Universidad Zaragoza

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Trabajo Fin de Grado

Comparación de algoritmos de anonimización: Mondrian y Datafly

Comparison between anonymization algorithms: Mondrian and Datafly

Autor: Alejandro Fernández Poza

Director: Elvira Mayordomo Cámara

Informática e Ingeniería de Sistemas

Lenguajes y Sistemas Informáticos

- 2017 -

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

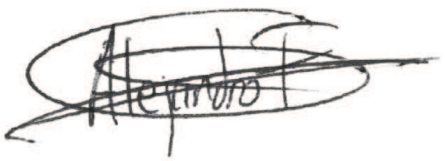
D./Dª. Alejandro Fernández Poza,

con nº de DNI 25204119Y en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Comparación de algoritmos de anonimización: Mondrian y Datafly,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 29 de agosto de 2017



Fdo: Alejandro Fernández Poza

RESUMEN

Comparación de algoritmos de anonimización: Mondrian y Datafly

El presente trabajo muestra una comparación de dos algoritmos para lograr la k-anonimización de un conjunto de datos. El primero es Datafly (1997-1998), un algoritmo heurístico cuyas principales herramientas son la generalización y la supresión de tuplas. El segundo algoritmo es Mondrian (2005), de desarrollo posterior, que basa su estrategia en la partición multidimensional de los datos, perdiendo en cierto modo el clásico enfoque tabular (filas-individuos y columnas-atributos).

Ambos algoritmos se han implementado desde cero, en un mismo lenguaje de programación (Java) y siguiendo lo más fielmente posible la idea expresada en los trabajos originales. Todo ello para poder realizar una comparación lo más justa posible. Como ambos algoritmos tienen algunas partes sin detallar, se han tomado ciertas decisiones de diseño e implementación que podrían afectar a la comparación, por ello se describen en este trabajo para que se pueda tener en cuenta a la hora de sacar las conclusiones oportunas. Entre las medidas utilizadas están: el coste temporal asintótico, el número de combinaciones totales finales, la k-anonimización media lograda y la varianza respecto a dicha media. No se ha tenido en cuenta en la comparativa el tiempo medido durante las pruebas, ya que se han ejecutado en una máquina multitarea y la carga de la misma podría afectar a la medición por ello se puede considerar una medida menos normalizada que las indicadas.

Cabe destacar el diseño e implementación de varios scripts para la herramienta MATLAB capaces de generar conjuntos de datos sintéticos con ciertas características modificables y para representar las estadísticas de los resultados obtenidos mediante gráficas.

Tras las pruebas realizadas se ha podido observar como Mondrian consigue mejores resultados en general, y con mayor regularidad de tuplas en cada una de las combinaciones finales. Sin embargo, Datafly a pesar de ser uno de los primeros algoritmos de k-anonimización consigue acercarse a los resultados de Mondrian en algunos de los conjuntos de datos, dependiendo de los valores y la distribución de estos.

1. ÍNDICE

2. INTRODUCCIÓN

Con el incipiente aumento de la necesidad de publicar y acceder a datos, es necesario proteger aquella información sensible que pueda identificar a individuos dentro de un conjunto. Por ello, se han desarrollado ciertos mecanismos que permitan publicar la máxima información posible pero manteniendo el anonimato legal, necesario y deseado. Dos de los campos que más sufren este problema son el financiero y el médico, aunque cada vez más sectores se están sumando, como la administración pública o las grandes empresas.

Como es lógico, la ley órganica de protección de datos ya tiene en cuenta dicho problema y una de las soluciones que propone es la anonimización. La anonimización es un método utilizado para evitar que los datos pertenecientes a un individuo lo identifique en un conjunto. Desde el punto de vista clásico de una base de datos tabular, el problema sería evitar que una tupla de una tabla sea identificable por uno, varios o todos sus atributos eliminando la mínima información posible.

Muchos expertos han reconocido que ningún método conocido de anonimización es invulnerable a ataques de desanonimización, que mediante inferencia estadística sumada a la adicción externa de información permite la identificación singular de individuos. Es bien conocido que el acceso a información privilegiada puede generar grandes beneficios. Es por esto, que es uno de los temas sobre los que se están llevando a cabo varias investigaciones. Uno ejemplo de esto es la investigación realizada en la universidad de Texas con el 'Netflix Prize Dataset' [3].

Por ejemplo, tomamos los datos en la tabla 2 con tres atributos y cuatro tuplas. Si se conoce el nombre y la edad de una de las personas se puede saber su deporte favorito. Tras anonimizar ambos atributos tal y como se puede ver en la tabla, ya no se puede identificar el deporte favorito de un individuo aún sabiendo su nombre y edad, aunque si se puede inferir que le gusta uno de los dos correspondientes a su identificador anonimizado.

Original		
Nombre	**Edad**	**Deporte favorito**
Alvaro	25	Baloncesto
Alejandro	22	Fútbol
Mario	12	Tenis
Marcos	18	Curling

Nombre y edad anonimizados		
Nombre	**Edad**	**Deporte favorito**
Al*	2X	Baloncesto
Al*	2X	Fútbol
Mar*	1X	Tenis
Mar*	1X	Curling

El término anonimización es muy amplio y por ello, existen diversos tipos y enfoques. Hemos elegido aquí el enfoque denominado"k-anonimización", muy frecuente en la literatura. La k-anonimización no es más que una restricción extra a la definición anterior que implica que tras la anonimización, cada identificador de cada tupla distinta, aparecerá al menos k veces. Encontrar una solución óptima a este problema es NP-difícil. En el ejemplo anterior se puede ver como se alcanza una 2-anonimización ya que cada par nombre-edad original puede corresponder (al menos) a 2 tuplas anonimizadas.

Los algoritmos elegidos son Datafly y Mondrian. Datafly es un algoritmo heurístico creado en 1997 [5][6] y posteriormente detallado para k-anonimización en 2002 [7] por Latanya Sweeney. Es unos de los llamados sistemas tempranos ya que junto a μ-Argus son los primeros sistemas completos de anonimización. Datafly se basa en la generalización del atributo con más valores distintos y la supresión individual de tuplas. En el lado opuesto, Mondrian es uno de los sistemas más recientes y data de 2005 [1]. Su heurística voraz se basa en la partición multidimensional de los datos. Este último posee dos modos de funcionamiento con el que se varía la forma en la que se realizan dichas particiones.

Ambos algoritmos se han implementado desde cero en un mismo lenguaje (Java) y se han refinado para que la comparación sea lo más justa posible. La evaluación se ha realizado de forma individual y la comparación se ha realizado utilizando las características propias de los algoritmos y los resultados obtenidos en los casos de prueba controlados. Todo el código desarrollado se puede encontrar en [4]. Ambos algoritmos se han desarrollado para que hagan uso solo de un proceso a pesar de que Mondrian mejora notablemente en tiempo haciendo uso de threads ya que cada partición es separable de las demás.

Un resumen de las características y los resultados obtenidos se puede ver en la tabla 1.

Cuadro 1: Síntesis de las principales características de ambos algoritmos.

Propiedad	**Datafly**	**Mondrian**
Fecha de creación	1997-2002	2005
Asegura k-anonimización	Sí	Sí
Asegura solución óptima	No	No
Supresión de datos	Sí	No
Forma de anonimizar	Generalización (Bottom-Up)	Partición (Top-Down)
Heurística	Generalizar el atributo con más valores distintos	Realizar una división por el atributo con el rango normalizado mayor
Coste asintótico temporal	O($\|PT\|$)	O($\|PT\| * log_2 \|PT\|$)
Resultados prácticos	Sobregeneralización	Cercanos a los óptimos en muchos casos

3. DATAFLY

Datafly es un algoritmo heurístico voraz que asegura la k-anonimización de los datos. Esta heurística consiste en generalizar el atributo con más valores distintos y una vez alcanzada una generalización suficiente, se eliminan las tuplas que siguen siendo identificables sin cumplir la k-restricción. Los resultados que se obtienen no son óptimos ya que tiende a sobregeneralizar como se podrá ver más adelante y su ratio de aproximación al resultado óptimo depende de los propios datos de entrada.

3.1. Resumen de funcionamiento

Comenzamos explicando el funcionamiento general del algoritmo datafly. El pseudocódigo asociado a la siguiente explicación puede encontrarse en la Figura 8, página 13 de [7]. En primer lugar, se hace una lista de frecuencias de las distintas combinaciones del ID existentes en los datos. En el caso de que haya k o más tuplas sin cumplir la k-anomización, se cuenta el número de valores distintos que tiene cada uno de los quasi-identificadores individualmente y se generalizan los valores del atributo con la mayor variación. Una vez hecho esto, se recalcula la lista de frecuencia de los identificadores y se vuelve a comprobar el número de tuplas que no cumplen la k-restricción. Esto se repite hasta que hay menos de k tuplas que no cumplen la k-restricción, las cuales son eliminadas.

Ejemplo: Vamos a tomar como ejemplo los datos mostrados en la Figura 7. La tabla de frecuencias inicial sería:

$$\{10, 10\} \Rightarrow 1 \text{ tupla}$$
$$\{10, 11\} \Rightarrow 1 \text{ tupla}$$
$$\{10, 21\} \Rightarrow 1 \text{ tupla}$$
$$\{19, 10\} \Rightarrow 1 \text{ tupla}$$
$$\{19, 11\} \Rightarrow 1 \text{ tupla}$$
$$\{19, 21\} \Rightarrow 1 \text{ tupla}$$

Con lo que hay 6 tuplas que no cumplen (por ejemplo, una 2-restricción). Así que se procede ha contabilizar el número de valores distintos de los quasi-identificadores:

$$A_0 \Rightarrow 2 \text{ valores}\{10,\ 19\}$$
$$A_1 \Rightarrow 3 \text{ valores}\{10,\ 11,\ 21\}$$

Por lo que se generaliza A_1, manteniendo las cifras de las decenas y superiores, y cambiando las unidades a 0. Se recalculan las frecuencias quedando:

$$\{10, 10\} \Rightarrow 2 \text{ tuplas}$$
$$\{10, 20\} \Rightarrow 1 \text{ tupla}$$
$$\{19, 10\} \Rightarrow 2 \text{ tuplas}$$
$$\{19, 20\} \Rightarrow 1 \text{ tupla}$$

Como sólo hay 2 tuplas sin cumplir la 2-anonimización, se pueden eliminar:

$$\{10, 10\} \Rightarrow 2 \text{ tuplas}$$

~~$\{10, 20\} \Rightarrow$ 1 tupla~~
$\{19, 10\} \Rightarrow 2$ tuplas
~~$\{19, 20\} \Rightarrow$ 1 tupla~~

De este modo ya se cumple la 2-anonimización por lo que se reconstruye la tabla con los nuevos valores de ID dando como resultado la tabla de la figura 1. Los valores tachados o eliminados no se guardan en los datos resultantes.

