INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE COMPUTACION

PROGRAMA DE MAESTRIA

Propuesta de un algoritmo para la selección democrática de individuos

Tesis para optar al grado de

Magister Scientiae en Computación

Rafael Monge Montenegro

Cartago, Costa Rica

Resumen

En el presente documento se introduce un algoritmo de muestreo inspirado en el muestreo estratificado. A diferencia de éste último, que solo funciona sobre estratos discretos con intersección nula (usualmente formados distribuyendo los individuos en estratos por los valores discretos de una propiedad de la población), el algoritmo propuesto está diseñado para funcionar sobre estratos discretos cuya intersección no es nula, creados al dividir los individuos en estratos a partir de los valores discretos disponibles en múltiples de las propiedades de la población.

Entre posibles aplicaciones para este algoritmo se encuentran la creación de muestras de lo más diversas (o, de forma un tanto eufemística "democráticas") posibles, a partir de una población dada. Esto puede resultar útil, por ejemplo, al seleccionar individuos para una encuesta en la que la diversidad de la muestra sea crítica. También, para detección de anomalías (al permitir obtener una mezcla con todo tipo de individuos, incluidos los más inusuales.)

El algoritmo propuesto es comparado contra el algoritmo SRS en términos de tiempo y efectividad de maximizar la diversidad, tomando en cuenta parámetros como el tamaño de la población, de la muestra, el número de propiedades de los individuos, etc. Los resultados indican que es más lento que el SRS, pero cumple más que satisfactoriamente el objetivo de brindar muestras diversas.

Abstract

This document presents an algorithm inspired in stratified sampling. Unlike stratified sampling, which only works on discrete strata with a null intersection (usually created by distributing individuals into strata by using the discrete values available in a single property of the population), the proposed algorithm is designed to work with discrete strata with a non-null intersection, created by dividing individuals into strata by using the discrete values available in several of the properties of the population.

Possible uses for this algorithm include creating samples that are as diverse (or, more euphemistically, "democratic") as possible from a population. This can be useful, for instance, when selecting individuals for a survey en which the diversity of the sample is key. It can also be useful in anomaly detection (since it will yield a sample with every kind of individual, even the more or unusual).

The proposed algorithm is thoroughly compared against the SRS algorithm in terms of performance and the effectiveness with which it increases diversity. Parameters such as size of the population, the sample and the number of properties in the sample are taken into account to perform this comparison. Results indicate that the algorithm, while slower than the SRS, meets the objective of yielding diverse samples.

An hypothesis that arises from this research is that the algorithm is a generalization of stratified sampling: executing it while taking into account only one property of the population is likely to return the same result. A formal proof of this hypothesis is an interesting avenue of research, that shouldn't prove too hard.

Keywords: Sampling, Variance reduction, Stratified sampling, SRS

www.tec.ac.cr

TEC | Tecnológico de Costa Rica
Programa de Maestría en Computación
Teléfono: 2550-2402

ACTA DE APROBACION DE TESIS

Con fundamento en lo que establecen los **Artículos 22-24-25** del "Manual de Normas y Procedimientos para optar por el título de "MAGÍSTER SCIENTIAE EN COMPUTACION", el Tribunal Examinador de Tesis (TET), nombrado con el propósito de evaluar la tesis de grado.

"Propuesta de un algoritmo para la selección democrática de individuos"

Habiendo analizado el resultado general del trabajo presentado por el estudiante:

Primer Apellido	Segundo Apellido	Nombre	No. de carné
MONGE	MONTENEGRO	RAFAEL ALONSO	200750423

Emite el siguiente dictamen:

Ø APROBADO

El TET, considerando que el trabajo realizado por el estudiante es SOBRESALIENTE, le otorga la siguiente MENCION HONORIFICA:

O CUM LAUDE O MAGNA CUM LAUDE Ø SUMMA CUM LAUDE

O REPROBADO

O SE RECOMIENDA O NO SE RECOMIENDA

Brindarle una nueva oportunidad para la DEFENSA PUBLICA de su Tesis

NUEVA FECHA: _____

Dando fe de lo aquí expuesto firmamos (*IDEM*: HOJAS DE APROBACION DE TESIS)

PhD. Carlos González Alvarado
Profesor Asesor

PhD. José Helo Guzmán
Profesor Lector

M.Sc. Patricia Madriz Huertas
Profesor Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

25 de junio de 2014

Sello

FT-07-MC

www.tec.ac.cr

TEC | Tecnológico de Costa Rica
Programa de Maestría
en Computación
Teléfono: 2550-2402

APROBACIÓN DE LA TESIS

"Propuesta de un algoritmo para la selección democrática de individuos"

TRIBUNAL EXAMINADOR

PhD. Carlos González Alvarado
Profesor Asesor

PhD. José Helo Guzmán
Profesor Lector

M.Sc. Patricia Madriz Huertas
Profesor Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

Junio, 2014

Dedicatoria

Dedico esta tesis a mis padres quienes me apoyaron todo el tiempo, así como a mi hermana que ha sido un modelo a seguir desde joven.

