i> .....	12
1.6. ALGORITMOS DE OCH PARALELOS .....	14
1.7. CONCLUSIONES .....	14
<b>CAPÍTULO 2. UN MODELO SEMÁNTICO PARA LA GENERACIÓN DE ESPACIOS DE BÚSQUEDA .....</b>	<b>15</b>
2.1 WEB SEMÁNTICA .....	15
2.2. RESOURCE DESCRIPTION FRAMEWORK .....	15
2.3. ESCRIBIENDO SENTENCIAS SOBRE RECURSOS .....	16
2.3.1. <i>Conceptos básicos</i> .....	16
2.3.2. <i>El modelo RDF</i> .....	17
2.4. SINTAXIS PARA RDF: RDF/XML .....	18
2.5. SQLITE .....	20
2.6. SEMWEB .....	20
2.7. RDF EN UNA BASE DE DATOS SQLITE .....	22
2.8. CONSULTAS Y CREACIÓN DEL ESPACIO DE BÚSQUEDA .....	25
2.8.1 <i>Consultas sobre la base de datos SQLite</i> .....	25
2.8.2 <i>Creación de los nodos</i> .....	26
2.8.3 <i>Funciones de evaluación</i> .....	26
2.8.4 <i>Obtención del Score</i> .....	27
2.8.5 <i>Generación del espacio de búsqueda</i> .....	28
2.8.6 <i>Modelo de clases</i> .....	28
<b>CAPÍTULO 3. UN MODELO DISTRIBUIDO PARA LA RESOLUCIÓN DE ALGORITMOS OCH 32</b>	<b>32</b>
3.1. ARQUITECTURA DEL SISTEMA .....	32
3.1.1. <i>Funcionamiento del sistema</i> .....	32
3.2. MULTIMEDIAQUERIER .....	33
3.2.1. <i>Funcionamiento del MultimediaQuerier</i> .....	33
3.3. MASTERQUEENSERVICE .....	33
3.3.1. <i>Funcionamiento del MasterQueenService</i> .....	34
3.3.2. <i>Modelo de clases</i> .....	34
3.3.3. <i>Métodos del servicio web</i> .....	35
3.4. HIVECLIENT .....	37
3.4.1. <i>Funcionamiento del HiveClient</i> .....	37
3.4.2. <i>Modelo de clases</i> .....	37
<b>CAPÍTULO 4. CASO DE ESTUDIO .....</b>	<b>41</b>
4.1. UNA BASE DE DATOS DE DISCOS .....	41

---

4.2. DEFINICIÓN DE LOS METADATOS .....	41
4.2.1. <i>El metadato género</i> .....	43
4.2.2. <i>El metadato artista</i> .....	44
4.2.3. <i>El metadato año</i> .....	45
4.3. EVALUACIÓN DEL SISTEMA.....	46
4.3.1. <i>Consulta: Folk = 100</i> .....	47
4.3.2. <i>Consulta: Trance = 100</i> .....	49
4.3.3. <i>Consulta: Folk = 50; Trance = 50</i> .....	51
4.3.4. <i>Consulta: Folk = 80; Trance = 20</i> .....	55
4.3.5. <i>Consulta: Folk = 50; Javier Krahe = 50</i> .....	59
4.3.6. <i>Consulta: Folk = 80; Javier Krahe = 20</i> .....	63
4.3.7. <i>Consulta: Folk = 50; Armin Van Buuren = 50</i> .....	67
4.3.8. <i>Consulta: Folk = 80; Armin Van Buuren = 20</i> .....	71
<b>CAPÍTULO 5. CONCLUSIONES .....</b>	<b>76</b>
5.1. DESARROLLO DEL PROYECTO .....	76
5.2. CONCLUSIONES DEL TRABAJO REALIZADO.....	76
5.3. TRABAJOS FUTUROS .....	77
<b>BIBLIOGRAFÍA .....</b>	<b>78</b>

## Índice de figuras

Figura 1. Varias sentencias sobre un recurso .....	17
Figura 2. Tabla rdf_entities .....	22
Figura 3. Tabla rdf_literals .....	23
Figura 4. Tabla rdf_statements .....	24
Figura 5. Modelo de clases .....	29
Figura 6. Modelo de clases de las consultas .....	30
Figura 7. Modelo de clases de los nodos .....	31
Figura 8. Modelo de clases de MasterQueenService .....	34
Figura 9. Modelo de clases de HiveClient .....	38
Figura 10. Frecuencia. Consulta: Folk = 100 .....	47
Figura 11. Saturación. Consulta: Folk = 100 .....	48
Figura 12. Frecuencia. Consulta: Trance = 100 .....	50
Figura 13. Saturación. Consulta: Trance = 100 .....	51
Figura 14. Frecuencia. Consulta: Folk = 50; Trance = 50 .....	52
Figura 15. Saturación. Consulta: Folk = 50; Trance = 50 .....	53
Figura 16. Frecuencia. Consulta: Folk = 50; Trance = 50 .....	54
Figura 17. Saturación. Consulta: Folk = 50; Trance = 50 .....	55
Figura 18. Frecuencia. Consulta: Folk = 80; Trance = 20 .....	56
Figura 19. Saturación. Consulta: Folk = 80; Trance = 20 .....	57
Figura 20. Frecuencia. Consulta: Folk = 80; Trance = 20 .....	58
Figura 21. Saturación. Consulta: Folk = 80; Trance = 20 .....	59
Figura 22. Frecuencia. Consulta: Folk = 50; Javier Krahe = 50 .....	60
Figura 23. Saturación. Consulta: Folk = 50; Javier Krahe = 50 .....	61
Figura 24. Frecuencia. Consulta: Folk = 50; Javier Krahe = 50 .....	62
Figura 25. Saturación. Consulta: Folk = 50; Javier Krahe = 50 .....	63
Figura 26. Frecuencia. Consulta: Folk = 80; Javier Krahe = 20 .....	64
Figura 27. Saturación. Consulta: Folk = 80; Javier Krahe = 20 .....	65
Figura 28. Frecuencia. Consulta: Folk = 80; Javier Krahe = 20 .....	66
Figura 29. Saturación. Consulta: Folk = 80; Javier Krahe = 20 .....	67
Figura 30. Frecuencia. Consulta: Folk = 50; Armin Van Buuren = 50 .....	68
Figura 31. Saturación. Consulta: Folk = 50; Armin Van Buuren = 50 .....	69
Figura 32. Frecuencia. Consulta: Folk = 50; Armin Van Buuren = 50 .....	70
Figura 33. Saturación. Consulta: Folk = 50; Armin Van Buuren = 50 .....	70
Figura 34. Frecuencia. Consulta: Folk = 80; Armin Van Buuren = 20 .....	72
Figura 35. Saturación. Consulta: Folk = 80; Armin Van Buuren = 20 .....	72
Figura 36. Frecuencia. Consulta: Folk = 80; Armin Van Buuren = 20 .....	73
Figura 37. Saturación. Consulta: Folk = 80; Armin Van Buuren = 20 .....	74

## Capítulo 1. Algoritmos evolutivos basados en colonias de hormigas

En este capítulo trataremos de describir el uso de algoritmos basados en colonias de hormigas para la resolución de problemas de optimización como el de la orientación o similares. Definiremos el problema que, en esencia, nuestro trabajo quiere resolver, que no es otro que el de encontrar soluciones con puntuación máxima, sin superar ciertas restricciones. Este problema suele describirse como el problema de la orientación, si bien su uso no se restringe sólo a este ámbito. Describiremos qué es una colonia de hormigas y qué podemos extraer de ellas para realizar nuestra tarea. Mostraremos cómo creamos hormigas artificiales a partir del comportamiento de estas colonias. Todo ello para poder solucionar problemas en contextos fijos, que trabajarán con unos datos fijos cuya descripción será tratada en el siguiente capítulo.

### 1.1. Descripción del problema

Como hemos comentado, el problema que trataremos de resolver en nuestro estudio suele ser conocido como el problema de la orientación, ya que es originario de las carreras deportivas dónde un grupo de participantes recorre un espacio desconocido con la ayuda de un mapa. Debido a que no sólo es importante en este problema el tiempo empleado en la llegada a la meta, sino también los puntos de control recorridos, se debe encontrar un balance entre la velocidad y las estaciones visitadas. El ámbito de uso del problema es amplio, pudiendo trasladarse a la generación de itinerarios, listas de reproducción, etc. Su formulación es como sigue.

Dado un grafo no dirigido  $G(V,A)$ , con  $|V| = n$  nodos en el espacio Euclídeo donde cada nodo tiene asociado un determinado beneficio  $S_i$ ,  $S_i > 0$  excepto los nodos  $v_1$  y  $v_n$  (inicial y final) que tienen asociado un beneficio  $S_1 = S_n = 0$ ; cada nodo puede ser visitado como máximo una vez; y cada arista  $(v_i, v_j)$   $v_i, v_j \in V$  tiene asociado un coste  $c_{ij} > 0$  el objetivo es encontrar un camino de beneficio máximo donde el coste asociado a dicho camino no supere un determinado valor máximo  $T_{max}$ .

Formulación matemática:

$$\text{Max} \sum_{i=1}^n \sum_{j=1}^n S_i * x_{ij}$$

Manteniendo las restricciones:

$$\sum_{j=2}^n x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1$$

$$\sum_{i=2}^{n-1} x_{ik} = \sum_{j=2}^{n-1} x_{kj} \leq 1, \quad k = 2, \dots, n-1$$

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} * x_{ij} \leq T_{max}$$

$$x_{ij} \in \{0,1\}, \quad i, j = 1, \dots, n$$

En nuestro caso, los elementos representados mediante nodos no vienen fijados previamente, sino que serán fijados por el usuario final de nuestra componente. Esto permitirá la resolución del problema expuesto en cualquiera de los ámbitos referidos.

## 1.2. Optimización basada en colonias de hormigas

Las colonias de hormigas, como otros tipos de sistemas provenientes de la naturaleza, son sistemas con una organización social perfectamente estructurada en las que existen mecanismos de coordinación que hacen que las mismas puedan realizar en conjunto tareas que posiblemente no podrían ser abordadas por ellas de forma individual.

Un ejemplo de esta coordinación está relacionado con la percepción visual de muchas de las especies de hormigas, muy poco desarrollada, lo cual hace que gran mayoría de sus mecanismos de comunicación se basen en el intercambio de feromonas, unas sustancias químicas producidas de forma natural. Concretamente, en las colonias de hormigas tienen una importancia vital las feromonas de rastreo “*trail feromones*” que algunas especies como la “*Lasius Niger*” o la “*Iridomyrmex humilis*” utilizan para marcar químicamente caminos en el terreno que recorren que le conduzcan desde su hormiguero hasta los lugares en los que haya alimentos. Mientras las hormigas se van desplazando desde el hormiguero hasta la fuente de alimento, van depositando feromonas en el terreno que recorren. Si una hormiga en su camino no encuentra rastro alguno de feromona, ésta se moverá de forma aleatoria; en cambio, si detecta dicha sustancia, tenderá a seguir con una mayor probabilidad el rastro detectado.

Existen gran cantidad de experimentos que han demostrado que las hormigas terminan tomando aquellos caminos que tienen una mayor concentración de feromonas y que, en la práctica, cuando hay varios caminos bifurcados, las hormigas terminan eligiendo qué camino tomar basándose en la intensidad de feromonas. De esta manera, al depositar estas mismas hormigas a su vez feromonas en el camino que acaban de elegir, se convierte este mecanismo en un proceso de refuerzo que termina construyendo caminos intensamente marcados. Este proceso tiene una razón de ser dado que, como han demostrado los experimentos del puente doble de *Deneubourg*, el mecanismo de autorefuerzo conduce a la selección de los caminos más cortos debido a la evaporación de las feromonas.

*Deneubourg* diseñó un puente doble con dos caminos (un el doble de largo que el otro) que conectaban a un hormiguero de la especie *Iridomyrmex humilis* con una fuente de alimento. En los experimentos realizados por *Deneubourg*, éste pudo observar consistentemente que



tras cierto periodo de tiempo transitorio en el cual las hormigas escogían ambos caminos aleatoriamente siguiendo un proceso que podíamos denominar de exploración, las hormigas terminaban intensificando el camino más corto.

Basándose en el comportamiento de las hormigas a penas descrito, *Dorigo* y sus colaboradores introdujeron por primera vez la optimización basada en colonias de hormigas (OCH) con el objetivo de resolver problemas de optimización combinatoria. La formulación de *Dorigo* se basa en la existencia de un modelo de hormiga artificial, agentes *software* que trabajan de forma cooperativa comunicándose mediante rastros de feromonas también artificiales. Se traslada al mundo de la computación el estudio del sistema natural descrito.

Para entender de una forma sencilla el funcionamiento de los algoritmos de OCH debemos ver el proceso de resolución como un ejercicio constructivo en el que, en cada iteración, cada hormiga construye una solución recorriendo el grafo subyacente al problema. Este proceso constructivo puede verse como un proceso de decisión en el que cada hormiga debe decidir individualmente a qué nuevo nodo dirigirse desde el último nodo del grafo que ya haya visitado. Para poder tomar estas decisiones, el algoritmo mantiene dos tipos de información, heurística y de rastros de feromonas. La información heurística mide la atracción (o el grado de preferencia) de moverse desde un nodo origen a un nodo destino. Dicha atracción no se modifica a lo largo de la ejecución del algoritmo. En cambio, la información de feromonas artificiales mide la atracción aprendida de moverse entre dos nodos y actúa como una especie de memoria a largo plazo. Esta atracción aprendida imita a las feromonas naturales y por tanto dicha información va siendo modificada a lo largo de la ejecución del algoritmo siguiendo unos criterios que serán detallados más adelante.

### 1.3. Problemas Resolubles mediante OCH

Los problemas resolubles mediante OCH pertenecen a la categoría de problemas de camino mínimo que están caracterizados por los siguientes aspectos:

- Un conjunto de restricciones posiblemente dependientes del tiempo  $\Omega(t)$
- Un conjunto finito de componentes  $C = \{c_1, c_2, \dots, c_{N_c}\}$

Los estados del problema se definen en términos de secuencias  $x = \langle c_i, c_j, \dots, c_h, \dots \rangle$  de longitud finita sobre los elementos de  $C$ . El conjunto de todos los posibles estados se denota como  $\chi$ .

- Un conjunto de soluciones candidatas  $S \subseteq \chi$ .
- Un conjunto de estados posibles  $\bar{\chi} \subseteq \chi$  definidos mediante un test que es dependiente del problema que verifica que no es imposible completar una secuencia  $x \in \bar{\chi}$  en una solución que satisfaga las restricciones  $\Omega$ .
- Un conjunto no vacío de soluciones óptimas  $S^* \subseteq \bar{\chi}$  y  $S^* \subseteq S$ .

- Una función de coste  $g(s, t)$  asociada con cada solución candidata  $s \in S$ .
- Una función de coste, o estimación de coste  $J(x, t)$  asociados a los estados que no son soluciones candidatas.

Se puede observar mediante un análisis comparativo que el problema de la orientación descrito anteriormente pertenece a la categoría de problemas que pueden ser solucionados mediante OCH. Pero como antes exponíamos, éste no es el único. Otros problemas que han sido abordados por OCH son el problema del viajante de comercio (*TSP*), el problema de la asignación cuadrática (*QAP*), el problema de la asignación generalizado, el problema del enrutamiento de vehículos, y otros muchos similares.

Con esta formulación una hormiga artificial construye soluciones realizando caminos de forma probabilista en el grafo completamente conectado  $G_C = (V, A)$  (grafo de construcción) donde los vértices son las componentes de  $C$  y las aristas de  $A$  conectan completamente las componentes de  $C$ . tanto los nodos como las aristas pueden tener asociada los dos tipos de información mencionados anteriormente: un valor heurístico de atracción ( $\eta_i, \eta_{ij}$  respectivamente) y un nivel de feromona ( $\tau_i, \tau_{ij}$  respectivamente). A continuación veremos una definición más formal del modelo de hormiga artificial y el mecanismo mediante el cual recorre el grafo de construcción y hace uso de la información asociada al mismo para encontrar soluciones.

#### 1.4. Hormigas artificiales

Como hemos mencionado, una hormiga artificial  $k$  es un proceso estocástico constructivo que presenta las siguientes propiedades:

Hace uso del grafo de construcción  $G_C = (V, A)$  en búsqueda de soluciones óptimas  $s^* \in S^*$ .

Tiene una memoria  $\mathfrak{M}^k$  para almacenar información sobre el camino seguido hasta un instante dado. Dicha memoria puede ser utilizada para:

- Construir soluciones factibles.
- Calcular valores heurísticos  $\eta_i, \eta_{ij}$ .
- Evaluar la solución encontrada.
- Recorrer el camino seguido hacia atrás.

Tiene un estado de partida  $x_s^k$  y una o más condiciones de terminación  $e^k$ . Normalmente el estado inicial se expresa como una secuencia vacía o como una secuencia con una única componente.

Estando en un estado  $x_r = \langle x_{r-1}, i \rangle$  y si no se satisface ninguna condición de terminación, se mueve a un nodo  $j$  en su vecindad  $\mathcal{U}^k(x_r)$ , esto es, al estado  $\langle x_r, j \rangle \in \chi$ . Si al menos alguna de las condiciones de terminación es cierta entonces la hormiga se detiene.

Selecciona un movimiento aplicando una regla de decisión probabilística. Dicha regla es función de:

- Los valores locales heurísticos  $\eta_i, \eta_{ij}$  y de feromonas  $\tau_i \tau_{ij}$ .
- La memoria privada de la hormiga en la que se almacena el estado actual.
- Las restricciones del problema.

Cuando se añade una componente  $c_j$  al estado actual se puede actualizar el nivel de feromona asociado a dicha componente o a la arista seleccionada.

Una vez se ha construido una solución puede volver hacia atrás el camino realizado y actualizar los niveles de feromonas de las componentes seleccionadas.

Es importante señalar que varias hormigas actúan de forma concurrente e independiente y que aunque cada hormiga puede encontrar una solución (probablemente pobre) las soluciones de mayor calidad emergen a partir de la interacción colectiva de varias hormigas. Esta interacción colectiva se obtiene mediante la comunicación indirecta que se da a través de las variables que almacenan niveles de feromonas y que las hormigas leen/escriben. Es pues, una forma de aprendizaje colectivo en el que cada hormiga modifica de forma adaptativa la manera en la que el resto de hormigas perciben el problema a resolver.

## 1.5. Modelos de OCH

Una vez descrita genéricamente la metaheurística OCH hay que destacar que existen diversos algoritmos que hacen uso de ella para problemas de naturaleza combinatoria. Describiremos brevemente el Sistema de Hormigas que tiene interés histórico y el Sistema de Colonia de Hormigas que es el utilizado en [1] para la resolución del problema de la orientación, el cual tomamos como referencia en nuestro trabajo.

Otros algoritmos similares pueden ser el Sistema de Hormigas con ordenación de *Bullnheimer*, el Sistema de la Mejor-Peor Hormiga de Cordón, y el Sistema Max-Min de *Stützle*, los cuales alcanzan resultados satisfactorios.

### 1.5.1. El Sistema de Hormigas

El Sistema de Hormigas fue el primer algoritmo de OCH propuesto por *Dorigo* y sus colaboradores en 1991. Los autores proponen tres variantes denominadas SH-ciclo, SH-cantidad y SH-densidad. En el SH-ciclo las hormigas depositan feromonas una vez completada la solución (“*off-line update*”). En cambio, SH-cantidad y SH-densidad realizan la deposición de feromona a cada paso de construcción de una solución (“*on-line update*”).

En SH-densidad la cantidad de feromona depositada es constante mientras que en SH-cantidad depende de la atracción heurística  $\eta_{ij}$  de la transición seleccionada.

Los estudios empíricos han demostrado que SH-ciclo es superior al resto y por esta razón y dado que su mecanismo de actualización ha servido de referencia para posteriores algoritmos lo detallaremos a continuación.

Como hemos comentado, la feromona se deposita una vez que todas las hormigas han completado una solución. En primer lugar los rastros de feromona de cada transición se evaporan mediante una tasa de evaporación de forma que los rastros se reducen en un factor constante:

$$\tau_{ij}^{new} \leftarrow (1 - \rho) * \tau_{ij}^{old}; \rho \in (0,1]$$

Una vez evaporadas las feromonas cada hormiga  $k$  recorre su memoria local  $\mathfrak{M}^k$  y deposita una cantidad de feromona  $\Delta\tau_{ij}^k = f(C(S_k))$  en cada arco transitado del grafo de construcción donde la cantidad de feromona depositada depende de la calidad  $C(S_k)$  de la solución  $S_k$  obtenida por la hormiga  $k$

$$\tau_{ij}^{new} \leftarrow \tau_{ij}^{old} + \Delta_{ij}^k, \quad \forall a_{ij} \in S_k$$

Por último, el proceso de construcción de la solución sigue el mecanismo descrito en el modelo de hormiga artificial donde la regla de decisión probabilista de dicho modelo es como sigue

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha * [\eta_{ij}]^\beta}{\sum_{u \in \mathfrak{U}_i^k} [\tau_{iu}]^\alpha * [\eta_{iu}]^\beta}, & \text{si } j \in \mathfrak{U}_i^k \\ 0, & \text{en otro caso} \end{cases}$$

Donde  $\mathfrak{U}_i^k$  es el vecindario alcanzable  $\mathfrak{U}^k(x_r)$  estando en el estado  $\langle x_r^k, i \rangle \in \chi$  y  $\alpha, \beta \in \mathfrak{R}$  son dos factores que ponderan la importancia de la información heurística y de los rastros de feromona. Si  $\alpha = 0$ , aquellos nodos con mejor atracción heurística son los seleccionados y nos encontramos ante un algoritmo probabilista de naturaleza voraz. Si, en cambio  $\beta = 0$ , sólo las feromonas dirigen el proceso de construcción de soluciones de manera que se llega a un rápido estancamiento en el que las hormigas construyen siempre las mismas soluciones que suelen ser óptimos locales.

Es importante resaltar que los autores de este trabajo propusieron una mejora final a este procedimiento denominada SH-elitista en la que una vez realizado el proceso anterior, la hormiga reina deposita un nivel de feromona adicional en aquellas aristas que pertenecen a la mejor solución encontrada hasta el momento en el proceso de búsqueda. Este incremento de feromona se produce en un factor  $E$  que indica el número de hormigas elitistas que se consideran

$$\tau_{ij}^{new} \leftarrow \tau_{ij}^{old} + E * f(C(S_{mejor})), \quad \forall a_{ij} \in S_{mejor}$$

### 1.5.2. El sistema de colonias de hormigas

El sistema de colonias de hormigas es una evolución del SH en el que se proponen diversas modificaciones y que detallaremos aquí pues es el algoritmo que se toma como base en [1] para la resolución del problema de la orientación. En primer lugar se añade una regla de selección probabilista de un valor aleatorio  $q_0 \in [0,1]$  de forma que el siguiente nodo a visitar por la hormiga  $k$  se elige según la siguiente distribución de probabilidad:

Si  $q \leq q_0$

$$p_{ij}^k = \begin{cases} 1, & \text{si } j = \arg \max_{u \in \mathcal{U}_i^k} \{[\tau_{iu}]^\alpha * [\eta_{iu}]^\beta\} \\ 0, & \text{en otro caso} \end{cases}$$

Si no

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha * [\eta_{ij}]^\beta}{\sum_{u \in \mathcal{U}_i^k} [\tau_{iu}]^\alpha * [\eta_{iu}]^\beta}, & \text{si } j \in \mathcal{U}_i^k \\ 0, & \text{en otro caso} \end{cases}$$

Como podemos observar, en el primer caso ( $q \leq q_0$ ) se explota la información disponible tanto de naturaleza heurística como de feromonas tomando como transición aquella que maximiza la atracción. Sin embargo en el segundo caso se aplica una exploración aleatoria siguiendo la distribución de probabilidad descrito en el algoritmo SH tradicional.

La segunda modificación de esta versión del algoritmo con respecto al algoritmo original estriba en el hecho de que únicamente la hormiga reina actualiza los rastros de feromonas cuando todas las hormigas han obtenido una solución. A esta actualización se le denomina *offline* y la solución que se tiene en cuenta para esta única actualización es la mejor solución global evaporando previamente los valores de feromonas de todas las aristas que participan en la mejor solución global. Además, en algunas ocasiones se pueden aplicar técnicas de búsqueda local para mejorar la solución obtenida antes de actualizar las feromonas.

