Estructuras de Datos y Algoritmos

Departamento de Informática
Universidad de Valladolid

Curso 2014-15

Grado en Ingeniería Informática
Grado en Estadística

Motivación (vida real)

- Detectar (medir) si la solución a un problema es **escalable** o no. Saber que en caso contrario pueden existir otras alternativas (otros algoritmos y/o estructuras de datos).
- Comprender cómo (y porqué) se estructuran las bibliotecas de contenedores más usadas.
  - Standard Template Library (STL) de C++
  - Collections en Java, .Net
  - Organización interna de lenguajes como JavaScript (hashtables), Python (dictionaries)
- Poder crear aplicaciones donde la eficiencia sea esencial.

Ejemplo #1: Shazam

- Aplicación para averiguar la canción que está sonando en un entorno (mediante el teléfono móvil).
- Se obtiene un fragmento de unos 10 seg. y se envía a un servidor.
- El servidor identifica las posibles canciones y envía la lista al teléfono móvil.
- El fragmento puede ser cualquiera.
- El entorno puede ser ruidoso.
- El número de posibles canciones supera el millón.
- La identificación debe tardar segundos.

Metodología

- Cada canción se representa por su huella acústica: Información comprimida de forma especial para facilitar su identificación.
- La base de datos almacena huellas, no canciones.
- Es necesario encontrar la huella adecuada:
  - Máxima compresión.
  - Mínima entropía (capacidad de identificación).
  - Resistente a ruido
  - Independiente de la posición dentro de la canción.
- El método se basa en el cálculo de puntos especiales (constelación) en el espectograma.

