Anonimización de citas médicas mediante programación con restricciones

Trabajo Fin de Máster

Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Curso 2015/2016
Convocatoria Septiembre 2015/2016 Nota:8

Alumno

Richard Jordan Cabana Ramírez

Tutor

Rafael Caballero Roldán

El abajo firmante, matriculado en el Máster en Ingeniería Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "*Anonimización de citas médicas mediante programación con restricciones*", realizado durante el curso académico 2015-2016 bajo la dirección de Rafael Caballero Roldán en el Departamento de Sistemas Informáticos y Computación, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en internet y garantizar su preservación y acceso a largo plazo.

28 de septiembre de 2016

Richard Jordan Cabana Ramírez

Me gustaría agradecer a todas las personas con las que he trabajado estos dos años y me han apoyado con los trabajos, exámenes y prácticas con las que he podido aprender muchas cosas. Amigos que he conocido durante el máster y que siempre me han acompañado y dado lo mejor de sí. A los profesores que han puesto sus esfuerzos en enseñarnos los conocimientos que hacen posibles trabajos como este. A mi familia por animarme a seguir estudiando para aprender más y más, sobretodo en los momentos difíciles.

Y en especial, a mi tutor Rafael Caballero Roldán que me ha apoyado desde el inicio hasta el final y que gracias a sus conocimientos e ilusión han hecho para mí que este proyecto sea tan interesante y motivador.

Richard

Resumen

El actual auge de internet y las comunicaciones genera inmensas cantidades de información. De entre toda esta información tiene especial interés aquella que es personal y privada, ya que puede comprometer la seguridad o intimidad de las personas. La información médica de una persona es una de las más vulnerables y es de vital importancia proteger la identidad de los pacientes, así como los resultados y diagnósticos a los que se someten.

Este trabajo se centra en intentar dificultar la identificación de una persona a partir de datos que se publican en las bases de datos médicas de los programas de screening, generando citas que agrupen individuos con mismos datos públicos: edad, género, etc... Se propone una solución basada en la programación con restricciones y se presenta una nueva medida de anonimato. Se compara experimentalmente el incremento en el anonimato que se obtiene en nuestra propuesta con respecto a la generación aleatoria de citas.

Palabras clave: programación con restricciones, screening, K-anonimato, optimización, Choco, Java, experimentos, vector de anonimato, cuasi-identificador, heurística.

Abstract

With the rise of Internet and communications that currently exist, the amount of information generated is increasingly growing. This is especially relevant in the case of personal and private data, which can compromise the security or privacy of individuals. Medical information is one of the most vulnerable and it is vital to protect the identity of patients, and the results and diagnoses.

This work focuses on trying to avoid the identify of people from data published in the medical databases of screening programs, generating appointments that group together individuals with same public data: age, gender, etc... We propose a solution based on constraint programming and a new measure of anonymity. We compare experimentally the Increased of anonymity of our proposal with respect to a random generation of appointments.

Keywords: constraint programming, screening, K-anonymity, optimization, Choco, Java, experiments, anonymity vector, quasi-identifier, heuristic.

Índice general

1. Introducción

1.1 Problema del anonimato

Hoy en día vivimos en una sociedad cada vez más tecnológica, en la que un número creciente de servicios conlleva depositar cierta información personal que debe ser utilizada y guardada cuidadosamente. Los que custodian la información deben asegurar que se utiliza únicamente para los fines pensados, garantizando la protección e integridad de esa información, así como que la identidad de las personas queda oculta.
Aunque este problema puede aparecer en todos los campos, es especialmente sensible la información relacionada con la salud de las personas. Cuando nosotros vamos al médico por alguna posible enfermedad o estudio, esa información queda guardada y protegida del público en general. Es decir, solo pueden acceder ciertas personas a esa información.

¿Por qué es importante proteger esa información? El problema de conocer la información médica de un paciente es que puede ser utilizada de varias formas en perjuicio suyo.
Por ejemplo, al optar a un puesto de trabajo, saber que una persona tiene una enfermedad o problemas de salud puede hacer que quede eliminada del proceso de selección, los seguros médicos pueden imponer precios superiores debido a esos problemas, puede haber personas que se aprovechen de esa situación para beneficio propio, etc.

Por estas razones es muy importante que dicha información quede protegida y sea anónima para el público en general. En España, el apartado 5.1.g) del Reglamento de desarrollo de la Ley Orgánica 15/1999, aprobado por Real Decreto 1720/2009, de 21 de diciembre, indica que se consideran datos de carácter personal relacionados con la salud “las informaciones concernientes a la salud pasada, presente y futura, física o mental, de un individuo”.

1.2 Algunas soluciones

Hay ocasiones en las que interesa publicar información médica para que investigadores y laboratorios puedan realizar estudios. Esto es algo muy positivo ya que estos estudios ayudan a detectar problemas sanitarios o encontrar patrones de algunas patologías, por ejemplo. Pero en dicha información compartida todo debe ser anónimo para evitar los problemas anteriormente dichos.

Normalmente se suelen eliminar datos como el DNI, o el nombre del paciente para no poder detectar a esa persona, dejando solamente los campos genéricos, como pueden ser, edad, raza, CP, y el diagnóstico.

	Race	Birth	Gender	ZIP	Problem
t1	Black	1965	m	0214*	short breath
t2	Black	1965	m	0214*	chest pain
t3	Black	1965	f	0213*	hypertension
t4	Black	1965	f	0213*	hypertension
t5	Black	1964	f	0213*	obesity
t6	Black	1964	f	0213*	chest pain
t7	White	1964	m	0213*	chest pain
t8	White	1964	m	0213*	obesity
t9	White	1964	m	0213*	short breath
t10	White	1967	m	0213*	chest pain
t11	White	1967	m	0213*	chest pain

Figura 1 Ejemplo de información publicada en estudios médicos [1]

Pero se ha comprobado que con esta información genérica es posible identificar a un individuo concreto cruzando información con otras tablas procedentes de otras fuentes.

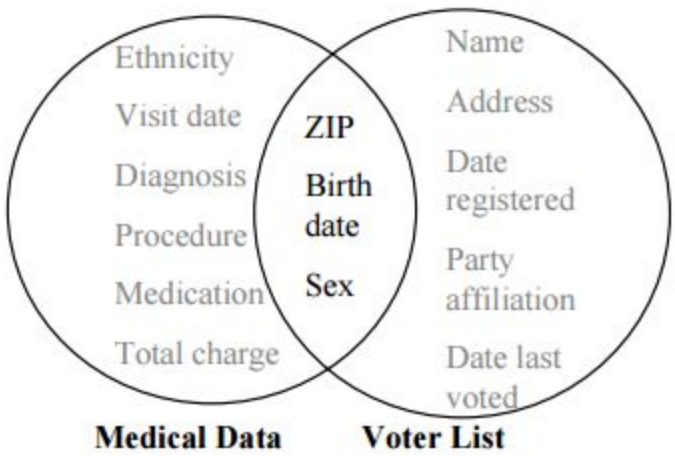


Figura 2 Cruce de información médica con lista de votantes [1]

Es decir, podemos encontrar a una persona en particular aunque no estén sus datos de identificación. Por ejemplo puede darse el caso que con el CP, edad y sexo se identifique a un individuo en concreto a partir del censo electoral. Para intentar solucionar esto, y que no quede nadie detectable se utilizan varias técnicas a posteriori.

1.2.1 Introducir Wild Cards

Esta solución elige un atributo de la tabla, por ejemplo el código postal, y lo modifica para no dar toda la información.

28040 ⇒ 280**

De esta forma conseguimos que no haya un único individuo con ese código postal y varios tengan ese mismo código modificado. El problema es que haciendo esto, perdemos información de la zona o de la información que daba ese campo, por lo que dependiendo del estudio que se quiere realizar la pérdida de información puede afectar a los resultados.

1.2.2 Intervalos

Consiste en cambiar un campo de la lista por un cierto intervalo [2].
Por ejemplo la edad:

45 años ⇒ > 40 y < 50 años

Con ello conseguimos tener más gente en ese intervalo ocultando a alguien que tenga una determinada edad. También perdemos información al utilizar este método.

1.2.3 Borrar Filas o Datos Falsos

Otra forma de evitar la detección de las personas es introduciendo datos falsos o copias [3]. También puede ocurrir que haya algún caso excepcionalmente raro, que destaque sobre todos los demás por lo que puede ser conveniente eliminarlo por seguridad. Este método tiene como contrapartida la pérdida de calidad en los datos.

1.3 K-anonimato

Partimos de una información referente a personas que podemos suponer que está organizada en forma de tabla. Llamamos *atributos* a las columnas de dicha tabla (por ejemplo, dni, nombre, apellidos, CP(código postal), edad, ...).

Llamamos *cuasi-identificador* a un subconjunto de atributos de la tabla que, en principio, no identifican unívocamente a la persona. Por ejemplo {CP, edad, sexo} es un posible cuasi-identificador. Para simplificar asumimos que la tabla tiene un cuasi-identificador ya conocido. Este cuasi-identificador tomará una determinada cantidad Q de valores diferentes $v_1 \ldots v_Q$ en la tabla. Si N es el número de individuos en la tabla y q_i representa el número de individuos para los que el cuasi-identificador toma el valor $\mathbf{v_i}$, $1 \leq i \leq Q$, se verifica:

1. $Q \leq N$
2. $q_1 + \ldots + q_Q = N$

Para poder cuantificar de alguna manera la privacidad de los pacientes en la tabla, en este trabajo utilizamos una medida conocida como K-anonimato [4][1] .

Se dice que una tabla verifica un nivel K de anonimato (K >= 1) cuando los valores asociados al cuasi-identificador de todo individuo se repiten en la tabla al menos K veces. Utilizando la notación anterior:

$$\mathbf{K = \min \{ q_i \mid 1 \leq i \leq Q \}}$$

Es decir, cuanto mayor sea K mayor nivel de privacidad tenemos. Por el contrario, para valores de K pequeños y en particular para K = 1 puede ser relativamente fácil detectar a un individuo y de esta forma revelar datos sobre su diagnóstico.

1.4 Screening

En este trabajo nos centramos en el problema de la privacidad dentro de los programas de screening [6] . Se trata de programas masivos de citas médicas para pruebas clínicas, que se aplican a grupos de personas que en teoría se encuentran sanas, pero que se sospecha que pueden tener algún tipo de enfermedad o patología. Normalmente se trata de segmentos de población con unas características determinadas, como por ejemplo, deportistas profesionales de entre 16 y 25 años a los que se desea realizar una prueba para la detección temprana de cardiopatías.

En principio asumimos que de la información personal de las personas a citar ya se ha eliminado los datos identificativos y solo nos quedan los cuasi-identificadores. Por tanto la población inicial parte de un cierto nivel de k-anonimato, al que determinamos *anonimato base*. Al repartir las citas entre las personas, tendremos que añadirles la información de las citas (por ejemplo lugar de la prueba médica y hora). Para algunos estudios puede ser relevante saber el lugar de la prueba por lo que necesitaremos publicar esa información, por lo que podemos considerar que esta información adicional se añade al conjunto de atributos que determinan el cuasi-identificador. Es fácil comprobar que al introducir nuevos atributos como parte del cuasi-identificador el nivel de k-anonimato se reduce: personas con el mismo valor de cuasi-identificador en la situación inicial, pueden pasar ahora a ser distinguibles si toman valores distintos en los nuevos atributos.

Pensemos en una persona (atacante) que sepa que un paciente ha ido al lugar de la cita en la fecha concreta para hacerse las pruebas. Puede ser un vecino, un pariente, un recepcionista, personal hospitalario, o incluso un periodista que sigue a un “famoso”. Para concretar pensemos en alguien del personal del centro hospitalario, que tiene acceso a la información de la cita y pueden ver a la persona justamente en ese lugar y hora y que pueden saber que el programa de screening probablemente pueda involucrar a un conocido suyo. Si el nivel de k-anonimato base es suficientemente alto, esta persona en principio no podrá saber el resultado de las pruebas médicas aunque tenga acceso a la documentación final, porque recordemos que esta información sólo contendrá datos como el CP, edad, y hay muchas personas con esa misma información en el programa de screening.

Pero si se añade la información de la cita, porque el atacante ve a su conocido en el centro médico a una hora determinada y solo hay una persona de la lista citada en ese centro a esa hora, con ese CP, edad, etc. se habría perdido el anonimato y se puede conocer el diagnóstico de dicho paciente. Es este riesgo el que intentamos disminuir en el presente trabajo.