Capitulo 1 Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Índice de Fórmulas

Capitulo 1 Planteamiento del problema

1.1 Descripción informal del problema

El proceso de seleccionar individuos (personas, carros, frutas, etc.) de una población con el objetivo de realizar algún análisis sobre este conjunto reducido, es un proceso muy común en la ciencia. La variante más conocida de estos procesos es el *muestreo*: la selección de unidades de una población con la intención de utilizar la muestra para estimar los valores de la población. (TILLE, Sampling Algorithms, 2006) Ejemplos de estos valores son la media, la varianza etc.

Un proceso muy similar al muestreo resulta útil en otros escenarios: seleccionar unidades, no para estimar propiedades de la población, sino para realizar análisis sobre estas unidades directamente. Claramente, lo óptimo es que las unidades seleccionadas sean "interesantes", palabra que solo cobra relevancia en contexto.

Un ejemplo de estos escenarios es la *detección de anomalías*[1]. Puede asumirse que existe un proceso relativamente largo y costoso que determina inequívocamente si un producto que está siendo manufacturado se encuentra en buen estado o no. Dado el volumen de los productos manufacturados, resulta inviable ejecutarlo sobre todas ellos. Sin embargo, ejecutarlo sobre un subconjunto de los productos sí es factible.

Realizar este largo proceso sobre una muestra aleatoria de los productos probablemente no daría buenos resultados. Pero, si se ejecuta sobre un conjunto de productos que (con un cálculo rápido) fueron catalogados como potencialmente defectuosos, puede dar excelentes resultados con un bajo costo. Este cálculo rápido permite en efecto seleccionar las unidades *interesantes*, aquellas en las que hay algo sugerente de una anomalía.

El problema bajo investigación es similar al anterior, con una variante: no hay forma de determinar si un individuo es probablemente anómalo de forma rápida. Solo se tiene el proceso largo y costoso. Sin embargo, se sabe de antemano que si un individuo es anómalo, los que compartan propiedades con él tienen alta probabilidad de también serlo. Por ejemplo, si se detecta que un automóvil es defectuoso, y este fue creado por Juan y Pedro, es altamente probable que algún otro producto en el que alguno de ellos participó también se encuentre defectuoso. Uno en que ambos estén involucrados sería *especialmente sospechoso*.

En estos escenarios, un subconjunto que contenga la mayor diversidad posible es lo óptimo. En el ejemplo de los automóviles, un subconjunto que contenga la mayor heterogeneidad de

[1] Detección de anomalías es la identificación de ítems u observaciones que no siguen el patrón usual de otros ítems en un conjunto de datos.

personas que participan en la creación de los automóviles permitiría maximizar la cantidad de defectos encontrados.

En concreto, el problema por resolver es cómo seleccionar un subconjunto de individuos de una población de tal forma que, a lo largo de cada una de las propiedades de los individuos, se dé la mayor diversidad posible. Para lograr esto, debe alterarse la probabilidad con la que individuos que pertenecen a una minoría cuentan para ser seleccionados. En efecto, la probabilidad de las minorías de ser seleccionada debe ser tan cerca como sea posible de ser *equiprobable* con la de las mayorías.

Este problema a primera vista parece ser equivalente a un simple muestreo, pero en la práctica es completamente distinto. La *muestra* seleccionada no es representativa de la población. No se espera que lo sea, ni se intenta con ella hacer ningún tipo de aserción sobre la población como tal, solo analizar los individuos. Por lo tanto, como se ejemplifica posteriormente, las decenas de algoritmos de muestreo tradicionales resultan insuficientes.

En la próxima sección, se elabora más formalmente sobre una opción de cómo lograr este objetivo. Se plantea como un problema más matemático, cuya solución presenta una forma atractiva de atacar el requerimiento aquí planteado. Sin embargo, este planteamiento más formal no es más que una de muchas formas en las que quizá se puede cumplir con este requerimiento, por lo que no se debe perder la perspectiva original del mismo.