PT	A_0	A_1	A_2	A_3	A_4	A_5
PT[0]	10	10	0	0	0	0
PT[1]	10	10	0	0	0	0
~~PT[2]~~	~~10~~	~~20~~	~~0~~	~~0~~	~~0~~	~~0~~
PT[3]	19	10	0	0	0	0
PT[4]	19	10	0	0	0	0
~~PT[5]~~	~~19~~	~~20~~	~~0~~	~~0~~	~~0~~	~~0~~

Figura 1: Ejemplo final de anonimización usando Datafly.

3.2. Detalles de implementación

Este algoritmo tiene dos partes importantes que marcan su coste, el cálculo y actualización de la tabla de frecuencias y el recuento de valores distintos de cada quasi-identificador. En los siguientes razonamientos se usará la nomenclatura presente en la sección 9.

Para la tabla de frecuencia se ha utilizado un diccionario Hash en el que la clave es una composición de todos los quasi-identificadores separados por guiones. De esta forma el coste promedio de acceso a un elemento es O(1), siendo el coste asintótico de la generación de la tabla completa O($|PT|$), O($|PT|$) para recorrerse todas las tuplas y O($|ID|$) para generar la clave de la tupla con lo que el coste total es O($|PT| * |ID|$). En la gran mayoría de los casos prácticos $|ID|$ es muy pequeño y lo podemos considerar constante, por lo que O($|PT| * |ID|$)=O($|PT|$).

Para el recuento de valores de los atributos se ha usado una estructura del tipo vector de diccionarios Hash, uno por quasi-identificador. El diccionario usa como clave el valor del atributo y como elemento el número de ocurrencias de dicho valor.

De este modo, el coste asintótico del recuento es O($|PT|$) para recorrer todas las tuplas y O($|ID|$) para cada atributo. Como se disponen de los índices de los QI el coste de acceso a cada elemento del vector es O(1) y puesto que solo se necesita saber el número de valores almacenados en cada diccionario, el tiempo para encontrar el atributo con más valores distintos es O($|ID|$). Así que el coste asintótico de esta parte es O($|PT| * |ID| + |ID|$) $\approx$ O($|PT|$).

Como ejemplo ilustrativo, el primer recuento de valores del ejemplo utilizado anteriormente, daría lugar a la estructura V (vector de diccionarios hash) mostrada en la figura 2.

Para contar el número de tuplas que no cumplen la k-restricción basta con recorrerse la tabla de frecuencias acumulando el número de tuplas de las secuencias con una frecuencia menor que k. La cota superior temporal es O($|PT|$), es el caso extremo en el que cada tupla tiene una combinación de ID distinta. Esto solo se cumple en la primera iteración y el coste es menor conforme las tuplas van compartiendo combinaciones.

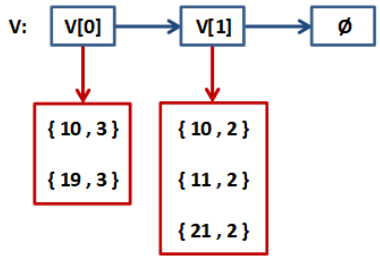


Figura 2: Ejemplo de vector de diccionarios Hash para el recuento de valores.

La generalización tiene un coste O($|PT|$) ya que hay que recorrerse cada uno de los datos del atributo designado para generalizarlo. En algunos casos puntuales es necesario realizar dos pasadas, como en el caso de la primera generalización de las cadenas o los números decimales (llamados dobles en el presente trabajo).

La eliminación de tuplas que no cumplen la k-restricción se puede hacer en O(k) = O(1) si se almacenan los indices de las tuplas que no han cumplido la condición de parada de Datafly mientras se genera la tabla de frecuencias.

Teniendo en cuenta todo lo anterior, el coste total del algoritmo es O(numero de iteraciones * (frecuencias y atributos + condición + obtención del atributo con más valores + generalizar) + eliminación), esto es, O($x * (|PT| + |PT| + |ID| + |PT|) + k$) = O($x * |PT|$) ≈ O($|PT|$). Esta última asunción de aproximación se debe al número de iteraciones necesario para la generalización total de cada tipo de datos. Dicho número es variable: los carácteres necesitan 1 iteración, los enteros un máximo de 10, los dobles cerca de 20, las fechas siempre 6 y las cadenas necesitan como máximo un número igual a la longitud máxima permitida. De este modo, la variable x tomaría el valor $|ID| * |cadenaMasLargaPermitida|$. Este valor máximo es inalcanzable en datasets no preparados para ello. En los conjuntos reales ningún atributo del tipo cadena y de gran longitud formaría parte de un identificador. Por ello, se puede asumir que $|ID| * |cadenaMasLargaPermitida|$ realmente es en general muy inferior al valor que toma $|PT|$ y por ello despreciable.

3.3. Forma de generalizar

El algoritmo original usa jerarquías de generalización de campo (DGH en inglés). Esto implica que para cada atributo de cada conjunto de datos distinto, es necesario crear una jerarquía de conceptos que abarque todos y cada uno de los valores que pueden aparecer en ellos, además de los conceptos extras para las generalizaciones intermedias.

Como se puede intuir, esto es muy costoso y rara vez reutilizable salvo para algunos atributos muy comunes. Además es sensible a los errores o a los valores extraños que puedan aparecer en los datos. Algunos de los problemas más comunes de las DGHs encontradas son: sensibilidad a las mayúsculas, falta de acentos, faltas ortográficas, etc. Algunos de ellos son fácilmente remediables, pero otros requieren un sistema más complejo para su reconocimiento. Por ello, en este trabajo se han sustituido las DGHs por jerarquías de generalización según el tipo de dato. Con este cambio, el tiempo que se necesita invertir para acondicionar un conjunto de datos es prácticamente cero a costa de perder precisión en las generalizaciones. Estas jerarquías por tipo de dato son el resultado de utilizar la misma DGH en los campos con el mismo tipo de dato.

Cuadro 2: Configuraciones de los datasets independientes.

NOMBRE	EXPRESIÓN REGULAR
CARÁCTER (CHAR)	.
ENTERO (INT)	[0-9]+
DOBLE (DOUBLE)	[0-9]+([.][0-9]+)?
FECHA (DATE)	[0-9]2[/][0-9]2[/][0-9]4
CADENA (STRING)	.+

Se han definido 6 tipos de datos que pueden contener los atributos, en la tabla 2 se pueden ver sus nombres y las expresiones regulares que los identifican.

La generalización de cada uno es bastante genérica y sencilla, independiente de errores ortográficos siempre que se cumpla la expresión regular. En el caso de que no se obtenga la precisión deseada, se pueden añadir fácilmente más tipos de datos o cambiar la generalización de estos.

Los caracteres simplemente se sustituyen por un *. Se han probado alternativas como usar la operación módulo (%) de los enteros, pero desde el punto de vista de la información que contienen, no es una generalización ya se modifica su significado, por ello se ha preferido usar el enmascaramiento total.

Los enteros se generalizan usando la operación módulo (%) con potencias de 10 crecientes. La función completa es la siguiente: $f(x, i) = x - (x \,\%10^i)$, donde 'x' es el entero a generalizar e 'i' es el número de generalizaciones sufridas por el atributo más uno. Por ejemplo: $1234 \Rightarrow 1230 \Rightarrow 1200 \Rightarrow 1000 \Rightarrow 0$.

Los dobles (punto flotante), se generalizan de forma parecida a los enteros. En la primera generalización, todos los números pasan a tener el mismo número de decimales que el valor que menos tiene, en las siguientes se van quitando decimales hasta que no quedan, luego se tratan como enteros. Por ejemplo: $1234.6789 \Rightarrow 1234.67$ (por la existencia de un valor con dos decimales) $\Rightarrow 1234.6 \Rightarrow 1234 \Rightarrow ... \Rightarrow 0$.

Las fechas se generalizan eliminando los días, meses y, una vez sólo quedan los años, se trata como los enteros. Por ejemplo: $10/02/1234 \Rightarrow 02/1234 \Rightarrow 1234 \Rightarrow ... \Rightarrow 0$.

Las cadenas se generalizan igualando la longitud en la primera generalización y eliminando el último carácter en las siguientes. Cuando sólo queda uno, se le trata como un carácter y se enmascara. Por ejemplo: anonimización $\Rightarrow$ anon (por la presencia de una cadena con 4 caracteres) $\Rightarrow$ ano $\Rightarrow$ an $\Rightarrow$ a $\Rightarrow$ *.

En general, esta forma de generalizar funciona bien, pero la generalización de cadenas usando DGH da un mejor resultado que la usada a cambio de necesitar trabajo extra tanto creando la jerarquía como corrigiendo las inexactitudes que pueden surgir entre la jerarquía y los datos. A pesar de lograr un mejor resultado, se ha optado por el método propuesto al ser más genérico.

4. MONDRIAN

Mondrian es un algoritmo voraz que cambia un poco la forma en que se ven los datos, en vez de tener una tabla como la del ejemplo anterior los datos se tratan como puntos en un hiperespacio cuyas dimensiones vienen dadas por los atributos. Inicialmente todos los datos están dentro de un único hiperespacio, y este algoritmo realiza divisiones a dicho hiperespacio de forma que los puntos en el interior de cada una de las particiones comparten ciertas características.