1.5 Generación de Citas

Por tanto, debemos encontrar una forma de repartir las citas de forma que la K tras añadir la información de las citas sea máxima. Dicho de otra forma, buscamos minimizar la pérdida de anonimato al incorporar información de la cita al cuasi-identificador. Vamos a ver distintas posibilidades para la generación de citas. Hay que señalar que en los programas de screening las citas se realizan de antemano y sin participación de los individuos implicados, que reciben una comunicación con los datos de la cita médica.

1.5.1 Aleatoria

En la práctica a menudo se usa un método aleatorio para repartir las citas, que como es obvio no tiene en cuenta la disminución de k y por tanto puede generar una importante reducción del anonimato.

1.5.2 Heurístico

Una primera posibilidad de mejora consiste en repartir las citas agrupando un conjunto de pacientes con características comunes en el mismo lugar y a la misma hora para que así haya varios con la información de la cita igual, lo que aumenta el K-anonimato. Cuanto el atacante intente detectar a un paciente según su información pública (por ejemplo por haberle visto acudir a un centro hospitalario y tener por tanto la información de la cita) se encontrará que varias personas tienen esa misma cita médica. El problema de este método, es que cuando se agrupa a los pacientes con este criterio, puede quedar alguno que no pueda ser incluido con otras personas de las mismas características (cada cita tiene una capacidad, el máximo de personas que se pueden atender en ese lugar y turno). En este caso la persona que queda "descolgada" queda expuesta a ser detectado fácilmente. Resolver este problema no es sencillo ya que se trata de un problema combinatoriamente complejo, ya que se ha demostrado que el problema de la búsqueda del k-anonimato óptimo es *NP-hard* [7].

1.5.3 Programación con restricciones

La programación con restricciones [8][11] es un paradigma de la programación en informática donde las relaciones entre las variables se expresan en términos de restricciones (ecuaciones). Se suele utilizar para resolver problemas combinatorios especialmente difíciles. Se trata de un paradigma declarativo, en el que se distingue entre el *modelo*, que define el problema mediante variables y restricciones, y el *resolutor*, que se encarga de buscar las soluciones del problema. Habitualmente se distingue entre problemas de satisfacción de restricciones (denominados CSP por las siglas en inglés de *Constraint Satisfaction Problem*), que buscan valores de las variables que hacen ciertas todas las restricciones, y *problemas de optimización*, que de entre todas las soluciones del CSP eligen aquellas que minimizan/maximizan una cierta función.

Los modelos suelen tener un dominio básico para las variables, y los resolutores suelen ser específicos para cada dominio. Dominios habituales suelen ser los booleanos (en este caso hablamos de problemas *SAT*), subrangos de los enteros (problemas de *dominios finitos*) o números reales. En ocasiones se tratan otros dominios más complejos, como conjuntos, dominios tipo grafo, o incluso combinaciones de varios dominios [9], pero en este trabajo utilizaremos variables de tipo entero.

En particular, utilizaremos la librería Java Choco [10]. Choco dispone de documentación, tutoriales, múltiples materiales, foros, y mucho más accesible en internet. Se trata de una librería disponible como código abierto y al estar implementado en Java tiene mayor portabilidad y acogida entre los programadores. Maneja variables de diversos tipos como variables enteras, reales, booleanos, vectores, conjuntos, etc… pero en lugar de como variables en el lenguaje nativo (Java), estas variables son variables Choco que contienen información adicional que utiliza el resolutor para poder encontrar las soluciones.

Choco dispone de más de 70 restricciones predefinidas que cubren la mayoría de dominios para resolver el problema (*equal*, *less*, *not equal*, etc…). Además puede trabajar con dominios como enumerados, acotados o listas. Tiene implementados varios algoritmos de búsqueda que podemos configurar según nuestro tipo de problema, ya que puede mejorar la eficiencia en algunos casos. Se pueden conseguir una o varias soluciones si existen, también tienen métodos de optimización de las soluciones (maximización y minimización).

Utilizaremos programación con restricciones para generar citas de forma que la pérdida de anonimato sea mínima. Para resolver este problema, tenemos que definir el modelo y usar las variables y relaciones que verifican las restricciones sin necesidad de pensar en la implementación.

1.6 Objetivos

Nuestro objetivo es poder publicar los resultados de un programa de screening médico, incluyendo datos de la cita, sin comprometer la identidad y privacidad de los participantes en el programa. Para ello intentaremos repartir las citas para dichas pruebas teniendo en cuenta la privacidad de los pacientes, y proponemos medidas que permitan detectar la pérdida de privacidad. Dicho de otra forma, hay que intentar que el K-anonimato no empeore al dar las citas (añadir información más restrictiva siempre baja el K) o que baje lo menos posible. El objetivo final es un sistema que asigne citas para programas de screening pero que también nos permite realizar experimentos para determinar la calidad de la propuesta.

Es importante señalar que en nuestro trabajo damos un enfoque diferente al método habitual que se utiliza para aumentar la privacidad porque intentamos mejorar la privacidad a priori, desde que repartimos las citas y no a posteriori, modificando los resultados de los estudios. La modificación a posteriori, una vez, conocido el resultado del diagnóstico es un problema diferente, y las técnicas ya conocidas siguen siendo aplicables [12][13].

1.7 Contribuciones

Las contribuciones del trabajo se pueden resumir en la siguiente lista:

- Definimos un nuevo concepto de anonimato que refina el concepto ya conocido del K-anonimato. Esto se hace mediante los llamados *vectores de anonimato.*
- Estudiamos las propiedades de los vectores de anonimato.
- Proponemos una solución para el problema de las citas en programas de screening basada en programación con restricciones.
- Estudiamos la eficiencia de la propuestas anteriores con respecto a:
    - La generación aleatoria de citas
    - La generación con una heurística que introducimos en el trabajo

1.8 Estructura del trabajo

Haremos una breve descripción de los apartados de este trabajo.

Capítulo 1. Introducción: Cuenta la importancia del problema de anonimato y explica algunas de las soluciones que se aplican actualmente. Define el concepto básico de K-anonimato y describe un caso real de screening y los problemas que se presentan. Por último hace una descripción de las distintas formas de generar citas y los objetivos para este trabajo.

Capítulo 2. Vectores de anonimato: Introduce el concepto de vector de anonimato y sus propiedades aplicadas al problema del anonimato. Se explica cómo utilizamos las particiones de teoría de números para definir una distancia entre vectores de anonimato y poder medir la calidad de las soluciones.

Capítulo 3. La asignación de citas como un problema de optimización con programación con restricciones: En este apartado se cuenta el algoritmo principal desarrollado en este proyecto usando el paradigma de programación con restricciones incluyendo el pseudocódigo del algoritmo.

Capítulo 4. Comparativa entre las diversas formas de generar las citas: Presentamos las 3 formas de generar citas utilizadas en este proyecto, la aleatoria, la de programación con restricciones y como alternativa práctica un algoritmo heurístico. Se presentan los resultados obtenidos de los experimentos con los distintos algoritmos.

Capítulo 5. Prototipo: Muestra el funcionamiento y la utilización de una interfaz desarrollada para poner a prueba el algoritmo con restricciones, donde se crean pacientes, recursos y como resultado final se reparten las citas de la forma óptima.

Capítulo 6. Conclusiones: Explicamos los resultados y conclusiones que hemos encontrado en el desarrollo del proyecto así como ideas para mejorar y ampliar este trabajo.

2. Vectores de Anonimato

2.1 El papel de los cuasi-identificadores

En el capítulo 1 hemos visto que una primera medida para incrementar la privacidad es eliminar los identificadores personales, sin embargo, la información obtenida después de eliminar los identificadores todavía no es segura desde el punto de vista del anonimato.

En [1] muestra cómo en 1990 más del 87% de los individuos de Estados Unidos pueden ser identificados unívocamente a partir de sus 5 dígitos del código postal, género y su fecha de cumpleaños. Por esta razón, las columnas de género, edad y código postal son conocidos como cuasi-identificadores. L. Sweeney propuso el K-anonimato como una forma sencilla de calcular el nivel de anonimato de un conjunto de datos que contiene cuasi-identificadores. Como vimos en el apartado 1.3, para hallar el nivel K hay que determinar el número de repeticiones de cada valor tomado por el cuasi-identificador, y calcular el mínimo de todos los valores. Este número se denomina K y el conjunto de datos se dice que verifica un nivel K de anonimato. Así, hablamos de 1-anonimato (el peor posible), 2-anonimato, etc.

Por ejemplo, podemos considerar la siguiente tabla con N = 15 individuos.

	Zip	Gender	Age	Test
1	88888	male	20-25	✔
2	11111	male	25-30	✔
3	11111	female	25-30	✔
4	88888	female	25-30	✘
5	88888	male	20-25	✔
6	11111	female	25-30	✔
7	11111	female	25-30	✔
8	11111	male	25-30	✔
9	11111	female	25-30	✔
10	88888	female	25-30	✔
11	11111	male	25-30	✘
12	11111	female	25-30	✔
13	88888	male	20-25	✔
14	88888	male	20-25	✘
15	11111	male	25-30	✔

Tabla 1: Datos de ejemplo y sus cuasi-identificadores coloreados.

Se muestran en distintos colores los Q = 4 valores de los cuasi-identificadores.

V_1 **= (88888, male, 20-25)**
V_2 **= (11111, male, 25-30)**
V_3 **= (11111, female, 25-30)**
V_4 **= (88888, female, 25-30)**

De esta tabla podemos sacar la información:

- $\mathbf{V_1}$ se repite 4 veces. Decimos que $\mathbf{q_1} = 4$
- $\mathbf{V_2}$ se repite 4 veces. Decimos que $\mathbf{q_2} = 4$
- $\mathbf{V_3}$ se repite 5 veces. Decimos que $\mathbf{q_3} = 5$
- $\mathbf{V_4}$ se repite 2 veces. Decimos que $\mathbf{q_4} = 2$

Con estos datos podemos obtener K:

$$\mathbf{K = \min \{ 4, 4, 5, 2 \} = 2}$$

Podemos decir que la tabla 1 tiene un nivel 2-anonimato.

Supongamos que un supuesto atacante conoce a una mujer de edad entre 25 y 30 años y que vive en la zona con código postal 88888, y tiene acceso a los resultados del screening realizado a los pacientes. Con esta información todavía no es posible concretar el resultado del test para esta persona, porque hay al menos otros 3 individuos que tienen el mismo cuasi-identificador.

Como nota aparte, se debe mencionar que en el caso de que las 4 mujeres tuvieran el mismo resultado del test el atacante sí conocería el resultado. Incluso si la distribución de probabilidades para este cuasi-identificador fuera notablemente diferente que para el resto, el atacante tendría “pistas” para determinar con cierta probabilidad el resultado del test para el individuo conocido. Estos son los problemas que afrontan las medidas de anonimato conocidas como t-closeness y ℓ-diversity [13][12]. Sin embargo, en nuestro caso nos centramos en el caso de las citas de screening, cuando aún no se conoce el resultado del test. Por ello nos centramos en la medida de k-anonimato a priori, asumiendo que los problemas que surjan al añadir los resultados del test con posterioridad serán tratados entonces.

2.2 K-Anonimato Generalizado

En la siguiente figura se muestran unas tablas de ejemplo para explicarlo mejor.

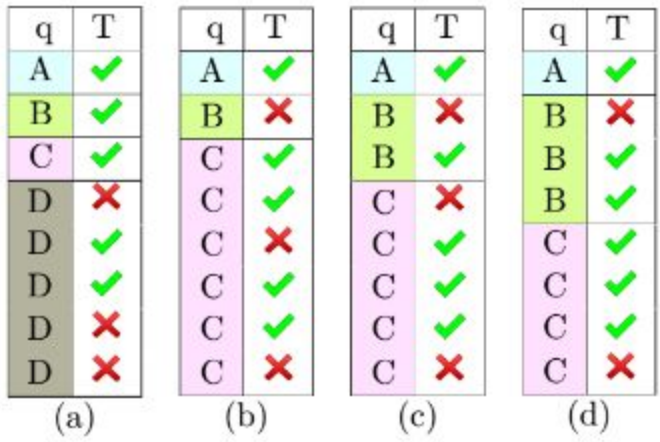


Figura 3. Tablas de ejemplo.