Espectograma

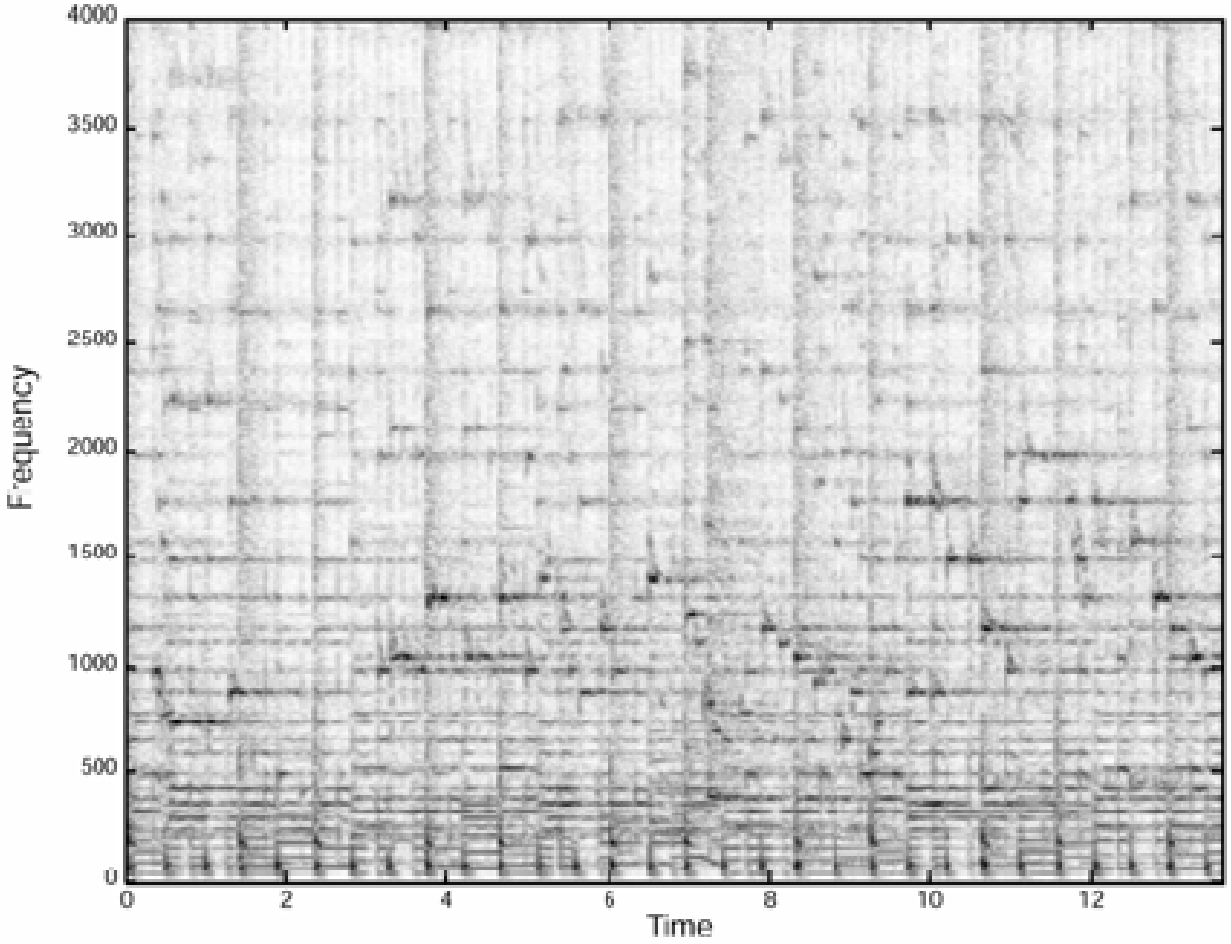


Fig. 1A - Spectrogram

Constelación

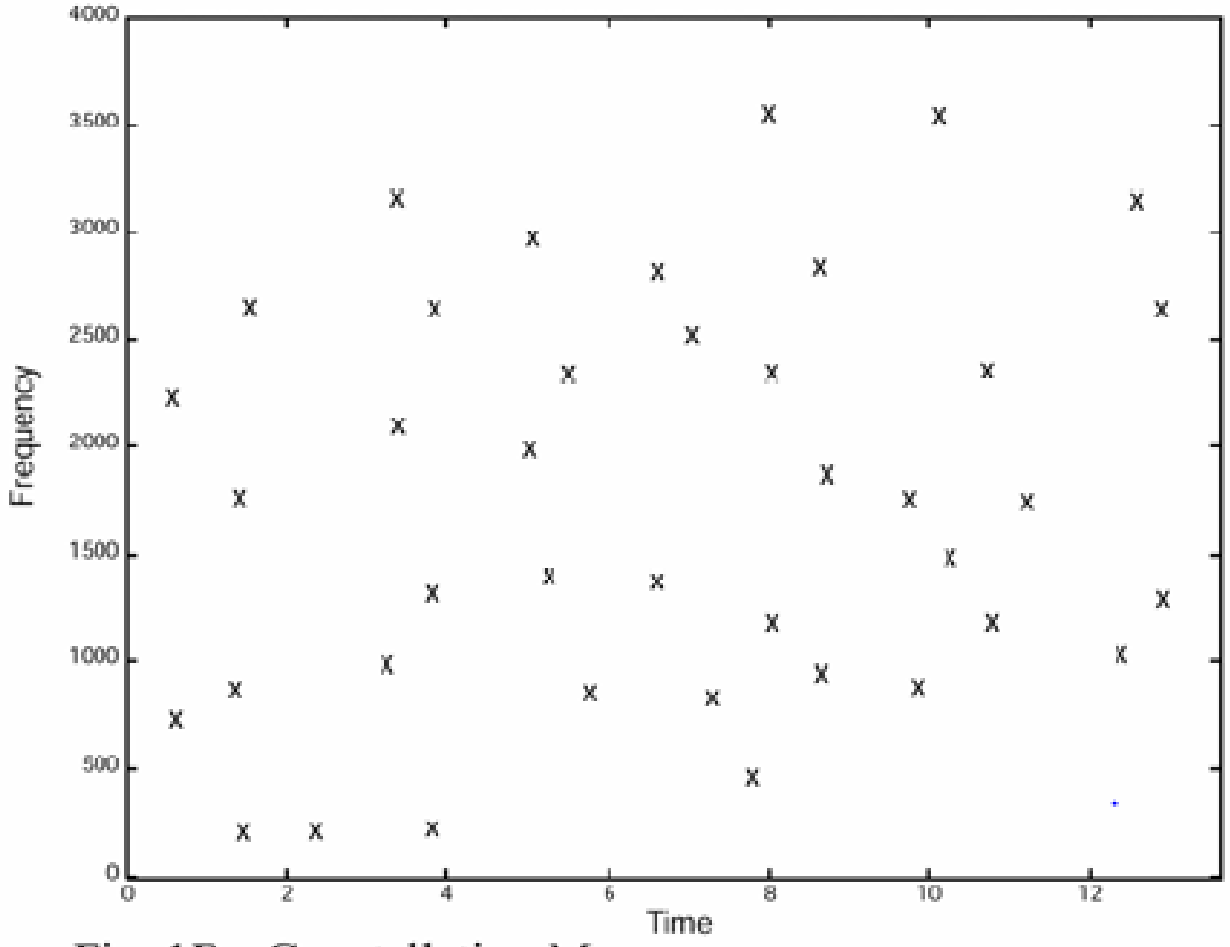


Fig. 1B - Constellation Map

Diferencial

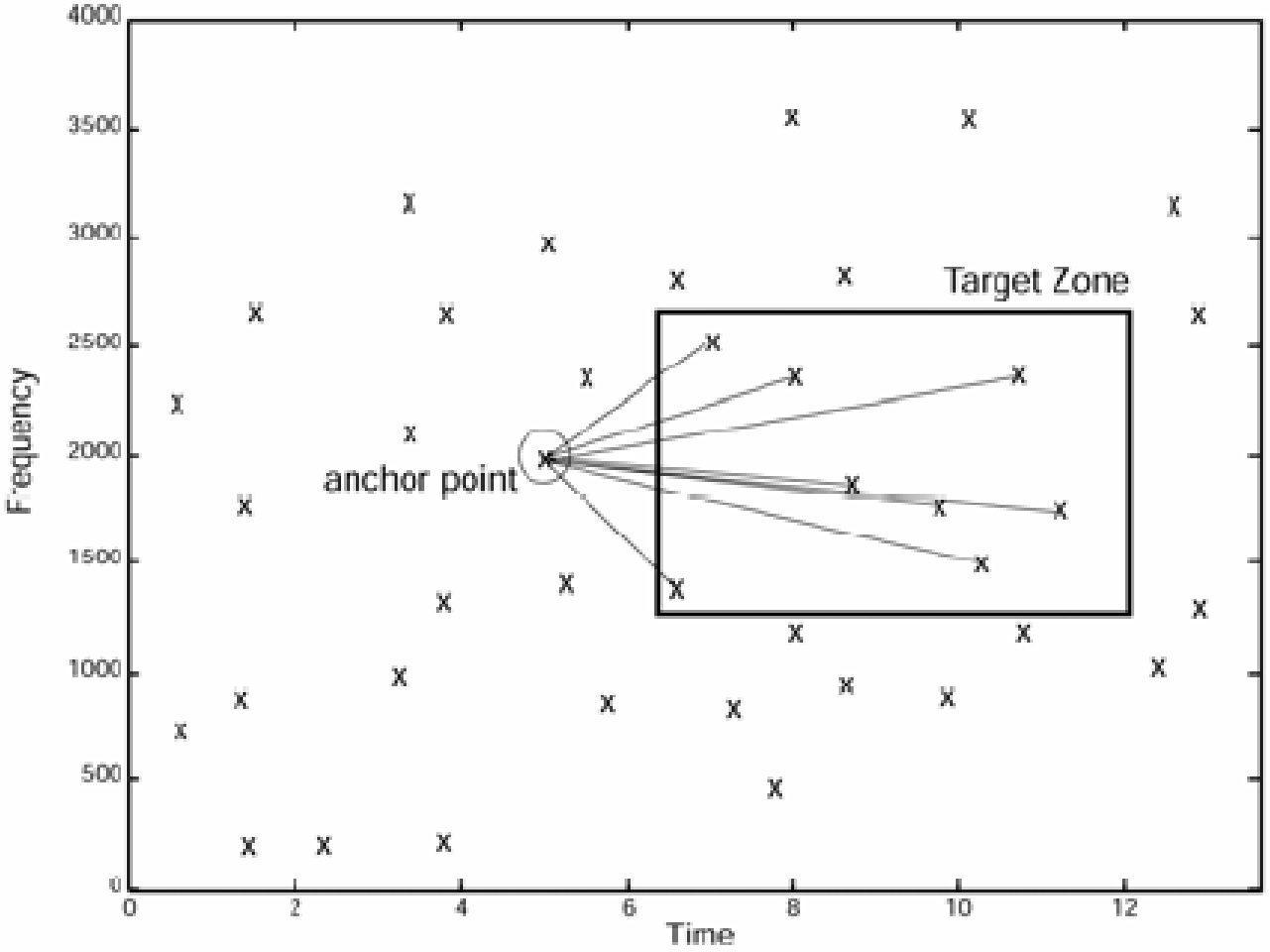


Fig. 1C - Combinatorial Hash Generation

Primera etapa