Por último, la tercera variación con respecto al algoritmo SH original deriva de la actualización de las feromonas paso a paso por cada hormiga cada vez que se ha seleccionado una arista  $a_{ij}$  como sigue:

$$\tau_{ij}^{new} \leftarrow (1 - \varphi) * \tau_{ij}^{old} + \varphi * \tau_0 : \varphi \in (0,1]$$

Donde  $\varphi$  es un parámetro más de decremento de feromona. La regla anterior aplica simultáneamente una evaporación y una aplicación de  $\tau_0$  unidades de feromona donde  $\tau_0$  es

un valor muy pequeño que sirve de límite inferior para los valores de feromona. Al aplicar esta regla aquellas aristas visitadas por muchas hormigas tendrán un valor de feromona menor de manera que su atracción para otras hormigas será cada vez menor y por lo tanto se facilitará que no todas las hormigas sigan el mismo camino. A este mecanismo de actualización se le denomina actualización *online*.

En el caso del algoritmo propuesto en [1] para resolver el problema de la orientación se sigue la estrategia planteada en esta versión y que de forma pseudo-algórica podría resumirse como sigue:

Inicialización de todos los parámetros

Repetir

    Repetir (para cada hormiga  $k$ )

        Construir solución según la regla de decisión

        Aplicar la modificación de feromonas *on-line*

    Hasta que todas las hormigas han sido generadas

    Aplicar la búsqueda local (no se realiza en nuestro trabajo)

    Evaluar soluciones y almacenar la mejor global

    Aplicar la modificación de feromonas *offline*

Hasta alcanzar el criterio de parada

En el caso particular del problema de orientación además se particulariza la definición matemática expresada anteriormente de manera que:

$$\eta_{ij} = \frac{S_j}{c_{ij}} \quad \tau_0 = \frac{1}{n * T_{max}} \quad S_{kp} = S_k * \frac{T_{max}}{T_k}$$

Donde, como ya definimos en la formulación matemática del problema de la orientación,  $S_j$  representa el beneficio (*score*) asociado a cada nodo del grafo de construcción;  $c_{ij}$  la distancia o coste asociado a la arista que une los nodos del grafo  $i$  y  $j$ ;  $n$  el número de nodos en el grafo, y  $S_{kp}$  es la penalización del beneficio asociado a una solución  $S_k$  no factible obtenida por la hormiga  $k$ . En esta versión del algoritmo se permiten soluciones no factibles porque se permite la exploración de éstas si se encuentran cercanas a la frontera de las soluciones factibles dado que en estas regiones es probable que se encuentren óptimos globales.

En el ámbito de nuestro proyecto cabe destacar por último que trabajaremos con *scores* fijos, que vendrán dados por funciones de evaluación definidas por el usuario, y que a su vez trabajarán dentro de unos contextos claramente determinados.

## 1.6. Algoritmos de OCH paralelos

La obtención de algoritmos paralelos de OCH ha sido recientemente sujeto de intensas labores de investigación. El objetivo básico a conseguir es la aceleración del proceso de construcción de soluciones sin comprometer la calidad de las soluciones finales obtenidas. Los algoritmos propuestos en este campo de investigación se clasifican en dos categorías según el paralelismo sea individualmente a nivel de hormiga o globalmente a nivel de colonia. En el primer caso, los algoritmos propuestos ubican a cada hormiga en un procesador dedicado obteniéndose de esta forma un paralelismo de granularidad fina. En el segundo caso, en cambio, varias hormigas e incluso una colonia entera comparten un mismo procesador o nodo de computación.

Nuestro objetivo, pues, es demostrar que es posible diseñar un algoritmo paralelo de grano grueso y multi-colonia pero en el que las colonias trabajen como islas aisladas e independientes trabajando únicamente en una región parcial del espacio de búsqueda completo.

## 1.7. Conclusiones

El objetivo de nuestro trabajo es, por tanto, dar solución a los dos problemas que las actuales implementaciones de los algoritmos OCH suelen incorporar.

Por un lado el espacio de búsqueda queda predeterminado en el momento de la implementación del algoritmo y por lo tanto la aplicación obtenida es completamente dependiente del tipo de datos.

Además, las implementaciones que se han venido realizando han solido seguir una estructura centralizada lo cual supone un claro impedimento a la hora de realizar estudios sobre espacios de búsqueda mayores.

Nosotros trataremos de dar solución a estos dos problemas mediante la realización de nuestra componente *software*, como explicaremos en los dos capítulos siguientes. Resolviendo en el primero el problema de los contextos fijos y en el segundo el de la centralización de los algoritmos.

## Capítulo 2. Un modelo semántico para la generación de espacios de búsqueda

### 2.1 Web semántica

La web semántica es una extensión de la web actual, a la que dota de un mayor significado, tomando como objetivo una mejor definición de la información que facilitará la obtención de la misma por parte del usuario. Al añadir a la web metadatos semánticos se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común mediante la que se comparte, procesa y transfiere información haciendo, en cierto sentido, razonar a la propia web. Estos metadatos se proporcionan formalmente para que las máquinas de procesamiento puedan evaluarlos automáticamente.

Para lograr todo esto, la *web* semántica hace uso de varios componentes, de los cuales podemos destacar como esenciales *RDF*, *SPARQL* y *OWL*, mecanismos destinados a convertir a la web en una infraestructura global dónde todos los datos sean reutilizables e interpretables por un gran número de aplicaciones. La función de cada componente es la siguiente:

- *RDF*: se encarga de proporcionar información descriptiva sobre los recursos existentes en la *web*. Hablaremos más a fondo de *RDF* a continuación.
- *SPARQL*: es el lenguaje de consulta sobre las sentencias de *RDF*, facilita las búsquedas sobre los recursos de la *web* semántica.
- *OWL*: es un mecanismo de desarrollo de vocabularios específicos para algunos de estos recursos. *OWL* proporciona un lenguaje de definición de ontologías.

### 2.2. Resource Description Framework

El Marco de Descripción de Recursos (a partir de ahora *RDF*, del inglés *Resource Description Framework*) es un lenguaje de representación de información acerca de recursos en la *Web*. En especial está orientada a la representación de metadatos sobre recursos *web*, tales como el título, autor y fecha de modificación de una página *web*, derechos de autor e información de la licencia de un documento en la red, o el calendario de disponibilidad de un recurso compartido. En cualquier caso, generalizando el concepto de “recurso *web*”, *RDF* puede usarse también para representar información sobre cosas que pueden ser identificadas en la red, incluso cuando no pueden ser recuperadas directamente de ella. Algunos ejemplos podrían ser información al respecto de artículos disponibles en instalaciones de compras *on-line*, o la descripción de las preferencias de envío de un usuario de las mismas.



*RDF* está pensado para situaciones en las cuales esta información tiene que ser usada por aplicaciones, en lugar de ser únicamente vista por gente. *RDF* provee un marco común para expresar esta información para que pueda ser intercambiada por aplicaciones sin pérdida de significado. Como es un marco común, los diseñadores de aplicaciones pueden habilitar la disponibilidad de *parsers* y herramientas de procesado de *RDF* comunes. La habilidad de intercambiar información entre distintas aplicaciones significa que la información puede ponerse a disposición de aplicaciones para las que no fue creada.

*RDF* se basa en la idea de identificar las cosas usando identificadores *web* (llamados *Uniform Resource Identifiers* o *URIs*), y describir los recursos en forma de propiedades simples y valores de las propiedades. Esto permite a *RDF* representar sentencias simples sobre recursos en forma de grafos de nodos y aristas representando los recursos y sus propiedades y valores.

*RDF* también proporciona una sintaxis basada en *XML* (llamada *RDF/XML*) para almacenar e intercambiar estos grafos, que es la que usaremos para transformar en *SQLite*. Un ejemplo de esta sintaxis con datos de nuestro corpus sería el siguiente:

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:music="http://www.example.org/music#">

  <rdf:Description rdf:about="Ten">

    <music:genre>Grunge</music:genre>

    <music:artist>Pearl Jam</music:artist>

    <music:year>1991</music:year>

  </rdf:Description>

</rdf:RDF>
```

## 2.3. Escribiendo sentencias sobre recursos

### 2.3.1. Conceptos básicos

Como hemos dicho antes, *RDF* se usa para proporcionar una forma simple de crear sentencias sobre recursos de red, como páginas *web*. Esta sección describe las ideas básicas que explican la forma en que *RDF* proporciona estas habilidades.

*RDF* se basa en la idea de que las cosas descritas tienen propiedades que tiene valores, y que los recursos pueden ser descritos mediante sentencias, similares a las de arriba, que especifican esas propiedades y valores. *RDF* usa una terminología particular para hablar de las distintas partes de las sentencias. Específicamente, la parte que identifica la cosa de la

que habla la sentencia se llama sujeto. La parte que identifica la propiedad o característica del sujeto que especifica la sentencia se llama el predicado, y la parte que identifica el valor de esa propiedad se llama el objeto.

### 2.3.2. El modelo RDF

Las sentencias se pueden modelar en *RDF* como nodos y arcos en un grafo. Según el modelo de grafos de *RDF*, una sentencia se representa mediante un nodo para el sujeto y el objeto y un arco dirigido desde el nodo sujeto al nodo objeto para el predicado. La figura 1 muestra el grafo correspondiente al trozo de código *XML/RDF* de la sección anterior.

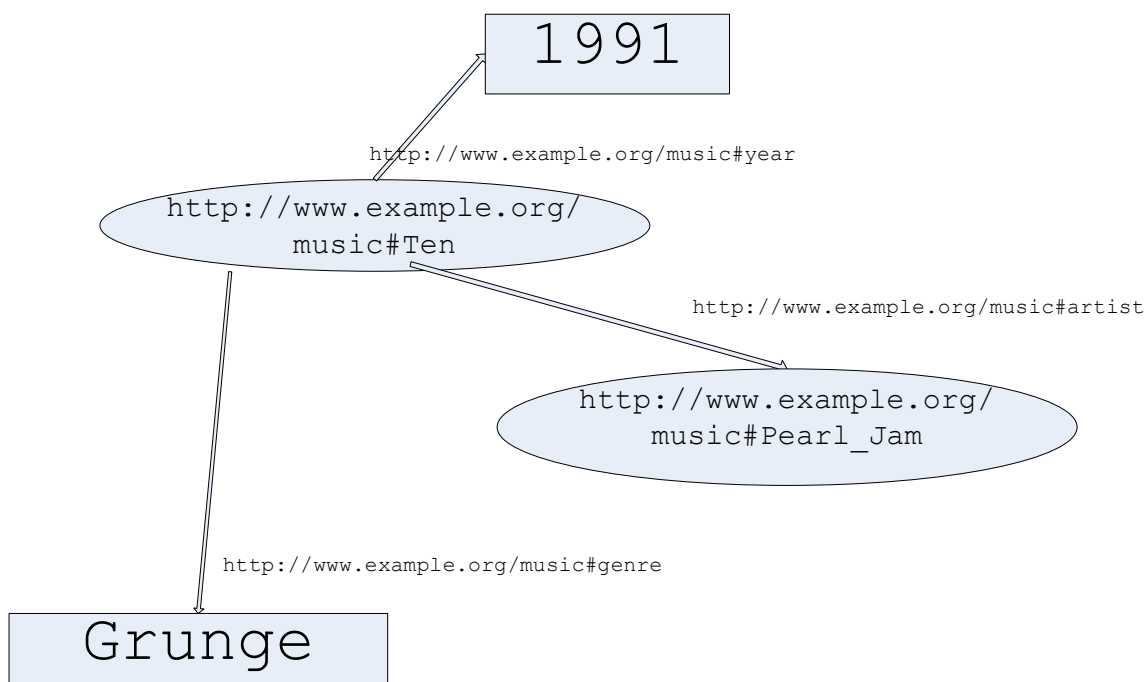


Figura 1. Varias sentencias sobre un recurso

Las *URIrefs* (*URI reference*) que representan las elipses del grafo, son *URIs* junto con unos identificadores al final. Las *URIrefs* son la forma habitual de representar sujeto, predicado y objeto de una sentencia.

Como se puede ver en la figura anterior, dos de los nodos objeto figuran como rectángulos en lugar de elipses. Esto se debe a que esos nodos tienen valores constantes (llamados literales) que son representados por cadenas de caracteres. Los literales no pueden ser usados como sujetos o predicados en *RDF*.

Cuando no es conveniente el uso de grafos para escribir sentencias se pueden usar **tripletras**. En esta notación cada sentencia del grafo se escribe como una simple tripleta de sujeto, predicado y nodo.

```
<http://www.example.org/music#Ten><http://www.example.org/
music#genre>"Grunge"
```

```
<http://www.example.org/music#Ten><http://www.example.org/
music# year>"1991"
```

```
<http://www.example.org/music#Ten><http://www.example.org/
music#artist><http://www.example.org/music#Pearl_Jam>
```

La notación completa en tripletas requiere que las *URIs* se escriban completamente, entre los símbolos ‘<’ y ‘>’, lo cual puede hacer, como se observa en el ejemplo anterior, que se escriban líneas muy largas en una página. Para mejorar esto, se suele usar una forma más corta de escribir tripletas (que también es usada en otras especificaciones, como veremos más adelante). Estas abreviaciones sustituyen un nombre cualificado *XML* (*qualified name* o *QName*, en inglés) sin los símbolos ‘<’ y ‘>’ por una abreviación de la *URIref* completa. Un *QName* contiene un prefijo que se asigna previamente a una *URI* de un espacio de nombres, dos puntos y un nombre local. Algunos ejemplos de *QNames* muy extendidos serían:

El prefijo `rdf:` para la *URI* del espacio de nombres:

```
http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

El prefijo `rdfs:` para la *URI* del espacio de nombres:

```
http://www.w3.org/2000/01/rdf-schema#
```

El prefijo `ex:` para la *URI* del espacio de nombres: `http://www.example.org/`

## 2.4. Sintaxis para RDF: RDF/XML

Como se comentó en el punto anterior, el modelo conceptual de *RDF* es un grafo. *RDF* proporciona una sintaxis *XML* para la escritura e intercambio de grafos *RDF*, llamada *RDF/XML*. Al contrario que las tripletas, que son una especie de notación taquigráfica, *RDF/XML* es la sintaxis normativa para *RDF*.

El código de la página 18 nos da un ejemplo del uso de *RDF/XML*. En este caso, los nodos de la **Figura 1** están también representados en él. Lo reproduciremos de nuevo a continuación para poder comentar sus componentes.

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:music="http://www.example.org/music#">

  <rdf:Description rdf:about="Ten">

    <music:genre>Grunge</music:genre>
```

```

    <music:artist>Pearl Jam</music:artist>

    <music:year>1991</music:year>

  </rdf:Description>

</rdf:RDF>

```

En la primera línea tenemos la declaración de *XML*, que indica básicamente que el código está escrito en *XML*, y la versión del mismo usada.

La segunda línea comienza con un elemento `rdf:RDF`. Esto quiere decir que el contenido *XML* que viene a continuación y que terminará con el `</rdf:RDF>` de la última línea trata de representar *RDF*. Justo después del `rdf:RDF`, en la misma línea hay una declaración de un espacio de nombres *XML*, representada como un atributo `xmlns` de la etiqueta inicial `rdf:RDF`. Esta especificación viene a significar que todas las etiquetas de este contenido que tengan el prefijo `rdf:` son parte de un espacio de nombres identificado por el *URIref* `http://www.w3.org/1999/02/22-rdf-syntax-ns#`. Las *URIrefs* que empiezan con `http://www.w3.org/1999/02/22-rdf-syntax-ns#` son usadas como términos del vocabulario *RDF*. Después de esa expresión, hay otra también representada como un atributo `xmlns` de la etiqueta inicial `rdf:RDF`, y especifica que el *URIref* `http://www.example.org/music#` se asociará con el prefijo `music:`. Los *URIrefs* que empiezan con la cadena `http://www.example.org/music#` se usan en nuestro caso para los términos de la base de datos que usaremos de ejemplo. El “>” al final de la línea indica el final de la etiqueta inicial `rdf:RDF`. Estas dos primeras líneas son necesarias para indicar que el contenido es *RDF/XML* y para identificar el espacio de nombres usado dentro de este contenido *RDF/XML*.

Las líneas de la tercera a la séptima proporcionan el código *RDF/XML* para las sentencias representadas en la Figura 1. Una forma sencilla de hablar de una sentencia *RDF* es decir que es una descripción, y que trata del sujeto de la sentencia (en este caso, trata de `http://www.example.org/music#Ten`), y esta es precisamente la forma en que *RDF/XML* representa las sentencias. La etiqueta inicial `rdf:Description` en la tercera línea indica el inicio de una descripción de un recurso, y continúa identificado el recurso acerca del cual trata la sentencia (que es el sujeto de la sentencia) usando el atributo `rdf:about` para especificar el *URIref* del recurso sujeto. La cuarta, quinta y sexta línea proporcionan un elemento de propiedad distinto cada uno, con los *QNames* `music:genre`, `music:artist` y `music:year` respectivamente como sus etiquetas, para representar el predicado y el objeto de la sentencia. El *QName* `music:genre` se elige de forma que al agregar el nombre local `genre` a la *URIref* del prefijo `music:` (`http://www.example.org/music#`) obtenemos la *URIref* del predicado de la sentencia `http://www.example.org/music#genre`. El contenido de este elemento de propiedad es el objeto de la sentencia, en este caso el literal `Grunge` (el valor de la propiedad género del recurso sujeto). El elemento de propiedad se anida dentro del elemento contenedor `rdf:Description`, que indica que esta propiedad se aplica al

recurso especificado en el atributo `rdf:about` del elemento `rdf:Description`. Esto se puede aplicar para cada una de las líneas de la cuarta a la sexta. La séptima línea indica el final de este elemento `rdf:Description`.

Para terminar, la octava línea indica el final del elemento `rdf:RDF` comenzado en la segunda línea.

Una base de datos puede por tanto describirse mediante *RDF/XML* creando un segmento `rdf:Description` para cada uno de los elementos que pueda contener esta base de datos. Cada línea dentro de esa etiqueta `rdf:Description` contendrá a su vez un atributo del elemento en cuestión así como el valor de ese atributo para el elemento tratado.

## 2.5. SQLite

Para facilitar el uso de los datos provenientes de archivos *RDF/XML* en nuestra componente, se decidió, como primer paso, la transformación de esos archivos en una base de datos relacional. El tener una base de datos relacional nos permite el uso de librerías ya fácilmente incorporables en el *Framework .NET* para acceder a los datos.

De entre los distintos sistemas de gestión de bases de datos relacionales sobre los cuales puede trabajar *.NET*, se decidió el uso de *SQLite*. *SQLite*, que es un proyecto de dominio público creado por *Richard Hipp* [2], está contenido en una librería en C de aproximadamente 500 kb, pudiendo llegar a bajar de 250 kb.

Al contrario que la mayoría de los sistemas de gestión de bases de datos, *SQLite* no tiene un proceso servidor separado, sino que lee y escribe directamente en archivos. Una base de datos *SQL* completa, con múltiples tablas, índices, disparadores y vistas, se contiene en un solo archivo. Además el formato del archivo que contiene la base de datos es independiente de la plataforma en que se use, motivos estos por los cuales se tomó este sistema para la creación de nuestra base de datos.

El hecho de que en su tercera versión se puedan usar bases de datos de hasta dos Terabytes de tamaño y se permita la inclusión de campos de tipo BLOB, no hace más que reforzar nuestra decisión.

## 2.6. SemWeb

Para poder almacenar en una base de datos *SQLite* nuestros datos, contenidos inicialmente en un archivo *RDF/XML*, formato visto más arriba, tratamos de buscar una herramienta, a ser posible de código libre, que pudiese ser incorporada en aplicaciones creadas en *.NET* como nuestra componente, y tuviese la suficiente potencia como para poder procesar bases de datos de un tamaño relevante. Y justo esto es lo que *SemWeb* nos ofrece.

*SemWeb* [3] es una librería creada para *Web Semántica/RDF* y escrita en *C#* para *.NET 1.1/2.0*. Esta librería puede ser usada para leer y escribir *RDF* (tanto en *XML*, como en una notación llamada *N3*), almacenar de forma persistente *RDF* (en memoria, mediante *MySQL*, mediante *SQLite*, como sucederá en nuestro proyecto, etc.), hacer consultas a estos almacenamientos persistentes mediante equiparación de grafos y *SPARQL* (que es una recomendación para crear un lenguaje de consulta dentro de la *Web* semántica), y para hacer consultas *SPARQL* a terminales remotas.

En nuestro caso, hacemos uso tan sólo de una pequeña porción de esta librería, como hemos indicado previamente, aquella que se encarga del almacenado de *RDF* de forma persistente.

Para la creación de la base de datos nos bastará con hacer uso del siguiente método:

```
Store store =
    Store.Create("sqlite:rdf:Uri=file:music.sqlite;version=3");
```

El método `Create` es usado para crear varios tipos de orígenes de datos, como pueda ser, en este caso, una base de datos *SQLite*. Si bien la implementación de un almacén (*store*) en *SQLite* se encuentra en el ensamblado `SemWeb.SQLiteStore.dll`, una vez se hace uso de `Create` basta con tener este ensamblado disponible en tiempo de ejecución, no hace falta referenciarlo en tiempo de compilación.

Una vez ha sido creada la base de datos, se pueden añadir fácilmente sentencias *RDF* (que serán incluidas en varias tablas, como se verá posteriormente) mediante el método `Store.Add`, como se verá a continuación (si bien nuestra aplicación hará primero un análisis sintáctico del archivo *XML/RDF* que posteriormente incluirá en variables auxiliares de tipo `Literal` o `Entity`, para facilitar la comprensión del uso de los métodos hemos incluido la parte de creación de las mismas variables):

```
const string URI = "http://www.example.org/music#";
static readonly Entity genre = URI + "genre";
store.Add(new Statement("Ten", genre, (Literal)"Grunge"));
```

La primera línea del código anterior sirve sólo para facilitar la vista de la *URIref* usada.

En la segunda línea se crea un objeto `Entity`, una entidad *RDF* de valor `http://www.example.org/music#genre`, que como se dijo anteriormente, son las únicas que pueden ser utilizadas como sujeto o predicado.

En la tercera línea se añade al almacén una nueva sentencia, creada *in situ*, que tiene por sujeto la entidad `"Ten"`, como predicado la entidad `genre` creada en la línea anterior, y como objeto el literal `)"Grunge"`, creado en la misma línea a partir de la cadena de caracteres.

Para cada una de las sentencias incluidas en el archivo *RDF/XML* de que se disponga, se ejecutará el presente método.


Una vez ya se han terminado de añadir campos, la base de datos deberá ser cerrada mediante el método:

```
store.Dispose();
```

## 2.7. RDF en una base de datos SQLite

La creación de bases de datos relacionales en SQLite a partir de SemWeb no sólo es sencilla, sino que el resultado obtenido es de muy fácil interpretación, lo cual facilitará nuestras consultas posteriores.

Una base de datos SQLite con contenido RDF obtenida a partir del código SemWeb explicado en la sección anterior consta de tres tablas, una para las entidades, otra para los literales, y otra para las sentencias. Las siguientes figuras, obtenidas a partir del *SQLite Database Browser* [4] facilitarán la explicación de la base de datos.




id	value
1	2 Ten
2	3 http://www.example.org/music#genre
3	5 http://www.example.org/music#artist
4	7 http://www.example.org/music#year
5	9 À Découvrir Absolutement
6	13 A Feast of Wire
7	17 A State of Trance 2005
8	21 A State of Trance 2006
9	23 A State of Trance 2007
10	25 Achtung Baby
11	28 Actos Inexplicables
12	32 Adore
13	36 Aparejo de fortuna
14	39 Black Market Music
15	42 Cábales y Cicatrices
16	44 Cajas de música difíciles de parar
17	45 Canciones desde palacio

Figura 2. Tabla *rdf\_entities*

En la Figura 2 se puede observar la tabla de entidades (*rdf\_entities*). En esta tabla se añade una nueva entrada por cada nuevo objeto de tipo *Entity* que es añadido en el *store* correspondiente a la base de datos.

La tabla de entidades tiene dos campos. El campo *value* contiene el valor de la entidad correspondiente, que son las *URIrefs* de cada uno de las entidades de la base de datos. El campo *id* es un identificador dentro de la base de datos, que se usa para asociar entidades y literales a las sentencias, sin necesidad de repetir *URIrefs* o cadenas de caracteres respectivamente. El *id* es asignado a medida que se añade una nueva entidad o literal a la base de datos y es único. Puesto que la asignación del *id* es la misma para entidades y literales, un *id* dado puede identificar o bien a un literal, o bien a una entidad, y sólo a una.

Table: *rdf\_literals* 

<i>id</i>	<i>value</i>	<i>language</i>	<i>datatype</i>	<i>hash</i>
1	0 ver:1			
2	4 Grunge			dxlmVttxR6phUKJoFij4TKa824=
3	6 Pearl Jam			1EawJK94oAdOMPcneSEPAM1v74g=
4	8	1991		YAOJjd76jBhpFVa4VhruyLHxR6Y=
5	10 Alternative			5aHwwbsV6epZ5QI7AIzY0+dqBA=
6	11 Various Artists			s0w3VuHJJHhgkGmcg3sacRu1tx0l=
7	12	2004		GLEeHhgbp17Yn1Etm2auvdWqTog=
8	14 Indie Rock			TrusZFxxaFhrjr1vXqpPHHJW9wo=
9	15 Calexico			T/d2AEz5PVuny54iz3JlPlUeUlg=
10	16	2003		LQjCMbJf5k3lk7CgTV2HmzCY90s=
11	18 Trance			rkp/2widiXggAGjW0PhdK9cvtl=
12	19 Armin Van Buuren			gev6f8T a3UC6qHdYq7Q2j9+LeHM=
13	20	2005		au/gzm0b7VnioAeuxYIEzlwVY4Y=
14	22	2006		WqH5jl2mptmQ0wlCoyXhtlqxH/o=
15	24	2007		BePW4H2codxiJUw/Ne0/SDR+TuA=
16	26 Rock			zdVUgIQ8ex15U7gCSlvW/rq7X9w0=
17	27 U2			6p7C2WvFww9SYRxfvicm+varlQ=

Figura 3. Tabla *rdf\_literals*

En la Figura 3 se puede observar la tabla de literales (*rdf\_literals*). En esta tabla se añade una nueva entrada por cada nuevo objeto de tipo *Literal* que es añadido en el *store* correspondiente a la base de datos.

La tabla de literales tiene cinco campos, si bien son los dos con valores idénticos a la tabla *rdf\_entities* aquellos que realmente nos interesan y de los cuales haremos uso en nuestra componente. El campo *id* tendrá el mismo uso que en la tabla *rdf\_entities* y como hemos dicho será único tanto en la tabla, como en el conjunto de las tablas *rdf\_entities* y *rdf\_literals*. El campo *value* en este caso contendrá la cadena de caracteres correspondiente al valor del literal, que se habrá introducido como objeto para una, o varias, sentencias de la



base de datos. Los otros tres campos los omitimos, pues no serán utilizados en nuestra componente.

	subject	predicate	objecttype	object	meta
1	2	3	1	4	1
2	2	5	1	6	1
3	2	7	1	8	1
4	9	3	1	10	1
5	9	5	1	11	1
6	9	7	1	12	1
7	13	3	1	14	1
8	13	5	1	15	1
9	13	7	1	16	1
10	17	3	1	18	1
11	17	5	1	19	1
12	17	7	1	20	1
13	21	3	1	18	1
14	21	5	1	19	1
15	21	7	1	22	1
16	23	3	1	18	1
17	23	5	1	19	1

Figura 4. Tabla *rdf\_statements*

En la Figura 4 se observa la tabla de sentencias (*rdf\_statements*). En esta tabla se añade una nueva entrada por cada nuevo objeto de tipo *Statement* que es añadido en el *store* correspondiente a la base de datos.

La tabla de sentencias tiene cinco campos de los cuales tan sólo tres van a ser utilizados por nuestra componente. Los campos *subject*, *predicate* y *object* representan respectivamente el sujeto, predicado y objeto de la sentencia a la que hacen referencia. Cada uno de estos tres campos tiene un valor numérico que se corresponderá con el valor *id* de alguna entrada, bien en la tabla *rdf\_entities*, bien en la tabla *rdf\_literals*. Como habíamos dicho antes no puede haber un literal y una entidad que compartan valor de *id*, por tanto, el valor de cada uno de los campos corresponderá a un solo valor en una de las dos tablas.

Pongamos por ejemplo que queremos recuperar la sentencia a la que hace referencia la primera entrada de la tabla *rdf\_statement*. Como vemos tiene los valores **2**, **3**, y **4** para los campos *subject*, *predicate* y *object* respectivamente.

Si miramos las tablas *rdf\_entities* y *rdf\_literals*, vemos que el valor de *id* **4** se encuentra en la tabla *rdf\_literals*, mientras que los valores **2** y **3** se encuentran en la tabla *rdf\_entities*. Para **2** el valor del campo *value* es “Ten”, para **3** el valor es “http://www.example.org/music#genre” y para **4** vemos en la tabla *rdf\_literals*

el valor “Grunge”. Por tanto la primera sentencia de nuestra base de datos es, en forma de tripleta:

```
<Ten><http://www.example.org/music#genre>“Grunge”
```

Con el uso de tan solo estas tres tablas podemos representar todos los datos que nos puedan ser proporcionados a través de un código *RDF/XML*.

Esta representación, como se verá en el punto siguiente, facilita ostensiblemente la realización de consultas sobre nuestra base de datos.

## 2.8. Consultas y creación del espacio de búsqueda

### 2.8.1 Consultas sobre la base de datos *SQLite*

Una vez hemos obtenido la base de datos *SQLite* a partir del código *RDF/XML* ya tenemos la componente básica para la realización de consultas.

Las consultas que realizará nuestra componente se basarán en los posibles distintos predicados de las sentencias que compongan nuestra base de datos. Así, si nuestra base de datos es en esencia una serie de sentencias de forma

$$(\text{suje}_i, \text{predic}_i, \text{objeto}_i)$$

Dónde el subíndice indica la posición en la tabla *rdf\_statements* de la sentencia en cuestión, podemos representar nuestras consultas como un conjunto de términos de la forma:

$$(\text{pred}, \text{obj}, \text{rel})$$

Siendo *pred* uno de los predicados existentes en nuestra base de datos, *obj* el valor esperado del campo objeto de las sentencias cuyo predicado sea *pred*, y *rel* la relevancia, medida desde 0 a 100, del término  $(\text{pred}, \text{obj}, \text{rel})$  dentro de la consulta total.

La funcionalidad de la relevancia es dar más importancia a ciertas partes de una consulta, sobre el resto de ellas. El valor de la relevancia no ejerce ninguna influencia en el resultado final cuando la consulta realizada tiene un solo término.

Un ejemplo de consulta vendría dado a continuación:

```
Q = { (http://www.example.org/music#genre, "Grunge", 80),
      (http://www.example.org/music#artist, "Pearl Jam", 60),
      (http://www.example.org/music#artist, "Soundgarden", 40) }
```

La consulta anterior, de tres términos, se podría interpretar como la búsqueda de (en el caso del ejemplo) discos que tuviesen como género *Grunge* con una relevancia sobre cien de 80, como artista *Pearl Jam* con una relevancia de 60, y como artista *Soundgarden* con una relevancia de 40.

Como se puede comprobar, en una misma consulta puede haber varios términos que traten sobre el mismo predicado. Esto es debido a que la relevancia sólo expresa la importancia dentro de la consulta del término de la consulta, y no es una restricción dentro de los resultados obtenidos por la misma.

### 2.8.2. Creación de los nodos

La función de las consultas no es otra que la generación del espacio de búsqueda sobre el cual aplicar nuestro algoritmo OCH explicado en el primer capítulo. Para esto, las consultas tienen dos funciones. Primero nos ayudan en la creación de los nodos del grafo que forma el espacio de búsqueda, y una vez estos han sido creados, gracias también a la consulta, se dota de un beneficio y un coste a cada uno ellos.

Para la realización de la primera de las dos funciones, la de creación de los nodos, hacemos uso del conjunto de todos los predicados de la consulta, así, si tenemos que una consulta  $Q$  podría ser representada como un conjunto de forma:

$$Q = \{ (pred_1, obj_1, rel_1), (pred_2, obj_2, rel_2), \dots, (pred_n, obj_n, rel_n) \}$$

Para crear los nodos tendremos que tomar el conjunto  $\{pred_1, pred_2, \dots, pred_n\}$ , y con este conjunto realizar una consulta a la base de datos *SQLite* de forma que seleccionemos todos aquellos distintos sujetos que formen parte de sentencias en las que el predicado sea alguno de los predicados del conjunto.

Para cada uno de estos sujetos crearemos un nodo que tendrá como atributos los valores del objeto de cada una de las sentencias del mismo que tengan predicados pertenecientes a las consultas. Estos atributos serán añadidos para ser utilizados como datos a la hora de asignar a cada nodo una puntuación o beneficio, como veremos en el punto siguiente.

De esta forma, habremos obtenido todos los nodos pertenecientes al espacio de búsqueda del problema, omitiendo aquellos sujetos que no sean relevantes para la consulta en cuestión, pudiendo así disminuir el número de nodos creados.

### 2.8.3. Funciones de evaluación

Como explicamos en el capítulo referente a OCH, todos los nodos del espacio de búsqueda deben tener asignados una puntuación (beneficio) y un coste. Para la obtención de la puntuación, es necesaria una forma de evaluación de cada uno de los atributos de los nodos de nuestro espacio de búsqueda con respecto a cada uno de los términos de nuestra consulta.

Esta puntuación se hace mediante funciones de evaluación que deberán ser proporcionadas por el usuario de nuestra componente. Puesto que es él el que conoce el dominio de los datos sobre los que trabaja el sistema.

Así, para cada uno de los distintos predicados que existan en la base de datos utilizada por el usuario, debería haber una función de evaluación que, para dos valores distintos de objeto asociados a ese predicado, devuelva una estimación de la similitud entre ambos.

Estas funciones se usarán entonces para evaluar, para cada nodo del espacio de búsqueda, la similitud de los valores de sus atributos con los valores de los objetos de los términos de la consulta que se le correspondan.

Veamos un ejemplo. Pongamos que tenemos la consulta de la página 27:

```
Q = { (http://www.example.org/music#genre, "Grunge", 80),
      (http://www.example.org/music#artist, "Pearl Jam", 60),
      (http://www.example.org/music#artist, "Soundgarden", 40) }
```

Ahora trataremos de evaluar el nodo que surgiría de la descripción de la página 20:

```
<rdf:Description rdf:about="Ten">
    <music:genre>Grunge</music:genre>
    <music:artist>Pearl Jam</music:artist>
    <music:year>1991</music:year>
</rdf:Description>
```

Para evaluar, tomamos, para cada uno de los términos de la consulta, el valor de su objeto, y comparamos con el valor del nodo para el mismo predicado. Así llamaríamos a la función de evaluación de `genre` para los valores `Grunge` (de la consulta) y `Grunge` (del nodo), y en dos ocasiones a la función de evaluación de `artist`, para los valores `Pearl Jam` (de la consulta) y `Pearl Jam` (del nodo) en primer lugar, y otra vez para los valores `Soundgarden` (de la consulta) y `Pearl Jam` (del nodo).

Puesto que las funciones de evaluación, como hemos dicho antes, son una medida de la similitud (o mejor dicho, de la distancia entre ambos valores), será de esperar que el valor para las dos primeras llamadas retorne cero, y un valor distinto de cero para la última (puesto que los dos parámetros de la función son distintos).

#### 2.8.4. Obtención del Score

Las funciones de evaluación explicadas en el punto anterior son la esencia a la hora de calcular el *score*, que es el beneficio asociado, de un nodo. La fórmula mediante la cual se calcula este beneficio para un nodo dado  $k$  es la siguiente:

$$S_k = \sum^{|Q|} (100 - d(v_{q_i}, v_{k_i})) * r_{q_i}$$

Esto quiere decir que para un nodo  $k$ , el beneficio del mismo en relación a una consulta  $Q$  es el sumatorio, para cada uno de los términos de la consulta del valor resultado de restar a 100 la distancia entre los valores del objeto del término de la consulta y el del nodo para el atributo asociado al predicado del término. Y este resultado multiplicarlo por la relevancia del término.

Al medir las distancias, que son devueltas por las funciones de evaluación, de cero a cien, se tiene que para aquellos nodos que compartan valores de sus atributos con los términos, el beneficio será máximo. Y en cambio, para aquellos cuya distancia entre valores sea considerada la máxima, el beneficio puede llegar a resultar cero.

### 2.8.5. Generación del espacio de búsqueda

Una vez obtenido el beneficio de cada nodo del espacio de búsqueda, sólo necesitaremos obtener los costes asociados tanto a los nodos, como a las aristas del mismo.

Para esto, el usuario proporcionará una función más con una estructura similar a las funciones de evaluación. Se trata de la función de evaluación del peso.

Esta función constará de dos métodos que devolverán sendos enteros, medida del coste (peso) de una arista o un nodo. Para el primer caso, la función tendrá como parámetros los dos nodos a los que llegue la arista cuyo coste queremos obtener. En el segundo caso, habrá un único parámetro que será el nodo cuyo coste nos es necesario.

El uso de estas funciones para cada uno de los nodos, y cada una de las aristas, del grafo del problema terminarán por generar el espacio de búsqueda final sobre el que podrá ser aplicado nuestro algoritmo OCH.

A continuación veremos la forma en que nuestra componente se estructura. La sección en que la generación explicada en los puntos anteriores se lleva a cabo, la comunicación del espacio de búsqueda a los procesos encargados de llevar a cabo el cálculo OCH. Y una visión de la generación y transmisión del resultado final.

### 2.8.6. Modelo de clases

En la Figura 5 se puede observar el modelo de clases asociado a la estructura de las consultas y los nodos dentro de nuestra componente. Para facilitar su comprensión, a continuación se explicarán todos los elementos que lo componen.

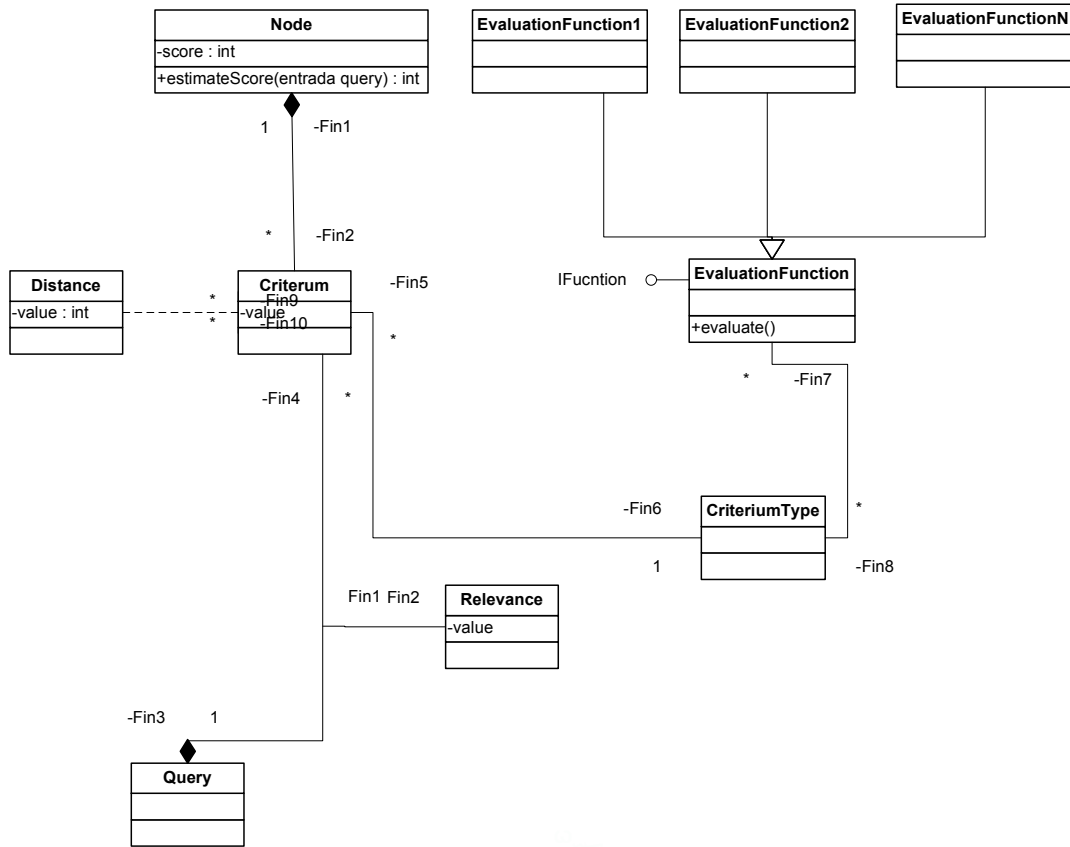


Figura 5. Modelo de clases

Para mejorar más si cabe la comprensión, dividiremos el modelo anterior en dos partes. Por un lado tendremos las clases relacionadas con las consultas, y por otro aquellas que son propias de los nodos.

La Figura 6 muestra las clases que contendrán toda la información que atañe a una consulta.

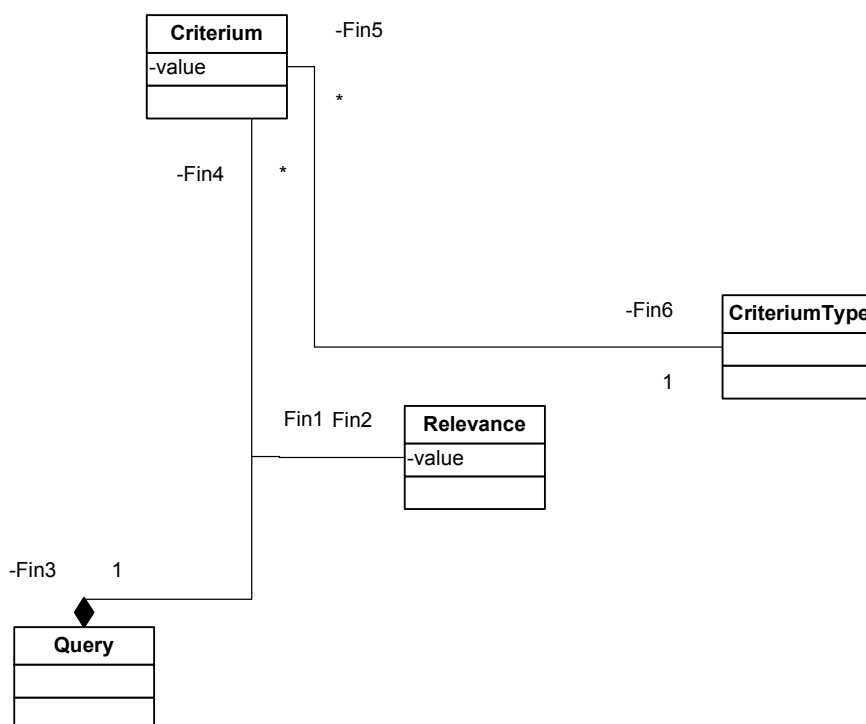


Figura 6. Modelo de clases de las consultas

Este modelo se compone de las siguientes clases:

- **Query**. Esta clase representa las consultas que el usuario pueda hacer a la base de datos *RDF* a través de nuestra componente.
- **Relevance**. Esta clase representa la relevancia que un término de una consulta tendrá con respecto al resto de los términos. Su valor, que debe fluctuar entre 0 y 100, hará que los términos puedan tener distinto peso dentro del resultado de la consulta. Cada término de una consulta tiene una *Relevance*.
- **Criterium**. Esta clase representa el objeto que tiene un término de una consulta. Su rango de valores vendrá dado por el *CriteriumType* (o predicado). Cada término de una consulta tiene un *Criterium*.
- **CriteriumType**. Esta clase representa el predicado que tiene un término de una consulta. Su rango de valores viene dado por los distintos predicados que tenga la base de datos *RDF* usada por la componente, y el valor que un *CriteriumType* en cuestión tenga condicionará el valor del *Criterium* de ése mismo término. Cada término de una consulta tiene un *CriteriumType*.

La Figura 7 muestra las clases que contendrán la información que corresponde a los nodos.

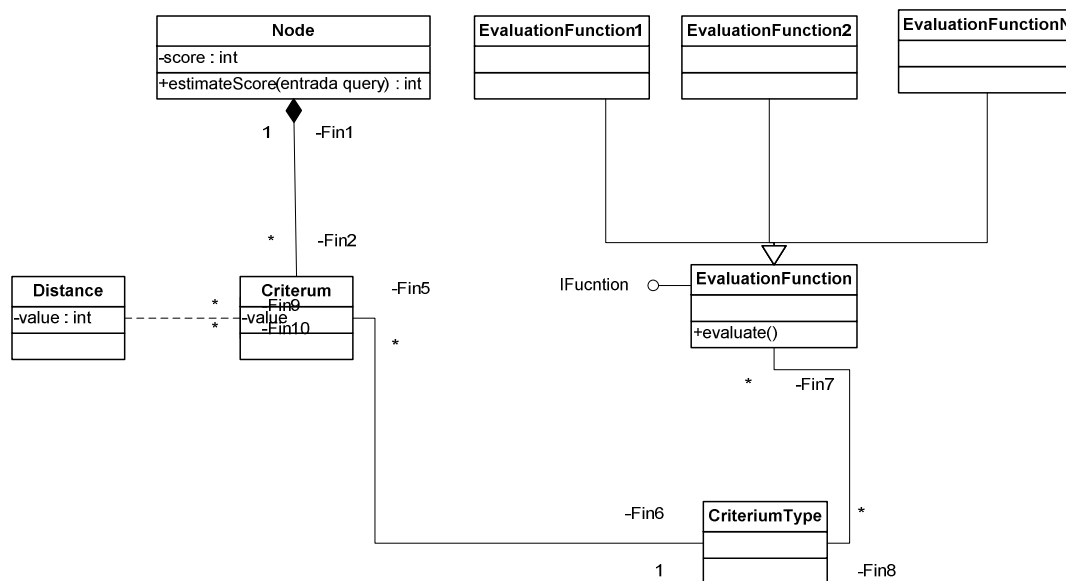


Figura 7. Modelo de clases de los nodos

Las distintas clases presentadas son:

- **Node**. Esta clase representa los nodos del espacio de búsqueda del problema. Un nodo es creado por cada sujeto en la base de datos *RDF*, y cada uno de ellos tendrá un beneficio (o *score*) asociado; calculado como hemos visto anteriormente.
- **Criterium**. Esta clase representa, como hemos explicado anteriormente los valores de los objetos, pero en este caso no de las consultas, sino de cada uno de los nodos (sujetos) para cada valor distinto de predicado que tengan.
- **CriteriumType**. Esta clase representa, como hemos explicado anteriormente los distintos valores de los predicados de la base de datos *RDF*. Cada *Criterium* de uno nodo, tendrá su *CriteriumType* asociado.
- **Distance**. Esta clase representa, para dos *Criterium* distintos, el valor de la distancia que hay entre ambos, medido de cero a cien.
- **EvaluationFunction(s)**. Estas clases representan las distintas funciones de evaluación que el usuario proporcione para una base de datos determinada. Si bien las funciones de evaluación deberán seguir una interfaz ya definida, cada una podrá obtener los resultados de cualquier forma, ya que esto es transparente a nuestra componente.



## Capítulo 3. Un modelo distribuido para la resolución de algoritmos OCH

### 3.1. Arquitectura del sistema

Nuestro sistema consta de tres partes diferenciadas, cuyas funciones e interconexión trataremos de explicar en profundidad en el transcurso del presente capítulo.

La primera de las componentes de nuestro sistema es el *MultimediaQuerier*, que será el encargado de leer la base de datos *RDF* y las consultas del usuario, creando así, con la ayuda de las funciones de evaluación de los criterios y del peso los nodos que formarán el espacio de búsqueda del sistema. Siendo la esencia del capítulo anterior, ya fue explicada en éste, y aquí tan solo referenciaremos aquello que ya comentamos anteriormente.

La segunda es el *MasterQueenService*, servicio *web* que será utilizado por ambas componentes, de forma transparente al usuario, y que hará las veces de comunicador entre el *MultimediaQuerier* y todos los *HiveClient* activos. Además está pensado este servicio *web* para gestionar el uso de los distintos *HiveClient* cuando el tamaño de espacio de búsqueda sea tan grande que una partición del mismo se haga necesaria para mejorar la calidad y la eficiencia en el cálculo de las soluciones mejores mediante el algoritmo OCH que usaremos.

La tercera componente es el *HiveClient*, se trata de procesos clientes que serán los encargados de aplicar el algoritmo OCH al espacio de búsqueda que le proporcionará el servicio *web MasterQueenService*. El número de clientes *HiveClient* dependerá del usuario de nuestro sistema, y el uso de los mismos dependerá del tamaño del espacio de búsqueda, y de la disponibilidad de cada uno en el momento de la solicitud de trabajo por parte del servicio *web*.

#### 3.1.1. Funcionamiento del sistema

Como hemos comentado en el punto anterior, nuestro sistema consta de tres componentes diferenciadas que interactúan entre sí en lo que sería el transcurso de una ejecución del mismo.

Las consultas son en sí la esencia de nuestro sistema, puesto que es la creación de una nueva consulta la que lanzará todo el proceso posterior de interacción entre las distintas componentes para obtener una solución final a la consulta que sea computacionalmente satisfactoria.

Una vez las funciones de evaluación y de cálculo del peso ya se encuentran en el directorio de ejecución correspondiente, y hemos transformado mediante *SemWeb* el archivo *RDF/XML* en una base de datos *SQLite* podremos empezar, a través del *MultimediaQuerier* la ejecución de nuestra aplicación.

La consulta que queramos realizar se deberá introducir término a término siguiendo la estructura explicada en el segundo capítulo del presente trabajo. Una vez esta consulta sea

definida deberá lanzarse la ejecución, que comenzará por la búsqueda en la base de datos de los sujetos relevantes para la consulta.

Una vez se tengan estos sujetos se procederá a la creación de los nodos correspondientes. Todo esto dentro del *MultimediaQuerier*. Una vez tengamos toda la estructura de nodos, que no es más que el espacio de búsqueda del problema representado mediante la consulta, se procederá a la creación de un grafo que pueda ser interpretado por nuestro algoritmo OCH. Este grafo será almacenado en un archivo de texto, para facilitar su almacenamiento.

Una vez este grafo haya sido creado se llamará a un método del servicio *web MasterQueenService* para registrar la localización del archivo (o archivos en caso de que el problema tenga un tamaño considerable); para terminar por hacer una llamada a otro método del servicio *web* que será el que se encargue de obtener el camino final.

En este momento el control pasará al servicio *web MasterQueenService*. Este servicio tiene dos funciones principales. Por una parte se encarga de la gestión de todas las instancias de *HiveClient* que puedan estar en ejecución en cada momento. Por otra parte será también quien envíe solicitudes a estos clientes de obtención de soluciones a los problemas que haya podido recibir de distintas instancias de *MultimediaQuerier*, devolviendo a posteriori a estos mismos las mejores soluciones obtenidas.

*HiveClient* contiene la implementación del algoritmo de hormigas. Son estos clientes los encargados del cálculo y obtención de las soluciones del problema. Han sido programados de forma que puedan trabajar de forma distribuida, cada uno de ellos con el mismo problema, distintos problemas, o distintas secciones de un mismo problema, para aquellos espacios de búsqueda de mayor tamaño.

## 3.2. MultimediaQuerier

### 3.2.1. Funcionamiento del MultimediaQuerier

Como hemos contado más arriba el *MultimediaQuerier* se encargará de la creación del grafo que contendrá el espacio de búsqueda asociado a cada consulta que se le realice sobre la base de datos *RDF/XML* previamente convertida a *SQLite*.

La estructura y funcionamiento del mismo ha sido explicada con extensión en el segundo capítulo, a la hora de describir nuestro modelo independiente de datos, con lo cual no repetiremos aquí el contenido del mismo.

## 3.3. MasterQueenService

### 3.3.1. Funcionamiento del MasterQueenService

El servicio web *MasterQueenService* realiza las funciones de comunicación entre el *MultimediaQuerier*, que definirá la consulta a realizar y por tanto el espacio de búsqueda, y los distintos clientes *HiveClient*, que serán los encargados de obtener una solución mediante OCH a esa consulta expuesta; también pondrá a disposición de cada una de estas dos componentes aquellos métodos que, para la facilitación de la comunicación recién comentada, sean necesarios.

Así, por lo tanto, *MasterQueenService* proporcionará a *MultimediaQuerier* un método de registro de los espacios de búsqueda obtenidos a partir de las consultas realizadas, y otro método de solicitud de obtención de una solución a partir de estos mismos espacios de búsqueda.

Por otro lado *MasterQueenService* proveerá a cada uno de los clientes *HiveClient* los métodos necesarios para su registro en la súper-colonia de hormigas que hará de “contenedor virtual” de los clientes en el servicio *web*; para el envío de los comandos a realizar por los clientes de manera independiente; para la posible devolución posterior de los resultados que los clientes pudiesen obtener; y para la desvinculación final de los clientes con el servicio *web* una vez su labor se haya dado por terminada.

Además de estas funciones, el servicio *web* tendrá, en la figura de las súper-reinas (*SuperQueen*) una clase que se encargará de la posible integración de las soluciones otorgadas por los clientes asociados a ellas en el caso de que éstos, por ejemplo, trabajasen con distintas secciones de un problema de mayor envergadura.

### 3.3.2. Modelo de clases

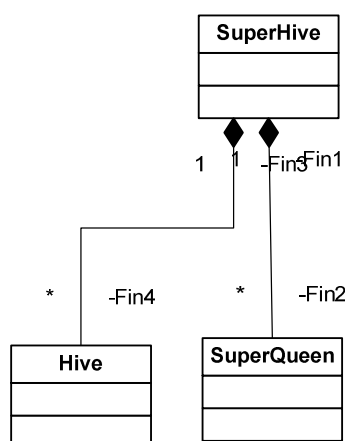


Figura 8. Modelo de clases de MasterQueenService

Las distintas clases presentadas son:

- **SuperHive.** Esta clase representa una estructura de súper-colmena, encargada de gestionar los distintos clientes *HiveClient* asociados al servicio *web*

*MasterQueenService*. Además contendrá también información de todas las súper-reinas del sistema, las cuales serán asignadas a un conjunto de clientes cada una.

- **Hive.** Esta clase representa las colmenas que serán en nuestra aplicación cada uno de los clientes *HiveClient* que hayan sido registrados en la súper-colmena *SuperHive*. Ellos recibirán la información relativa al espacio de búsqueda y resolverán el problema aplicando el algoritmo OCH descrito.
- **SuperQueen.** Esta clase representa la estructura de súper-reina que estará asociada a un conjunto de clientes registrados bajo su identificador. Su labor será la de integrar las distintas soluciones proporcionadas por cada uno de los clientes en una solución global que el servicio *web* pueda devolver a aquél proceso que realizase la consulta correspondiente.

### 3.3.3. Métodos del servicio web

Los métodos que el servicio *web* *MasterQueenService* proporciona son seis, y son utilizados por las otras dos componentes de nuestro sistema. El uso que del servicio hacen ambas componentes es bien diferenciado, y por tanto los métodos del mismo no serán nunca usados por ambas componentes.

Los primeros cuatro servicios hacen referencia a la comunicación del servicio *web* *MasterQueenService* con los clientes *HiveClient*. Tratarán de establecer una secuencia de actuación a la hora de trabajar con estos clientes, y su funcionamiento queda reflejado en las siguientes líneas:

- **hiveAlive.** Éste es el primer método utilizado por los clientes, *HiveClient*. La llamada al método se hace por parte de estos clientes en el momento en que quieren registrarse al servicio *web* *MasterQueenService* como clientes disponibles para posibles consultas.

El método devuelve un entero, identificador del cliente en la súper-colmena (*sHive*) que se encarga de gestionar los clientes en el servicio *web*. Este identificador será único para cada uno de los *HiveClients* registrados en el servicio *web*, y será utilizado por otros métodos del servicio.

- **hiveReady.** El método *hiveReady* es utilizado por los clientes cuando, una vez ya registrados en el servicio *web* mediante *hiveAlive*, quieren solicitarle al mismo la secuencia de comandos que deben llevar a cabo durante su ejecución.

Estos comandos habrán sido introducidos en la cola de comandos correspondiente al identificador del cliente en cuestión, de aquí la necesidad del uso del identificador, y serán transferidos al cliente para que pueda interpretarlos y ejecutarlos secuencialmente.

En caso de que el cliente en cuestión no tenga comandos asociados en su cola, un comando nulo será transferido al mismo.

- ***hiveCommandsDone***. El método *hiveCommandsDone* es utilizado por los clientes cuando quieren avisar al servicio *web* de que todos los comandos que previamente, mediante *hiveReady* le habían llegado, han sido ya realizados.