1.2 Planteamiento formal del Problema

Esta sección describe el problema descrito anteriormente desde un punto de vista matemático.

Sea U la población inicial, un conjunto de N individuos tal que $U = \{u_1, u_2 \dots u_N\}$

Todos los elementos del conjunto comparten un conjunto de propiedades (también llamadas comúnmente *factores*, *variables* etc. dependiendo del contexto) de tamaño M. Sea este conjunto de propiedades llamado X, tal que $X = \{x_1, x_2 \dots, x_M\}$.

Cada una de las propiedades dentro de X tiene a su vez un conjunto finito de valores. Sea este conjunto $\{x_i v1, x_i v2 \dots x_i vK\}$. Para hacer más sencilla la notación, se denota este conjunto utilizando la notación de dominio de función. Es decir $dom(x_i) = \{x_i v1, x_i v2 \dots x_i vK\}$ para $i = 1, \dots, N$.

El número de valores posibles puede ser distinto para cada propiedad, y será representado como $|dom(x_i)|$.

Sea S un subconjunto de U, de cardinalidad n. Este conjunto representa los elementos seleccionados.

Sea Sx_iv_j igual al subconjunto de elementos dentro de S tal que en la propiedad x_i poseen el valor de v_j. Sea a su vez $|Sx_iv_j|$ el total de elementos dentro de este conjunto.

Con estas definiciones, el problema planteado en la sección anterior se vuelve un algoritmo que partiendo de U genere el conjunto S de forma tal que minimice la "varianza" de los $|Sx_iv_j|$ para cada x_i.

Con esto se logra maximizar la cantidad de individuos con distintos valores en cada propiedad.

1.3 Minimizando la varianza y criterios de éxito

En la sección anterior se menciona minimizar la "varianza" de forma un tanto ambigua. Cualquier métrica que logre describir las diferencias entre el conteo valores dentro de cada propiedad es válida.

Las dos fórmulas siguientes cumplen con dicho requisito. La primera calcula la desviación relativa del conteo de valores dentro de cada propiedad y luego las suma. La segunda hace lo mismo, pero utilizando la desviación absoluta. Cualquiera de ellas es una buena candidata para la minimización (Justificado en la sección 1.4).

$$DesviacionRelativa = \sum_{i=1}^{M} \sum_{j=1}^{|dom(x_i)|} \left(\sqrt[2]{\left(\frac{n}{|dom(x_i)|} - |Sx_iv_j| \right)^2} \right) / |dom(x_i)|$$

$$DesviacionAbsoluta = \sum_{i=1}^{M} \sum_{j=1}^{|dom(xi)|} \sqrt[2]{(\frac{n}{|dom(x_i)|} - |Sx_iv_j|)^2}$$

Fórmula 1- Funciones objetivo

Estas fórmulas se basan en los conceptos de desviación absoluta, desviación relativa y desviación estándar. Fórmulas similares, pero basadas en la varianza y otras métricas, son perfectamente plausibles y la más adecuada puede depender de que se esté intentando minimizar.

1.4 ¿Por qué esta estrategia?

Para clarificar cómo el minimizar alguna de las funciones objetivo anteriores puede cumplir con el requerimiento de una muestra tan diversa/democrática como sea posible, se hace necesario un ejemplo. Se utiliza un conjunto de datos similar al clásico de los pasajeros de la tragedia del Titanic (DAWSON, 1995). La población original puede visualizarse en la Figura 1:

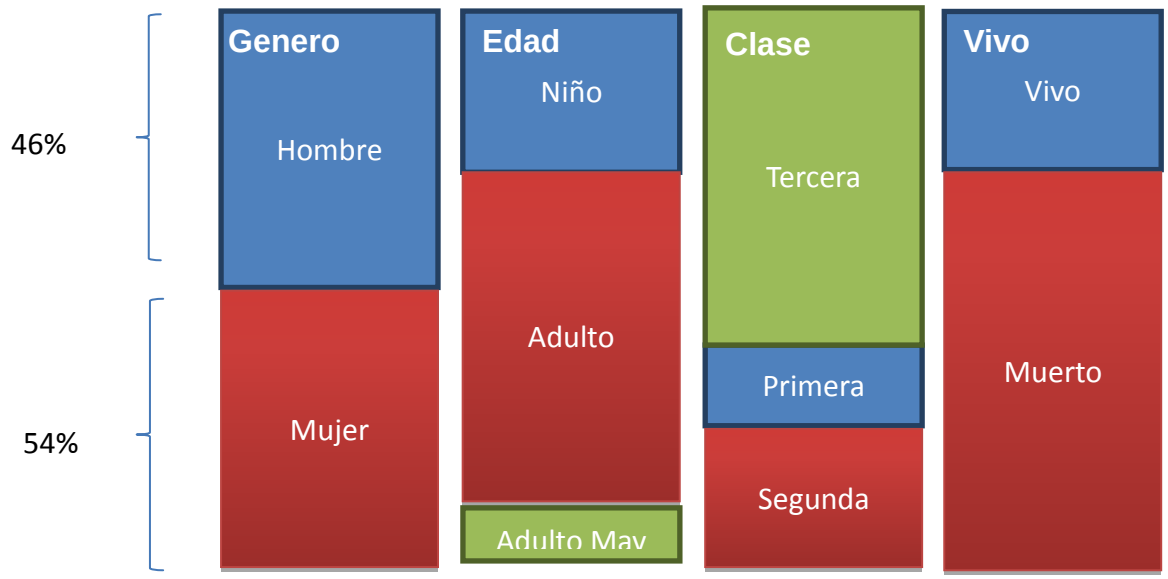

Figura 1 - Población inicial del Titanic.
Cada individuo cuenta con 4 propiedades: Género, Edad, Clase y Vivo. Los cuadros bajo cada propiedad indican los posibles valores que puede tomar dicha propiedad. El área representa el porcentaje de la población con dicho valor en la propiedad

En esta visualización, la altura de cada rectángulo es proporcional a la cantidad de elementos dentro de la población que poseen dicha propiedad. Por ejemplo, un 46% de la población es hombre, y un 54% mujer.

Si se crease una muestra aleatoria con un algoritmo como el S o el algoritmo de Floyd, (BENTLEY, 1987) la muestra resultante tendría una probabilidad alta de verse en esta visualización como una copia de la población, con pequeñas variantes en los porcentajes/altura de los rectángulos.

Dado que lo deseado es una muestra lo más diversa posible, una muestra ideal se debería ver según se aprecia en la figura 2

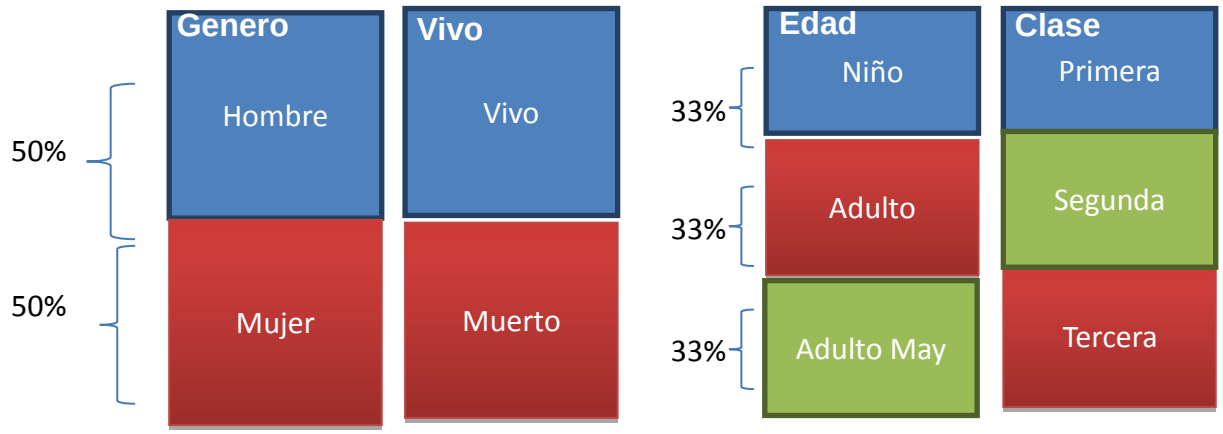

Figura 2 - Muestra ideal
En esta muestra, cada una de las propiedades tiene una distribución equitativa dentro de sus valores. Por ejemplo, hay un 50% de hombres y un 50% de mujeres.

Aplicando las fórmulas planteadas en la sección superior 1.3, se puede notar que conforme más se acerca una muestra a esta muestra ideal, menor es el valor resultante de la fórmula. Por ejemplo, asuma 3 muestras, A, B y C. Cada una de ella cuenta con 100 elementos, distribuidos según se muestra en la Tabla 1.