- Procesar un número grande de canciones para encontrar los parámetros óptimos de generación de constelaciones.
- Problema 1: Convencer a una discográfica para que te permita trastear con su base de datos de canciones.
- Problema 2: Encontrar un **algoritmo** que pueda tratar los datos en un tiempo razonable y en equipos modestos (4 meses de cómputo).

Segunda etapa

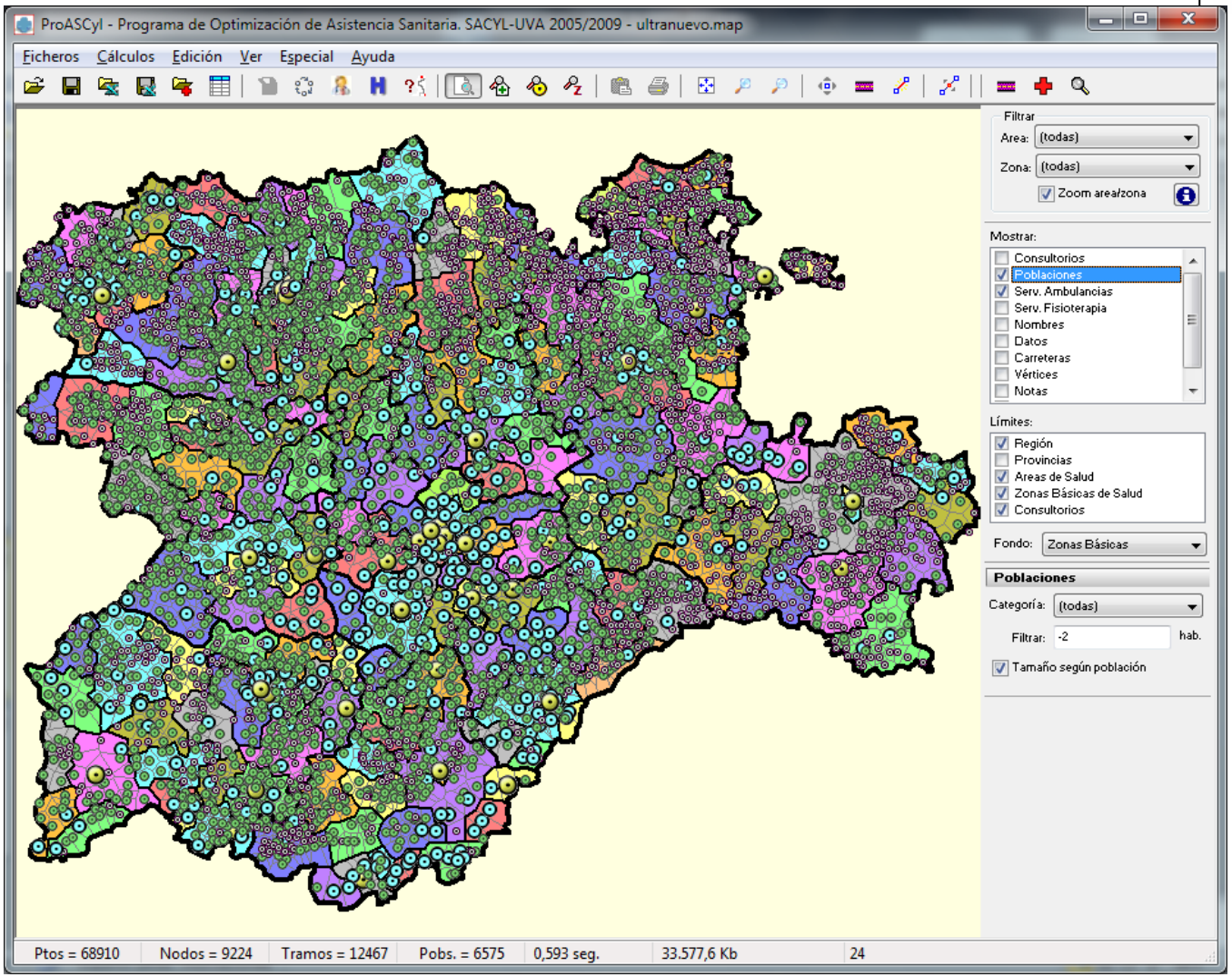
- Encontrar una **estructura de datos** adecuada para el servidor.
- Que almacene las huellas de las canciones de manera que la búsqueda de fragmentos sea lo más rápida posible.
- Que permita actualizar su contenido en tiempo razonable.
- Que ocupe un espacio de almacenamiento razonable.