En todos los casos consideramos el cuasi-identificador compuesto de una sola columna *q*. Podemos ver que las 4 tablas tienen un nivel 1-anonimato, porque el número mínimo de repeticiones de cada valor tomado por el cuasi-identificador {q} es uno. Pero, ¿es correcto decir que las 4 tablas son igual de anónimas? Fijándonos con más detalle podemos ver que:

- La tabla (a) incluye 3 personas(cuasi-identificadores A, B, C) cuyos resultados del test pueden darse a conocer si alguien conoce su cuasi-identificador.
- La tabla (b) contiene 2 personas(cuasi-identificadores A, B) en esta situación .
- Las tablas (c) y (d) solamente tienen un cuasi-identificador con una repetición (A).

Por tanto parece razonable decir que la tabla (a) es la menos anónima seguido de la tabla (b), y finalmente por (c) y (d). Para comparar las tablas (c) y (d) necesitamos fijarnos en el siguiente nivel de riesgo. La tabla (d) no tiene ningún cuasi-identificador con dos repeticiones, pero la tabla (c) tiene un cuasi-identificador con dos repeticiones (B), y por eso podemos decir que la tabla (c) es menos anónima que la tabla (d). En general, podemos establecer un orden que nos permita comparar el nivel de anonimato de diferentes tablas usando un nuevo concepto, los vectores de anonimato.

Definición 1. Vector de anonimato
Sea T una tabla con datos de N individuos y con un determinado cuasi-identificador (conjunto de atributos). Sea Q el número de valores diferentes v_1, …, v_Q que toma el cuasi-identificador, y q_1 … q_Q el número de repeticiones de cada valor. Entonces, llamamos:

- k(j) al número de diferentes valores v_1, …, v_Q de los cuasi-identificadores con j repeticiones en T:

$$k(j)= | \{ v_i | q_i = j, 1 \leq i \leq Q \} |$$

- Vector de anonimato de T, *avector(T) = (k(1),k(2), ...)*

El vector de anonimato cuenta cuántos valores del cuasi-identificador aparecen repetidos una, dos, tres, etc… En principio el vector puede ser infinito, pero es fácil ver para k(N+1) en adelante, se tiene k(N+1) = k(N+2) = … = 0, por lo que podemos representarlo de forma finita. Los vectores de anonimato para las tablas anteriores son:

	1	2	3	4	5	6
(a)	3	0	0	0	1	0
(b)	2	0	0	0	0	1
(c)	1	1	0	0	1	0
(d)	1	0	1	1	0	0

Tabla 2. Vectores de anonimatos para las tablas de la figura 3.

Se escribe en la fila superior 1,2,... en lugar de k(1), k(2), ... Por ejemplo, para la tabla (a), el vector es (3,0,0,0,1) que indica que la tabla contiene 3 valores del cuasi-identificador con una repetición, y un valor con 5 repeticiones (lo que corresponde al valor 1 en la posición 5). Veamos cómo estos vectores se pueden utilizar para comparar el nivel de anonimato de dos tablas.

Definición 2 K-anonimato generalizado
Sea T_1, T_2 dos tablas, cada una con su cuasi-identificadores conocido. Entonces, decimos que:

1. T_1 tiene el mismo K-anonimato generalizado que T_2 y lo indicamos como $anon_k(T_1)$ = $anon_k(T_2)$ cuando avector(T_1) = avector(T_2).
2. T_1 tiene mejor k-anonimato generalizado que T_2 y lo indicamos como $anon_k(T_1)$ > $anon_k(T_2)$ cuando avector(T_1) $<_{LEX}$ avector(T_2) con $<_{LEX}$ como orden lexicográfico.

Recordamos que en el orden lexicográfico se cumple $v_1 <_{LEX} v_2$ si existe una posición j tal que v_1[j] < v_2[j] y v_1[i] = v_2[i] para todo $i < j$, es decir si los dos vectores son iguales en

los primeros j -1 componentes, y el j-ésimo elemento de v_1 es menor que la misma posición en v_2.

Por ejemplo, en el ejemplo de la tabla anterior tenemos:

$$\underset{avector(d)}{(1,0,0,1)} <_{LEX} \underset{avector(c)}{(1,1,0,0,1)} <_{LEX} \underset{avector(b)}{(2,0,0,0,0,1)} <_{LEX} \underset{avector(a)}{(3,0,0,0,1)}$$

de estos vectores de anonimato podemos ver que :

$$anom_k(d) > anom_k(c) > anom_k(b) > anom_k(a)$$

2.3 Propiedades de los vectores de anonimato

Estas son algunas de las propiedades que verifican los vectores de anonimato. Aprovechamos también para introducir algo de notación que usaremos en el resto del trabajo.

P1. Dado un vector de anonimato v, el k-anonimato para la población representada en el vector, corresponde a la posición del primer valor de v distinto de 0 empezando por la izquierda.

P2.Podemos normalizar los vectores eliminando los ceros por la derecha, es decir se puede considerar (5, 2, 2, 0, 0) y (5, 2, 2) el mismo vector de anonimato. En general llamamos longitud |v| de un vector de anonimato v a la posición más a la derecha que tenga un valor distinto de 0. En el ejemplo v=(5,2,2,0,0) tenemos |v| =3.

P3. Dada una población de tamaño N, con un vector de anonimato v, se cumple que

$$\sum_{(i=1...|v|)} v[i] * i = N$$

Por ejemplo el vector v = (5,2,2,0,0) corresponde a una población de 5*1+2*2+2*3 = 15 (en otras palabras, cinco valores del cuasi-identificador se repiten una vez, dos valores se repiten dos veces, y otros dos valores del cuasi-identificador aparecen tres veces).

P4. Un vector menor lexicográficamente significa más anonimato como hemos visto en la definición de K-anonimato generalizado.

P5. Sean v_1, v_2 dos vectores de anonimato para dos tablas T_1, T_2 de tamaño N, tales que T_1 verifica k_1-anonimato y T_2 verifica k_2-anonimato. Entonces, si $v_1 < v_2$, se cumple $k_1 \geq k_2$.

La propiedad P1 indica que los vectores de anonimato refinan el concepto de k-anonimato. En particular, si dos tablas T_1 y T_2 verifican k_1-anonimato y k_2-anonimato respectivamente, con $k_1 > k_2$, entonces $anon_k(T_1) > anon_k(T_2)$. Por último podemos observar que los vectores de anonimato contienen información implícita sobre el nivel k-anonimato (P1), así como sobre el número de filas en una tabla (P3).
Si tenemos una tabla T y su vector de anonimato ($a_1, a_2, a_3, \ldots, a_n$) se verifica que:

$$a_1 x\ 1 + a_2 x\ 2 + a_3 x\ 3 + \ldots + a_n x\ n = N$$

Por ejemplo, si el vector de la tabla es (1,0,1,1,0,0). El nivel de k-anonimato es 1 (primera posición que no es cero) y la tabla contiene 1x1+0x2+1x3+1x4+0x5+0x6 = 8 filas.

2.4 Influencia de las Citas en el anonimato

Las columnas correspondientes a la cita de un proceso de screening pueden ser consideradas parte de un cuasi-identificador. Aunque sus valores para un individuo en particular no sean tan fáciles de descubrir como la edad, género o el código postal, son datos que un atacante puede descubrir con cierta facilidad. Por ello, la información de la cita concreta puede suponer un riesgo en la seguridad si la combinamos con el resto de cuasi-identificadores y el resultado final del test. Puede ocurrir que:

1. La información de la cita sea publicada como parte de los resultados, o al menos que esté disponible para algunos investigadores. Esto ocurre con frecuencia ya que así es posible detectar desviaciones según centros, turnos, etc..

2. Incluso si la información de la cita no es publicada en los resultados finales, esta información la maneja muchas personas durante el desarrollo de la prueba de screening. La gente que colabora con el screening puede encontrarse por casualidad a algún conocido en el centro médico y descubrir la cita que tenía. Si ellos tiene acceso a la información de la tabla siguiente y el nivel de anonimato es bajo entonces pueden deducir el resultado del test para esa persona.

	Zip	Gender	Age	Center	Hour	Test
1	88888	male	20-25	A	9	✔
2	11111	male	25-30	A	9	✔
3	11111	female	25-30	A	9	✔
4	88888	female	25-30	A	9	✘
5	88888	male	20-25	A	13	✔
6	11111	female	25-30	A	13	✔
7	11111	female	25-30	B	9	✔
8	11111	male	25-30	B	9	✔
9	11111	female	25-30	B	9	✔
10	88888	female	25-30	B	9	✔
11	11111	male	25-30	B	13	✘
12	11111	female	25-30	B	13	✔
13	88888	male	20-25	C	13	✔
14	88888	male	20-25	C	13	✘
15	11111	male	25-30	B	13	✔

Tabla 3. Citas para un programa de screening

Esta tabla muestra los cuasi-identificadores después de añadir la información de las citas. El vector de anonimato en esta tabla pasa a ser (11,2), lo que significa que hay 11 cuasi-identificadores que no se repiten (nivel 1, con fondo en blanco) y 2 de nivel 2 (fondo en verde y turquesa). Podemos ver que el vector de anonimato antes de incluir las citas (tabla 1) era (0,1,0,2,1) lo que corresponde a nivel 2-anonimato. Es decir hemos pasado de un anonimato base de (0,1,0,2,1) a (11,2), lo que supone una gran pérdida de anonimato en los datos finales.

Sin embargo, en este ejemplo se puede encontrar de manera sencilla una asignación de citas con K=2, aunque de todas las posibles asignaciones aleatorias hemos comprobado que aproximadamente el 99.94% corresponden a un nivel K=1.

En este ejemplo, podríamos pensar en encontrar la asignación óptima desde el punto de vista del k-anonimato generalizado en pocos segundos, simplemente probando todas las combinaciones posibles. Sin embargo, en un ejemplo real, en el que participan cientos o miles de personas la explosión combinatoria en el número de asignaciones de citas posibles hacen que el problema no se pueda resolver simplemente probando.

2.5 Distancia entre vectores de anonimato

Ya sabemos cómo comparar dos vectores de anonimato, cómo saber si un vector representa más o menos anonimato que otro. Pero nos gustaría determinar “cuánto” mejor es un vector con respecto a otro, es decir tener una noción de *distancia* entre vectores. Para ello la primera pregunta a considerar es:

¿cuántos vectores de anonimato existen para una población de n individuos?

Para responder a esta pregunta debemos ver los vectores de anonimato bajo un punto de vista nuevo, el de las particiones.

En teoría de números, una partición [14][15] de un número positivo n es una forma de escribir n como suma de varios enteros positivos, sin importar el orden de los sumandos. Más formalmente, las particiones de un entero positivo n es una secuencia finita no creciente de valores $\lambda_1, \ldots, \lambda_n$ tal que $\lambda_1 + \ldots + \lambda_n = n$

Las particiones han sido tratadas con atención en matemáticas, y hay mucha información al respecto. Se trata de un valor que tiene muchas aplicaciones en el campo de la combinatoria.

Por ejemplo, dado el número 4, tenemos que hay 5 particiones:

- 4
- 3+1
- 2+2
- 2+1+1
- 1+1+1+1

El número de particiones de n es dado por la función p(n), por lo que en este caso p(4)=5. Si nos fijamos detenidamente podemos ver que coinciden con los vectores de anonimato ya que estos vectores también suman n. Veamos la analogía.

- 4 **v1 = (0,0,0,1)**
- 3+1 **v2 = (1,0,1)**
- 2+2 **v3 = (0,2)**
- 2+1+1 **v4 = (2,1)**
- 1+1+1+1 **v5 = (4)**

No olvidemos que nuestro vector de anonimato tiene su peso correspondiente en cada posición, por ejemplo:

v1 = (0,0,0,1) = 1x0+2x0+3x0+4x1 = 4
v3 = (0,2) = 1x0+2x2 = 4

No existe una fórmula explícita para calcular p(n), pero ha sido calculado para valores grandes de n, por ejemplo, p(100) = 190,569,292 o para n =10000 p(10000) es aproximadamente 3.61673 x 10^{106}. Sí existen fórmulas para aproximar p(n) para valores n muy grandes. Una expresión asintótica para p(n) es la dada por:

$$p(n) \sim \frac{1}{4n\sqrt{3}} \exp\left(\pi\sqrt{\frac{2n}{3}}\right) \text{ as } n \to \infty.$$

Esta fórmula asintótica fue obtenido por primera vez por G. H. Hardy y S. Ramanujan [17] en 1918. Por ejemplo, para p(1000) la fórmula da un valor 2.4402 x 10^{31} que es 1.415% mayor que el valor exacto.