		A	B	C
Genero	Hombre	50	49	49
	Mujer	50	51	51
	DesviacionRelativa	0	0.04	0.04
Vivo	Vivo	50	50	50
	Muerto	50	50	50
	DesviacionRelativa	0	0	0
Edad	Niño	33	33	33
	Adulto	33	33	33
	Adulto Mayor	34	34	34
	DesviacionRelativa	0.030303	0.030303	0.030303
Clase	Primera	33	33	33
	Segunda	33	33	32
	Tercera	34	34	35
	DesviacionRelativa	0.030303	0.030303	0.090909
	DesviacionRelativaTotal	0.060606	0.100606	0.161212

Tabla 1 - Desviación Relativa

En este caso se puede ver cómo observarse cómo minimizar el valor de la función *desviación relativa* acercaría una muestra hacia el ideal. Incluso, en una muestra completamente ideal, el valor resultante de la función es 0.

La función *desviación absoluta* se desempeña de forma similar, como se puede observar en la Tabla 2

		A	B	C
Genero	Hombre	50	49	49
	Mujer	50	51	51
	DesviacionAbsoluta	0	2	2
Vivo	Vivo	50	50	50
	Muerto	50	50	50
	DesviacionAbsoluta	0	0	0
Edad	Niño	33	33	33
	Adulto	33	33	33
	Adulto Mayor	34	34	34
	DesviacionAbsoluta	1	1	1
Clase	Primera	33	33	33
	Segunda	33	33	32
	Tercera	34	34	35
	DesviacionAbsoluta	1	1	3
	DesviacionAbsolutaTotal	2	4	6

Tabla 2 - Desviación Absoluta

Se decidió utilizar la fórmula de *desviación relativa* como métrica final debido a que siempre brinda un número entre 0 y 1, lo cual permite fácilmente comparar el resultado entre distintas muestras. La *desviación absoluta* no posee dicho beneficio. Por facilidad, el resultado de dicha fórmula será llamado de ahora en adelante *Índice democrático*.

1.5 Variantes del problema

Anteriormente, se ha asumido que es igual de importante minimizar la varianza a lo largo de todas las propiedades de la población. Esto no es necesariamente cierto. Puede resultar necesario un mecanismo de priorizar la diversidad dentro de una propiedad comparada con la diversidad dentro de otra propiedad.

Por ejemplo, para los objetivos de un estudio hipotético, puede ser más relevante garantizar una muestra en la que haya exactamente 50% de mujeres y 50% de hombres, pero que la distribución del estado civil no sea relevante. Expresado de otra forma, la prioridad de minimizar la varianza en la propiedad género es mayor en este caso que la prioridad de minimizar la varianza de la propiedad estado civil.

El algoritmo debe poseer algún mecanismo para soportar esta priorización. Un mecanismo de pesos a cada variable probablemente sería lo óptimo. Las fórmulas objetivo planteadas anteriormente no contemplan priorización, por lo que deben ser extendidas.

1.6 Objetivos

1.6.1 Objetivo principal

Crear un algoritmo que partiendo de una población U de elementos, cada uno de los cuales posee una o más propiedades con un conjunto finito de valores discretos, retorne una muestra de tamaño n en la que la varianza de los valores $|Sxiv|$ sea la menor posible. (Por brevedad, será llamado como selección democrática de individuos)

1.6.2 Objetivos específicos

1. Crear un algoritmo que minimice el valor de las funciones objetivo.
2. Diseñar y ejecutar un experimento estadístico para comparar el desempeño de las funciones objetivo.
3. Extender al menos una de las funciones objetivo para soportar un mecanismo de priorización de propiedades.
4. Crear una librería de código libre que permita ejecutar el nuevo algoritmo bajo una plataforma con alto uso en la industria (i.e. JVM, CLR).
5. Comparar experimentalmente el algoritmo de selección democrática contra el algoritmo SRS en términos de rendimiento (en tiempo) y en su capacidad para minimizar la varianza de las propiedades

1.7 Hipótesis

Dado el problema planteado con anterioridad, se plantean las siguientes hipótesis:

1. Un algoritmo voraz es suficiente para resolver el problema
2. Minimizar el valor de las funciones objetivo o debe ser la meta de dicho algoritmo voraz.
3. Asignar pesos a cada propiedad, combinado con las métricas mencionadas anteriormente, es suficiente para permitir dar prioridad a la minimización de la diferencia de ciertas propiedades con respecto a otras.