www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

Ejemplo #2: ProasCyL

- Aplicación para optimizar la localización de recursos sanitarios en Castilla y León.
- Objetivos:
  - Localizar poblaciones donde situar servicios sanitarios.
  - Que cubran la mayor demanda posible.
  - En el menor tiempo de acceso.
  - Con otras restricciones secundarias.
  - Problema parametrizable
- Dificultades:
  - Más de 6.000 poblaciones en CyL.
  - Problema con características combinatorias

ProasCyL


ProASCyl - Programa de Optimización de Asistencia Sanitaria. SACYL-UVA 2005/2009 - ultranuevo.map
Ficheros Cálculos Edición Ver Especial Ayuda
Filtrar
Area: (todas)
Zona: (todas)
Zoom area/zona
Mostrar:
Consultorios
Poblaciones
Serv. Ambulancias
Serv. Fisioterapia
Nombres
Datos
Carreteras
Vértices
Notas
Límites:
Región
Provincias
Areas de Salud
Zonas Básicas de Salud
Consultorios
Fondo: Zonas Básicas
Poblaciones
Categoría: (todas)
Filtrar: -2 hab.
Tamaño según población
Ptos = 68910
Nodos = 9224
Tramos = 12467
Pobs. = 6575
0,593 seg.
33.577,6 Kb
24

Dos soluciones

Basada en GIS	ProasCyL
ED y algoritmos “estandar”	ED específica, algoritmos adaptados al problema
Resuelve un problema restringido	Resuelve el problema general
Ejecución: Supercomputador “Origin”, 18.000 €	Ejecución: PC normal
Tiempo de cómputo:	Tiempo de cómputo:

Dos soluciones

Basada en GIS	ProasCyL
ED y algoritmos “estandar”	ED específica, algoritmos adaptados al problema
Resuelve un problema restringido	Resuelve el problema general
Ejecución: Supercomputador “Origin”, 18.000 €	Ejecución: PC normal
Tiempo de cómputo: **4 dias**	Tiempo de cómputo: **3 seg**.