Aunque no se disponga de una fórmula explícita, sí que es posible determinar una fórmula recursiva que permite calcular el número de particiones, es decir el número de vectores. En formato del lenguaje Java, tenemos que p(n) será el valor devuelto por el siguiente método para una llamada p(n,n)

```
Public long p(int n, int m)
{     long resultado;
      if(n<0)
            resultado=0L;
      elseif (n==0 || m==1)
            resultado=1L;
      else
            resultado=p(n-m,m)+p(n,m-1);
      return resultado;
}
```

La idea es que p(n,m) calcula el número de particiones de n en sumandos que sean menores o iguales que m (por eso el valor que nos interesa para hallar las particiones de n es p(n,n)).
Los casos base se justifican de la siguiente forma:

- Si n < 0 se conviene que no hay ninguna forma (r =0)
- Si n = 0 se conviene que sólo hay una forma, el propio 0 (por definición, es el único caso en que se permite usar 0)
- Si m = 1 también hay sola una partición: 1+1+1+1+1+..... n veces

Para el caso recursivo dividimos p(n,m) en dos partes, las particiones que realmente utilizan m y las que no.

- Las que no lo utilizan son simplemente p(n,m-1).
- Las que sí lo utilizan tienen particiones de la forma n = m + X pero ¿cuántas?. Pasando m al otro lado tenemos de (n-m) = X es decir, p(n-m ,m).

Llamamos *posición* de un vector de anonimato v para una población de tamaño N a su posición dentro de los vectores válidos para el tamaño N comenzando desde 0, y lo representamos como pos(v). En particular se tendrá que pos((N)) = 0, y que si v es el vector formado por N-1 ceros seguidos de un uno (v = (0,...,0,1) se tiene pos(v) = p(N)-1.

Ahora definir la resta entre dos vectores v_1 y v_2 para el mismo tamaño N de la siguiente forma:

$$v_1 - v_2 = pos(v_1) - pos(v_2)$$

y la distancia dist(v_1, v_2) se define ahora de forma natural como:

$$dist(v_1,v_2) = | v_1 - v_2 |$$

Es importante comentar que no existe una forma eficiente de obtener esta distancia: debe hacerse enumerando los posibles vectores de anonimato y “contando” el número de vectores entre ambos.

Dado que en este trabajo nos interesa comparar el anonimato entre distintas formas de generar citas, mediremos la mejora/pérdida de anonimato con respecto al vector de anonimato inicial (antes de incluir la cita), al que llamaremos *vBase*. Por tanto, si queremos comparar dos vectores de anonimato v_1 y v_2 podemos calcular

$$mejora(v_1, v_2, vBase) = (pos(v_1) - pos(v_2)) / pos(vBase)$$

que representa el tanto por 1 de mejora que supone v_1 con respecto a v_2. Obsérvese que la cantidad no está definida si pos(vBase) = 0; lo que significa que no tiene sentido comparar pérdidas de anonimato si partimos del peor anonimato posible. El valor puede ser negativo, representando que v_1 supone una pérdida de anonimato con respecto a v_2.

Con el siguiente caso concreto podemos verlo con más facilidad:

- N=30 número de personas.
- vBase={2,3,2,4} vector de anonimato sin repartir citas.
- vRandom= {24,3} vector de anonimato después de repartir citas aleatoriamente.
- vAnom={9,4,3,1} vector de anonimato con programación con restricciones que veremos en el capítulo siguiente.

El mejor vector de anonimato de esos tres es vBase ya que después de repartir las citas el vector de anonimato solo puedo quedar igual o peor que con el que hemos comenzado.
Además, en general vRandom será el peor de los tres, ya que no sigue ninguna estrategia inteligente y hace la repartición aleatoriamente.

En el mejor de los casos debería ser tan buena como la solución heurística o restricciones. vAnom, en general estará en medio de vBase y vRandom. A partir de esta noción podemos hablar de proporciones, es decir podemos “cuantificar” cuánto mejor es un vector mejor que otro, es decir, ganancia..

Podemos verlo gráficamente en el siguiente diagrama:

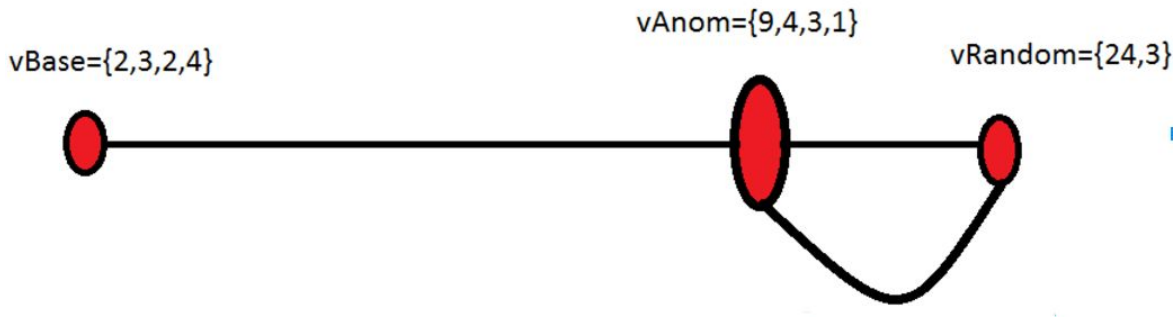


Figura 4. Distancia entre vectores de anonimato del ejemplo 1

Primero determinamos las posiciones absolutas de vBase, vRandom y vAnom. Para este ejemplo:

pos(vBase) = 3157
pos(vRandom) = 8
pos(vAnom) = 642

Nuestro resultado será:

mejora(vAnom,vRandom,vBase) = (642 - 8) / 3157 = 0.20

Es decir, en este caso el valor vAnom es un 20% mejor que el valor vRandom. vRandom es un vector de anonimato aleatorio por lo que simplemente nos sirve como comparativa y ver cuánta mejora se consigue con los otros dos algoritmos.
Si ahora consideramos vAnom={2,6,4,1}, que es mejor que {9,4,3,1} conseguimos un valor mucho más cercano al mejor posible.

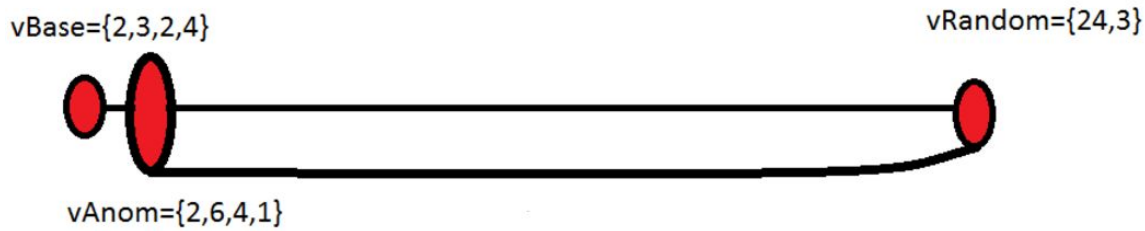


Figura 5. Distancia entre vectores de anonimato del ejemplo 2

Utilizaremos las proporciones siguientes en este caso:

pos(vBase) = 3157
pos(vRandom) = 8
pos(vAnom) = 3045

Nuestro resultado será:

mejora(vAnom, vRandom, vBase) = (3045-8) / 3157 = 0.96

Lo que supone una mejora del 96%.
En el siguiente capítulo, proponemos usar la programación con restricciones para generar la asignación óptima desde el punto de vista del anonimato.

3. La asignación de citas como un problema de optimización en programación con restricciones

Nuestro objetivo es conseguir una asignación de citas que minimice el decremento del anonimato. Para ello comenzamos definiendo la solución óptima mediante el paradigma de la programación con restricciones. El problema de la asignación de citas se corresponderá con un modelo en este paradigma, en el que se minimizaran unas variables de decisión que corresponderá con el vector de anonimato.

Para explicar nuestro modelo, empezamos con un ejemplo de una tabla con poca población (tabla 1). De esta tabla es fácil conseguir los valores de los cuasi-identificadores con sus frecuencias que podemos ver en la tabla 4, representados por la columna f.

	Zip	Gender	Age	f
1	88888	male	20-25	4
2	11111	male	25-30	4
3	11111	female	25-30	5
4	88888	female	25-30	2

Tabla 4. Frecuencias de los cuasi-identificadores de la tabla 1.

Asumimos que existe una tabla de recursos que contiene el centro médico, las horas de apertura y la capacidad (número de personas que pueden ser admitidas) en cada slot tabla 5. La tabla 5 contiene los recursos para el ejemplo actual.

	Center	Hour	Capacity
1	A	9	4
2	A	13	2
3	B	9	4
4	B	13	3
5	C	13	2

Tabla 5. Recursos suficientes para la tabla 4.

De ahora en adelante representaremos cada cuasi-identificador y cada recurso por su posición, las tablas 4 y 5, respectivamente. Estas dos tablas corresponden a los parámetros iniciales de nuestro modelo. Llamamos:

- Q: Número de valores de cuasi-identificadores, que es el número de filas en la tabla 4.
- f: Frecuencia de cada valor que toma el cuasi-identificador. Corresponde a la columna f de la tabla 4 y a los valores que hemos venido denominando $q_1 \ldots q_Q$ en capítulos anteriores. f es un vector, así que usaremos la notación f[i] para referirnos a la frecuencia del i-ésimo cuasi-identificador en la tabla, $1 \leq i \leq Q$.
- R: número de recursos, que es el número de filas de la tabla 5.
- c: Capacidad de cada recurso. Corresponde a la columna capacidad de la tabla 5. Usamos la notación c[j] para referirnos a la capacidad del j-ésimo recurso en la tabla, $1 \leq j \leq R$.

Es importante observar que en el modelo no manejamos los conceptos de hora de apertura o centro, sino que cada valor que puedan tomar estos atributos se considera simplemente un *recurso*. Con estos parámetros podemos definir un array bi-dimensional *a* de dimensión Q x R conteniendo variables enteras de decisión, que representan la solución de las citas de nuestro problema. Las variables de decisión son variables cuyos valores serán computados por el resolutor de restricciones, que satisfacen los requerimientos del modelo.
En este array a[i,j] representa el número de personas con el i-ésimo valor del cuasi-identificador i ($1 \leq i \leq Q$) que son asignados al recurso j ($1 \leq j \leq R$).

Para obtener la cita con el mejor k-anonimato generalizado necesitamos encontrar la cita con el vector de anonimato más pequeño con respecto al cuasi-identificador *q* U *r*, con *q* el cuasi-identificador inicial, y *r* los atributos que identifican a la cita (recurso).

El algoritmo 1 presenta una técnica para conseguir el vector de anonimato óptimo iterativamente. Empieza calculando en el parámetro *p* la población total, obtenido como la

suma de las frecuencias de todos los identificadores. La variable l indica que la siguiente iteración del bucle calculará el primer componente del vector de anonimato v.

La condición en el bucle while comprueba si el valor actual en v cubre a la población total. Si es así entonces podemos asegurar que el resto de componentes son cero y el algoritmo para. En otro caso, nuevos componentes son añadidos. Para asegurar que el vector de anonimato óptimo es completado, el algoritmo busca por el mínimo número de cuasi-identificadores con l repeticiones, empezando con l = 1. Esto hace que el vector es mínimo con respecto al orden lexicográfico, y en consecuencia que la cita es óptima con respecto al k-anonimato generalizado.

El problema de optimización empieza definiendo dos variables, k que será el que contenga el nuevo elemento del vector, y a, que contiene las variables que constituirán la repartición de citas óptima. Podemos observar que el array a es nuevo en cada iteración, es decir, si había un array a anterior se elimina y se define otro array a nuevo.

El problema de optimización se define por cuatro restricciones principales. Las restricciones C1, C2, definen citas válidas. C1 indica que cada fila i de a debe proveer citas para cada persona con el i-ésimo valor del cuasi-identificador. Esto es, que toda persona consigue una cita. C2 se asegura que las citas no superan la capacidad de los recursos. C3 y C4 hacen que el asignamiento de las citas no sea sólo válido sino también óptimo. C3 fuerza que la nueva solución mantenga los valores para v[n] ya calculados con $1 \leq n \leq l-1$. C4 indica que la k es el nuevo valor para v contando el número de repeticiones del valor l. El objetivo del problema de optimización es minimizar el nuevo valor para v. Finalmente, el nuevo valor se almacena en v. Después de la última iteración del bucle, a contiene la asignación de citas óptima.