4.1. Resumen de funcionamiento

El algoritmo comienza creando un hiperespacio o partición única en el que se introducen todos los datos, pudiéndose ver como puntos. Después coge la partición y la divide en dos, por la mediana de la dimensión con el rango de valores normalizado más amplio. Cada uno de los puntos del hiperespacio original pasa a una de las dos particiones. En este punto se tienen dos particiones, de las que se elige una y se vuelve a partir. De este modo se siguen dividiendo las regiones mientras se pueda, es decir, mientras haya al menos 2*k puntos en su interior. El algoritmo termina cuando ya no hay más particiones que se puedan dividir. Entonces se reconstruye la tabla original con las mismas tuplas pero sustituyendo los valores de los atributos por los rangos de las dimensiones de la partición a la que pertenece.

Ejemplo: En la Figura 3 se puede ver una tabla con un solo atributo. Esta se convierte en un 1-espacio finito o segmento. Al dividirla en dos por la mediana, da lugar a dos segmentos de menor tamaño. Después cada tupla de la tabla toma el rango de valores de la partición a la que pertenece. De esta forma se ha conseguido una 3-anonimización.

Si se desease una 2-anonimización, el resultado sería el mismo, dado que ninguna de las dos semirrectas se puede dividir más sin que ningún valor se quede aislado en una partición. 3 valores siempre solo se pueden dividir en 1-2 o 2-1, violando la 2-restricción en ambos casos. De modo que aunque existe una 2-anonimización exacta ([0-2], [3-4], [6-9]), el algoritmo no la encuentra, es decir, no siempre da una solución óptima.

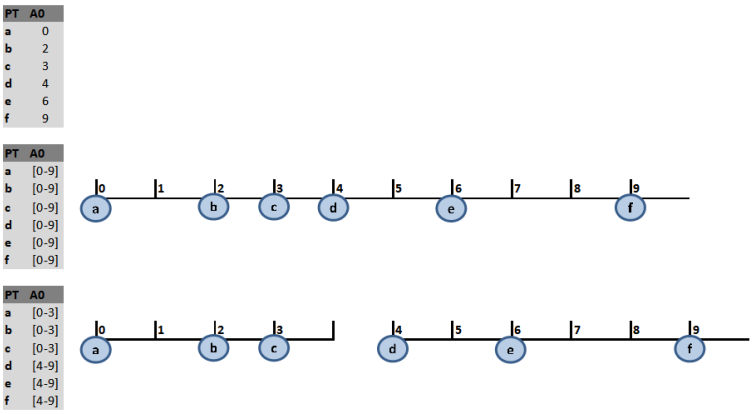


Figura 3: Ejemplo unidimensional de Mondrian.

Mondrian tiene dos formas de realizar la división y estas aportan un comportamiento algo distinto. En primer lugar esta el modo “strict” o estricto el cual pasa todos los valores

coincidentes con la mediana a uno de las dos particiones en su totalidad. Por contra, el método "relaxed" o permisivo reparte los datos de la mediana entre ambas particiones, procurando generar particiones del mismo tamaño.

4.2. Detalles de implementación

En primer lugar se ha desarrollado una clase para representar una partición. Los datos que almacena son a grandes rasgos dos vectores, el primero almacena los índices de las tuplas que pertenecen a la partición (los puntos del hiperespacio) mientras que el segundo almacena los rangos de las distintas dimensiones. Esto se hace guardando el valor mínimo y máximo de cada atributo además de otra información con el fin de evitar tener que recalcularla.

Esta clase tiene dos operaciones claves, la primera es la que calcula los rangos de cada dimensión. Su coste es intuitivamente $O(|P| * |ID|) \approx O(|P|)$, hay que recorrerse cada atributo de cada punto perteneciente a la partición. El coste depende directamente del tamaño de la partición que cada vez es menor. La partición inicial posee $|PT|$ índices, mientras que las últimas menos de 2*k, ya que en caso contrario se podrían seguir dividiendo.

La otra operación clave es la de división. En primer lugar se necesita $O(|ID|)$ para obtener el rango mayor. El rango mayor se considera el que tiene un ratio mayor entre la amplitud del rango en la partición actual y la amplitud que tenía en la partición inicial. Matemáticamente se puede definir como:

$$f(P) = \left\{ ID_{amplio} \;\middle|\; \frac{\text{máx}_{1\leq j\leq |P|}\, PT[j][ID_{amplio}] - \text{mín}_{1\leq j\leq |P|}\, PT[j][ID_{amplio}]}{\text{máx}_{1\leq j\leq |PT|}\, PT[j][ID_{amplio}] - \text{mín}_{1\leq j\leq |PT|}\, PT[j][ID_{amplio}]} = \max_{1\leq i\leq |ID|} \frac{\text{máx}_{1\leq j\leq |P|}\, PT[j][ID_i] - \text{mín}_{1\leq j\leq |P|}\, PT[j][ID_i]}{\text{máx}_{1\leq j\leq |PT|}\, PT[j][ID_i] - \text{mín}_{1\leq j\leq |PT|}\, PT[j][ID_i]} \right\}$$

Por ejemplo: Dados inicialmente los siguientes valores de dos atributos $A_0 = \{100, 101, 102, 103, 104\}$ y $A_1 = \{1, 2, 3, 4\}$. Tras varias iteraciones la partición P tiene los siguientes valores en los dos atributos $A_0 = \{103, 104\}$ y $A_1 = \{1, 2\}$ y tenemos que elegir que atributo de la partición es el más amplio. A_0 tiene un valor de $\frac{104-103}{104-100} = 0{,}25$ mientras que A_1 tiene un valor de $\frac{2-1}{4-1} = 0{,}33$, $f(P) = A_1$. Por ello, A_1 tiene un rango más amplio y es el que se usará para dividir la partición.

Después hay que obtener la mediana del rango, para ello se han probado dos métodos: En primer lugar, y el elegido, es la ordenación de valores y obtención del valor en la posición $|P|/2$. El coste es $O(|P| * \log |P|)$ ya que ese es el coste promedio de ordenar un vector mediante Quicksort de doble pivote. El segundo es conocido como Median of Medians, un algoritmo aproximado que obtiene la mediana (o el elemento en la posición que se quiera) en un vector no ordenado con coste $O(|P|)$. A pesar de su menor coste asintótico, el valor que debe tomar $|P|$ para ser más rápido que la ordenación es demasiado grande por lo que no se suele utilizar en la práctica.

Siguiendo con la división, hay que recorrer todos los indices de la partición para asignarlos a una de las dos nuevas, lo que tiene un coste $O(|P|)$. En el modo "strict", los valores iguales o menores a la mediana, van a una partición y los mayores a otra mientras que en el modo "relaxed" se utiliza una tercera partición temporal para los valores iguales a la mediana que posteriormente se reparten entre las otras dos, con el fin de equilibrar los tamaños de ambas particiones. Por último se generan los nuevos rangos de las dos particiones mediante el método anterior. Así que el coste total de la división es $O(|ID| + |P| + |P| + 2 * |P|) = O(|P|)$.

El coste del algoritmo es $O(|PT| * \log |PT|)$. Las divisiones se pueden ver como un árbol

binario cuyos nodos representan las particiones y el número en su interior corresponde al de elementos que contienen. Ya que el coste de dividir las particiones es O($|P|$), el de dividir un nivel completo del árbol es siempre $|PT|$. Así que el coste total viene dado por el producto de la altura del árbol por el número inicial de elementos $|PT|$. Dicha altura es siempre $\log_2 \frac{|PT|}{k}$.

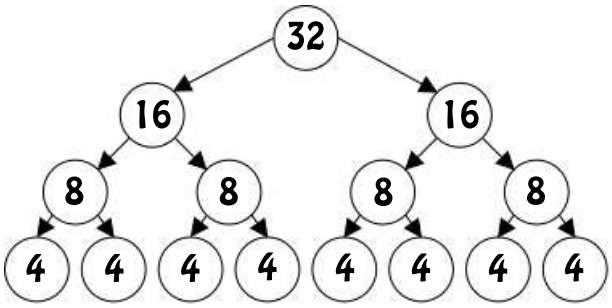


Figura 4: Ejemplo de árbol binario de particiones.

En la Figura 4 se pueden ver las particiones de un conjunto de datos con 32 tuplas y 4-anonimización. La suma de elementos por nivel siempre es 32. Y la altura del árbol es $\log_2 \frac{32}{4} = 3$. Obviamente hay un desajuste cuando no se tratan de potencias de 2, pero esto no cambia el coste asintótico.

5. PRUEBAS

Las pruebas experimentales realizadas a ambos algoritmos tienen tres partes: generación u obtención de los datos de entrada, ejecución y evaluación. Las pruebas realizadas cuentan tanto con datasets sintéticos generados como con el dataset más usado para probar algoritmos de anonimización.

El tipo de dato elegido para las pruebas sintéticas ha sido el ENTERO. Esta decisión no es relevante ya que no influye en el funcionamiento de Mondrian. Datafly tiene una generalización parecida en todos los tipos de datos. Además, la generalización utilizada para los enteros coincide con la DGH más común en la literatura.

Los datasets generados se han dividido en dos grupos, los independientes y los dependientes. Los independientes poseen quasi-identificadores que podrían ser identificadores por sí mismos, independientemente de los demás. Por su parte, los dependientes no son identificadores por sí mismos, se necesitan todos los atributos del dataset para formar el identificador.

Para generar los conjuntos de entrada se ha utilizado la herramienta MATLAB para desarrollar dos scripts que, dados unos parámetros de configuración, escriba uno o varios ficheros con las características deseadas y directamente ejecutables por el programa de anonimización desarrollado. La generación de los valores no es aleatoria, por lo que ejecutando los scripts mencionados anteriormente, se obtienen los mismos conjuntos de valores que los utilizados para las pruebas. Ambos scripts son fácilmente modificables para la generación aleatoria de valores.