Este método tiene dos parámetros. Por un lado se le dirá qué reina está asociada al cliente que haga la llamada y por otro lado se le dará la solución (en caso de haberla) que haya obtenido el cliente al aplicar el algoritmo de colonias de hormigas, que será facilitado a esa reina (*SuperQueen*), para una posible integración en el caso de tener un espacio de búsqueda lo suficientemente grande como para que haya sido fraccionado entre distintos clientes.

- ***hiveDead***. El método *hiveDead* es el último de los métodos del servicio *web* *MasterQueenService* que será utilizado por los clientes *HiveClient*. Éste método servirá para avisar al servicio *web* de la *muerte* del proceso cliente.

Este aviso hará que el cliente correspondiente, marcado por su identificador numérico, sea eliminado de la cola de colmenas que manejará el servicio *web* en su clase *SuperHive*. Evitando así que posibles nuevos comandos sean añadidos a la cola de este cliente, y por tanto se queden sin ejecutar.

Una vez estudiados los métodos del servicio *web* *MasterQueenService* que serán utilizados por los clientes *HiveClient* pasaremos a ver los dos métodos que serán utilizados por la componente *MultimediaQuerier*. En este caso los métodos tratarán de solventar la localización de los grafos correspondientes al espacio de búsqueda del problema y la posterior solicitud del *MultimediaQuerier* de obtención de una solución para la consulta que haya sido formulada. Una explicación del funcionamiento de ambos métodos viene a continuación:

- ***registerGraph***. El método *registerGraph* es un método del servicio *web* *MasterQueenService* utilizado por la componente *MultimediaQuerier*. Es la forma en la cual el *Querier* comunica al servicio *web* la localización del (o de los) grafo que contiene el espacio de búsqueda.

Esta información es almacenada por la clase *SuperHive* dentro de un diccionario al que accederá en el momento en que facilite a cada cliente información al respecto de dónde se encuentra el grafo del problema sobre el que debe aplicar el algoritmo de colonias de hormigas.

- ***obtainPath***. El método *obtainPath* es un método del servicio *web* utilizado por la componente *MultimediaQuerier* para enviar al servicio la solicitud de obtención de un resultado.

Esta solicitud se hará proporcionando el nodo inicial y el final, el  $TMax$ , que es el coste máximo al que puede llegar la solución y el identificador del espacio de búsqueda que habrá devuelto la llamada al método *registerGraph*.

La llamada a *obtainPath* hará que *SuperHive* asigne a esa llamada una *SuperQueen*, que será la encargada de añadir a la cola de comandos aquellos que hagan que los clientes asociados a la *SuperQueen* realicen la ejecución del algoritmo de colonias de hormigas con los datos adecuados.

Para terminar *obtainPath* devolverá a la componente *MultimediaQuerier* la solución obtenida y seleccionada por la *SuperQueen* encargada.

## 3.4. HiveClient

### 3.4.1. Funcionamiento del HiveClient

*HiveClient* representa cada uno de los clientes distribuidos de los que puede hacer uso el servicio *web MasterQueenService* a la hora de resolver las consultas que hayan sido formuladas en la componente *MultimediaQuerier*.

La función principal de cada uno de los clientes *HiveClient* será la de poner en ejecución cierta cantidad de hormigas durante un número determinado de iteraciones (que llamaremos generaciones) cada una de las cuales obtenga mediante un algoritmo OCH una solución para el problema representado por el espacio de búsqueda.

Después de cada generación, la reina (*Queen*) del cliente deberá seleccionar la mejor solución de entre las obtenidas para todas las hormigas y sobre esa solución realizar la actualización offline correspondiente.

Una vez todas las generaciones a realizar hayan terminado, el mejor resultado final será el que devuelva el cliente al servicio *web*.

### 3.4.2. Modelo de clases

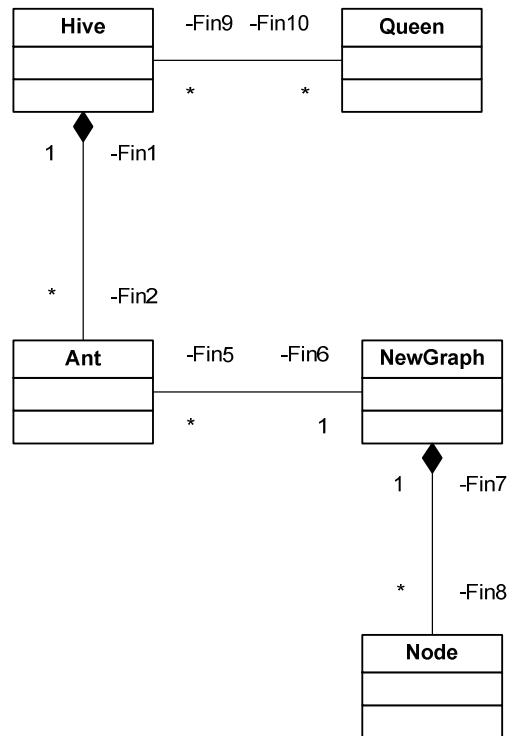


Figura 9. Modelo de clases de HiveClient

Las distintas clases presentadas son:

- **Hive.** Esta clase representa el núcleo de las aplicaciones cliente *HiveClient*. Por un lado es la encargada de la generación de las llamadas a los métodos del servicio web *MasterQueenService* en relación al registro del cliente y de la posterior solicitud de comandos a realizar.

Una vez estos comandos sean recibidos, es función también de la clase *Hive* la interpretación de los mismos, realizando, dependiendo del comando en cuestión, la acción prevista asociada a éste. Como pueda ser la ejecución del algoritmo OCH implementado.

También se encargará *Hive* de realizar la llamada a los métodos del servicio web *MasterQueenService* asociados con la entrega de resultados una vez terminado el cálculo de los mismos y de la desvinculación del cliente con el servicio web una vez toda su labor haya sido llevada a cabo exitosamente.

- **Queen.** Esta clase representa la reina de la colmena que es el proceso *HiveClient*.

Su función principal será la de comparar las distintas soluciones dadas por cada una de las hormigas de la colmena en una generación dada, y extraer de ellas aquella que sea mejor.

- **Ant.** Esta clase representa la esencia de nuestro trabajo. Las hormigas de la colmena formada por cada proceso cliente *HiveClient*. Ellas serán las encargadas de aplicar el algoritmo OCH propuesto para obtener soluciones sobre el espacio de búsqueda proporcionado a cada uno de los clientes.

Cada una de las hormigas trabajará en paralelo en hilos distintos dentro de la ejecución del programa cliente. Será la clase *Hive* la encargada de poner en ejecución cada una de las instancias de *Ant* que vayan a ser lanzadas. El número de hormigas en acción vendrá dado por una variable de *Hive* llamada *NUMANTS*.

La comunicación de la solución al hilo principal se hará a través de un *Delegate* desde cada uno de los hilos *Ant*.

- **NewGraph.** Esta clase representa el grafo del espacio de búsqueda del problema a resolver por el cliente *HiveClient* correspondiente. Contendrá este espacio de búsqueda en una matriz simétrica (además de en una lista de nodos) que habrá creado a partir del archivo de texto creado por *MultimediaQuerier* y cuya localización le habrá sido proporcionada por el servicio *web MasterQueenService*.

Además de la creación del grafo del espacio de búsqueda inicial, una instancia de *NewGraph* deberá contener en todo momento el grafo actual utilizado por cada instancia de *Ant*, siendo este modificado por cada hormiga mediante las actualizaciones *offline* y *online*.

También dentro de esta clase se obtendrá el tipo de transición que una hormiga deberá dar en un momento dado a partir de un nodo para alcanzar otro dentro de una solución, ya sea esta por exploración o por explotación.

- **Node.** Esta clase representa cada uno de los nodos del espacio de búsqueda del problema o de la sección del problema asignada al cliente *HiveClient* correspondiente.





## Capítulo 4. Caso de estudio

### 4.1. Una base de datos de discos

A la hora de tratar de probar la validez de los resultados obtenidos por nuestra infraestructura nos decidimos por la utilización de una base de datos *RDF* que contuviese una serie de discos de música e información relativa a ellos.

Así, las consultas que se podrían hacer sobre esta base de datos consistirían en crear listas de reproducción que se adaptasen en la medida de lo posible a las especificaciones dadas por el usuario, ciñéndose siempre a una duración máxima de estas listas expresada en minutos.

Un ejemplo de la utilidad de la creación de listas de reproducción basándose en ciertos criterios viene dado en [5]:

“Imagina la siguiente situación: una persona deportista sale de su casa pronto para su carrera matutina diaria. Esta persona siempre hace deporte con música, pero no le gusta escuchar la misma música una vez y otra. Así que tiene una gran colección de música en su reproductor portátil.

>> Elige la música que se ajusta a su estado de ánimo actual, sin tener que retrasar el ejercicio: mientras está corriendo, navega por la colección. Después de oír la canción que ha elegido, el reproductor seguirá con temas similares. Al final de la ruta, elige de improviso algunos sonidos relajantes mientras trota hacia casa.”

La idea detrás de esta explicación es el interés que genera el hecho de que el usuario pueda, mediante la creación de esta lista de reproducción de forma automática, omitir el proceso de selección de álbumes musicales tras el término de cada uno de ellos cuando lo que pretende el usuario es escuchar música que se ciñe a ciertos criterios que son conocidos por él mismo a priori.

Otro ejemplo de utilización sería la creación de una lista de reproducción por parte de un usuario de cara a utilizarla en una fiesta o reunión con una temática determinada. Así, por ejemplo, si un usuario quisiese preparar la música para una fiesta cuyo tema fuese, por ejemplo, “los años sesenta” y que tiene pensado hacer durar ocho horas, tan sólo tendría que realizar una consulta con estos datos y podría por tanto despreocuparse de la música de esa reunión, sabiendo que en todo momento ésta se ceñirá, en la medida de lo posible, a la temática correspondiente.

### 4.2. Definición de los metadatos

A continuación estudiaremos el contenido de esta base de datos, haciendo hincapié en el significado y los rangos de valores de los distintos criterios de la misma para cada disco.

La base de datos sobre la que trabajamos tiene como unidad esencial álbumes musicales. Al contrario que en otros trabajos se ha utilizado a los álbumes como objeto sobre el que tratar

en lugar de a las canciones puesto que el objetivo esencial de las listas de reproducción que obtendremos como resultado sería el segundo que ha sido explicado en el punto anterior. Ya que se pretende que estas listas de reproducción tenga una duración de varias horas, y en un afán de tener cierta homogeneidad entre las distintas canciones, hemos seleccionado a los álbumes para ser el centro de nuestra base de datos.

Por supuesto esto es sólo el enfoque que hemos dado para este caso de estudio, ya que nuestra infraestructura permite el uso de cualquier tipo de datos.

Para cada uno de los álbumes hemos seleccionado tres tipos de metadatos distintos, que son los predicados de la base de datos **RDF** original, de los cuales obtendremos valores para cada uno de los álbumes. Estos son el género, el artista, y el año de lanzamiento.

La base de datos total, que consta de 71 títulos, es la siguiente.

<b>Álbum</b>	<b>Género</b>	<b>Artista</b>	<b>Año</b>
Ten	Grunge	Pearl Jam	1991
À Découvrir Absolutement	Alternative	Various Artists	2004
A Feast of Wire	Indie Rock	Calexico	2003
A State of Trance 2005	Trance	Armin Van Buuren	2005
A State of Trance 2006	Trance	Armin Van Buuren	2006
A State of Trance 2007	Trance	Armin Van Buuren	2007
Achtung Baby	Rock	U2	1991
Actos Inexplicables	Folk	Nacho Vegas	2001
Adore	Alt. Rock	The Smashing Pumpkins	1998
Aparejo de fortuna	Folk	Javier Krahe	1983
Black Market Music	Alt. Rock	Placebo	2000
Cábalas y Cicatrices	Folk	Javier Krahe	2002
Cajas de música difíciles...	Folk	Nacho Vegas	2003
Canciones desde palacio	Folk	Nacho Vegas	2003
Canciones para el tiempo...	Pop	Iván Ferreiro	2005
City Zen	Folk	Kevin Johansen	2004
Clandestino	Latin	Manu Chao	1998
Colour the Small One	Pop	Sia	2004
Contra la la ley de...	Pop	Los Planetas	2004
Core	Grunge	Stone Temple Pilots	1992
Corral de cuernos	Folk	Javier Krahe	1985
Creamfields 2005	Trance	Ferry Corsten	2005
Darklands	Alt. Rock	The Jesus & Mary Chain	1987
Desaparezca aquí	Folk	Nacho Vegas	2005
Different Class	Britpop	Pulp	1995
DJ-Kicks	Electronica	Erlend Øye	2004
El hombre pez	Rock	Elefantes	1997
Eligeme	Folk	Javier Krahe	1988
Genetic World	Electronic	Télépopmusik	2001
Hunky Dory	Pop Rock	David Bowie	1971
Hypnotica	Electro	Benny Benassi	2003
In Search of Sunrise 5...	Trance	Tiësto	2006
International Velvet	Pop	Catatonia	1998
Kid A	Alt. Rock	Radiohead	2000
Kittenz and Thee Glitz	House	Felix da Housecat	2001

La leyenda del espacio	Pop	Los Planetas	2007
Last Splash	Alternative	The Breeders	1993
Live at Innercity...	Trance	Tiësto	1999
Maestro	Rock	Kaizers Orchestra	2005
Meds	Alt. Rock	Placebo	2006
Melody A.M.	Electronica	Röyksopp	2001
OK Computer	Alt. Rock	Radiohead	1997
Pet Sounds	Baroque pop	The Beach Boys	1966
Placebo	Alt. Rock	Placebo	1996
Quiet Is the New Loud	Indie Pop	Kings of Convenience	2001
Rated R	Hard Rock	Queens of the Stone Age	2000
Seal IV	Pop	Seal	2003
Selected Ambient Works... 2005	Ambient Hip Hop	Aphex Twin SFDK	1992 2005
Songs of Faith and...	Synth Rock	Depeche Mode	1993
Summer Sun	Indie Rock	Yo La Tengo	2003
Supa Dupa Fly	Hip Hop	Missy Elliott	1997
Tales from Turnpike House	Synthpop	Saint Etienne	2005
The Best of 1980-1990	Rock	U2	1998
The Best of 1990-2000	Rock	U2	2002
The Big Bang	Hip Hop	Busta Rhymes	2006
The Campfire Headphase	I.D.M.	Boards of Canada	2005
The Coming	Hip Hop	Busta Rhymes	1996
The DJ 3 in the Mix	Dance	ATB	2006
The Life Pursuit	Chamber Pop	Belle & Sebastian	2006
The Magnificent Tree	Trip hop	Hooverphonic	2000
The Politics of Dancing 2	Trance	Paul Van Dyk	2005
The Understanding	Electronica	Röyksopp	2005
Tri-State	Trance	Above & Beyond	2006
Triángulo de Amor Bizarro	Noise	Triáng. de Amor Bizarro	2006
Un soplo en el corazón	Pop	Family	1993
Una semana en el motor...	Pop Rock	Los Planetas	1998
Unidad de desplazamiento	Pop Rock	Los Planetas	2000
Vain Glory Opera	Power Metal	Edguy	1998
Veneer	Indie	José González	2003
Vitalogy	Grunge	Pearl Jam	1994

A continuación veremos el significado de cada uno de los metadatos que son las características de los álbumes además de hablar de las funciones de evaluación de distancia entre los distintos valores que puedan tomar dichos metadatos en nuestra base de datos.

#### 4.2.1. El metadato género

Un género musical es una categoría que reúne temas musicales que comparten ciertos criterios. Estos criterios pueden ser musicales, como por ejemplo ritmo, instrumentación, o basarse en características tales como la región de origen, período histórico, o aspectos sociales y culturales.

Los géneros pueden dividirse en subgéneros, pero para nuestro estudio haremos uso del término género tanto cuando hablemos de géneros como cuando estemos tratando subgéneros. Además, si bien podría darse la situación de que un álbum determinado

pudiese ser interpretado como perteneciente a varios géneros, en nuestro caso hemos simplificado la labor dando un solo género a cada álbum de la base de datos.

Nuestra base de datos cuenta con 29 géneros distintos algunos de ellos bastante similares y otros totalmente distintos al resto. Los géneros son los siguientes:

Grunge, Alternative, Indie Rock, Trance, Rock, Folk, Alternative Rock, Pop, Latin, Britpop, Electronica, Electronic, Pop Rock, Electro, House, Baroque pop, Indie Pop, Hard Rock, Ambient, Hip Hop, Synth Rock, Synthpop, Intelligent dance music, Dance, Chamber Pop, Trip hop, Noise, Power Metal, Indie.

### **Evaluación de las distancias entre géneros**

Una vez hemos definido qué es un género, cuántos y cuáles son los que tenemos en nuestra base de datos, es hora de ver cómo hemos creado la función de evaluación para este criterio.

Cabe recordar que las funciones de evaluación devuelven una medida de la distancia (de cero a cien) que existe entre dos valores distintos de un criterio. Así, cuando dos valores son idénticos, en este caso el género es el mismo, la medida devuelta será cero. En cambio, cuando dos géneros sean muy dispares, antagónicos en cierto modo, la medida devuelta por la función de evaluación será cien.

Hemos decidido, dada la disparidad siempre habitual entre géneros, por símiles que estos puedan ser, que, salvo que el género sea el mismo, el valor mínimo de distancia será de 50. Así, dos géneros similares en principio, pero con claras distinciones para el oyente habitual, como puedan ser los géneros Baroque Pop y Synthpop, tienen una distancia de 50. En cambio géneros como el Power Metal y el Trance, tendrán una distancia cercana a 100.

#### **4.2.2. El metadato artista**

El artista es sencillamente el / los intérpretes del disco. Es el encargado de dar su nombre al álbum, ya sea por haber compuesto las canciones que en él se interpretan, por haberlas interpretado también, o por haber realizado la mezcla de la sesión que contiene el disco (sería el caso de algunos de los discos cuyo artista es un DJ).

Es cierto que un álbum puede tener varios artistas, incluso uno por tema. Para poder simplificar nuestro estudio hemos decidido que aquellos álbumes cuyo artista no pudiese ser identificado como un solo intérprete o grupo musical, tomen como artista el nombre genérico 'Various Artists'. Nombre además que es comúnmente usado en bases de datos musicales.