Algoritmo 1
Entrada: Q, f, r, c, definidos anteriormente.
Salida: un array a con el k-anonimato generalizado óptimo, y v, su vector de anonimato
Definimos:

$p \Leftarrow \sum_{i=1}^{Q} f[i]$ % población total
$l \Leftarrow 1$ % siguiente nivel del vector de anonimato para calcular
$v \Leftarrow [\,]$ % vector de anonimato, inicialmente con 0 componentes.

While $\sum_{n=1}^{l-1} v[n] * n < p$ do % calcular l-ésimo componente del vector

Definimos y resolvemos las siguientes restricciones del problema de optimización.

Variables:

- *a*: array bidimensional que contiene Q x R variables de decisión con el dominio en los enteros no negativos.
- *k:* variable de decisión de tipo entero que contiene el siguiente componente del vector de anonimato.

Restricciones:

(C1) For $i = 1...Q : \sum_{j=1}^{R} a[i,j] = f[i]$

(C2) For $j = 1...R : \sum_{i=1}^{Q} a[i,j] \leq c[j]$

(C3) For $n = 1...l-1$:
$| \{ a[i,j] \mid i = 1...Q, j = 1...R, a[i,j] = n \} |$ = v[n]

(C4) $k = | \{ a[i,j] \mid i = 1...Q, j = 1...R, a[i,j] = l \} |$

Objetivo: minimizar k;

Incrementamos el tamaño de v,
añadimos un nuevo componente a la derecha (posición 1) $v[l] \leftarrow k$
$l \leftarrow l + 1$
end While
return *a* como la mejor asignación de citas, con v como vector de anonimato.

	Zip	Gender	Age	Center	Hour	Test
1	88888	male	20-25	B	9	✔
2	11111	male	25-30	A	9	✔
3	11111	female	25-30	B	13	✔
4	88888	female	25-30	A	13	✘
5	88888	male	20-25	B	9	✔
6	11111	female	25-30	B	13	✔
7	11111	female	25-30	B	13	✔
8	11111	male	25-30	A	9	✔
9	11111	female	25-30	C	13	✔
10	88888	female	25-30	A	13	✔
11	11111	male	25-30	A	9	✘
12	11111	female	25-30	C	13	✔
13	88888	male	20-25	B	9	✔
14	88888	male	20-25	B	9	✘
15	11111	male	25-30	A	9	✔

Tabla 6. Asignación óptima de citas en la prueba de screening

El resultado del algoritmo para este ejemplo es:

$$a = \begin{pmatrix} 0 & 0 & 4 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 2 & 0 & 0 & 0 \end{pmatrix} \qquad v = (0, 2, 1, 2)$$

La primera fila de a indica que los 4 individuos con cuasi-identificador 1 están asignados al tercer recurso, lo que corresponde al centro médico B a las 9 horas. La tabla 8 muestra esta asignación incluyendo el cuasi-identificador y el dato del recurso.
El nuevo vector de anonimato (0,2,1,2) es mucho mejor que el vector de anonimato (11,2) de la asignación aleatoria de la tabla 3, y mantiene el nivel 2 de k-anonimato de la tabla después de la asignación cuyo vector de anonimato era (0,1,0,2,1).

Para acabar debemos mencionar que aunque el algoritmo 1 es iterativo, calculando componente a componente del vector de anonimato, en algunos sistemas de restricciones es posible prescindir del bucle while indicando que las soluciones deben cumplir un cierto orden lexicográfico. Esto se consigue con opciones de búsqueda. Cuando es posible indicaremos que en cada variable del vector de anonimato se debe comenzar con el menor valor posible, y que se debe seguir un cierto orden indicado.En este caso transformamos el problema de optimización en uno de satisfacción, sabiendo que la primera solución será justo la que deseamos encontrar (mínima desde el punto de vista lexicográfico).

4. Comparativa entre las diversas formas de generar las citas

4.1 Generación de la población

Para poder realizar las pruebas lo primero que necesitamos en tener una representación de un conjunto de individuos a los cuales podemos aplicar los algoritmos. En esta parte se ha desarrollado un método que genera una población aleatoria a partir de un parámetro n que es la población total para la prueba. Al terminar la generación de la población tenemos los parámetros para las siguientes fases:

- Q: Número de valores de cuasi-identificadores (valor entero).
- f: Frecuencia de cada cuasi-identificador (vector).

De esta forma se consigue tener una gran cantidad de casos distintos para poder comparar los distintos algoritmos y comprobar cual de ellos es mejor en cada caso. Dependiendo del número de cuasi-identificadores se puede ver la diferencia de rendimiento y la calidad de la solución.

4.2 Generación de los recursos

Dado una población es necesario generar los recursos o citas para poder satisfacer a dicha población. Una condición necesaria es que al menos haya suficientes recursos para toda la población. Los recursos generados también afectarán al rendimiento y a la solución. Después de haber generado los recursos tendremos los parámetros:

- R: número de recursos (valor entero).
- c: Capacidad de cada recurso (vector).

Se ha introducido la posibilidad de poder ponerles rangos a los valores aleatorios que se generan para intentar que se parezca más a un caso real en el que las distribuciones no tienen tanta diferencia unas de otras.

4.3 Citas aleatorias

Una forma de repartir las citas entre la población es seleccionando uno de los recursos de forma aleatoria y asignándoselo a un individuo. De esta manera todos los individuos tienen una cita asignada ya que el número de recursos es al menos suficiente para cubrir a la población. Una vez que todos los individuos tengan asignado su cita podemos calcular su vector de anonimato asociado a esa distribución. Los valores de este vector depende de los números generados aleatoriamente por lo que puede ser valores buenos o malos. No se aplican ningún tipo de mejora a esta distribución.

4.4 Citas con algoritmo con restricciones

En esta opción se utiliza el algoritmo explicado en el apartado 3 que fue implementado en java usando la librería Choco para las restricciones. Para poder aplicarlo necesitamos todos los parámetros anteriores Q,f,R,c. Como se puede ver el resultado de este algoritmo da lugar a una repartición de citas óptima, dando el mejor vector de anonimato posible que también nos sirve para comparar cómo de buenos son otros algoritmos respecto a la solución ideal. En las pruebas se puede comprobar que este algoritmo tiene el inconveniente del gran coste que supone para poblaciones no muy elevadas (del orden de 40 o más individuos) ya que a partir de este umbral el algoritmo tarda mucho en generar la solución final. Para poblaciones de cientos o miles de pacientes puede ser inviable aplicar esta solución debido al tiempo que necesitaría o los recursos necesarios.

Se ha implementado un algoritmo lexicográfico que es una variante al algoritmo con restricciones pero tiene en cuenta el orden lexicográfico que tienen los vectores de anonimato para ir generando el óptimo. La librería Choco en su documentación ofrece varios parámetros para configurar el resolutor principal y además ofrece varias estrategias y condiciones que se pueden adaptar según el problema que se está intentando resolver. Esta forma de resolver el problema la utilizamos para comparar la solución con la obtenida en la anterior, tiene mejor rendimiento aunque no puede con poblaciones grandes tampoco.

4.5 Método heurístico

Vamos a definir un algoritmo que informalmente se puede decir que se encuentra a medio camino entre la asignación aleatoria, que no tiene en cuenta el anonimato, y la asignación mediante optimización en programación restricciones, que busca el vector óptimo pero es poco eficiente en la práctica.

El algoritmo que presentamos a continuación intenta mejorar el anonimato del caso aleatorio, buscando repartir las citas en grupos de mayor tamaño. La razón que nos ha llevado a desarrollar este algoritmo es que, como vamos a ver, el algoritmo óptimo resulta para poblaciones de tamaño medio/alto sumamente ineficiente.

La idea es sencilla, vamos recorriendo los distintos valores q1...qn, que toma el cuasi-identificador q en orden descendente, es decir, de mayor a menor. Para cada cuasi-identificador que elegimos le buscamos un recurso que permita cubrir a los individuos de ese cuasi-identificador. Se tiene un valor booleano que nos indica si en alguna iteración no se ha podido encontrar una solución para poder parar.

Función:

int[] heurístico(n,Q,f,R,c)

Parámetros de entrada:

n=nº de personas

Q=nº de cuasi-identificadores.

f= vector con las frecuencias de los cuasi-identificadores

R= nº de recursos

c= vector con la cantidad de cada recurso

Devuelve:

v= vector de anonimato de la solución heurística.

Pseudocódigo en java:

```
int []v = new int[n];
int [][]a = new int[Q][R];
//Ordenamos f en orden descendente y lo guardamos en ff
int [] ff = sort(f);
boolean b = true;
para (int i=0; b && i<Q; i++)
        // Encontrar huecos
         b = findSlots(i,ff,c,a);
fin para
//dado a con la solución calculada anteriormente podemos hallar su vector de anonimato
v =getV(a, n, Q, R);
return v;
```

La función devuelve como resultado final el vector de anonimato que será lo que necesitaremos para poder compararlo posteriormente con los otros algoritmos. Si en algún momento findSlots devuelve false significa que no hay solución mediante este algoritmo.

Función:

boolean findSlots(pos, f, c, a)

Parámetros de entrada:

pos = posición del cuasi-identificador actual

f = vector con las frecuencias de los cuasi-identificadores

c= vector con la cantidad de cada recurso

a=matriz con la solución del reparto de citas. Tamaño QxR

Devuelve:

encontrado = devuelve si ha true si ha podido encontrar hueco y false si no lo ha encontrado.

Pseudocódigo en java:

=== fase 1 ===

bool encontrado = true

-Marcar todos los recursos como no visitados (se usar un array auxiliar para llevar estas marcas)

sofar = 0 (variable para saber cuántos recursos llevamos)

while sofar < q

Elegir el recurso j de mayor capacidad no visitado

sofar += c[j]

Marcar c[j] como visitado

fin while

si sofar < q entonces encontrado = false //no se encontró hueco

=== Fin fase 1 ===

/*Cuando acaba la primera fase tenemos un conjunto de recursos que cubren el valor q; puede que incluso se excedan. Para corregir esto se pasa a la fase 2*/

=== fase 2 ===

-Debemos decrementar los recursos marcados hasta que cubran exactamente a q y liberar el resto para siguientes selecciones.

=== Fin fase 2 ===

return encontrado

El algoritmo principal es sencillo y admite muchas optimizaciones que pueden acelerar el proceso, pero es una buena aproximación y en general presenta un buen rendimiento como veremos en la siguiente sección.

4.5 Medida para la comparativa de anonimatos

El objetivo final del capítulo es comparar el nivel de anonimato obtenido con la asignación de citas aleatorias y la generada mediante nuestro programa con restricciones. Para establecer esta comparación introducimos el concepto de *distancia* entre dos vectores de anonimato, que se define como la diferencia de las posiciones que ocupan ambos vectores en el orden lexicográfico, contando desde el mayor valor hacia el menor valor.

Por ejemplo, dada una población de 4 personas tenemos que hay 5 posibles vectores de anonimato:

1.- (4,0,0,0)
2.- (2,1,0,0)
3.- (1,0,1,0)
4.- (0,2,0,0)
5.- (0,0,0,1)

donde el menor anonimato es (4,0,0,0) que indica que las 4 personas tienen cuasi-identificadores distintos, mientras que el (0,0,0,1) es el mejor, al implicar que todos los usuarios tienen el mismo cuasi-identificador (un 1 en la posición 4, significa que hay un solo cuasi-identificador, que se repite 4 veces). La distancia, por ejemplo entre (0,2,0,0) y (2,1,0,0) es d((0,2,0,0), (2,1,0,0)) = 4-2 = 2.

Nuestro concepto de distancia no verifica la propiedad simétrica, y no es, por tanto una distancia en el sentido matemático sino una distancia dirigida. Se cumple que dados dos vectores de anonimato, v1 <= v2 si y solo si d(v1,v2) >= 0.