Las características configurables de los conjuntos de datos independientes son las siguientes:
-Función de rango de los atributos (func): La función desarrollada tiene dos posibilidades, 'linear' que genera un conjunto de datos con valores crecientes uniformemente y 'quadratic' los genera aumentando su valor exponencialmente. Estas son las dos opciones utilizadas pero la función es fácilmente modificable para soportar otros tipos de crecimiento.
-Número de tuplas (N).
-Número de atributos (A).

En la figura 5 se puede ver un ejemplo de como se distribuyen los valores. En ambas imágenes se muestra un conjunto de 10.000 datos, a la izquierda con una función lineal y a la derecha con una cuadrática.

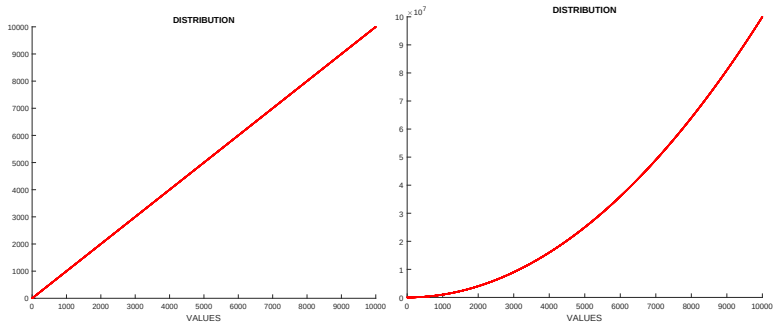


Figura 5: Ejemplo de valores usando una función lineal (izda.) y una cuadrática (dcha.)

Cuadro 3: Configuraciones de los datasets independientes.

N	**A**
1000	1
1000	2
1000	5
1000	10
10000	1
10000	2
10000	5
10000	10
100000	1
100000	2
100000	5
100000	10

Cuadro 4: Configuraciones de los datasets dependientes.

V	**A**	**N**
1:300	2	90.000
1:10	5	100.000
1:3	10	59.049
$1^2, 2^2, ..., 300^2$	2	90.000
$1^2, 2^2, ..., 10^2$	5	100.000
$1^2, 2^2, 3^2$	10	59.049

Las configuraciones que se han usado se pueden ver en la tabla 3. Para la función lineal se han usado todas ellas mientras que para la cuadrática solo hasta N=10000 ya que al elevar al cuadrado producen valores demasiado grandes. En total se han generado 20 conjuntos de datos.

Las características configurables de los atributos dependientes son únicamente dos:
-V: Vector con los diferentes valores que puede tomar cada atributo.
-A: Número de atributos.

Las configuraciones elegidas para la generación de datasets dependientes son las que se pueden ver en la tabla 4. En total 6 conjuntos. El numero de tuplas de un conjunto viene dado por las distintas variaciones con repetición de V tomando A elementos.

Un ejemplo de la diferencia entre atributos dependientes e independientes se puede ver en la tabla 5. Para Datafly no tiene ninguna repercusión utilizar uno u otro más allá de obligar a generalizar al menos una vez por cada atributo en los independientes. Sí que la tiene en Mondrian, ya que al usar los independientes ambos modos de ejecución generan el mismo resultado debido a que la mediana es única en todos los atributos y en todas las particiones que se realicen.

Una vez generados todos los datasets, cada uno se ha utilizado 4 veces con cada uno de los algoritmos. Cada una de esas ejecuciones se ha utilizado un valor de k-anonimización distinto, siendo estos 2, 8, 32 y 128. Mondrian se ha ejecutado tanto en modo estricto como en modo permisivo. Esto hace que el total de ejecuciones haya sido de (20 + 6) datasets * 4 valores de k * 3 algoritmos = 312. De cada ejecución se ha obtenido el número de combinaciones distintas que posee el ID (o particiones), el número de tuplas máximo y mínimo encontrado en

Cuadro 5: Atributos identificadores independientes (izda.) y atributos identificadores dependientes (dcha.)

Independientes		**Dependientes**	
A1	**A2**	**A1**	**A2**
1	1	1	1
2	2	1	2
3	3	2	1
4	4	2	2

una combinación, la media de tuplas por combinación y su varianza. Estas han sido las cinco medidas que se han utilizado para realizar la comparación de las pruebas.

Una vez se han obtenido todos los ficheros de resultados, se ha procedido a su evaluación y generación de gráficas para poder observarlos. En la sección 9 se pueden ver las gráficas generadas a partir de las medidas obtenidas. Como se ha dicho anteriormente, las gráficas de ambos modos de Mondrian (mostradas en las figuras 9 y 10) son idénticas por el uso de datasets independientes.

Además se ha realizado la prueba con el dataset real sobre la población adulta (Adult) disponible en el repositorio de la UC Irvine [2]. Dicho dataset posee 32561 tuplas y 15 atributos todos ellos con información financiera real de una parte del censo estadounidense. Los quasi-identificadores elegidos han sido todos los numéricos ya que los demás no se han considerado información sensible.

6. COMPARACIÓN

Datafly fue uno de los primeros sistemas de anonimización creados (1997), junto con μ-Argus. Mondrian es posterior, cuando ya había varios algoritmos que solucionaban los problemas que se habían encontrado anteriormente por lo que se puede considerar algo más avanzado.

Ambos algoritmos poseen una característica muy deseable en algoritmos de k-anonimización, la seguridad de que los datos resultantes cumplen la k-anonimización. Obtener una óptima k-anonimización es un problema NP-difícil por lo que ninguno de ellos la asegura (aunque puede darse, depende de los datos), y en su lugar se ha óptado por heurísticas voraces.

La estrategia de Datafly es Bottom-Up, esto quiere decir que se parte de múltiples elementos pequeños y se van agrupando en elementos de mayor tamaño, en este caso, los elementos son las tuplas. La estrategia de Mondrian es completamente opuesta, Top-Down, donde inicialmente solo se tiene un conjunto que se va dividiendo en elementos de menor tamaño.

La generalización de Datafly propuesta es la basada en DGH. La calidad de los resultados está muy relacionada a la DGH usada. Cuanto más detallada es la jerarquía mejores resultados se consiguen, pero el tiempo empleado en crearla es mayor, así como el de ejecución. Además de esto, hay que proponer un sistema adicional que controle los datos que no están en la jerarquía ya sea por error del conjunto de datos o por no haberlo tenido en cuenta. La solución adoptada para subsanar este problema es utilizar un conjunto de jerarquías de generalización según el tipo de dato del que se trate y que tengan en cuenta cualquier posible valor soportado por el tipo.

La división de particiones de Mondrian se realiza por la mediana del atributo con un rango normalizado mayor. Esta heurística voraz funciona bastante bien incluso con atributos repetidos, aunque si los hay y no se tiene en cuenta, el modo permisivo de este algoritmo puede dejar de asegurar la k-anonimización. Por ello se suele usar la pre-condición que impida haber identificadores repetidos en los datos de entrada.

La heurística de Datafly tiende a sobregeneralizar más conforme el número de generalizaciones sobre un atributo crece, por ello se utiliza la supresión cuando quedan pocas tuplas que no cumplan la k-anonimización. Con dicha supresión se evitan generalización extra cuando hay algunos datos muy alejados de los demás. Esto mitiga un poco la sobregeneralización pero no la evita. Una de las razones por la que esto pasa es que Datafly utiliza información externa y fija para generalizar (DGHs) mientras que Mondrian utiliza rangos, información obtenida dinamicamente de los datos. Por ejemplo, la generalización de los enteros agrupa los valores de 10 en 10, luego de 100 en 100, de 1000 en 1000, etc. Mondrian los agrupa mediante rangos no fijos y de diferente amplitud.

El coste asintótico de Datafly es $O(|PT|^2)$, polinómico de segundo grado (aunque en la práctica es mucho menor). Mondrian por su parte tiene un coste $O(|PT| * log_2|PT|)$, aunque con los números de tuplas manejados en las pruebas, se usa un algoritmo para encontrar la mediana que eleva el coste a $O(|PT| * (log_2|PT|)^2)$.

Mediante la pruebas realizadas se ha podido comprobar como Datafly realmente sobregeneraliza y Mondrian consigue unos resultados muy cercanos a los óptimos en muchos de los casos.

En las gráficas de las figuras de la subsección 'Gráficas de los conjuntos con atributos independientes' se puede ver los resultados obtenidos de las ejecuciones de los datasets con

Cuadro 6: Tabla con el número de combinaciones logradas con los datasets de 100000 tuplas y atributos independientes

K	Óptimo	Datafly	Mondrian
128	781.25	100	512
32	3125	1000	2048
8	12500	10000	8192
2	50000	10000	34464

Cuadro 7: Tabla con los valores óptimos del número de combinaciones del ID de los datasets con atributos dependientes.

Valores distintos por atributo	Atributos	k	Óptimo
3	10	2	29524.5
10	5	2	50000
300	2	2	45000
3	10	8	7381.13
10	5	8	12500
300	2	8	11250
3	10	32	1845.28
10	5	32	3125
300	2	32	2812.5
3	10	128	461.32
10	5	128	781.25
300	2	128	703.13

atributos independientes.

Con atributos independientes, lineales y un gran número de tuplas, el ratio de aproximación al óptimo de Mondrian es constante con los diferentes valores de k (65.5 %), mientras que el de Datafly es muy variable desde un 12.8 % logrado con k=128 hasta un 80 % logrado con k=8 (ver tabla 6). En media Mondrian logra un ratio algo mayor.

Con pequeños números de tuplas y atributos lineales, ambos algoritmos no funcionan tan bien, con los datasets de 1000 tuplas, Datafly suele anonimizar por completo los datos debido a la proximidad entre todos los valores, mientras que el ratio de aproximación del número de combinaciones de Mondrian empeora y se vuelve algo inestable como se puede ver en la gráfica inferior de la gráfica 9 para k=8.