4. El algoritmo SRS será significativamente más rápido que el algoritmo voraz propuesto, dado que realiza una selección aleatoria (y completamente desinformada) de los elementos.

1.8 Justificación

La *detección de anomalías* es vital en la industria moderna. Desde detección de fraudes hasta detección de fallas en procesos de manufactura, todas dependen de estas técnicas. Precisamente, el tema de esta tesis, surge como una necesidad planteada por una de las empresas que más contribuyen al PIB del país: Intel.

Sin ahondar en detalles, por motivos de confidencialidad, el proceso para determinar si una unidad dentro de Intel está compuesto de gran cantidad de pruebas, realizadas para cada uno de los procesadores y demás componentes creados en sus fábricas. Estos datos por unidad son almacenados por períodos de años, tanto para detección temprana de errores, como para análisis más profundo realizado meses o años después. La cantidad de datos es muy grande, por lo que las bases de datos gigantes, la minería de datos y las técnicas de la visualización de los mismos se vuelven una necesidad.

Claramente, la detección de fallas temprana es vital. Si un procesador progresa mucho en la línea de producción hasta que la falla en éste sea detectada, se ha perdido dinero. Tomando en cuenta el volumen de unidades que manejan las fábricas de Intel, esto puede representar fácilmente pérdidas de millones de dólares al año.

Hay técnicas modernas que permiten adelantar el momento en que se detecta una falla dentro de la línea de producción. Usualmente se basan en minería de datos, con modelos predictivos. Sin embargo, son demasiado costosas como para aplicarlas sobre todas y cada una de las unidades.

Estas técnicas solamente pueden usarse en la práctica sobre un subconjunto de todas las unidades, por lo que este subconjunto debe ser cuidadosamente seleccionado. Sin embargo, se presentan dos situaciones que pueden complicar el proceso de selección de esta muestra:

- **Diversidad de productos:** Intel maneja múltiples tipos de productos, CPUS, chipsets, módems y múltiples variantes de cada uno de ellos. Si bien es factible desarrollar un algoritmo que permita decidir si una unidad de estos tipos es sospechosa, el implementarlo para todos y cada una de las variantes de productos es un proceso que durará años. Dadas las potenciales ganancias de detectar fallas antes en el proceso de manufactura, se requiere algún método de seleccionar unidades sin estos algoritmos de *sospecha*, al menos durante el tiempo en que estos son desarrollados.

- **Falta de estandarización:** Existen algunos productos que ya cuentan con soporte para una elección inteligente de las unidades a seleccionar. Sin embargo, los datos generados no están accesibles de una forma estándar a lo largo de todos los productos. Mientras en algunos productos están en la base de datos A, para otros están en la base de datos B.

Actualmente se está trabajando en estandarizar e implementar este proceso a lo largo de toda la línea de productos. Sin embargo, mientras esta adaptación del proceso de manufactura toma lugar (probablemente años), aún hace falta realizar una selección semi-inteligente de los datos por muestrear.

La técnica de selección de individuos democrática planteada en el transcurso de este documento, surge como una propuesta para atacar este problema. Si tiene éxito, podrá ser aplicada dentro de Intel, así como dentro de otras compañías que enfrentan situaciones de la misma índole.

Capitulo 2 Marco Teórico

El problema planteado es muy similar al muestreo tradicional, en el que se intenta calcular valores de la población partiendo de una muestra. Dada la importancia del muestreo para gran cantidad de disciplinas científicas, hay una plétora de algoritmos para llevarlo a cabo (TILLE, Sampling Algorithms, 2006). Sin embargo, ninguno busca realmente lo mismo que el algoritmo de selección de individuos democrática.

Algunos de los algoritmos similares, o que de alguna forma resultan para la realización del proyecto se presentan las siguientes secciones:

2.1 Muestreo aleatorio Simple - Floyd

El muestreo aleatorio simple es el muestreo más familiar y comúnmente utilizado. Dada una población inicial, se escogen aleatoriamente N elementos. Cada uno de ellos tiene exactamente la misma probabilidad de ser escogido. (PATHAK, 1988)

Si un individuo puede ser seleccionado más de una vez durante el muestreo, entonces se realizó un muestreo *con reemplazo* (es decir, se "reinserta" el individuo dentro de la población luego de que es seleccionado)

Por el contrario, si los individuos no pueden ser seleccionados más de una vez, se llevó a cabo un muestreo *sin reemplazo*.