En nuestra base de datos hay 51 artistas distintos, algunos de los cuales tienen una sola aparición entre los discos de la base de datos mientras que otros tienen varias. La lista completa de artistas de nuestra base de datos es la siguiente:

Pearl Jam, Various Artists, Caalexico, Armin Van Buuren, U2, Nacho Vegas, The Smashing Pumpkins, Javier Krahe, Placebo, Iván Ferreira, Kevin Johansen, Manu Chao, Sia, Los Planetas, Stone Temple Pilots, Ferry Corsten, The Jesus and Mary Chain, Pulp, Erlend Øye, Elefantas, Télépopmusik, David Bowie, Benny Benassi, Tiësto, Catatonia, Radiohead, Felix da Housecat, The Breeders, Kaizers Orchestra, Röyksopp, The Beach Boys, Kings of Convenience, Queens of the Stone Age, Seal, Aphex Twin, SFDK, Depeche Mode, Yo La Tengo, Missy Elliott, Saint Etienne, Busta Rhymes, Boards of Canada, ATB, Belle & Sebastian, Hooverphonic, Paul Van Dyk, Above & Beyond, Triángulo de Amor Bizarro, Family, Edguy, José González.

### **Evaluación de las distancias entre artistas**

Ahora que ya tenemos claro el concepto de artista, y cuáles forman parte de nuestra base de datos, veamos cómo se creó la función de evaluación para este criterio.

Puesto que es muy difícil encontrar a dos artistas que sean realmente similares hemos decidido que el valor mínimo de distancia para dos artistas distintos sea de 50. Por lo tanto dos artistas en cierto modo similares, como puedan ser Armin Van Buuren y Tiësto (ambos son DJs del mismo género), tendrán una distancia de 50, mientras que artistas completamente distintos, como puedan ser The Beach Boys y Benny Benassi, tendrán una distancia cercana a 100.

#### **4.2.3. El metadato año**

Como último dato de nuestra base de datos tenemos el metadato año, siendo éste el año de publicación del disco.

En nuestra base de datos los discos datan desde 1966 hasta 2007, siendo la mayor parte de ellos de los años 90 y de los primeros años del siglo XXI.

### **Evaluación de las distancias entre años**

La creación de la función de evaluación de este criterio ha sido bastante simple. Teniendo en cuenta el rango de años del que hemos hablado antes, para calcular la distancia entre dos años nos bastará con ver la distancia que hay entre dos y compararla con 41, que es la diferencia entre 2007 y 1966. Así si dos álbumes son del mismo año su distancia será cero. Si un álbum es de 1985 y otro de 2005 su distancia será prácticamente 50, y si un álbum es de 1966 y el otro de 2007 su distancia será 100.

Con esto terminamos la explicación de los distintos criterios de nuestra base de datos, así como del significado de los valores de distancia que son devueltos por las funciones de evaluación y el cómo se han calculado todos estos.

Ahora pasaremos a hacer un estudio de los resultados obtenidos por nuestro sistema, haciéndolo a partir del resultado promedio obtenido tras la repetición de varias consultas que consideramos significativas de aquello sobre lo que el usuario final querrá obtener información.

### 4.3. Evaluación del sistema

Para evaluar nuestro sistema hemos decidido hacer varias consultas. Así podremos ver con mayor claridad la frecuencia de aparición en la solución final obtenida de los distintos álbumes en relación con la distancia que hay entre ellos y el valor del criterio (o de los criterios) de búsqueda, así como poder también medir la saturación, término que explicamos a continuación. Estas son las dos variables que iremos estudiando a lo largo de la evaluación del sistema.

Todas las consultas que vamos a realizar tuvieron un valor común en el TMax, que se tomó de 1200 minutos o, lo que es lo mismo, 20 horas. La idea tras este valor era el tener siempre soluciones que no pudiesen ser presentadas con valores que se ciñesen a los criterios de la consulta de forma exclusiva por el hecho de tener la totalidad de éstos una duración conjunta inferior a estas veinte horas.

Dicho de otra forma, queríamos evitar tener soluciones en las que tan solo los criterios seleccionados hiciesen aparición.

Como hemos dicho habrá dos valores a estudiar, la frecuencia y la saturación.

La frecuencia es el porcentaje de la solución que se encuentra dentro del rango delimitado en lo que a distancia con el criterio de consulta se refiere. Así, por ejemplo, si tenemos que la solución tiene 10 álbumes y 5 de ellos tienen una distancia con respecto al criterio de la consulta mostrado de cero, la frecuencia para este rango (rango de distancia cero) será de 0.5 (un 50% de los álbumes de la solución tienen distancia cero con respecto al criterio de la consulta).

La saturación es en cambio una media en relación a los álbumes de cada rango. Así la saturación es el porcentaje de los discos totales de un rango dado con respecto al criterio de la consulta mostrado, que aparecen en la solución. Así si en el rango de distancia de 50 a 59 encontramos para un criterio dado 20 discos, y 5 de ellos aparecen en la solución, la saturación total será 0.25 (un 25% de los discos de distancias entre 50 y 59 con respecto al criterio de la consulta seleccionado aparecen en la solución).

En nuestro estudio hemos realizado ocho consultas distintas. Todas ellas se han ejecutado un mínimo de cinco veces para facilitar en las gráficas un promedio de resultados obtenidos, que se pueda ajustar de la mejor forma posible a los resultados reales que se obtendrían a través del sistema que hemos creado.

A continuación veremos cada una de las consultas realizadas con los gráficos correspondientes a frecuencias y saturación para, posteriormente, comentar todos y cada uno de ellos.

Además de mostrar los datos citados, en las gráficas también aparecerán los promedios de duración de los álbumes separados por rango puesto que, como comentaremos más adelante, la duración resulta un factor relevante a la hora de estudiar las soluciones proporcionadas por el sistema.

#### 4.3.1. Consulta: Folk = 100

La primera consulta que realizamos consta de un solo factor, `genre = folk` y tiene una relevancia de 100. Cabe destacar que la relevancia sólo es significativa cuando hay más de un factor en la consulta y por tanto los resultados son de idéntica índole sea cual sea la relevancia señalada en una consulta de un solo factor.

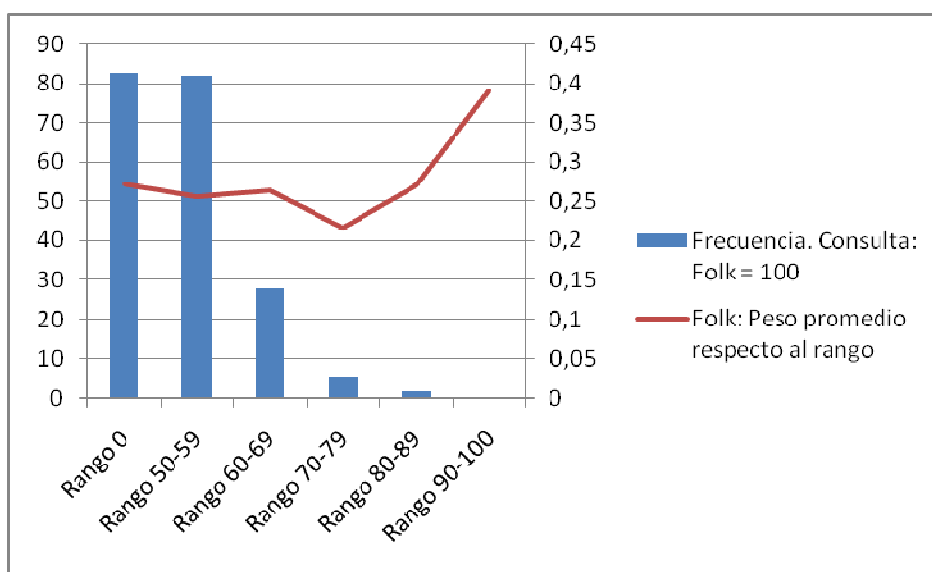


Figura 10. Frecuencia. Consulta: Folk = 100

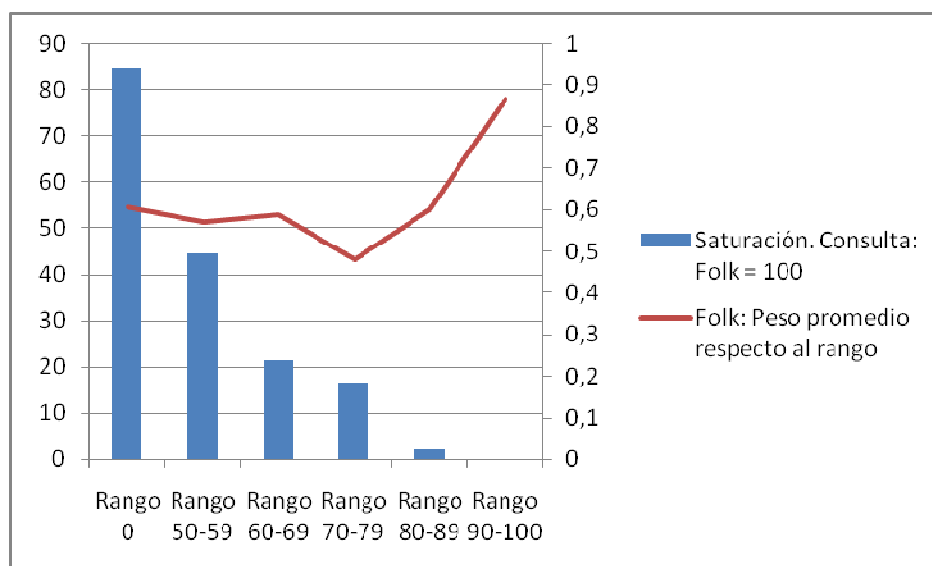
Como comprobamos en la Figura 10, la frecuencia es muy similar tanto en el Rango 0 como en el Rango 50-59 y es en ambos casos ligeramente superior al 40%. El hecho de que la frecuencia no sea mayor para aquellos álbumes que tienen un Rango 0 es debido a que la cantidad de álbumes de Rango 0 (nueve en este caso) es muy inferior al número promedio de álbumes devueltos por las soluciones (más de 20). Con lo cual es imposible que la frecuencia para esta consulta del Rango 0 supere el 45% que resulta de dividir nueve entre veinte.

Sí es cierto que puesto que las 17 ocurrencias de discos con Rango 50-59 sumadas a las nueve de Rango 0 hacen un total de álbumes superior al total devuelto por las soluciones, la frecuencia con la que los álbumes devueltos pertenecen a alguno de estos dos rangos, que supera el 80% no es todo lo grande que ésta podía ser.

Por último, de la gráfica se desprende también que, como era de esperar, a medida que aumenta el rango, la frecuencia con que álbumes de éste aparece será mucho menor, llegando a un 0% para el Rango 90-100.



La línea que marca el peso promedio en este rango no parece relevante por el hecho de que las grandes diferencias se encuentran en rangos muy alejados del criterio mostrado, aunque sí que podría haber esta misma situación ayudado a que los rangos finales desaparezcan de la solución.



**Figura 11. Saturación. Consulta: Folk = 100**

En lo que a la saturación respecta, varias conclusiones pueden sacarse viendo la Figura 11 y una de ellas es bastante importante ya que la veremos reflejada en posteriores resultados.

Como vemos la saturación para el Rango 0 sobrepasa el 90%, pero no llega al 100% a pesar de haber sólo 9 discos de este rango y obtener soluciones con el doble de álbumes. Esto se debe a que uno de los álbumes del género `folk` tiene una duración muy superior a la del resto de álbumes del mismo género.

El problema con la duración, que viene a ser el peso para el algoritmo de hormigas, es que a la hora de calcular el itinerario (lista de reproducción en nuestro caso) de mejor puntuación un disco que doble en duración a otros dos sólo tendrá preferencia con respecto a estos otros dos si su puntuación es mayor que la suma de las puntuaciones de los dos discos con la mitad de duración.

Así si tenemos un disco con duración 120 y con una puntuación, pongamos de 5000, otro disco con una duración de 60 y una puntuación de 2500 y un tercero de duración igual a 60 también y de puntuación 2600, aunque el primer disco supera en prácticamente el doble la puntuación de los otros dos, el algoritmo tomaría a los dos discos de 60 minutos a la hora de dar la solución por encima del de 120 ya que con igual duración conjunta, los discos de una hora tienen una puntuación combinada de 5100, superior a los 5000 del disco de dos horas.

Una vez esto está claro, no es de extrañar que en algunas soluciones el disco de mayor duración de `folk`, que supera el promedio de duración de los discos en casi un 90%, no

esté presente, dando lugar a una saturación cercana al 100%, pero no exactamente del 100%.

El resto de la gráfica vuelve a demostrar que los resultados que podrían ser intuitivos han sido obtenidos, dando así una saturación muy inferior, cercana al 50% a los discos de Rango 50-59, y menor aún en el resto de los casos.

#### 4.3.2. Consulta: Trance = 100

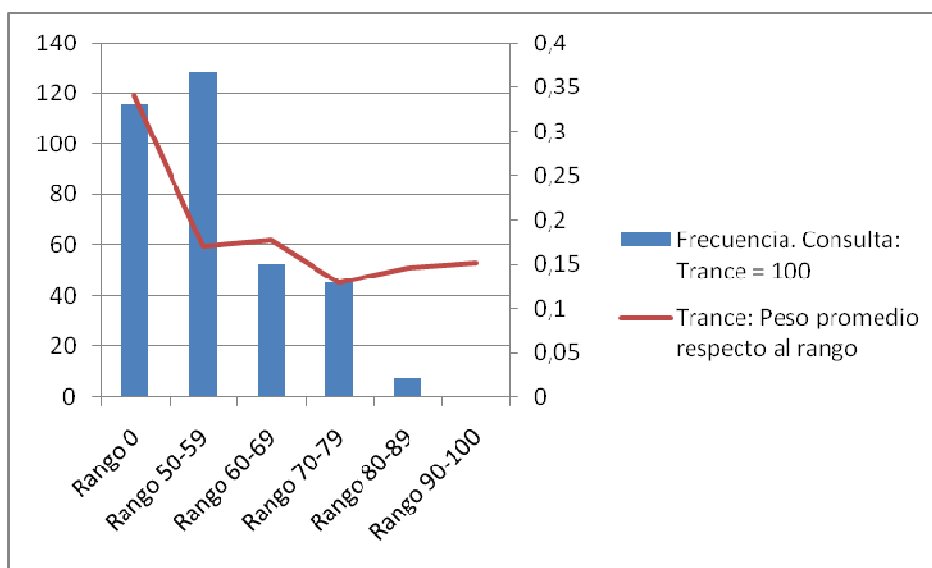
La segunda consulta sobre la que hemos trabajado consta también de un solo factor, `genre = Trance` con relevancia 100. Hemos hecho dos consultas tan similares con el fin de demostrar empíricamente la relevancia que tiene el peso (duración) de los álbumes en el cálculo de las listas de reproducción.

Como se puede comprobar en la Figura 12, los álbumes pertenecientes al `rango 0` tienen una duración promedio cercana a los 120 minutos, muy por encima del resto de rangos que fluctúan entre los 40 y los 60 y pocos minutos. Esto significa que en promedio un álbum del `rango 0` dura el doble que uno del resto de rangos, y por lo tanto el score debe ser el doble para que a efectos de interés para el algoritmo sea mejor que dos álbumes de los otros rangos.

El número de álbumes de `trance` es de ocho, y el promedio de álbumes de las soluciones es de 17, con lo cual la relación entre número de álbumes del género y el total de álbumes de las soluciones es prácticamente idéntica al de la consulta `folk = 100`, por esto podemos intuir que el motivo por el que la frecuencia baje más de un 5% será seguramente debido al aumento del peso de los álbumes para el `rango 0`.

En cualquier caso el comportamiento continúa siendo similar al comportamiento en la consulta anterior, si bien el `rango 70-79` que apenas participaba de las soluciones para `folk`, sí que tiene más de un 10% de representación en las soluciones de `trance = 100`, seguramente debido al hecho de que sea el rango con menor peso de todos.

Un total del 70% de los álbumes de la solución pertenece a los dos primeros rangos, un 12% menos que para la consulta anterior, pero aún un porcentaje bastante alto de la solución. Con el estudio de estas dos consultas podemos por tanto confirmar la relevancia de la duración, puesto que la media para `trance` es 118 y para `folk` de tan solo 55.



**Figura 12. Frecuencia. Consulta: Trance = 100**

En lo que a la saturación se refiere, viendo la Figura 13, comprobamos que la relevancia del peso se muestra con más intensidad si cabe. Mientras que la saturación para el rango 0 de la consulta de folk era de un 94%, para esta consulta el porcentaje baja drásticamente hasta un 72%, 22 puntos. De los ocho discos del género, en las consultas con mejores resultados sólo aparecen seis, siendo en los peores casos sólo el 50% de ellos los que forman parte de la solución. Además son los discos con mayor duración los que más se resisten a la hora de aparecer en las soluciones.

Es tanto el descenso de la saturación, que para el rango siguiente, rango 50-59, el índice de saturación es sólo un punto menor, 71%, y para el resto de rangos la saturación siempre es mayor a la saturación de la consulta anterior.

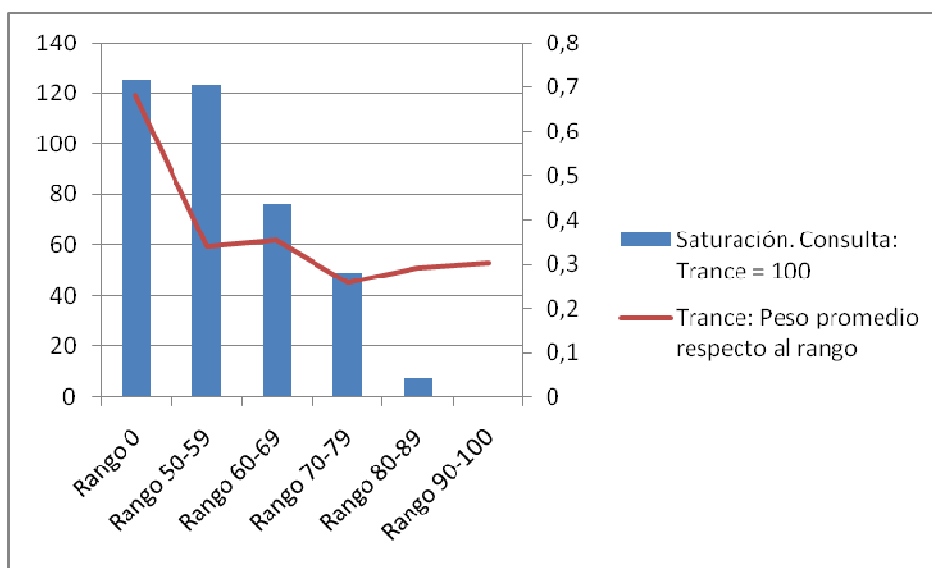


Figura 13. Saturación. Consulta: Trance = 100

#### 4.3.3. Consulta: Folk = 50; Trance = 50

A partir de esta tercera consulta, las siguientes seis que hemos realizado ya constan de dos factores.

En la primera consulta hemos tratado de comprobar qué sucede cuando elegimos dos factores sobre el mismo criterio para realizarla, y además le damos la misma relevancia a ambos. En este caso los factores han sido elegidos sobre el tipo de criterio género y sus valores son los dos valores utilizados en las consultas anteriores, *folk* y *trance*. Ambos con una relevancia de 50.

Estos dos géneros han sido elegidos por varios motivos. El primero de ellos es que son los géneros más representados en nuestra base de datos con 9 ocurrencias para *folk* y 8 ocurrencias para *trance*. El segundo es su disimilitud. La distancia entre ambos es de 95, por lo tanto los álbumes de un género estarán en el rango 90-100 de la gráfica del otro género. El tercer motivo es ver, si cabe una vez más, cómo el hecho de que *trance* tenga una duración promedio que dobla la de *folk* hace que su frecuencia en las soluciones, así como su saturación se vea afectada negativamente respecto a la del otro género.

Para estudiar los resultados mostraremos las mismas gráficas que en las consultas anteriores, pero haciendo una gráfica de secuencia y otra de saturación distinta para cada uno de los factores de las consultas. Así veremos primero los resultados desde el punto de vista del género *folk*, para posteriormente contrastarlos con los resultados obtenidos desde el punto de vista del género *trance*.

Si observamos la frecuencia desde el punto de vista del género *folk* en la Figura 14, vemos resultados similares a los obtenidos cuando en la consulta tan solo teníamos al género *folk*, con una única, pero importante excepción. Ha habido una disminución importante en la frecuencia de los rangos más cercanos al rango 0. Empezando con este mismo rango, que ha sufrido una disminución en frecuencia de un 0,5%, el rango 50-59 ha bajado un 6,3% y el rango 60-69 un 12,9%. Un total de un 19,7% de disminución en frecuencia que ha ido prácticamente en su totalidad (un 19,4%) a parar al rango 90-100, que es dónde se sitúa el género *trance* con respecto al género *folk*.

Así podemos concluir que los resultados son satisfactorios en la medida que el añadir un segundo factor del mismo tipo de criterio sólo influye en soluciones obtenidas anteriormente de forma relevante en los rangos en los que se sitúa el nuevo criterio de búsqueda.

Seguramente el hecho de que el rango en el que se encuentra el segundo factor sea, con diferencia, aquél que más peso tiene en promedio, ha hecho que la frecuencia del rango finalmente sea de menos de un 20%, pudiendo haber sido mayor si los promedios de pesos fuesen similares.

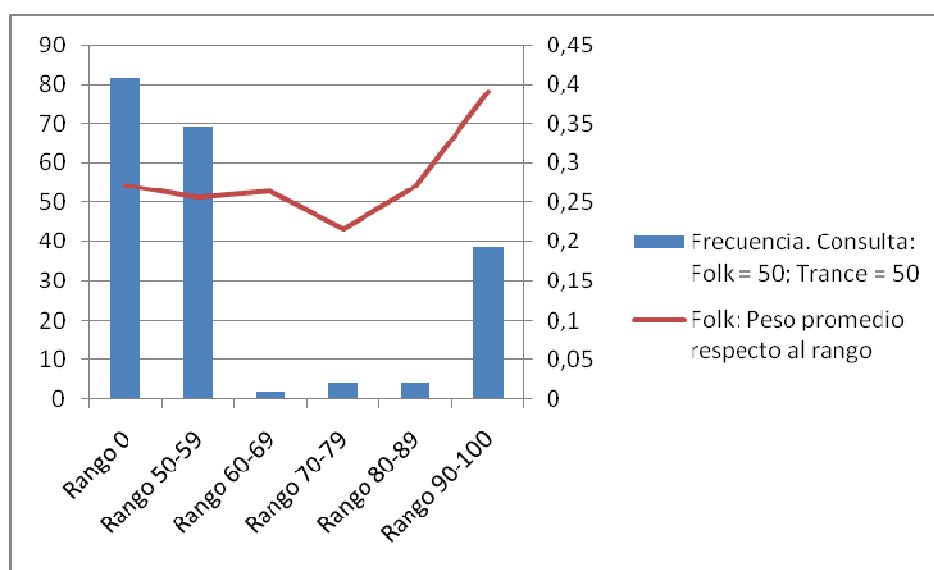


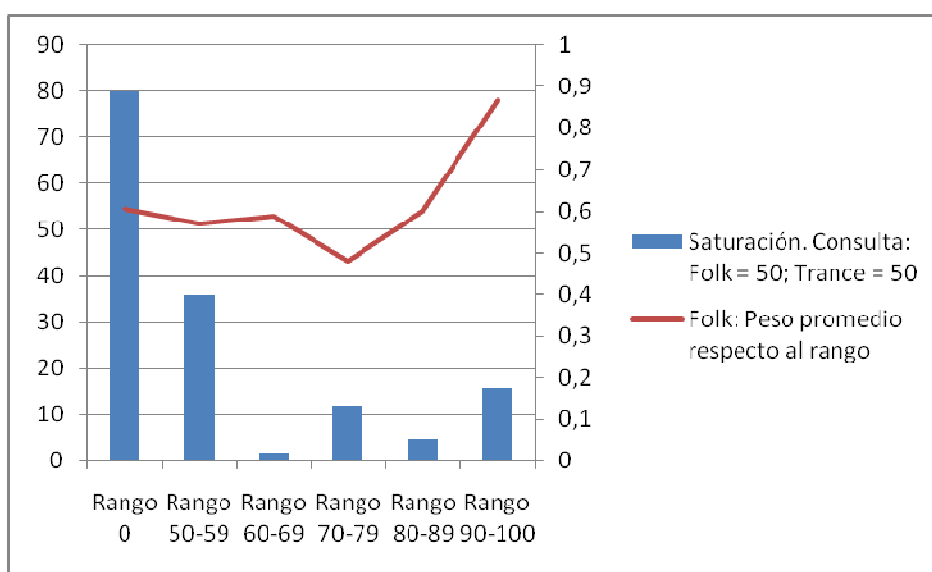
Figura 14. Frecuencia. Consulta: Folk = 50; Trance = 50

Cuando tomamos la saturación vista desde el criterio *folk*, comprobamos para empezar, como muestra la Figura 15, que hay grandes variaciones con respecto a los resultados obtenidos para la consulta *folk* = 100, variaciones mucho más significativas que aquellas observadas al estudiar la frecuencia.

Es cierto que *folk* sigue teniendo una saturación alta, sólo cinco puntos por debajo de la saturación del rango 0 en la consulta *folk* = 100, pero los rangos cercanos sufren una gran caída debido a la aparición de tantos discos pertenecientes al rango 90-100. Así el rango 50-59 tiene una caída del 9,4%, el rango 60-69 del 21,1% y el rango

70-79 del 5%. El hecho de que el rango 70-79 tenga mayor representación tiene que ver exclusivamente con el bajo número de álbumes que tiene este rango, tan sólo 3, con lo que su saturación es alta aún teniendo en promedio menos álbumes en las soluciones que el rango 60-69.