Una de las características que más distinguen a ambos algoritmos es la diferencia entre los resultados de los datasets lineales y cuadráticos. Mondrian es completamente insensible a las distribuciones de los valores de los datos, mientras que Datafly se comporta de manera muy distinta. Tanto el número de combinaciones como el número de tuplas en ellas es muy distinto en Datafly, por ello aparecen los picos en todas las gráficas de la figura 8, correspondiendo a los datasets con atributos cuadraticos. Las gráficas de Mondrian son completamente horizontales en los datasets con igual número de tuplas.

Con los datasets con atributos dependientes, las estadísticas son algo más variables como se puede ver en las gráficas de la subsección 'Gráficas de los conjuntos con atributos dependientes'. El número óptimo de combinaciones del ID de los datasets se puede ver en la tabla 7.

Con los datasets dependientes, la diferencia entre lineal y cuadrática es mucho menor

ya que el número de valores distintos de un atributo es mucho menor, la diferencia se puede ver entre los datasets de exactamente 100000 tuplas y 5 atributos. En los independientes variaban entre 1 y 100000 mientras que en los dependientes solo de 1 a 10. La diferencia entre los cuadrados consecutivos es mucho menor, casi despreciable $100000^2 - 99999^2 = 199{,}999 VS 10^2 - 9^2 = 19$.

Mondrian en modo restrictivo logra la mejor media de aproximación en número de combinaciones de ID, pero con valores de k grandes, el número de tuplas por combinación se vuelve irregular como se puede observar en la gráfica de la variación de la figura 12.

Por otro lado Mondrian en modo permisivo, es el más regular, sobre todo conforme mayor es la variación de los valores de los atributos y mayor es la k utilizada. Esto se puede ver en las gráficas de la figura 13, cuyas lineas son bastante horizontales. En la gráfica correspondiente a la variación se puede ver como con el k menor no se sigue la misma tendencia decreciente que con los demás valores de k.

Datafly se acerca a los números de combinaciones de Mondrian cuanto menor es el número de valores distintos que toman los conjuntos de datos. En los demás casos consigue unos pobres resultados ya que necesita generalizar demasiado. Además, cuanto mayor es el número de tuplas, mayor es la variación de tuplas por combinación.

En la subsección 'Gráficas comparativas con el dataset real 'Adult'' se puede ver como con un dataset con información real, Datafly logra unos malos resultados en comparación con los de Mondrian. El modo permisivo logra unos buenos resultados y regulares mientras que el mismo algoritmo en modo estricto obtiene unos resultados algo mejores que los de Datafly pero más regulares.

En resumen, Mondrian logra unos resultados mejores y con mayor regularidad que los conseguidos por Datafly en las pruebas realizadas tanto con conjuntos de datos sintéticos como con los reales.

7. CONCLUSIÓN

Tras obtener una visión global del problema de la anonimización leyendo los trabajos publicados por distintas universidades y grupos de investigación, se puede observar la gran cantidad de formas de abordar dicho problema. Por eso se ha querido comparar dos sistemas conocidos de anonimización, uno de los más antiguos y otro de los más modernos.

Datafly es uno de los primeros algoritmos de este estilo que se crearon, mientras que Mondrian es algo más novedoso y, por ello, avanzado al disponer de más información ya publicada. En las pruebas realizadas se ha podido ver como Mondrian es sus dos modos de ejecución, lograba mejores resultados y era más constante aunque se le varíen las propiedades de los datos.

Datafly usa la generalización del atributo con mayor número de valores distintos, lo que no tiene en cuenta la dependencia con los demás atributos, su generalización es unidimensional. Mondrian si lo tiene en cuenta a la hora de hacer sus particiones ya que son multidimensionales. Por esto, las divisiones de Mondrian son más precisas que las divisiones de Datafly si tomamos en conjunto todos los atributos del identificador.

Datafly en promedio es algo más rápido que Mondrian pero el tiempo no se ha tenido en cuenta en la presente comparación. Esto puede cambiar ya cada partición de Mondrian se puede ver como un problema independiente y puede mejorar mucho su tiempo haciendo uso de la concurrencia. Por todo lo anterior (ver en el resumen de la tabla 1) se puede concluir que Mondrian consigue mejores resultados con un coste asintótico mayor pero sin necesidad de eliminar o suprimir tuplas.

Por ello, concluímos que en caso de querer un algoritmo que no requiera ningún tipo de construcción extra o análisis previo de los datos, se debería elegir Mondrian. Sin embargo, si se pueden tratar los conjuntos de datos y crear jerarquías muy precisas, los resultados pueden ser parecidos en ambos casos.

Cabe destacar que ambos algoritmos se pueden adaptar para funcionar como el otro, Datafly utilizando rangos y Mondrian haciendo uso de DGHs. Pero no es lo propuesto en los trabajos originales y por ello no se ha tenido en cuenta.

Otro de los resultados de realizar este trabajo es el código desarrollado, el cual se puede encontrar en [4], es fácilmente modificable y ampliable según las necesidades del usuario final. También hay varias mejoras de rendimiento que se pueden implementar como el ya citado uso de threads.

Como trabajo futuro, se pueden añadir otros sistemas de anonimización tempranos como el ya mencionado μ-Argus o sistemas más novedosos como Incognito para realizar una comparación entre los cuatro o tempranos contra novedosos.

Otra de las opciones es llevar a cabo una comparación más exhaustiva modificando los algortimos desarrollados y ejecutandolos en una máquina dedicada para así poder tener en cuenta el tiempo de ejecución u otras medidas. Algunos ejemplos de modificaciones serían: permitir el uso de DGHs precisas y concretas en el caso de Datafly o usar threads en el de Mondrian.

También se pueden comparar los resultados obtenidos entre algoritmos que tratan la k-anonimización y los que emplean la l-diversidad.

8. TIEMPO INVERTIDO

El tiempo invertido en este trabajo se puede dividir en 5 puntos, la recopilación de información, la implementación tanto de los algoritmos como de las herramientas auxiliares, las pruebas, la comparación y la elaboración de la documentación, entre la que se incluye el presente documento.

Para la recopilación de información se han invertido 40 horas aproximadamente. En la recopilación de información se ha incluido tanto el tiempo invertido en la búsqueda y lectura de la información utilizada cómo la no utilizada. Por ejemplo, otros algoritmos descritos en la literatura, métodos alternativos de anonimización como la l-diversidad, otras formas de medir los resultados prácticos, etc. En general, todo aquello que ha contribuido a tener una visión más amplia del problema y la solución.

El tiempo invertido en la implementación engloba todo aquel que se ha invertido escribiendo el código de los algoritmos y las herramientas creadas para llevar a cabo las pruebas. Además se incluye el tiempo de la pruebas de corrección del algoritmo que comprueban su correcto funcionamiento y las pequeñas mejoras. En esta parte se han empleado cerca de 110 horas.

En las pruebas se han utilizado alrededor de 65 horas, entre las que se incluyen las invertidas en buscar datasets reales, generar los sintéticos y ejecutar los propios conjuntos de datos con los diferentes algoritmos y valores de k.

La comparación ha requerido de 25 horas. Es el tiempo utilizado en aplicar las herramientas de evaluación antes mencionadas a los resultados obtenidos en las pruebas y obtener otras informaciones útiles de los propios trabajos o del trabajo desarrollado en el presente trabajo.

Por último, el generar la documentación entre la que se incluye este documento ha costado 65 horas aproximadamente. Entre este tiempo se incluye el aprendizaje de LaTeX, el procesador de textos y la realización de las imágenes utilizadas para las explicaciones.

La duración total del trabajo ronda las 305 horas, el desglose de la utilización de estas se puede ver en la figura 8.

Figura 6: Desglose del tiempo invertido.

TAREA	TIEMPO APROX. (h)
Recopilación de información	40
Implementación (Datafly)	35
Implementación (Mondrian)	60
Implementación (Herramientas extra)	15
Pruebas (Búsqueda y generación de datasets)	20
Pruebas (Ejecución y obtención de resultados)	45
Comparación	25
Memoria y documentación	65
TOTAL	**305**

9. NOMENCLATURA

En este trabajo se utilizará la siguiente nomenclatura y abreviaturas:

- PT (private table): Conjunto de datos (dataset) de entrada (tuplas-atributos).
- N ó $|PT|$: Longitud de PT. Número de tuplas.
- PT[i]: Tupla i de PT.
- PT[i][j]: Valor del atributo j en la tupla i.
- A_i: Atributo i de PT.
- Identificador ó ID: Conjunto de atributos que identifican a una única tupla en el conjunto.
- Quasi-identificador ó QI: Cada uno de los atributos que conforman el ID.
- ID_i: QI i del ID.
- $|ID|$: Número total de QIs.
- Jerarquía de generalización por campo ó DGH: Estructura arborescente que indica cómo generalizar un campo (atributo) concreto.
- Partición ó P: Conjunto de puntos que forman un espacio multidimensional.
- $|P|$: Número de puntos o índices que posee P.

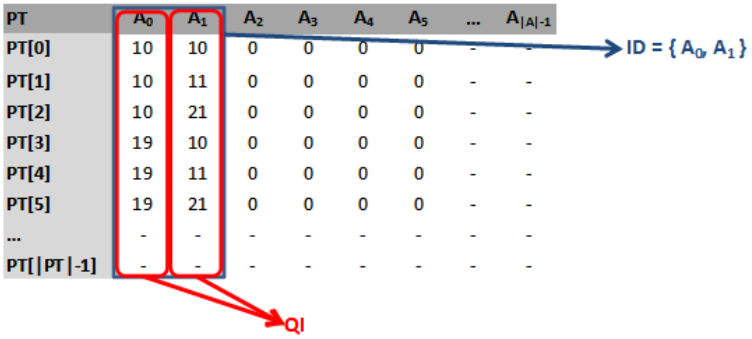


Figura 7: Ejemplo ilustrativo usando la nomenclatura.

En la Figura 7 hay dos atributos que conociendo su valor, permiten identificar una tupla. Este par de atributos forma el ID, y por lo tanto A_0 y A_1 son QIs.