Hay múltiples formas de implementar el muestreo aleatorio sencillo. Una particularmente eficiente es conocida como el *algoritmo de Floyd, cuyo pseudocódigo se muestra en la figura 3* :

```
public static <T> Set<T> randomSample4(List<T> items, int m){
   HashSet<T> res = new HashSet<T>(m);

   int n = items.size();

   for(int i=n-m;i<n;i++){
      int pos = rnd.nextInt(i+1);
      T item = items.get(pos);

      if (res.contains(item))
      {
         res.add(items.get(i));
      }
      else
      {
         res.add(item);
      }
```

```
    }
    return res;
}
```

Figura 3 - Pseudocódigo del algoritmo de Floyd

Este algoritmo es el más eficiente y rápido conocido para ejecutar un muestreo simple. Está diseñado para estructuras de datos de acceso aleatorio (BENTLEY, 1987). Existen algoritmos equivalentes, si bien más lentos, que funcionan sobre listas enlazadas (VITTER, 1985) así como algoritmos equivalentes para una cadena de datos en "streaming" (GROTHAUS, 2007) finita.

2.1.1 Relación con la investigación

Dado que el muestreo aleatorio sencillo es totalmente aleatorio, la muestra resultante es muy similar a la población, dentro de cierto margen de error. Esto lo hace poco útil para el requerimiento planteado.

Sin embargo, resulta útil como un punto de referencia en cuanto a velocidad y en cuanto a varianza de las propiedades. Por lo tanto, será utilizado como línea base para comparar contra el algoritmo planteado. La implementación de Floyd será utilizada para llevar a cabo esta comparación.

2.2 Muestreo estratificado

El *muestreo estratificado* intenta minimizar el margen de error respecto a la distribución de la población que se encuentra en el muestro aleatorio sencillo (como en el algoritmo de Floyd). Es un proceso bastante sencillo. La variante más común es el *muestreo estratificado proporcional*. (YATES & Grundy, Selection without replacement from within strata with probability proportional to size, 1953)

1. Sea N el tamaño de la población.
2. Dividir la población en estratos E_1, E_2 ... E_M (e.g. hombre/mujer, soltero/casado/divorciado)
3. Calcular la proporción de cada estrato dentro de la población (e.g. 40% de hombres, 60% de mujeres). Sean estas proporciones E_1P, E_2P ... E_mP
4. Ejecutar muestreo aleatorio sencillo (como Floyd) dentro de cada estrato, hasta obtener $E_kP * N$ individuos
5. Esto brinda una muestra con exactamente las mismas proporciones que la población.

Una variante es el *muestreo estratificado desproporcional*, en el cual los valores de EkP no son relativos al tamaño del estrato respecto a la población, sino asignados por algún otro criterio

(Generalmente a criterio de la persona generando la muestra, aunque puede ser un proceso más automatizado). (TILLE, Sampling Algorithms, 2006)}

2.2.1 Relación con la investigación

El muestreo estratificado desproporcional es bastante cercano al requerimiento planteado. Tiene un inconveniente. Los estratos están conformados por valores de una sola propiedad. Esto lo vuelve no factible para el problema planteado.

Se debe mencionar que este problema tiene una solución aparentemente trivial: crear combinaciones de propiedades (E.g. crear estratos como Hombre-Soltero, Mujer-Divorciada, etc.)

Sin embargo, esto presenta una complicación que lo vuelva poco factible en la práctica. Se da un crecimiento combinatorio de los estratos respecto a la cantidad de propiedades y el número de valores posibles dentro de cada uno de ellos. Esto lo vuelve inmanejable incluso para conjuntos de datos con relativamente pocas propiedades y valores de datos, más aun para conjuntos con decenas de ellos.

Otro problema complejo que surge al intentar utilizar el muestreo estratificado para atacar el problema planteado es cómo calcular las proporciones adecuadas para cada estrato. Por estos motivos, se descartó como opción para el problema por resolver.

2.3 Muestreo a 2 fases

Este algoritmo de muestreo es de los más utilizados. Originalmente, fue creado por Neyman colaborando con Milton Friedman. (NEYMAN, 1938)

Generalmente, resulta muy útil en detección de anomalías. Usualmente, hay una propiedad de los individuos que es costosa de calcular (por ejemplo, si es anómalo o no) así como una propiedad rápida (la que causa sospecha de anomalías) de calcular que está altamente correlacionada con la anterior. Partiendo de esto, los pasos del algoritmo son los siguientes:

1. Calcular la variable rápida para toda la población
2. Crear estratos basados la variable rápida de calcular
3. Realizar muestreo estratificado, proporcional o desproporcional, basado en dichos estratos.