Para la mejora de anonimato se parte de 3 valores:

- vBase: vector de anonimato que se tiene antes de incluir las citas, es decir teniendo en cuenta solo los cuantificadores iniciales:
- vRandom: vector de anonimato que se tiene tras incluir las citas generadas aleatoriamente, incorporando las columnas que señalan la cita.
- vPR: vector de anonimato obtenido con programación con restricciones

Es fácil comprobar que el anonimato es monótono decreciente en cuanto al número de cuasi-identificadores: si a una población dada le incorporamos nuevos cuasi-identificadores tendremos un vector de anonimato igual o peor. Tenemos por tanto que vRandom>=vPR>=vBase (recordemos que mayor vector, menor anonimato).

Para estudiar la mejora vamos a definir la cantidad d(vRandom, vPR) / (vBase-1).
Esta es una cantidad entre 1 y -1 que nos indicará el grado de mejora logrado al comparar vRandom y vPR.

4.6 Experimentos

Para los experimentos se utilizan las funciones explicadas en el apartado 4.1 y 4.2 que nos generan distintos valores de :

- Q: Número de valores diferentes que toma el cuasi-identificador (valor entero).
- f: Vector con los valores $q_1...q_Q$ que identifican la frecuencia que toman los Q diferentes valores que puede tomar el cuasi-identificador.
- R: número de recursos (valor entero).
- c: Capacidad de cada recurso (vector).

Con esto podemos probar los diferentes algoritmos con varias poblaciones y con diferentes cuasi-identificadores y recursos.
Se ha implementado una función test que recibe los parámetros necesarios para realizar las pruebas.

test(poblacion, nº de ejecuciones, valor optimo,
valor heurístico,minQ,maxQ,minF,maxF)

Podemos ir incrementando la población hasta valores altos pero a partir de 25 o superior solo se calcula el algoritmo heurístico, ya que el algoritmo óptimo por restricciones empieza a ser ineficiente. minQ, maxQ, minF, maxF son parámetros que indican sobre que rango generar los recursos y los cuasi-identificadores.

Teníamos vBase, vAnom y vRandom. vAnom será el vector de anonimato del algoritmo con restricciones o heurístico. vBase el vector de anonimato inicial, sin aplicar ningún algoritmo y vRandom el vector de anonimato de repartir las citas aleatoriamente.
En cada iteración se calculará la ganancia del algoritmo con respecto a la versión aleatoria.

El primer caso tenemos el algoritmo con restricciones. Nos devolverá la ganancia del algoritmo con restricciones respecto al aleatorio (columna 1). El segundo caso tenemos el algoritmo heurístico. Nos devolverá la ganancia del algoritmo heurístico respecto al aleatorio (columna 2). Después de realizar todas las ejecuciones se hace la media de las ganancias obtenidas para el algoritmo con restricciones y heurístico.

Valor óptimo contienen la ganancia media del algoritmo con restricciones de todas las ejecuciones realizadas y valor heurístico la ganancia media del algoritmo heurístico.

Por último, es interesante comparar la ganancia del algoritmo con restricciones con respecto al algoritmo heurístico, aunque solamente para valores pequeños de la población (columna 3).

Valores medios tras 100 ejecuciones

Población	Ganancia Óptimo/Aleatorio	Ganancia Heurístico/Aleatorio	Ganancia Óptimo/Heurístico
5	0.3815	0.3681666666666667	0.013333333333333332
10	0.6484488727858294	0.6034437490850536	0.045005123700775876
25	0.880872186821432	0.7105770931283475	0.1702950936930843
50	X	0.7727358356488846	X

Tabla 7. Tabla con los resultados de las ganancias obtenidas experimentalmente entre los distintos algoritmos

Se puede observar que el óptimo consigue ganancias mejores que el heurístico, aunque para poblaciones pequeñas el heurístico también consigue soluciones muy buenas. Cuando comparamos el óptimo con el heurístico podemos ver que la ganancia no es muy amplia aunque siempre se consigue mejor vector de anonimato con el óptimo. Cabe destacar que a medida que aumenta la población la ganancia con respecto a la generación aleatoria se hace mayor por lo que ambos algoritmos consiguen muchos mejores resultados.

Para estas pruebas en las que utilizamos las distancias para calcular las ganancias, para poblaciones muy grandes el coste de calcular las particiones empieza a ser muy grande. Recordemos que para valores de 100 la función de partición vale 190,569,292 y para 1000 es del orden $\approx 2.4 \times 10^3$, por lo que el coste computacional crece enormemente.
No se ha podido obtener la ganancia para valores mayores, porque este valor depende del llamado índice lexicográfico, posición del vector entre todos los vectores válidos. Este valor se calcula "enumerando", lo que hace que sea muy lento.

Para evaluar la eficiencia en valores altos elaboramos la tabla 8, que muestra el valor k a partir del vector de anonimato.
Si dejamos de lado el cálculo de las particiones y nos quedamos con el vector de anonimato el algoritmo heurístico es bastante más rápido que el algoritmo óptimo y funciona para poblaciones muy grandes (10000 o más).

Para ello en la siguiente tabla se puede ver el K-anonimato obtenido de los distintos algoritmos. En este caso no calculamos las ganancias por lo que podemos incluir poblaciones de por ejemplo 10000 personas y el algoritmo heurístico proporciona soluciones mejores que la aleatoria.

Población	K aleatorio	K óptimo	K heurístico
5	1.01	1.05	1.05
10	1.0	1.07	1.06
25	1.0	1.43	1.26
100	1.01	X	2.16
1000	1.02	X	6.82
10000	1.15	X	17.31

Tabla 8. Tabla con los K obtenidos para poblaciones de distintos tamaños.

Recordemos que cuanto mayor es K mejor nivel de anonimato tenemos. El algoritmo heurístico soporta valores muy grandes de población dando muy buenos resultados en comparación con el aleatorio.
Para un caso práctico el heurístico ofrece una buena solución(no siempre óptimo) con buenas ganancias y un coste mucho menor.

5. Prototipo

Para poder utilizar el entorno de pruebas es necesario disponer previamente de una base de datos SQL con los permisos y la información necesaria para conectarnos. La base de datos que elijamos se utilizará para crear las tablas y conseguir la información que necesitemos para los distintos procesos y estadísticas que haremos.
Al ejecutar la aplicación aparecerá la pantalla inicial donde tenemos las siguientes opciones:

- Connect to Database: En esta opción iniciaremos la conexión con la base de datos que indicaremos e introduciendo usuario/contraseña.
- GetK: Nos permite obtener el K de una tabla dada y sus cuasi-identificadores.
- Generate Tables: Genera tablas con “pacientes” aleatorios o siguiendo una distribución demográfica.
- Generate Resources: Genera los recursos necesarios para distribuirlos entre los pacientes de una tabla.
- Assign Appointment : Asigna a cada paciente una cita entre los recursos generados.
- Close Connection: Cierra la comunicación con la BD.

Figura 6. Menú principal de la aplicación.

Al principio la única opción disponible es Connect to Database. Una vez inicializada la conexión todas las demás opciones quedan habilitadas y se aplicarán a esa base de datos especificada. Si se quiere seleccionar otra base de datos, se debe cerrar la conexión con Close Connection y volver a conectarse a la BD que se quiera.

Connect to Database

En esta ventana debemos introducir la información de usuario y contraseña para conectarnos a la base de datos especificada.

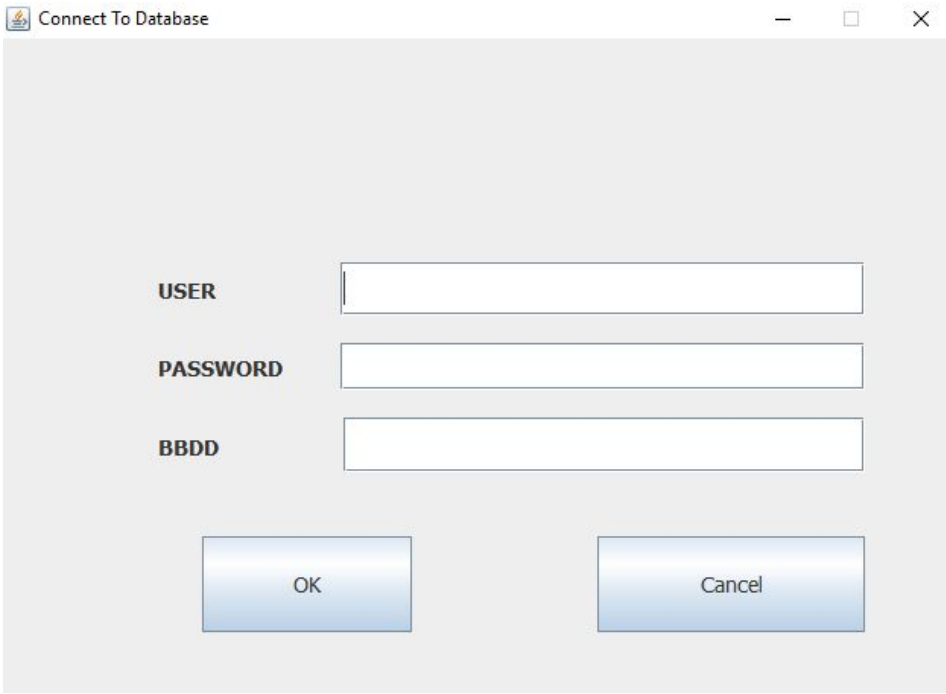


Figura 7. Datos de conexión a la base datos.

Cuando se haya comprobado la conexión a dicha base de datos aparecerá un mensaje avisando que todo ha ido correctamente. Habitualmente en MySql el usuario por defecto es *root* y no tiene contraseña. La base de datos que se quiera utilizar debe ser creada con anterioridad.

Figura 8. Éxito en la conexión.

A partir de este momento el resto de opciones quedan habilitadas y se aplicarán para la base de datos introducida.

Figura 9. Menú con opciones habilitadas.

GetK

Con esta opción podremos hallar el k-anonimato de una tabla o N tablas. Para ello debemos introducir la información y los cuasi-identificadores que utilizaremos para el cálculo.

- Table: introducimos el nombre de la tabla o tablas.
- Number of Tables: número de tablas.
- Attributes: son los cuasi-identificadores que queremos utilizar para calcular el K. Se introducen separados por comas. Ej(AGE,CP,SEX)

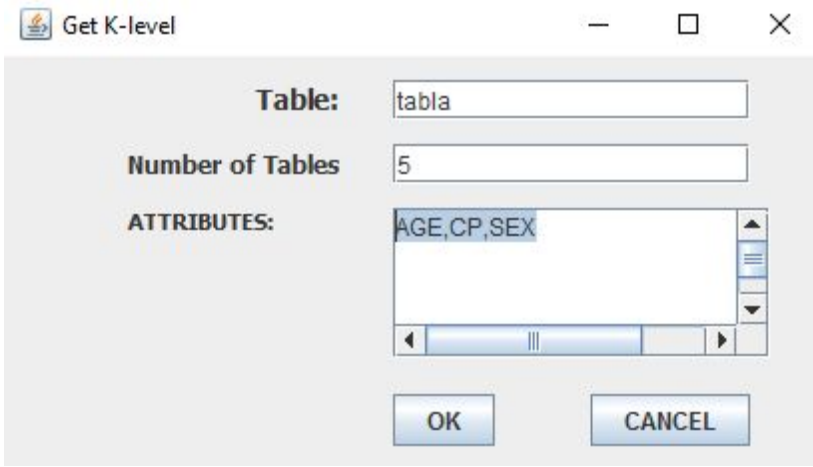


Figura 10. Datos para el cálculo del K-anonimato de varias tablas.

Necesitamos que las tablas estén creadas previamente o podemos crearlas nosotros mismos en la opción Generate Tables. Estas tablas deben contener la información de los cuasi-identificadores de cada individuo. Si nuestros atributos son age, cp y sex entonces las tablas tendrán que contener esas tres columnas con la información correspondiente a cada persona.
Es importante que los nombres de los atributos coincidan con el nombre de las columnas de las tablas, si no dará error por no haber podido encontrar las columnas para calcular el K-anonimato de las tablas.
Tenemos dos opciones para calcular el K-anonimato:

1. Si queremos hallar el K-anonimato de una tabla en número de tablas debemos colocar 1 y en Table el nombre completo de la tabla.
2. Si queremos hallar el K de N tablas, al nombre se le concatena el índice para formar el nombre de las tablas.

En el ejemplo de la imagen el nº de tablas es 5 y el nombre de la tabla es tabla, entonces hallará el K-anonimato de las tablas con nombre tabla0, tabla1, tabla2, tabla3, tabla4. Es decir, concatena un índice numérico al nombre de la tabla introducido.
El resultado de hacer la consulta nos muestra el K-anonimato de esa tabla y si son varias tablas, la media, el máximo y mínimo (se queda con el último min y max encontrados).