En el lado positivo de la balanza se sitúan los rangos más lejanos, que se ven influidos por la situación del segundo factor de la consulta (como ya hemos mencionado en el rango 90-100). El último rango pasa de tener un 0% de saturación a tener un 17.3%, pese a su alto peso promedio. Además el rango 80-89 sube hasta el 5%, justo el doble.



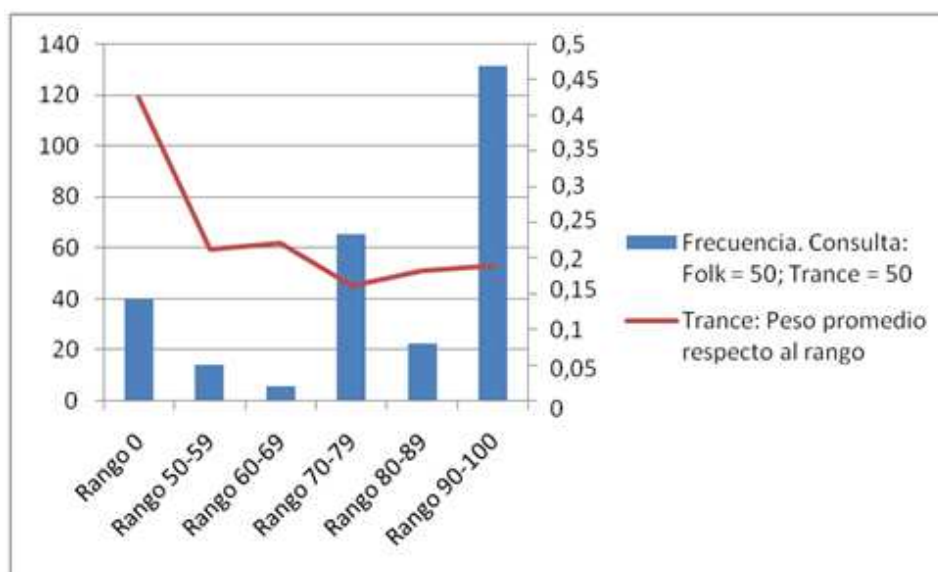
**Figura 15. Saturación. Consulta: Folk = 50; Trance = 50**

Cuando cambiamos la perspectiva y nos centramos en el género trance en la Figura 16, nos encontramos con la confirmación de todo aquello que habíamos visto desde el género folk.

El hecho de que el peso promedio para el rango 0 sea prácticamente el doble que el de cualquier otro rango ha hecho que la frecuencia se quede por debajo del 15%. Menos de uno de cada cinco discos de la solución era de *trance*, mientras que casi la mitad lo eran de *folk*.

La solución promedio tenía alrededor de 20 discos, de los cuales, el máximo de discos del rango 0 fue de cuatro.

Completamente opuesta es la situación para el rango más grande, rango 90 - 100 en el que se encuentra el género *folk* y que representa entre 9 y 10 discos de la solución en todas nuestras pruebas.



**Figura 16. Frecuencia. Consulta: Folk = 50; Trance = 50**

Pasamos a la saturación de la Figura 17 para ver mejor si cabe cómo ha afectado el peso del rango 0 en la solución.

Tenemos sorprendentemente que el único rango que tiene una saturación superior al 50% es el rango 70 - 79. Aquí es donde el problema del peso del rango 0 ha hecho desvirtuarse más si cabe la solución. Y es que estos datos pueden parecer bastante malos si no somos conscientes de que en realidad, simplemente la balanza se inclina más hacia los rangos más lejanos por la influencia de la distancia con el folk.

Al tener el rango 90 - 100 treinta y un discos, es imposible ver en la saturación el volumen de este, que ya vimos en la frecuencia, y nos tenemos que quedar con una gráfica claramente desequilibrada hacia el centro.

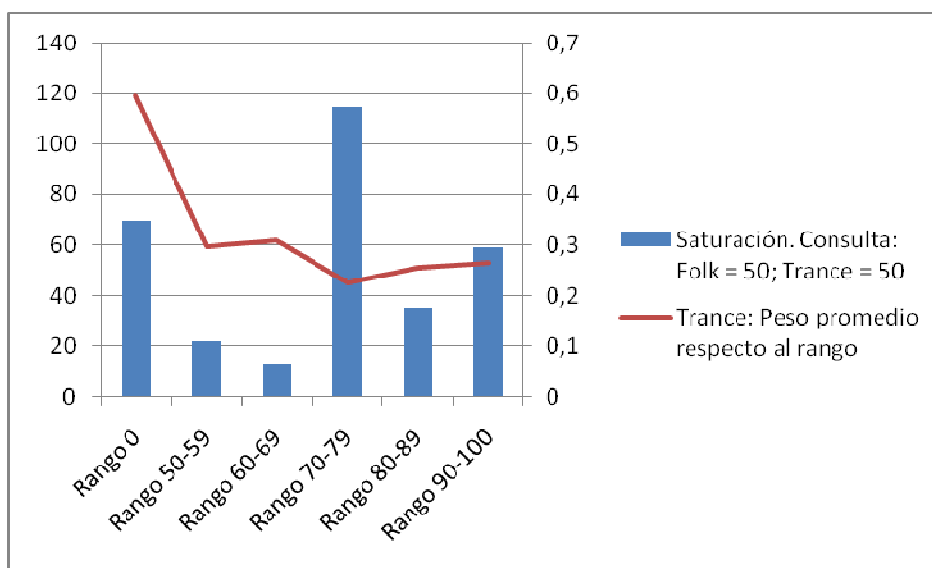


Figura 17. Saturación. Consulta: Folk = 50; Trance = 50

#### 4.3.4. Consulta: Folk = 80; Trance = 20

En la siguiente consulta realizada decidimos, partiendo de la consulta anterior, comprobar el verdadero peso de la relevancia en los resultados obtenidos.

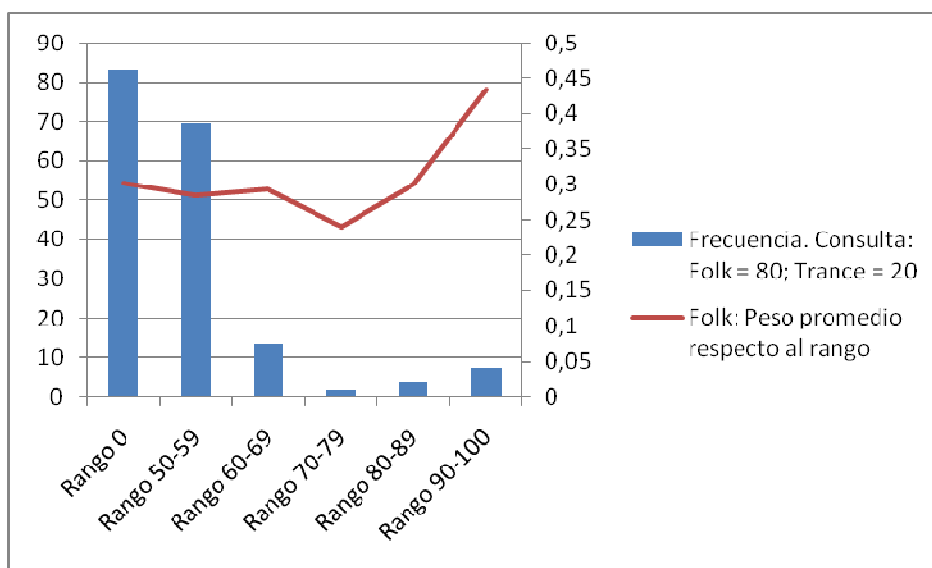
Si en la anterior consulta tanto `Folk` como `Trance` tenían una relevancia de 50, en esta ocasión hemos decidido dar al primer género una relevancia cuatro veces mayor a la del segundo término (80 frente a 20) con la esperanza de que esto decante más si cabe la balanza del lado del `folk` que, recordemos, ya se beneficiaba de su menor peso promedio.

Al tomar la frecuencia vista desde el término `folk` nos damos cuenta de que efectivamente ésta ha subido considerablemente para los rangos más cercanos al género con mayor relevancia.

La frecuencia del rango 0 era del 40,8% cuando ambos criterios compartían relevancia, pero ahora, el rango que determina los discos de género `Folk` sube hasta el 46.2%, casi seis puntos que le dejan muy cerca de la mitad de los discos de la solución.

El siguiente rango, el más cercano a `Folk` pero sin serlo sube también una cantidad similar, un 4.0% hasta el 38.7% mientras que el gran perjudicado es sin duda el rango 90-100, rango en el que se encuentra el género `Trance`, cuya disminución en relevancia le hace pasar de una frecuencia del 19.4% a una frecuencia del 4.3%, más de 15 puntos perdidos al pasar de 50 a 20 en relevancia.





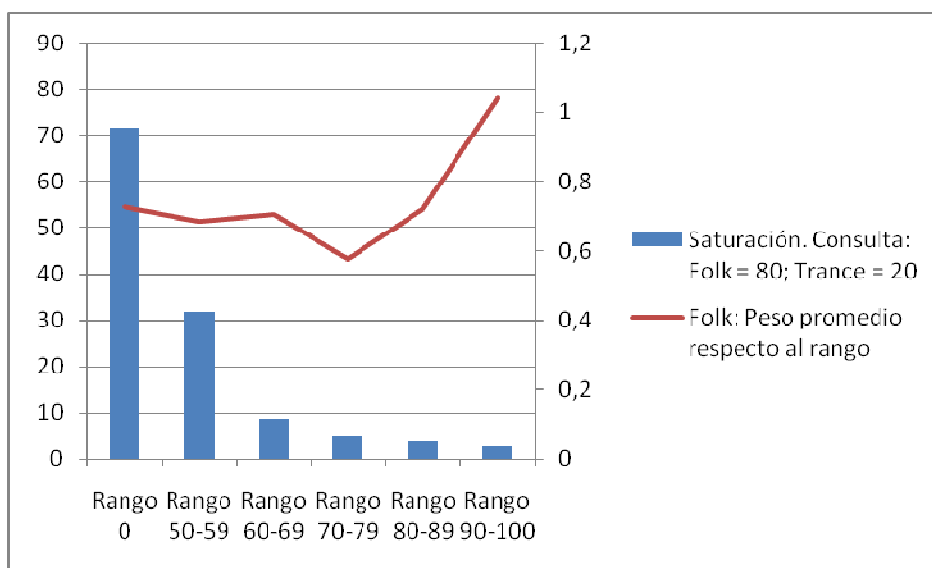
**Figura 18. Frecuencia. Consulta: Folk = 80; Trance = 20**

Similares son los resultados del rango de Trance en lo que a saturación se refiere. Si antes se incluía en la solución casi uno de cada cinco discos del rango máximo de separación con respecto a Folk (un 17,2%), ahora esta cifra cae en picado hasta el 3,6%, ligeramente menos de 15 puntos perdidos.

La cara de la moneda es folk, el género que ya contaba con un 88,9% de saturación la lleva ya hasta el 95,6%. De los nueve discos del género, en más de la mitad de las ocasiones no queda ninguno fuera de la solución final.

Aumentar la relevancia a folk ha hecho que el género haya aumentado de una forma muy notable, casi diríamos que drástica, su presencia en la solución en detrimento del género que baja su relevancia hasta una cuarta parte de la de folk, el Trance.

Vamos ahora a ver ambas gráficas desde el punto de vista del género que ha sufrido las pérdidas, para ver si efectivamente son tan desoladoras como el rango 90-100 sobre Folk hacen intuir.

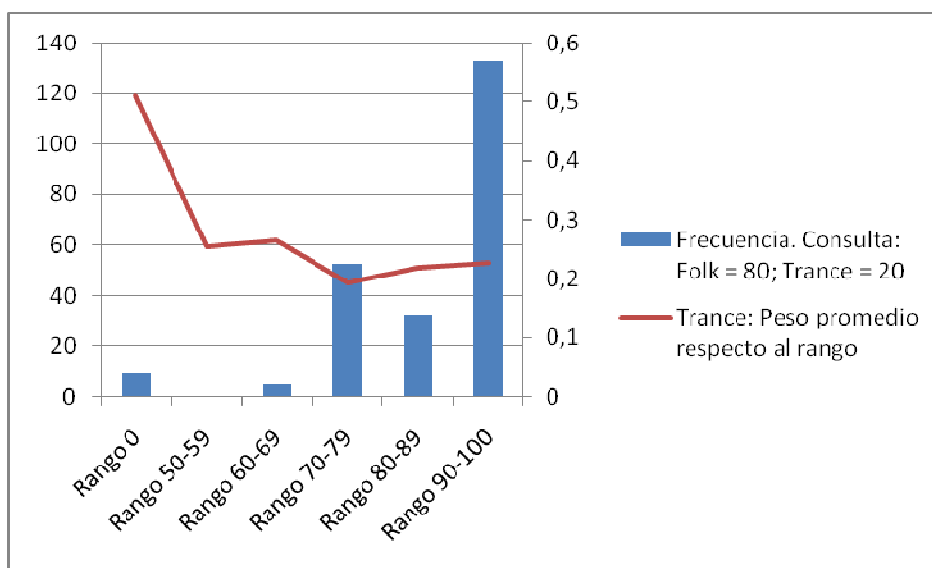


**Figura 19. Saturación. Consulta: Folk = 80; Trance = 20**

Efectivamente, como podemos comprobar en la gráfica de la frecuencia de la consulta vista desde el género Trance prácticamente no hay presencia alguna de los rangos más cercanos al género.

Si ya nos parecía poca la frecuencia para el rango 0 en la consulta que tenía una relevancia de 50 para el género Trance, la frecuencia en esta ocasión baja casi diez puntos más para quedarse en un 4,3%. Además el rango más cercano a Trance sin serlo baja hasta quedarse en el 0,0%. Ni un solo disco de este rango ha aparecido en ninguna de las soluciones obtenidas. Y en esta ocasión no es porque el rango tuviese una cantidad irrelevante de discos, ya que un total de nueve se encontraban en él.

Prácticamente son las modificaciones en la solución son simétricas ya que los 9,98 puntos que ha perdido el rango 0 y los 5,10 que ha perdido el rango 50-59 los han básicamente ganado los dos últimos rangos. El rango 80-89 ha subido un 5,82% mientras que el rango 90-100 se lanza hasta el 10,05% más, quedándose con más del 50% de los discos de la solución.



**Figura 20. Frecuencia. Consulta: Folk = 80; Trance = 20**

La saturación nos ayuda a completar el cuadro que nos ha dejado la frecuencia con un dato que ya de por sí es suficiente para ver la importancia de la relevancia. La saturación del rango 0 que estaba en un escaso 35% cuando la relevancia de Trance era 50 debido, como ya comentábamos, a la mayor duración de los discos de este rango, se despeña completamente hasta caer al 10%.

Una bajada de 25 puntos que ya no se le puede atribuir a la duración pues obviamente los discos son los mismos que en la anterior consulta. Una disminución de 30 puntos en la relevancia del término unida a un aumento por la misma cantidad de la relevancia de Folk hace que la saturación en la solución del género trance pierda dos veces y media la cantidad en la que acaba quedando, pasando de ser el segundo rango con mayor presencia de la solución a ser el tercero, solo por encima de los rangos de 50-59 y 70-79 con un 0% y un 6,7% respectivamente.

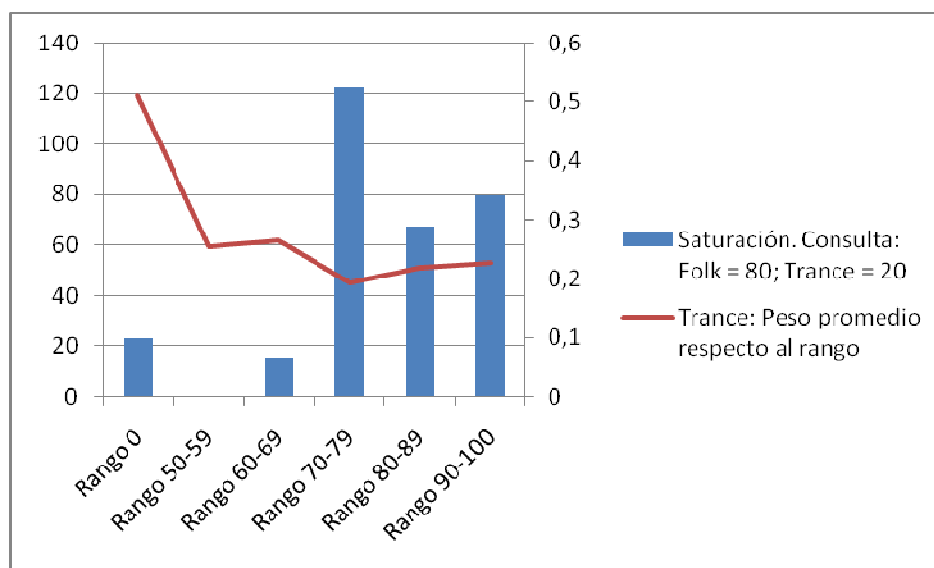


Figura 21. Saturación. Consulta: Folk = 80; Trance = 20

#### 4.3.5. Consulta: Folk = 50; Javier Krahe = 50

Las cuatro consultas realizadas que quedan por revisar ya no tendrán factores sobre el mismo criterio una vez hemos visto la influencia de los pesos y la relevancia cuando dos factores de idéntico criterio se encuentran.

Ahora nos centraremos en consultas con dos factores, uno del criterio género y otro del criterio artista. Para ello hemos decidido probar en las primeras dos consultas con un artista cuyo género principal sea el mismo que en el criterio y en las dos segundas con uno cuyo género principal sea completamente diverso. Jugando de nuevo con la relevancia en las dos consultas de cada tipo.

A pesar de que no hay relación entre ambos criterios en lo que al cálculo de las distancias se refiere (ambos tienen su propia tabla de distancias independiente) resulta evidente que aquellos artistas que comparten género musical se encuentren a mucha menor distancia que aquellos que tengan un género completamente distinto. De esta manera, esperamos que las soluciones de las dos primeras consultas sean mucho más homogéneas que aquellas de las dos consultas finales en las que el artista del criterio se aleja completamente del género elegido para la consulta.

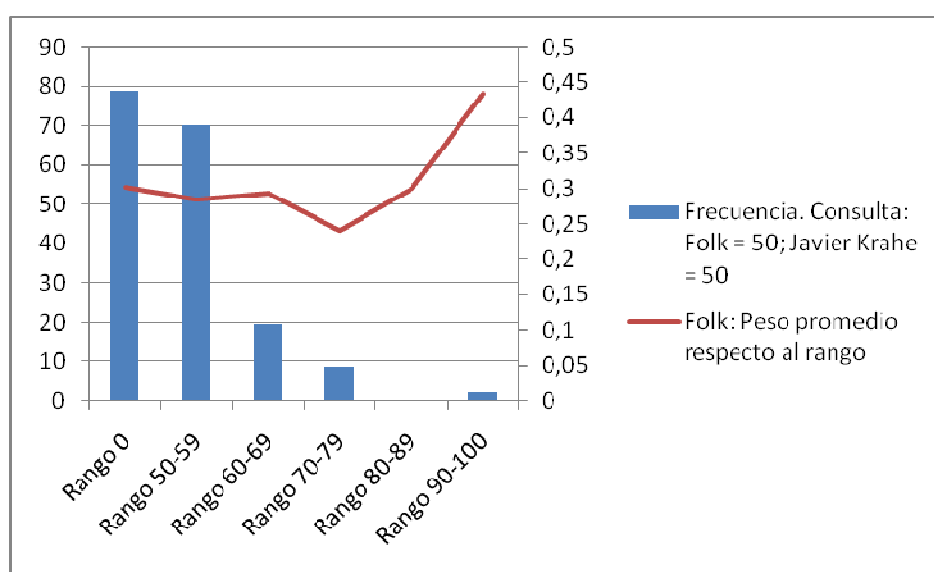
La primera de las consultas tiene como primer factor Folk, del criterio género, con relevancia igual a 50, mientras que el segundo factor es Javier Krahe, del criterio artista, con igual relevancia.

Javier Krahe tiene 4 discos en la base de datos, todos ellos dentro de los 9 discos del género Folk. Como única nota destacar que uno de estos cuatro discos tiene una duración muy por encima de los otros, superando en más de 20 minutos al siguiente. Veremos si esto también influye en los resultados.

Veamos pues, para empezar las gráficas de los resultados desde el punto de vista del primer factor, Folk = 50.

Decíamos que esperábamos resultados homogéneos para esta consulta especialmente, y no nos equivocábamos. De los 16/17 discos que tiene en promedio la solución, un 82,9% de ellos se incluyen en los rangos 0 ó 50-59. Y solo un 1,2% están en el rango final 90-100.

De los nueve discos del rango 0, se incluyen en la solución más de siete en promedio y no llegan a tres los discos con rango de 60 o superior que formar parte de la solución. Desde luego las soluciones satisfarían completamente a aquellos que buscaban folk en las mismas.

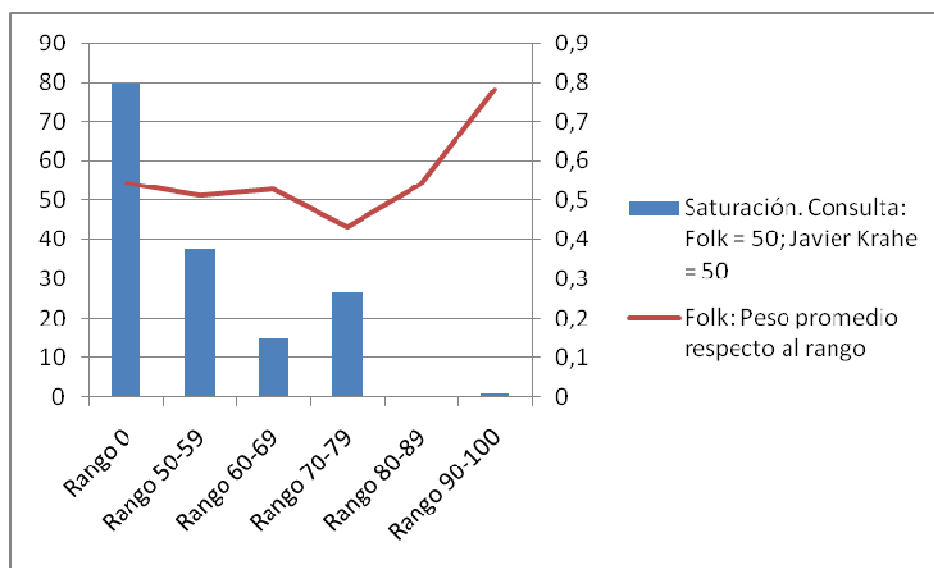


**Figura 22. Frecuencia. Consulta: Folk = 50; Javier Krahe = 50**

Pasamos ahora a analizar la saturación para ver si, aún a pesar de la homogeneidad de la solución aún nos han quedado muchos discos pertenecientes a rangos inferiores fuera de la lista de reproducción generada.

Los resultados son igualmente buenos, aunque es cierto que nos sorprende que la saturación para el rango 0 sea inferior a aquella que encontramos en la consulta Folk = 50; Trance = 50. Aún así un 80% es una saturación altísima, y la diferencia entre ambas puede ser causada simplemente por la varianza en las soluciones.

El rango 50-59 también tiene una saturación muy alta, con un 37,7% mientras que los rangos de 80-89 y 90-100 tienen una saturación del 0% y del 0,9% respectivamente. Básicamente no tienen presencia alguna. Una confirmación más de la validez y homogeneidad de las soluciones obtenidas para esta consulta.



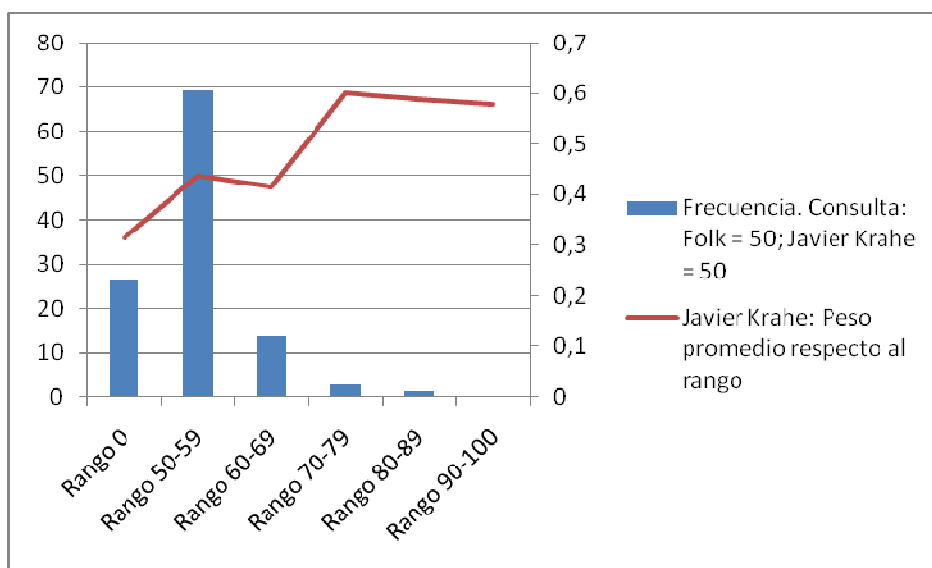
**Figura 23. Saturación. Consulta: Folk = 50; Javier Krahe = 50**

Cuando pasamos a comprobar los resultados desde el punto de vista del segundo término de la consulta, Javier Krahe = 50 tenemos que tener en cuenta que por primera vez pasamos a trabajar sobre un término con pocas ocurrencias. Hasta ahora tanto Trance como Folk habían contado con prácticamente diez ocurrencias, pero Javier Krahe solo tiene cuatro.

Esto se nota en especial cuando vemos la frecuencia del rango 0. Sobre los más de 16 discos de la solución promedia, el rango 0 no podía aspirar ni tan siquiera a llegar a un 25% de frecuencia.

Conociendo esto, por tanto, vemos que su frecuencia del 23,2% es absolutamente tremenda. De hecho en prácticamente todas las soluciones se han incluido los cuatro discos de Javier Krahe. Siendo el disco de mayor duración del que hablamos más arriba el único que quedó fuera menos de un 20% de las veces.

El temor que podría haber es que la escasez de discos en el rango 0 hiciese que la solución se esparciese a lo largo de los otros rangos, pero como vemos en la gráfica no ha sido exactamente así, y es el rango siguiente, 50-59 el que cuenta con el grueso de la frecuencia con un 61%. Un 84% de los discos de la solución están, por tanto, en estos primeros dos rangos.

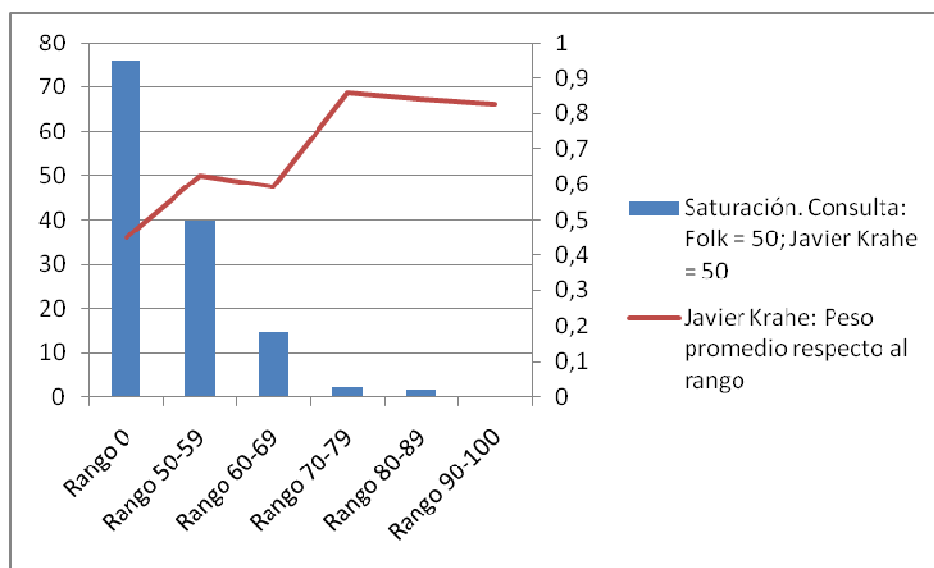


**Figura 24. Frecuencia. Consulta: Folk = 50; Javier Krahe = 50**

La saturación en cambio sí que es un punto bastante relevante. Teniendo en cuenta que además de tener un artista con tan solo cuatro discos, cuando las soluciones promedio tienen 16 discos, éste artista tiene todos sus discos pertenecientes al género que es el primer término de la consulta, el Folk.

Por este motivo, parece lógico esperar que la saturación, es decir, la presencia en las soluciones de la totalidad de los discos, en este caso, de Javier Krahe sea cercana al 100%. Y así es. Un 95% de saturación que unida al 50% de saturación del rango 50-59 confirman el comentario anterior al respecto de la homogeneidad de las soluciones obtenidas con esta consulta.

Observando las saturaciones del resto de los rangos vemos como éstas disminuyen drásticamente a medida que nos distanciamos del rango 0. Un 18% para el rango 60-69 y apenas un 3 y un 2% para los rangos 70-79 y 80-89 para terminar con un 0% en el rango final, 90-100.



**Figura 25. Saturación. Consulta: Folk = 50; Javier Krahe = 50**

#### 4.3.6. Consulta: Folk = 80; Javier Krahe = 20

Volvemos a repetir aquello que hicimos para la consulta cuyos términos eran Folk y Trance para la consulta actual, modificando las relevancias para que éstas dejen de ser idénticas.

En esta ocasión hemos decidido volver a dar a Folk la mayor relevancia en detrimento, en este caso, de Javier Krahe. Con esto, la presencia del artista Javier Krahe puede verse perjudicada, pero el género Folk debería tener una presencia aún mayor en las soluciones de la que tenían en la anterior consulta. Haciendo por tanto decantarse más hacia el rango 0 la gráfica de frecuencia del término.

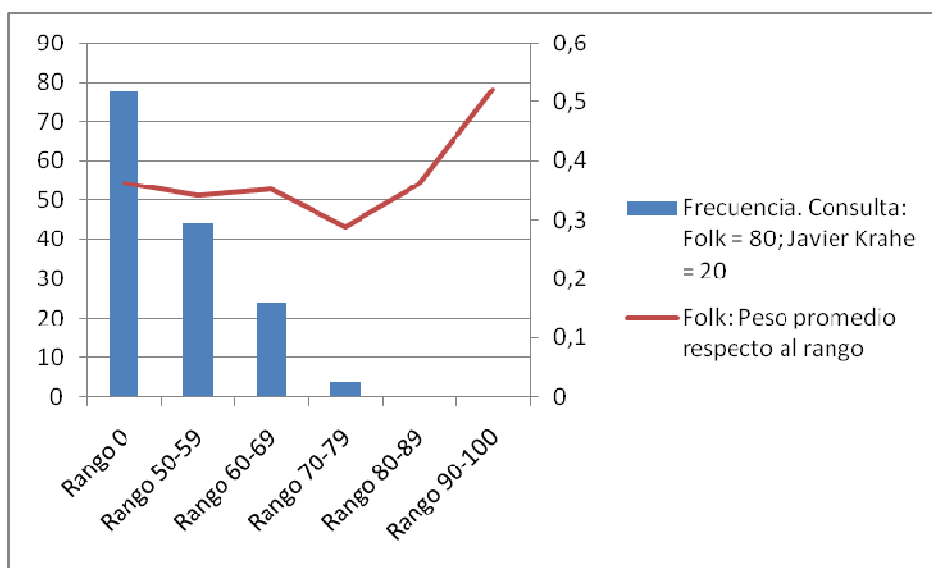
Y esto es lo primero que comprobamos con dicha gráfica, la gráfica de frecuencia de la consulta vista desde el valor Folk.

Es curioso, al estudiar la gráfica y compararla con aquella de la consulta Folk = 50; Javier Krahe = 50, ver que hay dos desplazamientos de frecuencia en la misma.

Por una parte, y como habíamos previsto, la frecuencia del rango 0 se dispara, superando el 50% para terminar con un 51.9%. Más de la mitad de los discos de la solución son de género Folk. Y parece que es sencillamente el rango 50-59 aquel que ha perdido los discos que ahora son de Folk, ya que baja de un 39.0% a un 29.6%, casi un 10% que concuerda prácticamente con el 8% que ha mejorado el rango 0.

El otro desplazamiento de frecuencias ha sido curiosamente hacia el rango 60-69. Este rango ha subido del 11.0% hasta el 16.0%, cinco puntos que básicamente se deben a la práctica desaparición (más si cabe) de los rangos posteriores que pasan de tener un 4.9%, 0.0% y 1.2% a tener un 2.5%, 0.0% y 0.0% respectivamente. En cualquier caso este 16% sigue estando muy por debajo del 29.6% del rango 50-59, que tiene una presencia claramente mayor.



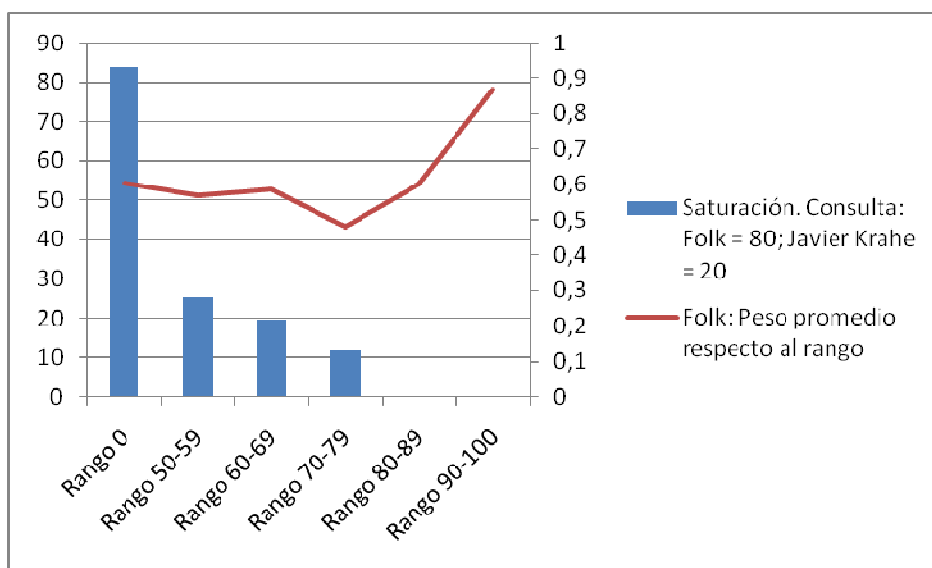


**Figura 26. Frecuencia. Consulta: Folk = 80; Javier Krahe = 20**

Si ya estábamos muy satisfechos en la consulta anterior con la saturación del término Folk, que había llegado hasta un 80%, gracias al aumento de su relevancia conseguimos alcanzar una saturación aún mucho mayor. Nada menos que un 93.3%. Los nueve discos del género Folk aparecieron en el 40% de las soluciones obtenidas, mientras que en ninguna de las soluciones faltaron dos o más discos de este género.

En lo que al resto de los rangos se refiere, vemos resultados que nos resultan más satisfactorios que en la consulta anterior, muestra del aumento en relevancia también. Nos referimos a que la saturación siempre desciende a medida que nos alejamos del rango 0. Cosa que no sucedía en el caso anterior donde el rango 60-69 tenía menor saturación que el rango 70-79, al igual que el rango 80-89 tenía menor saturación que el 90-100.

En este caso las saturaciones a partir del rango 50-59 son respectivamente 28.2%, 21.7%, 13.3% 0.0% y 0.0%.

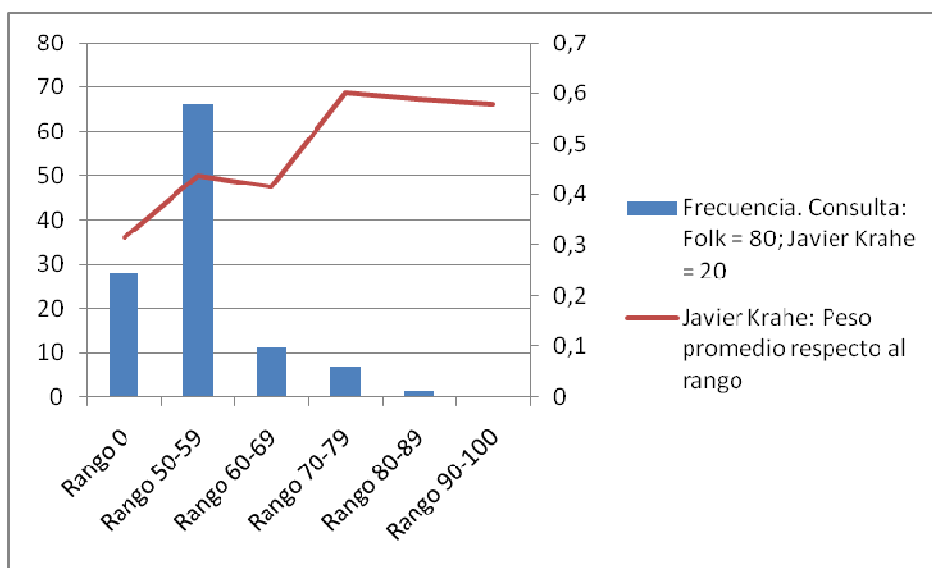


**Figura 27. Saturación. Consulta: Folk = 80; Javier Krahe = 20**

Una vez ya hemos comprobado los beneficios que la modificación de la relevancia ha tenido sobre el término Folk, tenemos que comprobar los perjuicios que pueda haber sufrido el término Javier Krahe.

Y lo cierto es que ya desde inicio se comprueba que estos perjuicios han sido más bien nulos. De hecho, para el rango 0, la consulta ha aumentado su frecuencia con respecto a aquella que tenía idéntica relevancia en 1.5 puntos.

El hecho de que, al contrario de lo que sucediese en la consulta de folk y trance, en esta ocasión los resultados no suponen pérdidas para el término desfavorecido, aunque sí que suponen mejoras para el término favorecido, nos hace intuir que el tener una consulta con términos relacionados (como ya hemos comentado, todos los discos del autor Javier Krahe pertenecen al género folk) es mucho más importante de cara a la adecuación de la solución final a aquello esperado. Algo que por otro lado es lógico, pues si nuestra consulta es heterogénea, la lista de reproducción que resulte de aplicar nuestro algoritmo lo será igualmente.



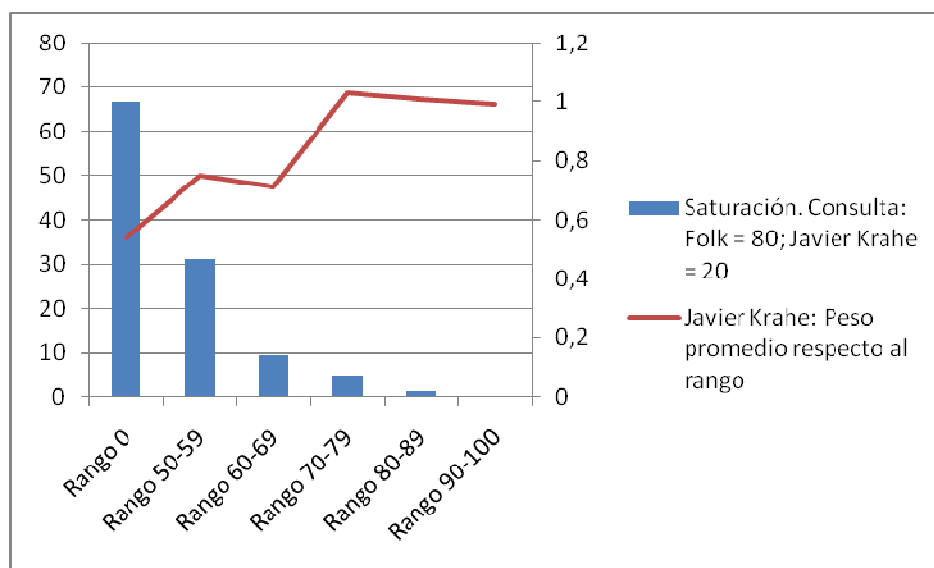