REFERENCIAS

[1] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. 2005.

[2] M. Lichman. UCI machine learning repository, 2013. University of California, Irvine, School of Information and Computer Sciences. Adult Data Set, donors: Ronny Kohavi and Barry Becker.

[3] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. 2008. 15 pages. The University of Texas at Austin.

[4] Alejandro Fernández Poza. Código desarrollado para el presente trabajo. https://bitbucket.org/AlexFP/anonymization/overview. Accedido: 27-08-2017.

[5] Latanya Sweeney. Datafly: a system for providing anonymity in medical data. 1997. 20 pages. Laboratory for Computer Science, Massachusetts Institute of Technology. Cambridge, MA 02139 USA.

[6] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. 1997. 5 pages. Laboratory for Computer Science, Massachusetts Institute of Technology. Cambridge, Massachusetts.

[7] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. 2002. 18 pages. School of Computer Science, Carnegie Mellon University. Pittsburgh, Pennsylvania, USA. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10 (5), 2002; 571588.

ANEXO: GRÁFICAS DE LOS RESULTADOS

En esta sección se exponen las gráficas generadas utilizando las medidas tomadas en la ejecución de los diferentes datasets con los diferentes algoritmos y valores de k.

Las figuras correspondientes a las pruebas sintéticas poseen 5 gráficas: la superior izquierda corresponde al número de las distintas combinaciones (particiones) presentes en los resultados, la superior derecha y central izquierda son los valores mínimo y máximo de tuplas que hay en una sola partición, la central derecha es el número medio de tuplas por partición (se puede ver como la k-anonimización media alcanzada) y por último en la parte inferior está la varianza del número de tuplas respecto a la media. Cada gráfica posee cuatro líneas correspondiente a los cuatro valores de k probados. El eje X corresponde al nombre de los datasets utilizados. Estos contienen sus propiedades codificadas: los independientes utilizan la siguiente regla dataset[func]-[N]-[A] y los dependientes dataset[func]-[$|V|$]-[A]. El significado de cada parámetro se puede consultar en la sección 5.

Las figuras de las pruebas del dataset real siguen la misma distribución que las anteriores pero en los ejes X no está el nombre de los datasets sino los valores de k y las líneas ahora corresponden a cada uno de los algoritmos.

Gráficas de los conjuntos con atributos independientes

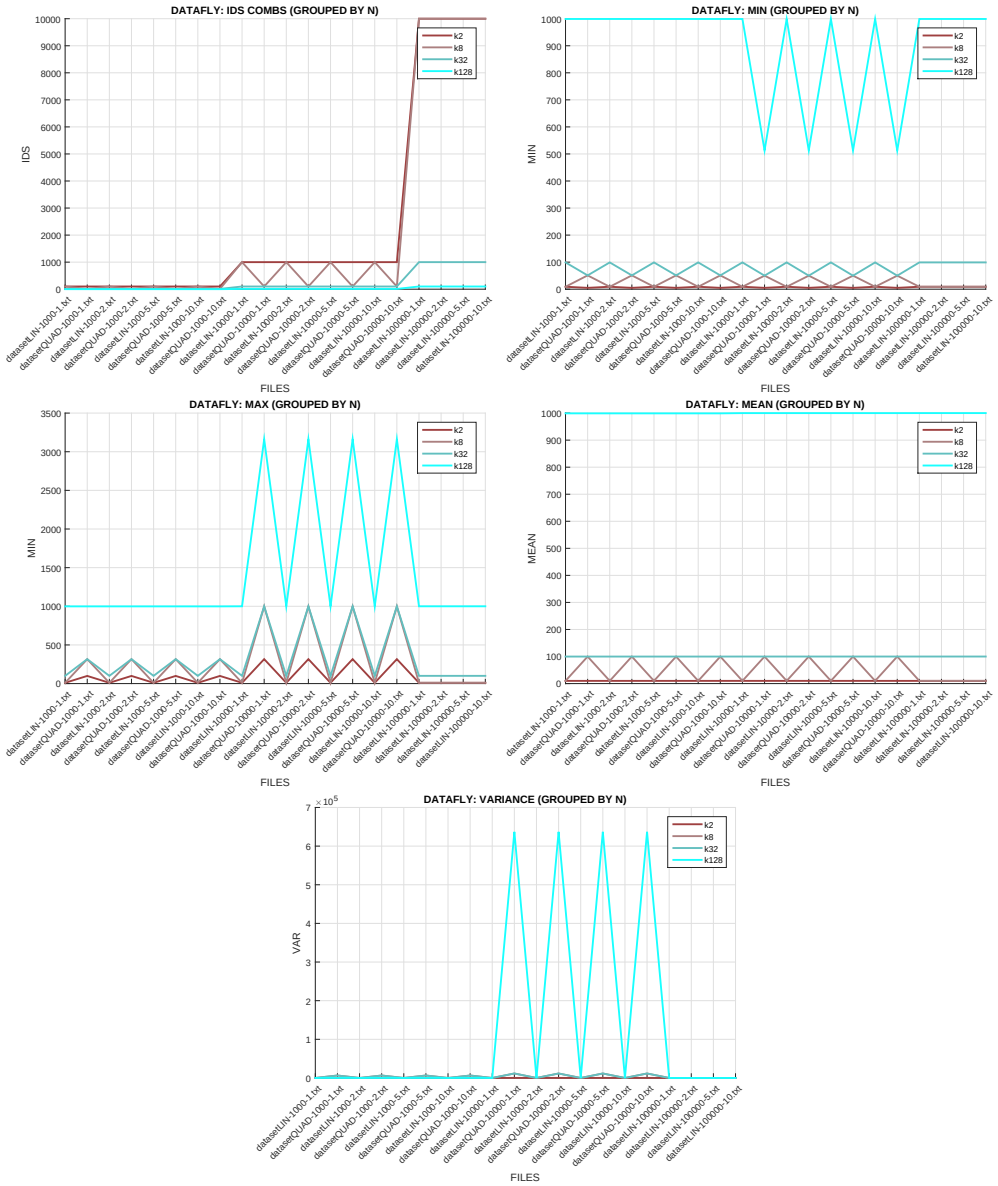


Figura 8: Gráfica de los resultados obtenidos por Datafly con atributos independientes.

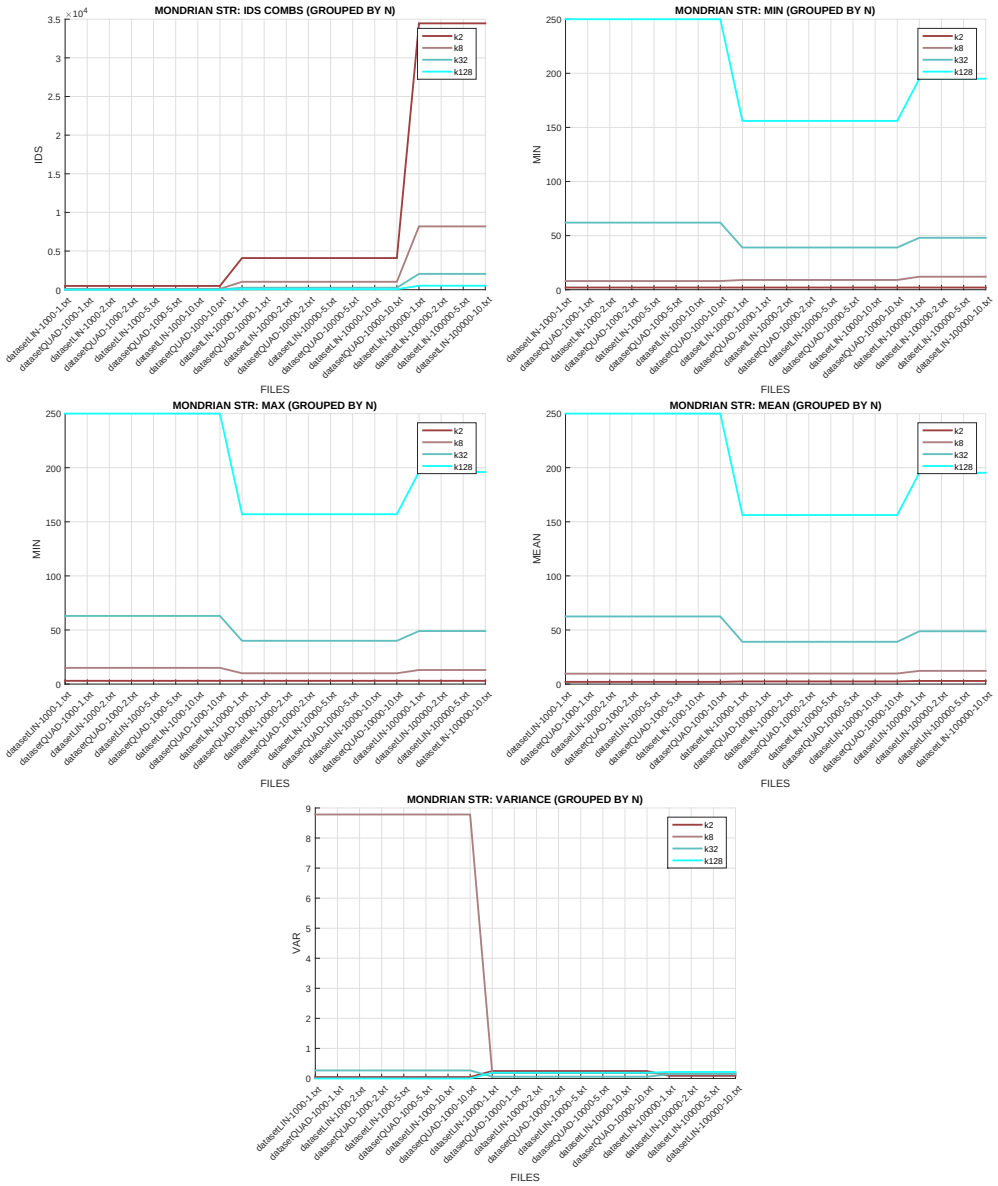


Figura 9: Gráfica de los resultados obtenidos por Mondrian estricto con atributos independientes.