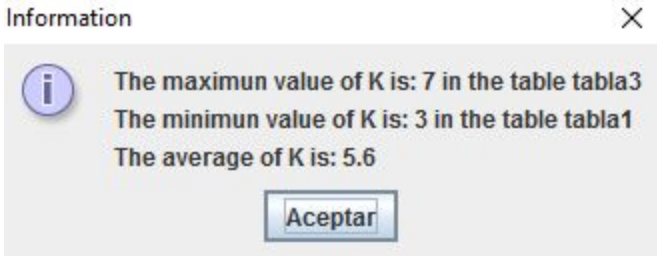


Figura 11. Cálculo del K-anonimato de varias tablas.

Generate Tables

En este apartado podemos generar una o varias tablas con datos de personas tanto aleatoriamente como siguiendo una distribución demográfica.

- Table Name: nombre de las tablas a generar.
- Number of Tables: número de tablas que generar.
- Age Randomization: distribución aleatoria o no.
- Sex: sexo de las personas, puede ser man, woman o vacío para ambos sexos.
- Min Age: mínima edad de las personas generadas.
- Max Age: máxima edad de las personas generadas.
- Population: número de personas por tabla.
- Number of CP: número de códigos postales aleatorios disponibles.

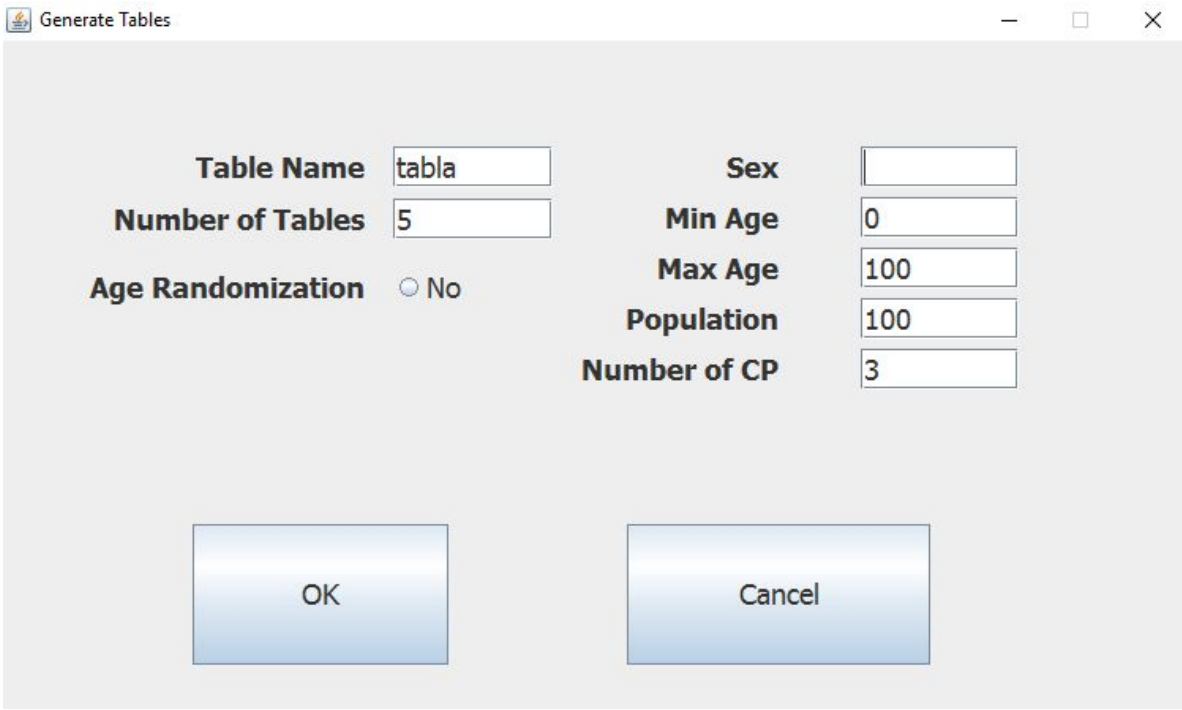


Figura 12. Introducción de los datos para generar tablas con poblaciones.

Se pueden crear tablas de dos formas:

1. Si queremos crear solo una tabla se utilizará el nombre explícito de la tabla.
2. Si queremos crear N tablas se utilizará el nombre y será concatenado con un índice, por ejemplo, tabla0, tabla1, tabla2, tabla3, tabla4. Está hecho para facilitar el cálculo de K y en la asignación de citas.

Si no marcamos AgeRandomization entonces la generación será aleatoria en edad. En cambio, si marcamos la opción entonces la distribución generada será la de la población española con la información obtenida del ministerio sobre la situación demográfica actual. Las tablas generadas no aleatorias tienen un K-anonimato más realista a una población real.
MinAge, MaxAge, NumberOfCp, Population y NumberOfTables deben tener valores positivos para una correcta generación.

Hay que tener en cuenta que si se introducen valores muy altos puede tardar en calcular todas las tablas y crearlas en la base de datos. Depende del ordenador en el que se ejecute y el número de tablas y población generadas.
Al terminar de crear todas las tablas saldrá un mensaje indicando que se ha completado.
Las tablas generadas en este proceso tienen unas columnas predeterminadas que nos serán útiles en nuestro problema.

+ Opciones

ID	AGE	CP	SEX	REC_RAND	REC_INTELLIGENT
0	14	28004	man	-1	-1
1	33	28003	woman	-1	-1
2	66	28006	man	-1	-1
3	95	28005	woman	-1	-1
4	15	28005	woman	-1	-1
5	52	28002	man	-1	-1
6	88	28008	man	-1	-1
7	48	28008	man	-1	-1
8	33	28003	woman	-1	-1
9	87	28007	woman	-1	-1

Figura 13. Tabla generada en la opción generate table con los parámetros introducidos..

Los campos REC_RAND y REC_INTELLIGENT son dos columnas que se rellenaran con la asignación de citas tanto aleatorio como de manera inteligente. Por defecto valen -1, que significa que aún no tienen citas asignadas. Una vez se hayan generado recursos para esta población y realizado la asignación de citas estas columnas se actualizarán con el recurso asignado para cada individuo.

Para la generación demográfica se ha utilizado información obtenida del ministerio que se pueden ver en las siguientes figuras.

Crecimiento poblacional por grupos de edad durante 2014

	Población a 1 de enero		Crecimiento anual(*)	
Grupos de edad	2015(*)	2014	Absoluto	Relativo (%)
TOTAL	46.439.864	46.512.199	-72.335	-0,16
0 a 4 años	2.256.137	2.320.612	-64.474	-2,78
5 a 9 años	2.484.228	2.478.498	5.730	0,23
10 a 14 años	2.307.748	2.267.843	39.905	1,76
15 a 19 años	2.152.888	2.140.570	12.317	0,58
20 a 24 años	2.318.277	2.374.617	-56.339	-2,37
25 a 29 años	2.637.741	2.749.308	-111.566	-4,06
30 a 34 años	3.267.325	3.456.208	-188.883	-5,47
35 a 39 años	3.948.602	4.032.770	-84.168	-2,09
40 a 44 años	3.888.532	3.858.819	29.713	0,77
45 a 49 años	3.690.385	3.689.866	520	0,01
50 a 54 años	3.409.097	3.333.372	75.725	2,27
55 a 59 años	2.978.760	2.877.803	100.958	3,51
60 a 64 años	2.508.107	2.491.892	16.215	0,65
65 a 69 años	2.357.956	2.327.434	30.522	1,31
70 a 74 años	1.949.490	1.809.958	139.531	7,71
75 a 79 años	1.553.295	1.652.238	-98.944	-5,99
80 a 84 años	1.425.513	1.403.260	22.253	1,59
85 a 89 años	854.988	825.182	29.807	3,61
90 a 94 años	356.755	333.079	23.676	7,11
95 y más años	94.038	88.871	5.166	5,81

(*) Datos provisionales

Figura 14.Información demográfica de la población española utilizada al generar las poblaciones no aleatorias [19].

Población residente en España	Población a 01/01/2014	Población a 01/01/2015	Variación %
Población total	46.512.199	46.439.864	-0,16
Hombres	22.877.461	22.820.775	-0,25
Mujeres	23.634.738	23.619.089	-0,07
Españoles	41.835.140	41.992.012	0,37
Extranjeros	4.677.059	4.447.852	-4,90

Figura 15. Información demográfica de la población española por sexos [20].

Generate Resources

En este apartado nos encargaremos de generar el número necesario de recursos o citas para las personas de cada tabla. Como queremos incorporar algo de realismo la generación de citas no será única sino que se crearán con valores intermedios entre un rango descrito obtenidos aleatoriamente.

- Number of Resources: número de tipos de cita diferentes.
- Number of Persons: número de personas para asignar cita.
- Min Capacity: el mínimo número de citas generadas para cada tipo.
- Max Capacity: .el máximo número de citas generadas para cada tipo.
- Table Name: nombre de las tablas donde están las personas.
- Number of Tables: número de tablas.

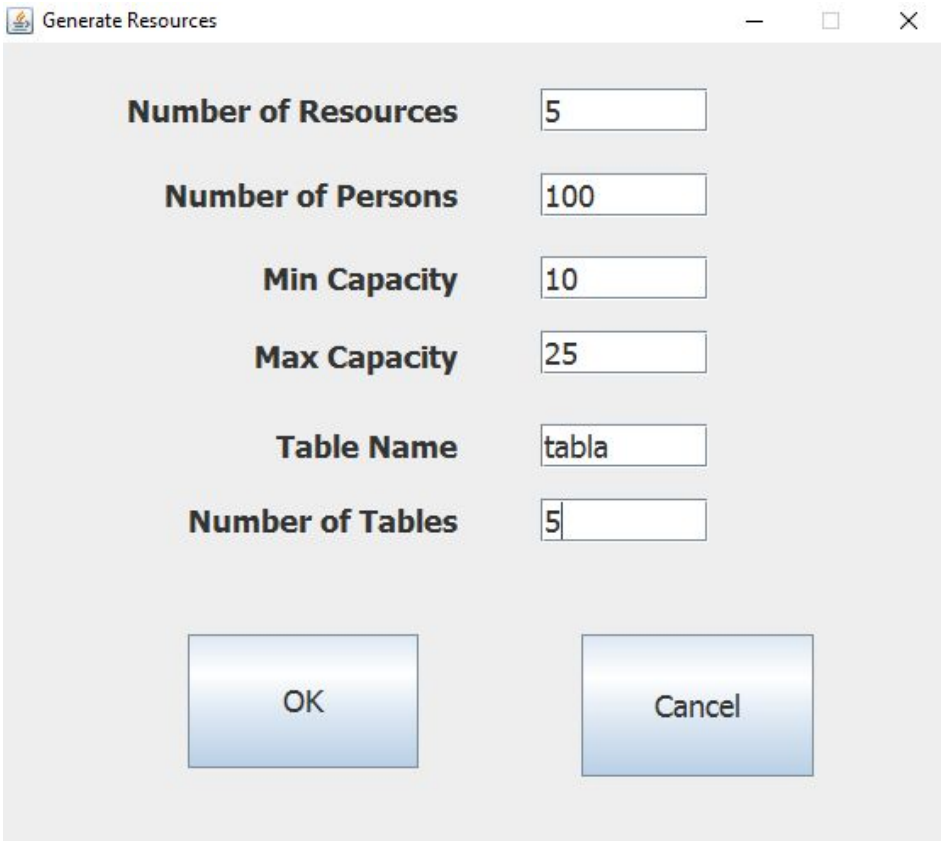


Figura 16. Introducción de los datos para generar los recursos necesarios para las poblaciones.

Hay dos formas de generar recursos:

1. Si queremos crear recursos solo para una tabla se utilizará el nombre explícito escrito.
2. Si queremos crear recursos para N tablas se utilizará el nombre y será concatenado con un índice seguido de resource, por ejemplo, tabla0_resource, tabla1_resource, tabla2_resource, etc.

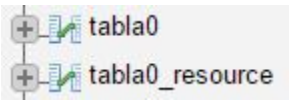
El nombre elegido debe coincidir con la tabla que contiene la población.
La primera comprobación que hay que hacer es si NumberOfResources x MaxCapacity es mayor que NumberOfPersons ya que si esto no se cumple entonces no hay suficientes citas para todas las personas. Si esto ocurre, automáticamente se incrementará el número de citas hasta tener suficientes para todas las personas.