**Figura 28. Frecuencia. Consulta: Folk = 80; Javier Krahe = 20**

Al comprobar la saturación de esta consulta nos damos realmente cuenta de qué ha sucedido. En absolutamente todas las soluciones que hemos obtenido han aparecido los cuatro discos cuyo autor es Javier Krahe, sin excepción.

Así, se han obtenido las frecuencias y saturaciones máximas para el rango 0 por primera vez en nuestro estudio, a pesar de que el término que ha conseguido esto tuviese una relevancia cuatro veces inferior al término dominante.

Puede surgir la duda de por qué esta frecuencia/saturación máxima no ha sucedido cuando los dos términos de la consulta tenían la misma relevancia.

La explicación seguramente venga del hecho de que, mientras que el rango 0 visto desde el término Javier Krahe contiene solo 4 de 9 discos de género Folk, el rango 0 desde el término Folk sí contiene los 4 discos de autor Javier Krahe.



**Figura 29. Saturación. Consulta: Folk = 80; Javier Krahe = 20**

#### 4.3.7. Consulta: Folk = 50; Armin Van Buuren = 50

Pasamos a la última consulta, de la cual veremos dos variaciones de nuevo basadas en la relevancia. Se trata de una consulta con dos términos, uno de la categoría género y otro de la categoría autor, como en la ocasión anterior.

En este lugar la diferencia es que el autor es Armin Van Buuren, autor cuyos discos figuran en el género Trance, que se encuentra en el rango 90-100 respecto a Folk.

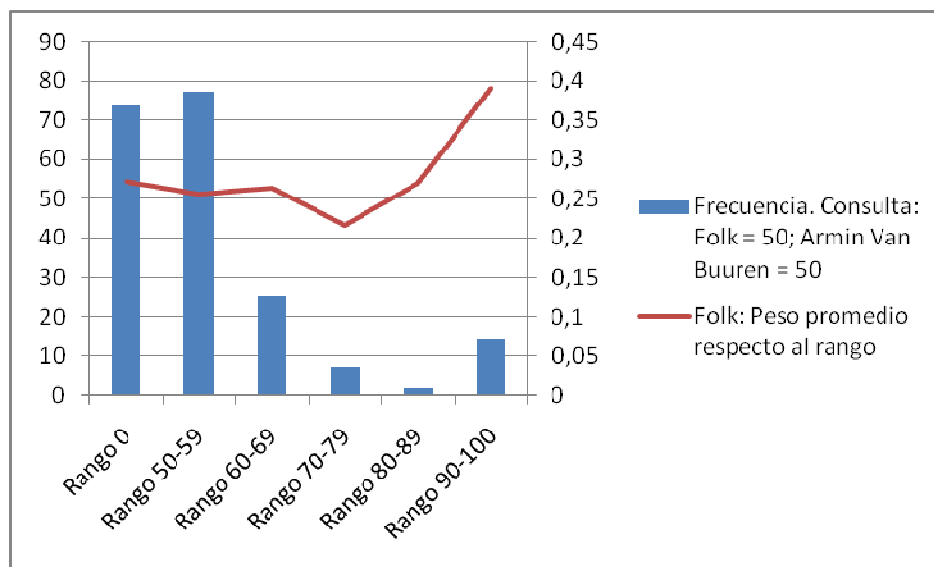
En esta consulta esperamos por tanto resultados similares a los que tuvimos para la consulta Folk = 50; Trance = 50, con la diferencia de que Armin Van Buuren apenas cuenta con tres discos de los ocho con los que cuenta el género Trance.

Además, si el género Trance ya contaba con un peso promedio de alrededor de 120, en esta ocasión aún va a más este factor con un peso promedio de casi 150.

La primera cosa que cabe destacar de los resultados de esta consulta es que con respecto a la variante que incluía a Folk y a Trance hemos pasado a incluir 22.2 discos en la solución promedia, por 19.6. Esto, unido al aumento descartado del peso promedio del rango 0 del término Armin Van Buuren nos hace prever en cierto modo el camino que tomarán las soluciones obtenidas.

La frecuencia del 37% del rango 0 del término Folk no tiene que confundirnos comparada con el 41% para la consulta con que lo estamos comparando, y es que el aumento en tres puntos del número de discos de la solución ha hecho que esta frecuencia sea menor, aun a pesar de haberse aumentado la cantidad de discos de Folk en la solución promedia.

El rango 90-100, en el que se encuentran los discos del otro término también sufre una disminución de su frecuencia, en este caso mucho más importante, pues en lugar de cuatro puntos estamos hablando de 12 puntos menos.

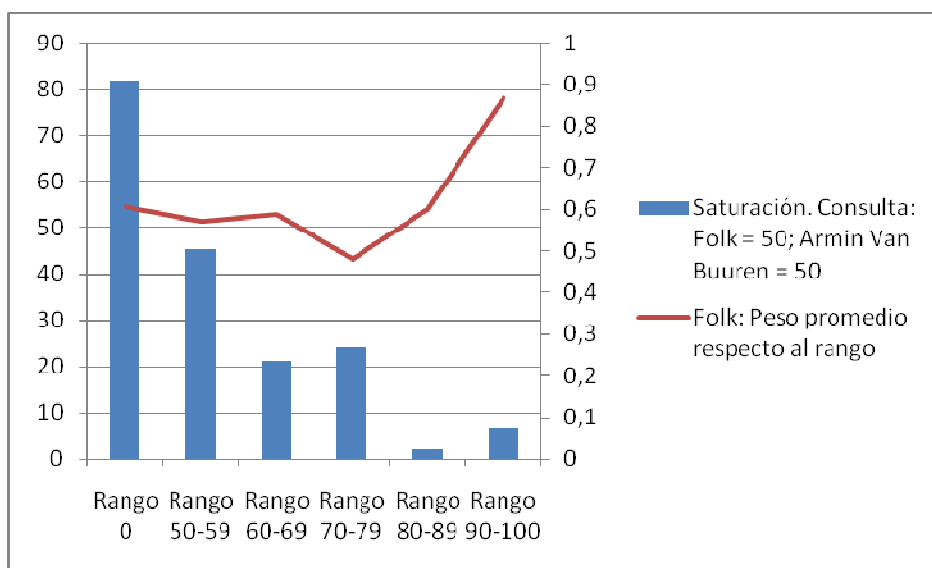


**Figura 30. Frecuencia. Consulta: Folk = 50; Armin Van Buuren = 50**

Al ver la saturación comprobamos como efectivamente los resultados de la frecuencia del rango 0 no tenían que preocuparnos. Esta ha aumentado hasta el 91.1%. Un pequeño aumento, pero no obstante un aumento.

El resto de los rangos, con la excepción de los dos últimos también aumentan su saturación, además de forma considerable, con 11, 21 y 13 puntos respectivamente, mientras que el rango 90-100 también baja en esta ocasión un total de 10 puntos.

Todos estos datos del rango 90-100 unidos al peso de los discos de Armin Van Buuren, por encima del doble del peso medio, nos hacen prever que el segundo término de la consulta va a verse gravemente dañado en esta ocasión.



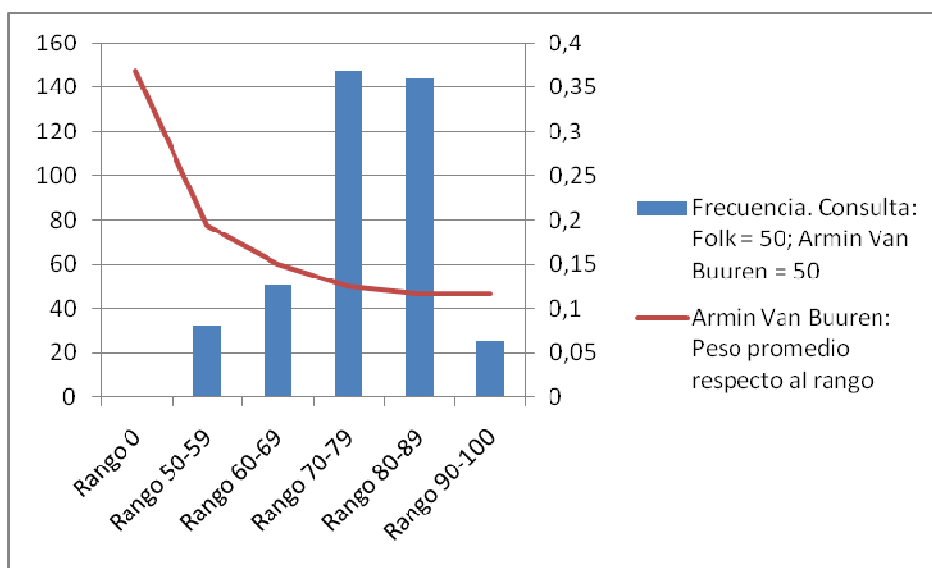
**Figura 31. Saturación. Consulta: Folk = 50; Armin Van Buuren = 50**

Y efectivamente nos encontramos con un panorama que seguramente supera aquello que habíamos podido llegar a pensar. El peso de los discos que tienen como autor a Armin Van Buuren es un factor negativo tan relevante que la presencia en las soluciones de los discos en cuestión es nula.

Ninguno de los tres discos aparece en ninguna de las soluciones obtenidas. La llegada a un punto tan extremo del peso promedio de un rango hace que éste, a pesar de formar parte de la consulta, no aparezca.

De aquí que sea tan importante darle el valor necesario al peso a la hora de definir un ámbito de trabajo. Es mejor incluir, en este caso, discos de rangos cercanos al rango 0 de la consulta si estos discos tienen un peso que no sea tan prohibitivo.

Así, para el rango 50-59, ya tenemos una frecuencia del 8.1%, gracias a que su peso promedio baja más de 60 puntos, un 40% aproximadamente.

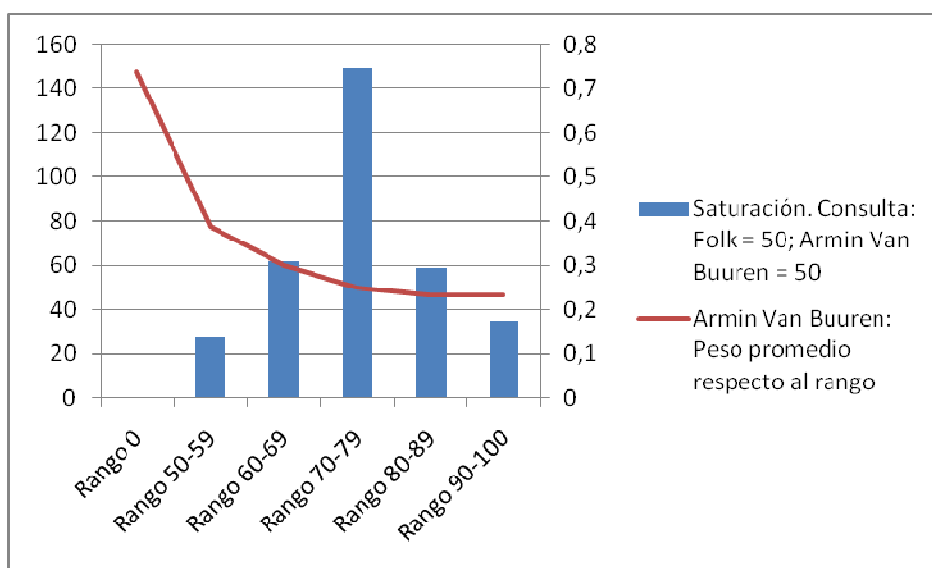


**Figura 32. Frecuencia. Consulta: Folk = 50; Armin Van Buuren = 50**

Al enfocar la saturación vemos cómo el peso en esta consulta ha tenido realmente una relevancia que va mucho más allá de hacer desaparecer al rango 0 de la solución.

Podemos comprobar cómo, en cierto punto la gráfica de saturación tiene una curvatura casi simétrica a la del peso hasta que llegamos al rango 70-79 que, a pesar de tener 11 discos, consigue una saturación del 74.5%.

A partir de ahí las saturaciones a ambos lados se asemejan algo, con el rango 60-69 y el rango 80-89 saturando en un 31.1% y 29.6% respectivamente y los rangos 50-59 y 90-100 un 13.8% y un 17.5% respectivamente.



**Figura 33. Saturación. Consulta: Folk = 50; Armin Van Buuren = 50**

#### 4.3.8. Consulta: Folk = 80; Armin Van Buuren = 20

Terminamos finalmente nuestras consultas con la realizada en el punto 4.3.8 aplicando las modificaciones que hemos venido usando en las consultas anteriores. Esto es, dar a Folk una relevancia de 80 a la vez que rebajamos la relevancia de Armin Van Buuren hasta 20.

Si el peso de los discos de Armin Van Buuren ya hizo que perdiese toda presencia el autor en la consulta con relevancia idéntica, no esperamos mejora alguna en esta situación con la nueva consulta.

Todo lo contrario, queremos ver cuánto más favorecido aparece el género Folk en esta consulta. Comparándolo también con los resultados obtenidos para la consulta Folk = 80; Javier Krahe = 20 y Folk = 80; Trance = 20.

Presumiblemente esta tercera consulta tendrá mejores resultados que la que implicaba a Trance debido al mayor peso de los discos de autor Armin Van Buuren. Cabe también ver si el tener un acompañante favorable en la consulta (como es Javier Krahe, cuyos discos son de género Folk) será mejor para el género que el tener una competencia claramente desfavorecida como sucede en este caso.

En el caso de la frecuencia. Vemos que los resultados del rango 0 son muy similares a los de la consulta Folk = 80; Trance = 20. Pero las soluciones cuentan con 1.2 discos más en promedio. Así, el resultado obtenido es cuantitativamente mejor en esta consulta.

Si bien queda algo lejos del 51.9% de frecuencia para la consulta Folk = 80; Javier Krahe = 20, tener dos de cada cinco discos en la solución en el rango 0 es satisfactorio.

Más si cabe cuando vemos que el rango 50-59 es favorecido más aún que en la consulta con Javier Krahe. Con idéntica frecuencia al rango 0, entre ambos tienen un 86.8% de los discos de la solución.

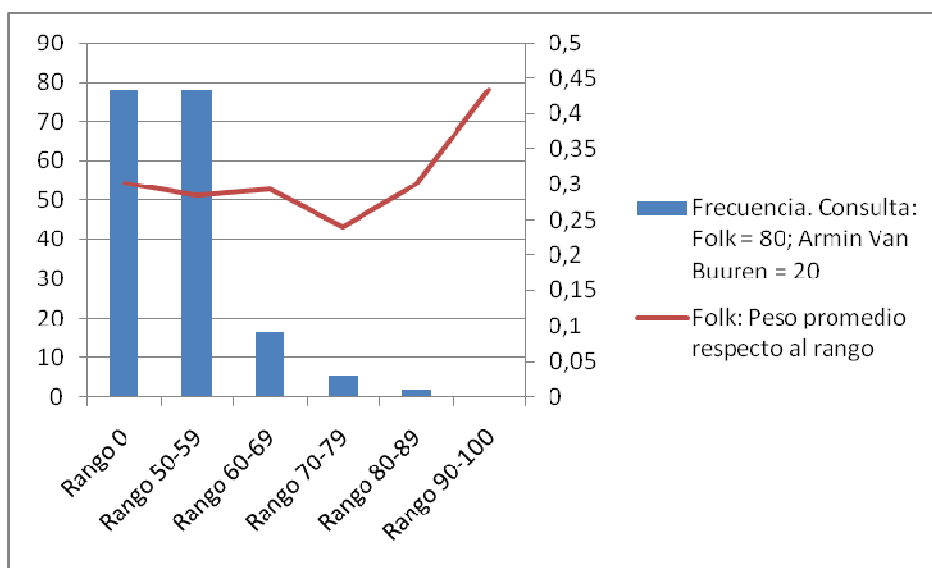
Este número es el mayor que hemos visto hasta ahora, comparemos con otras tres consultas:

- Consulta Folk = 80; Trance = 20. Frecuencia de los rangos 0 y 50-59: 84.9%
- Consulta Folk = 80; Javier Krahe = 20. Frecuencia de los rangos 0 y 50-59: 81.5%
- Consulta Folk = 50; Armin Van Buuren = 50. Frecuencia de los rangos 0 y 50-59: 75.6%

La cruz de la moneda es el rango 90-100, en el que se encuentran, entre otros, los discos que tienen a Armin Van Buuren por autor. La frecuencia es cero, o lo que es lo mismo, no solo ningún disco de Armin Van Buuren ha hecho aparición en ninguna de las pruebas que hemos realizado, sino que los otros discos de rango 90-100, discos con género completamente opuesto al Folk, han estado ausentes también.

Vemos, por tanto que la frecuencia de los dos primeros rangos es la mayor obtenida hasta ahora. Continuemos pues con la saturación y comparemos de nuevo los resultados obtenidos.

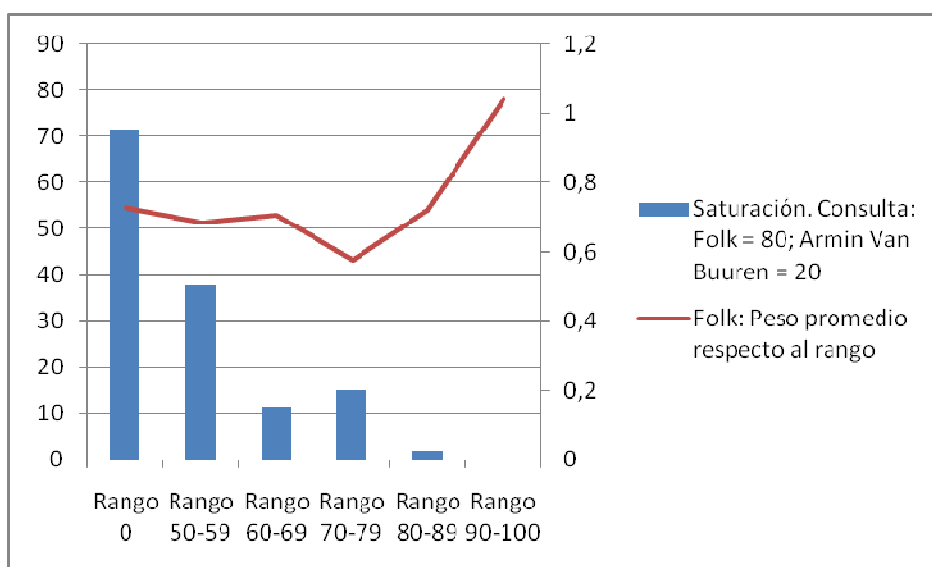




**Figura 34. Frecuencia. Consulta: Folk = 80; Armin Van Buuren = 20**

Resultado tremendo el que obtenemos de la saturación. Volvemos, como hicimos en la consulta Folk = 80; Trance = 20 a una saturación del 95.6%, además también tenemos saturación máxima para el rango 50-59 con un 50.6% (resultado idéntico esta misma consulta con relevancias idénticas para ambos términos).

Obviamente la saturación para el resto de rangos baja o se mantiene con respecto a la consulta con relevancia 50 para ambos términos. Y, como ya veíamos al observar la frecuencia, la peor parte de esta bajada se la ha llevado el rango 90-100 que ha desaparecido completamente de las soluciones obtenidas.

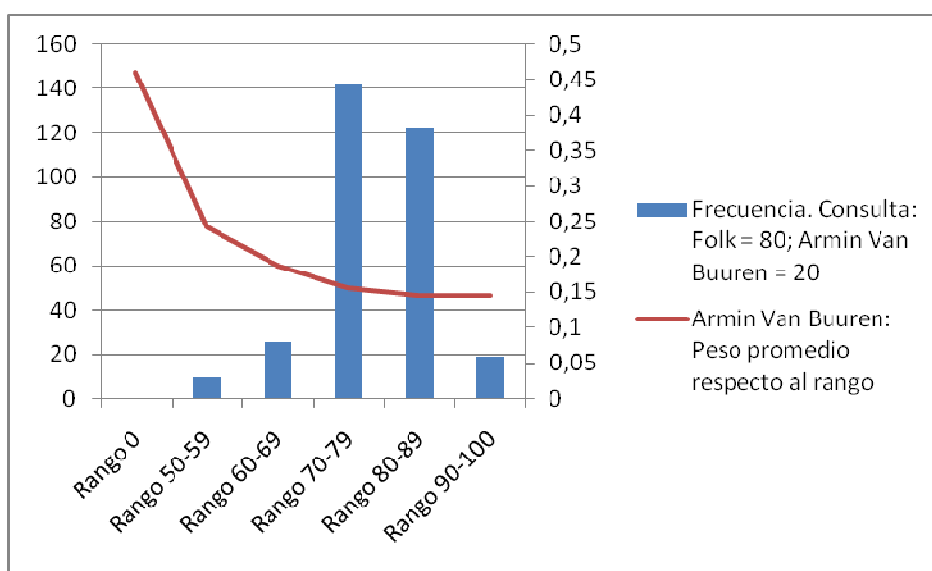


**Figura 35. Saturación. Consulta: Folk = 80; Armin Van Buuren = 20**

Para terminar con la evaluación, veremos la frecuencia y la saturación de esta consulta respecto del término Armin Van Buuren, comparando los resultados con los obtenidos en la consulta en la que las relevancias eran 50 para ambos términos.

Por supuesto, tanto la frecuencia como la saturación para el rango 0 vuelven a ser un 0%, en esto nada ha cambiado, pero sí es cierto que el siguiente rango más cercano al artista Armin Van Buuren, el rango 50-59, ha perdido más del 50% de su frecuencia a pesar de que las soluciones tengan menor cantidad de discos en esta consulta.

La frecuencia del rango 50-59 baja por tanto del 8.1% hasta el 3.0%. Además, el rango siguiente, 60-69, también baja considerablemente desde un 12.6% hasta un 8.1%, siendo el rango 70-79 el gran beneficiado, pues su frecuencia aumenta acercándose al 50% de los discos de la solución: Un 44.4%.



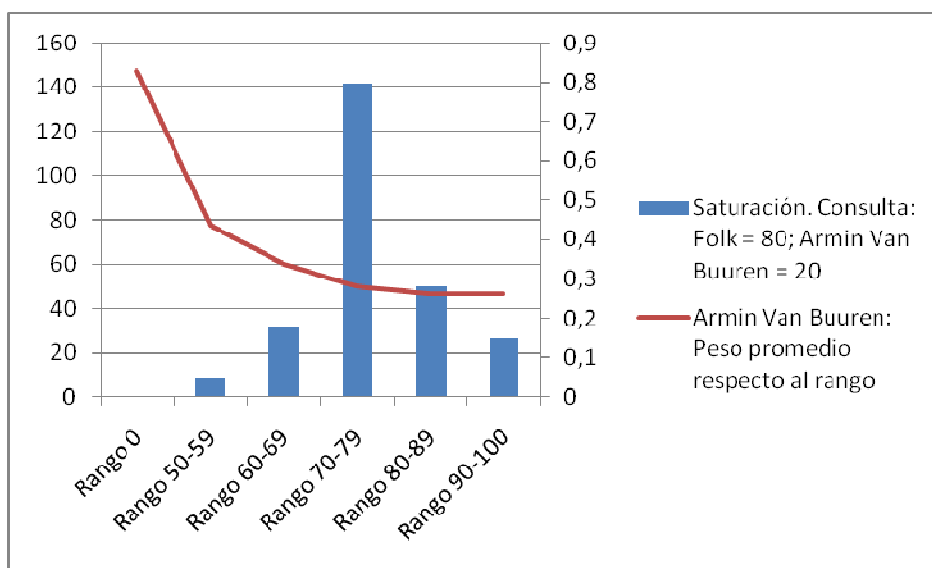
**Figura 36. Frecuencia. Consulta: Folk = 80; Armin Van Buuren = 20**

Por último, la saturación desde el punto de vista del término autor = Armin Van Buuren muestra de una forma aún más clara cómo los rangos cercanos al autor han caído en picado.

El rango 50-59 baja su saturación del 13.8% al 4.6%. En ninguna de las distintas repeticiones de la consulta ha conseguido incluir dos o más discos en la solución, e incluso en el 40% de las soluciones obtenidas ni tan siquiera ha aparecido.

Algo similar ha sucedido con el rango 60-69, cuya saturación ha bajado prácticamente a la mitad, desde un 31.1% hasta un 17.8%, con uno o dos discos en cada una de las soluciones de los nueve discos que tiene este rango.

A pesar de tener una buena cantidad de discos, once, el rango 70-79 vuelve a tener una saturación muy por encima de los otros rangos, e incluso superior al de la consulta Folk = 50; Armin Van Buuren = 50. Un 80% de discos en promedio con hasta 10 discos en alguna de las soluciones y un mínimo de siete.



**Figura 37. Saturación. Consulta: Folk = 80; Armin Van Buuren = 20**

De esta forma terminamos con la evaluación de nuestro trabajo y procedemos por tanto, en el siguiente capítulo a enumerar las conclusiones que podemos sacar de todos los resultados obtenidos en nuestras consultas además de aquello que durante el desarrollo del proyecto nos ha parecido clave.



## Capítulo 5. Conclusiones

### 5.1. Desarrollo del proyecto

El proyecto fue realizado en el Departamento de Sistemas Informáticos y Computación como parte de un año en el mismo trabajando a media jornada como becario a la vez que realizaba las últimas asignaturas de la especialidad de Ingeniería del Software en la antigua Facultad de Informática.

La realización del mismo fue en el Entorno de Desarrollo Integrado Microsoft Visual Studio 2005. Programándose todo el código en el Lenguaje C#, para el cual está especialmente preparado el entorno Visual Studio, y que era uno de los lenguajes más en alza en el momento del desarrollo.

Siendo este lenguaje en esencia una combinación de C++ y Java, lenguajes ya conocidos previamente, el tiempo de adaptación a este nuevo lenguaje de programación no fue demasiado grande.

El tiempo de documentación previo a la fase de programación fue relativamente grande. El proyecto partía de la base establecida por la Tesis Doctoral de Francisco Javier Jaén, director del proyecto. Éste fue el primer documento a estudiar.

Posteriormente la fase de documentación se centraría en el estudio del Resource Description Framework. Un Framework relativamente nuevo sobre el que había que profundizar para decidir, en primera instancia, si era conveniente para la solución del problema que queríamos abarcar con él.

Una vez concluida la conveniencia de RDF para éste trabajo, la fase de familiarización con el mismo se unió a la búsqueda y posterior tratamiento de un sistema de bases de datos adecuado a RDF, el SQLite.

Una vez terminado el desarrollo, las últimas semanas previas a la redacción del proyecto se dedicaron a la fase de pruebas que se explica detalladamente en el capítulo 4 que, como era de esperar, implicó también una pequeña depuración del código en los primeros intentos realizados.

### 5.2. Conclusiones del trabajo realizado

Eran dos los problemas que planteamos al inicio de este texto y que queríamos abarcar y en la medida de lo posible resolver con la realización de este trabajo.

Por un lado, sabíamos que los algoritmos evolutivos basados en colonias de hormigas era una buena forma de encontrar soluciones con puntuación máxima sin superar restricciones, pudiendo así resolver el llamado problema de la orientación; pero las implementaciones que se habían realizado hasta el momento solían seguir una estructura centralizada que suponía

un claro impedimento a la hora de realizar estudios sobre espacios de búsqueda de gran tamaño.

Mediante el uso del algoritmos de OCH paralelos, con el uso de súper colonias de hormigas compuestos por colonias que solucionan segmentos del problema de forma detallada y la posterior unión de estas soluciones parciales para formar una general, conseguimos, sin sacrificar en tiempo de computación, resolver este primer problema.

El segundo problema era que el espacio de búsqueda quedaba predeterminado en el momento de la implementación del algoritmo y por tanto la aplicación obtenida era dependiente del tipo de datos, haciendo imposible su reutilización.

La solución a este problema llegó mediante el uso de RDF para generalizar la definición del espacio de búsqueda y posteriormente independizar la creación del mismo de la implementación del algoritmo OCH.

Así, nuestra aplicación puede trabajar resolviendo cualquier otro problema de éste tipo sin sufrir cambio alguno a nivel de código. Basta con que se cree una definición en RDF de la base de datos sobre la que se trabajará y se implementen las componentes del calculo de distancias y pesos.

Un problema que vemos que ha surgido a la hora de obtener resultados a nuestras consultas en el cuarto capítulo del texto es la importancia del peso de los ítems con que se trabaja. Si bien este problema es seguramente dependiente del problema planteado en cuestión, es importante tener en cuenta esto para ver si se mantuviese esta influencia en futuros resultados obtenidos.

### **5.3. Trabajos futuros**

En trabajos futuros sería interesante realizar la implementación de nuestro sistema en productos reales de distintos ámbitos, para ver su posible funcionamiento.

Teniendo en cuenta el estado actual del desarrollo de aplicaciones para dispositivos móviles basadas en localización, ámbitos como el generador de itinerarios de viajes, además del ya tratado en nuestro trabajo de reproducción de listas musicales pueden tener una gran aceptación a nivel social si los resultados son tan buenos como nuestras pruebas han parecido mostrar.

Por supuesto no se tiene que limitar a este tipo de trabajos el desarrollo futuro de nuestro proyecto, pues son muchos los ámbitos en los que un algoritmo que solucione el problema de la orientación para volúmenes de datos muy amplios pueden ser usados.

## Bibliografía

- [1] Liang, Y, Smith, E. A. “An Ant Colony Approach to the Orienteering Problem”. IEEE Transactions On Evolutionary Computation. 2001
- [2] SQLite Home Page. <http://www.sqlite.org/index.html>
- [3] Semantic Web/RDF Library for C#/.NET.  
<http://razor.occams.info/code/semweb/>
- [4] SQLite Database Browser. <http://sqlitebrowser.sourceforge.net/>
- [5] Pohle, T., Pampalk, E., Widmer G. “Generating Similarity-Based Playlists Using Traveling Salesman Algorithms”. Proc. Of the 8<sup>th</sup> Int. Conference on Digital Audio Effects (DAFx’05), Madrid, Spain, September 20-22, 2005.