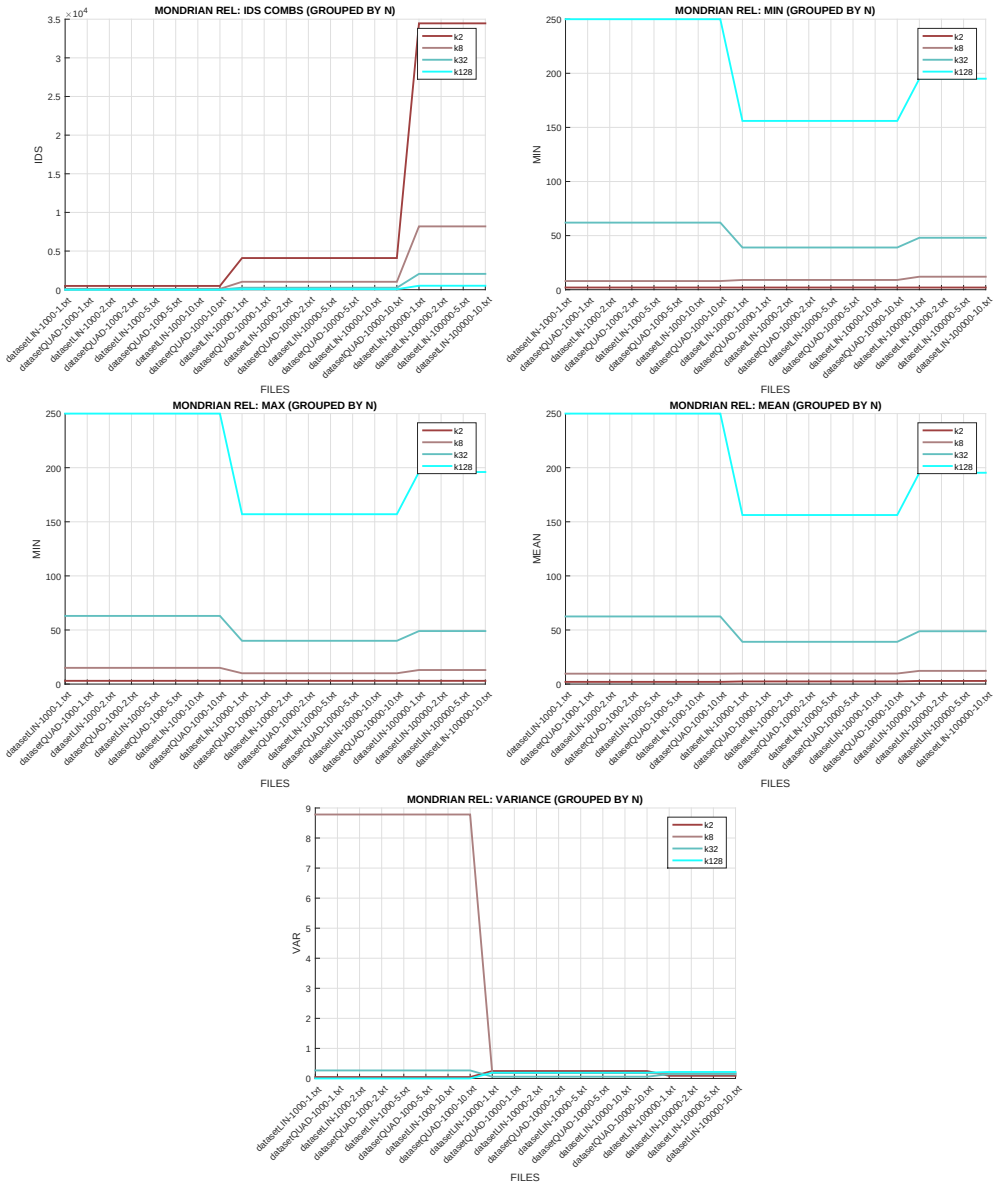


Figura 10: Gráfica de los resultados obtenidos por Mondrian permisivo con atributos independientes.

Gráficas de los conjuntos con atributos dependientes

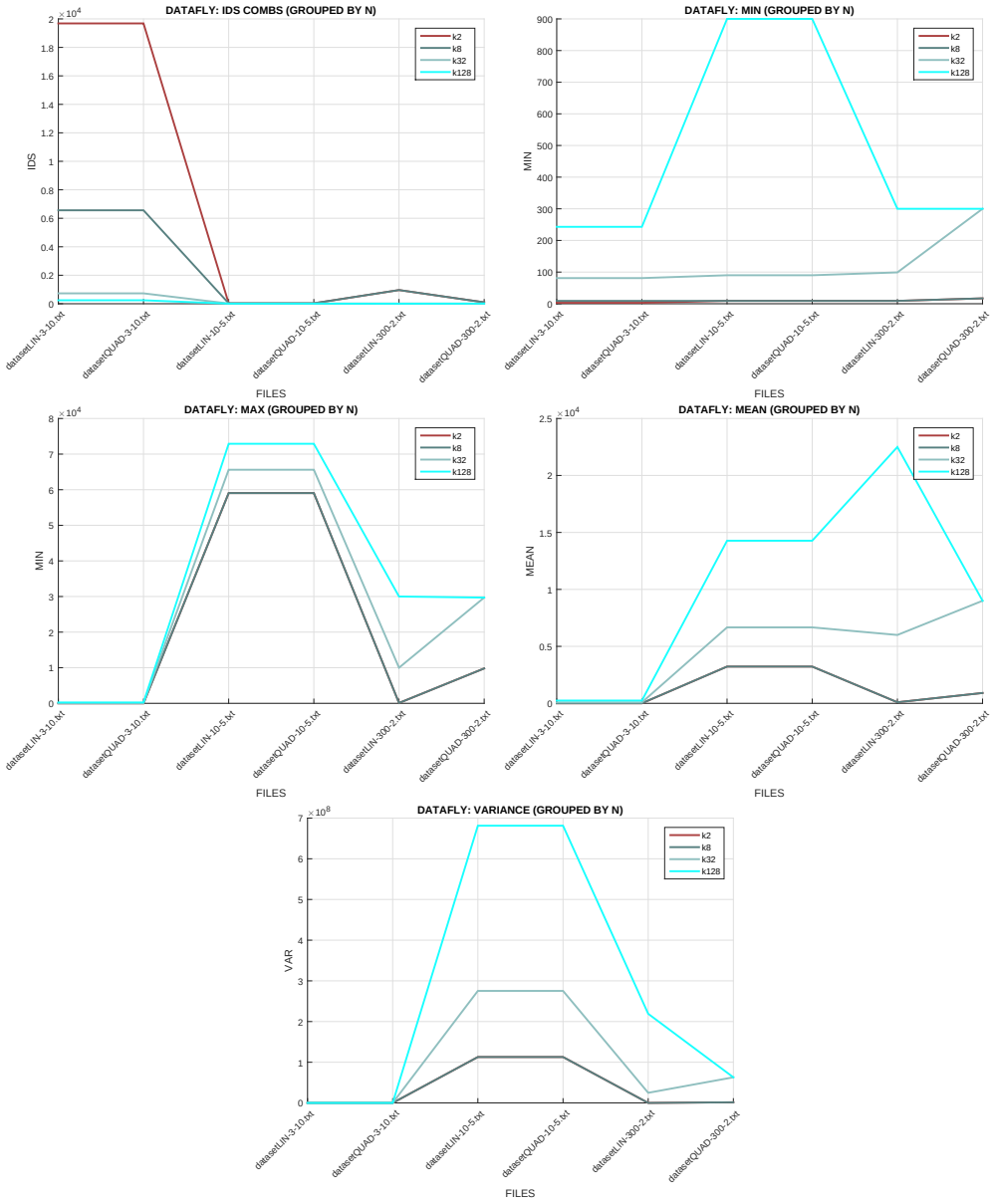


Figura 11: Gráfica de los resultados obtenidos por Datafly con atributos dependientes.