Las tablas generadas tendrán este aspecto y se utilizarán en la fase de asignación de citas. Pueden haber más citas de las necesarias pero siempre habrá suficientes.

+ Opciones

RESOURCE	CAPACITY
0	4
1	6
2	6
3	6
4	6

Figura 17. Tabla de recursos generados



Llegados a este punto deberíamos tener una pareja de tablas, una en la que están las personas y otra con los recursos generados para esa tabla.

Assign Appointment

Finalmente podemos realizar la asignación de citas a las personas que hemos generado con las opciones anteriores:

- Table Name: nombre de las tablas de personas y recursos.
- Number of Tables: número de tablas.
- Level of Search: limita el nivel de búsqueda del algoritmo con restricciones.

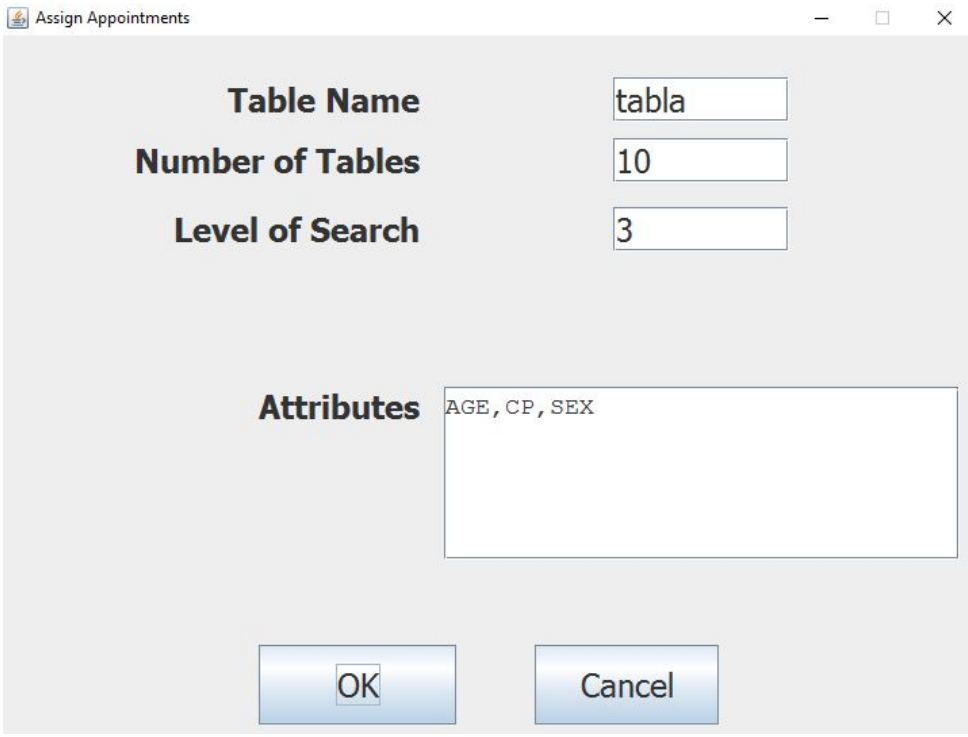


Figura 18. Datos para asignación de citas.

Tenemos dos formas de asignar citas.

1. Si queremos asignar citas solo para una tabla se utilizará el nombre explícito de la tabla.
2. Si queremos asignar citas para N tablas se utilizará el nombre concatenado con un índice seguido de resource a su tabla de personas correspondientes, por ejemplo, las citas de tabla0_resource se asignarán a tabla0.

El _resource del nombre de tabla de los recursos se añaden automáticamente y es muy recomendable que el número tablas de recursos y de personas sea el mismo.

Los nombres de las tablas de personas y recursos deben coincidir, es decir, si la de personas se llama tablaX, se buscará una de recursos tablaX_resource (que debe tener recursos suficientes para la tablaX).
El campo LevelOfSearch limita al algoritmo explicado en el apartado 3. Cuando el algoritmo empieza a encontrar ceros en el vector hace que pare antes de seguir calculando los siguientes valores para que tarde menos.

Después de generar las asignaciones se actualiza el campo REC_RAND y REC_INTELLIGENT en la tabla de personas con el valor del recurso asignado.

ID	AGE	CP	SEX	REC_RAND	REC_INTELLIGENT
0	52	28003	woman	5	5
1	51	28002	man	3	2
2	57	28000	woman	5	1
3	59	28002	man	4	2
4	52	28003	man	2	3
5	52	28003	woman	2	5
6	56	28003	man	3	3
7	57	28000	woman	1	1
8	45	28000	man	1	3
9	47	28002	man	4	2
10	47	28002	man	1	2
11	48	28003	woman	1	5
12	49	28000	woman	2	1
13	50	28001	man	3	4
14	46	28001	woman	5	2
15	50	28001	man	2	4
16	48	28003	woman	2	5
17	58	28001	man	5	4
18	52	28003	man	4	3
19	47	28002	woman	3	1

Figura 19. Tabla de personas con sus respectivas citas finales.

Close Connection

Antes de cerrar la aplicación es conveniente cerrar la conexión o si se quiere iniciar en otra base de datos elegir esta opción. Deshabilita todas las opciones hasta que se vuelva a conectar de nuevo con las credenciales.

6. Conclusiones y trabajo futuro

El equilibrio entre la confidencialidad de la información sobre la salud de las personas y la importancia de compartir información con fines científicos es un problema complejo, con una gran variedad de factores a tener en cuenta.

En este trabajo nos hemos centrado en los programas masivos conocidos como screening y hemos presentado posibles soluciones para el problema de la publicación de los datos de las citas médicas en este tipo de programas. También hemos evaluado la eficiencia de las medidas propuestas a base de experimentos y casos de ejemplo.

El algoritmo basado en programación con restricciones consigue el resultado óptimo de repartir las citas, pero en los experimentos hemos podido ver que cuanto mayor sea la población de muestra mayor es la ineficiencia del algoritmo hasta el punto de que a partir de cierto umbral empieza a ser impracticable debido al tiempo para calcular la solución. Se hicieron varias mejoras al algoritmo para intentar hacer más eficiente, tales como, distintas formas de búsquedas, asignando pesos, varias máquinas diferentes, etc … Pero este tipo de solución sigue siendo ineficiente para poblaciones grandes. Podemos decir que la principal virtud de este método es definir una solución óptima que puede servir de medida comparativa para otros métodos, aunque con la limitación de solo poder aplicar esta comparativa a valores de población pequeños.

En un caso práctico real, el número de pacientes suele ser más elevado de lo que los resolutores con restricciones son capaces de tratar. Por ello, surgió la idea de añadir a este trabajo un sencillo algoritmo heurístico, que aunque a menudo no alcance el resultado óptimo, sí que suponga una solución mejor que la simple repartición aleatoria de citas. Este algoritmo tiene una eficiencia mucho mayor que el método de las restricciones y consigue dar solución al problema para poblaciones grandes. Como en todo problema, se debe estudiar qué algoritmo utilizar según las necesidades y costes disponibles.
La idea de desarrollar un algoritmo alternativo al de restricciones surgió en la parte final del proyecto por la necesidad de dar una solución al resto de casos. Como se dijo anteriormente, el algoritmo heurístico puede ser optimizado de varias formas y este sería el principal campo de trabajo futuro. Otra línea consistiría en la búsqueda de un algoritmo eficiente que permita calcular el índice lexicográfico, necesario para calcular la distancia en los experimentos, en lugar del algoritmo actual que se limita a “contar” la distancia enumerando todos los vectores de anonimato y que al ser sumamente ineficiente limita el rango de los experimentos tal y como vimos en la tabla 7.

En cuanto a trabajo futuro, algunas ideas para mejorar el proyecto pueden ser:

- Mejorar el algoritmo heurístico con optimizaciones para conseguir mayor ganancia en los resultados.
- Verificar la corrección de los algoritmos, tanto en el caso de programación con restricciones como en el algoritmo heurístico.
- Implementar una interfaz gráfica para el apartado de pruebas de cara a facilitar la generación de pruebas y la comparación de resultados de los distintos algoritmos y mejoras.
- Extender el proyecto para que admita otros tipos de bases de datos, como postgresql o bases de datos que se utilicen en centros donde se realicen las pruebas de screening e investigación. Además de mejorar las consultas a la base de datos o creación de índices para acelerar el proceso.
- Implementar algún tipo de coloreado en tablas para así poder visualizar con mayor facilidad los diferentes cuasi-identificadores, su K-anonimato, etc…
- Estudiar en mayor detalle las características de los vectores de anonimato, en particular en relación con la partición de enteros.
- Buscar métodos eficientes para calcular el índice lexicográfico de un vector de anonimato. Este valor se utiliza en la fase de pruebas y se obtiene generando explícitamente el número de vectores menores que el dado, lo que resulta poco eficiente y pone un límite al tamaño de las pruebas.
- Proponer nuevas medidas de anonimato. Los vectores de anonimato pueden ser fuente de nuevas medidas no basadas en el k-anonimato, por ejemplo proponiendo medidas que tengan en cuenta todos los valores (valor medio ponderado en el vector).

7.Bibliografía

[1] L. Sweeney. “K-anonymity: a model for protecting privacy”. International Journal on Uncertainty,Fuzziness and Knowledge-based Systems, 10 (5), 2002; 557-570.

[2] Ciriani V, De Capitani di Vimercati SSF, Samarati P. “K-anonymity. In: Secure Data Management in Decentralized Systems”, 2007; Springer.

[3] Kisilevich, Slava; Rokach, Lior; Elovici, Yuval; Shapira, Bracha ”Efficient Multidimensional Suppression for K-Anonymity", Knowledge and Data Engineering, IEEE Transactions on, On page(s): 334 - 347 Volume: 22, Issue: 3, March 2010

[4] Pierangela Samarati and Latanya Sweeney. “Generalizing data to provide anonymity when disclosing information (abstract)”. Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems 0-89791-996-3. Seattle, Washington, USA, 1998.

[5] http://gpd.sip.ucm.es/rafa/docencia/pr/presenta/CHOCO%20-%20Roberto%20de%20Miguel%20y%20Carlos%20Ruiz.pdf

[6] James Maxwell Glover Wilson and Gunnar Jungner - “Principles and practice of screening for disease” World Health Organization. Public Health Paper, 1968 - Geneva

[7] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, An Zhu. Anonymizing Tables.Database Theory - ICDT 2005 Volume 3363 of the series Lecture Notes in Computer Science pp 246-258. 2005

[8] Krzysztof Apt. “Principles of Constraint Programming”. Cambridge University Press; 1 edition, 2009.

[9] Sonia Estévez Martín, Maria Teresa Hortalá-González, Mario Rodríguez-Artalejo, Rafael del Vado Vírseda, Fernando Sáenz-Pérez, Antonio J. Fernández. On the cooperation of the constraint domains , R, and F in CFLP. TPLP 9(4): 415-527, 2009

[10] <http://choco-solver.org/>

[11] https://es.wikipedia.org/wiki/Programación_con_restricciones

[12] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, Muthuramakrishnan Venkitasubramaniam “l-Diversity: Privacy Beyond k-Anonymity” Department of Computer Science, Cornell University (ICDE 2006)

[13] Ninghui Li, Tiancheng Li, Suresh Venkatasubramanian “t-Closeness: Privacy Beyond k-Anonymity and -Diversity” Department of Computer Science, Purdue University

[14] “The theory of partitions” George E. Andrews Addison-Wesley, 1976

[15] https://en.wikipedia.org/wiki/Partition_(number_theory)

[16] “Introducción a la teoría de números” Tom. M. Apostol editorial reverté, S. A.

[17] G. H. Hardy and S. Ramanujan, “Asymptotic formulae in combinatory analysis”, Proc. London Math. Soc.,17(2) (1918) páginas 75–115

[18] http://gpd.sip.ucm.es/rafa/docencia/pr/presenta/CHOCO%20-%20Roberto%20de%20Miguel%20y%20Carlos%20Ruiz.pdf

[19] http://www.ine.es/prensa/np917.pdf

[20] http://www.ine.es/inebaseDYN/cp30321/cp_inicio.htm

[21] https://github.com/richardjordancabana/anom

[22] https://github.com/richardjordancabana/Anonimizacion