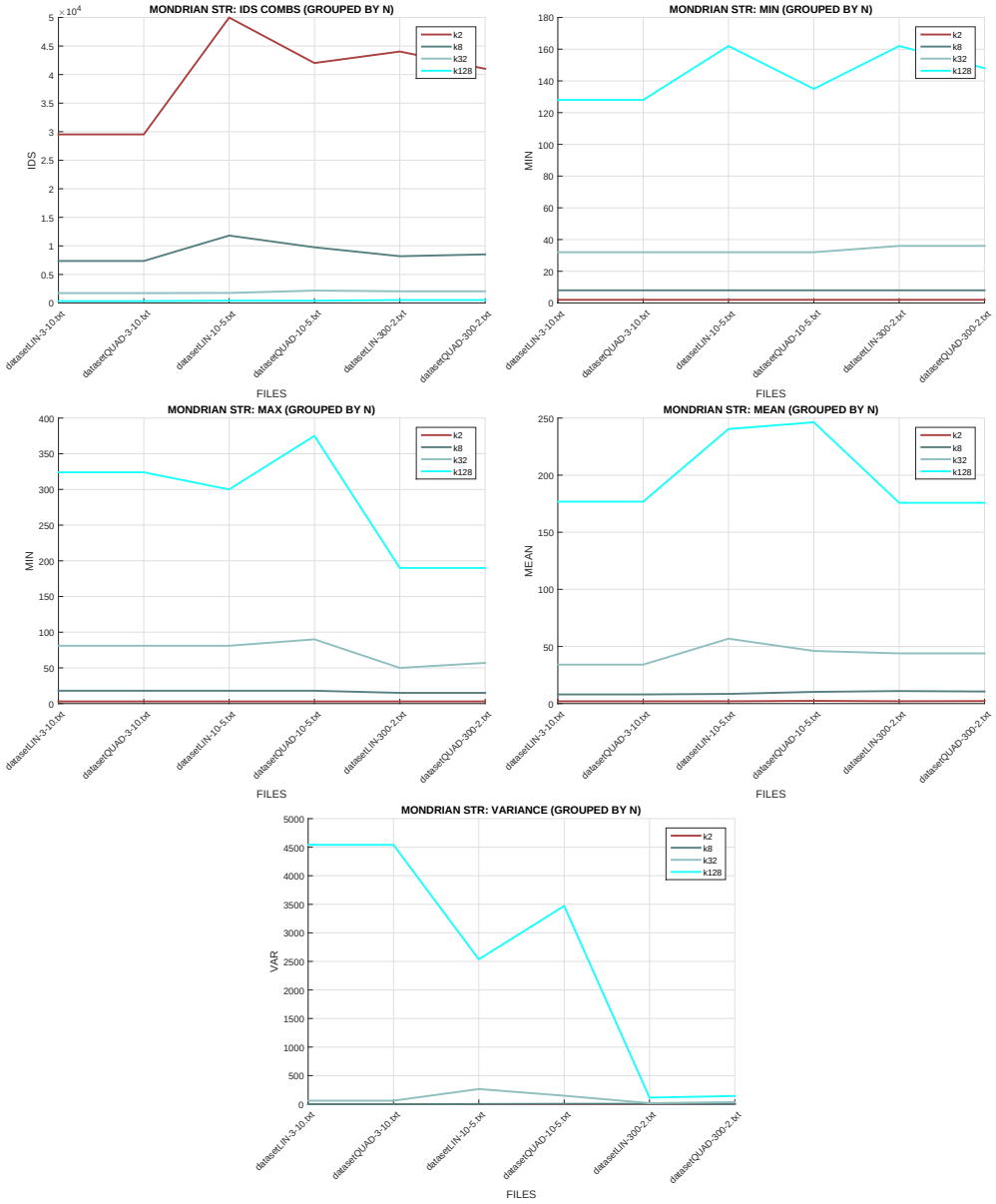


Figura 12: Gráfica de los resultados obtenidos por Mondrian estricto con atributos dependientes.

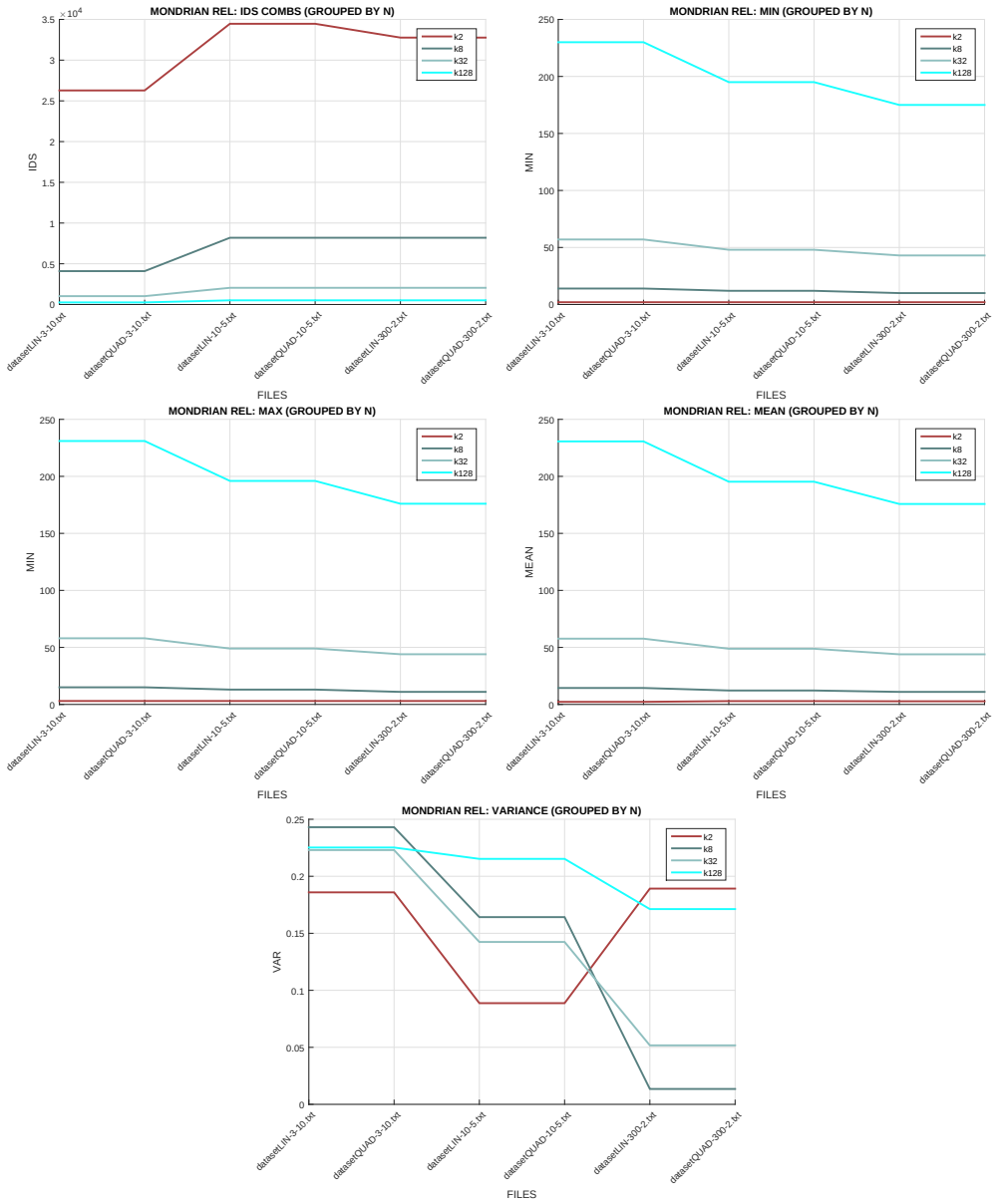


Figura 13: Gráfica de los resultados obtenidos por Mondrian permisivo con atributos dependientes.

Gráficas comparativas con el dataset real 'Adult'

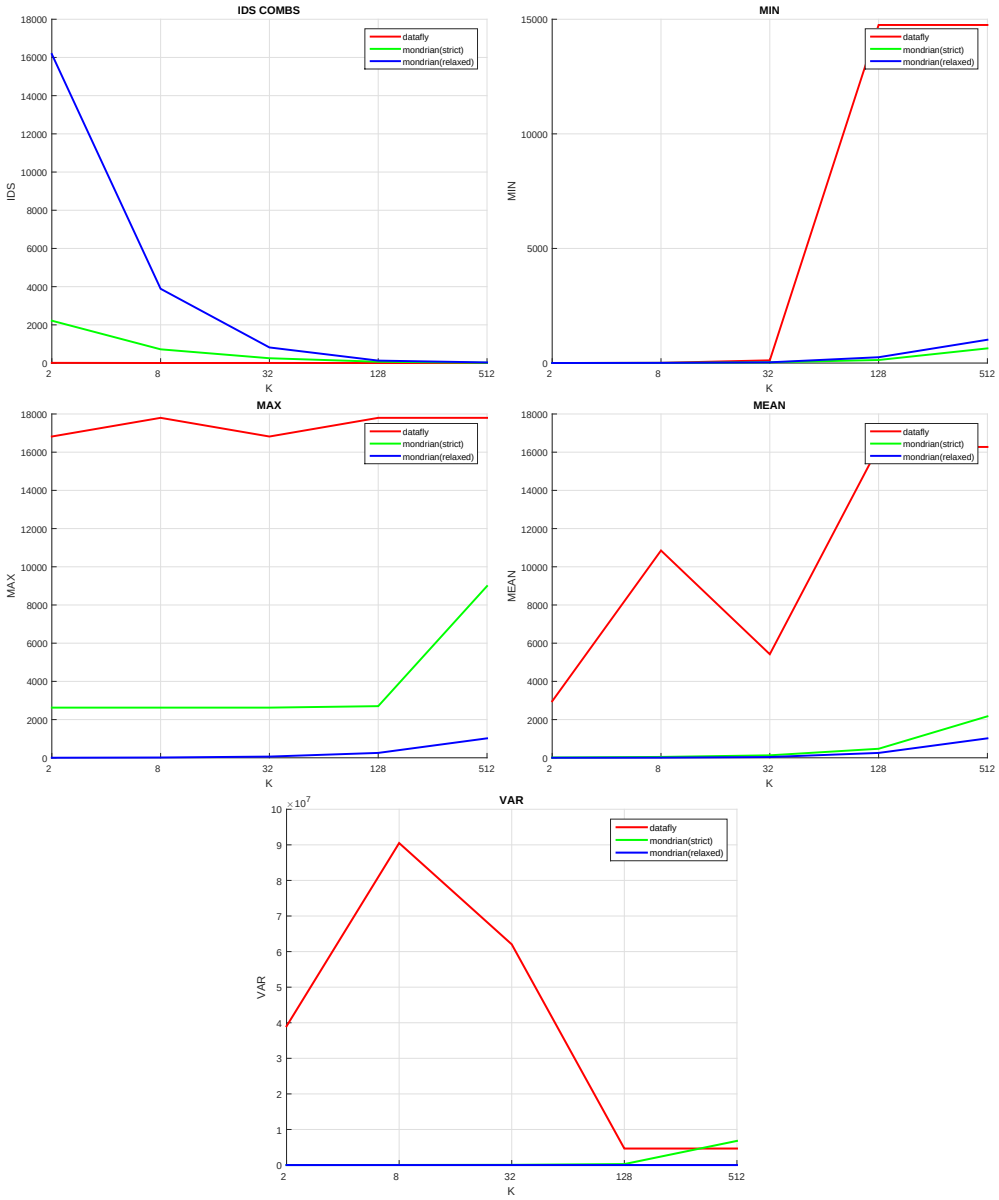


Figura 14: Gráfica de los resultados de los algoritmos con el dataset Adult y diversos valores de k.