Esta es la variante de muestreo que más se acerca a lo planteado en el requerimiento original, con 2 problemas que impiden su utilización:

1. Las desventajas ya mencionadas del muestreo estratificado desproporcional.
2. Dependencia de una variable de cálculo rápido. (en la justificación se explica el motivo por el cual no se cuenta con esta)

2.3.1 Relación con la investigación

Si bien no sirve exactamente para solucionar el problema planteado, la técnica de utilizar 2 o más fases bien puede utilizarse para la selección democrática de individuos.

Por ejemplo, puede usarse una primera fase que cree una muestra aceptablemente democrática y luego utilizar una variante del procedimiento de Yates para realizar mejoras a la muestra.

2.4 Procedimiento de Yate

El *procedimiento de Yate* intenta conseguir una muestra más balanceada, la misma meta que el algoritmo estratificado proporcional: el margen de error respecto a la distribución de la población que se encuentra en el *muestro aleatorio sencillo*. El procedimiento en si consiste en lo siguiente (YATES, A Review of Recent Statistical Developments in Sampling and Sampling Surveys, 1946)

1. Seleccione una muestra (SRS, estratificada etc.) de tamaño N.
2. Seleccione un nuevo individuo utilizando el mismo método.
3. Si el individuo contribuye a mejorar el balance de la muestra, reemplácelo por el individuo que afecte de forma más negativa el balance de la muestra.
4. Repetir paso 2 y 3 hasta llegar a M iteraciones o hasta que las mejoras en el balance sean triviales o cíclicas.

2.4.1 Relación con la investigación

Este procedimiento se vuelve relevante para el problema en investigación porque un método similar puede ser utilizado para mejorar la selección democrática de individuos.

Por ejemplo, luego de completar la muestra, si aún no es lo suficientemente “democrática”, se puede comenzar a sustituir elementos que estén sobre-representados por aquellos que estén supra-representados (es decir, sustituir los elementos de los que hay muchos por aquello de los que hay pocos). Varias repeticiones de este proceso, hasta dejar de observar ganancias, pueden potencialmente mejorar la muestra inicialmente creada de una forma sencilla y utilizando las mismas fórmulas objetivo.

2.5 Muestreo balanceado por programación lineal

Cualquier proceso de muestreo deseado puede ser teóricamente resuelto usando programación lineal. (TILLE, Balanced sampling by means of cube method, 2010) (WYNN, 1977)

Por ejemplo, se puede definir el costo de una muestra cualquier como $C(muestra)$. El costo es pequeño si la muestra está cerca de ser balanceada (en casos tradicionales) o democrática (en el caso presentado en este documento).

Se puede resolver de esta forma el problema, aplicando un procedimiento como Simplex (REVELIOTIS, 1997) u otros algoritmos de programación lineal, con una función objetivo que minimice el costo esperado.

Este método funciona muy bien con datos limitados. Sin embargo, el problema para utilizarlo de una forma más generalizada implica enumerar todas las posibles muestras. Dado un conjunto de elementos $U = \{u1, u2, \dots uN\}$ sobre el que se desea tomar una muestra de tamaño P, esto implica enumerar $\frac{N!}{P!(N-P)!}$ posibles muestras. Esta explosión combinatoria lo torna poco práctico, ya que el proceso de enumerar tantas muestras se vuelve prohibitivamente lento para valores relativamente pequeños de N y P.

Este método resultaba particularmente atractivo dada su capacidad de (teóricamente) resolver el problema del muestreo democrático. Sin embargo, la limitación de ser usado para escenarios pequeños lo vuelve impráctico para el escenario descrito en la introducción, el cual requiere al menos miles de elementos por muestra.

Existe una variante llamada el *método del cubo* (TILLE, Balanced sampling by means of cube method, 2010), la cual permite utilizar la programación lineal sin realizar la enumeración de todas las combinaciones posibles. Logra reducir las muestras a analizar planteándolas como un cubo en el que cada vértice es una muestra y simplemente evita recorrer los vértices que cumplen ciertos criterios.

Las muestras se convierten a vértices del cubo mediante los individuos que la conforman. Por ejemplo, si en una en población de 3 elementos hay una muestra que posee el primer y el tercer elemento, esto se representa como 101, lo cual al ser graficado en un plano tridimensional crea un punto. Al realizar el mismo proceso para todas las posibles muestras, se crea un cubo. Claramente, si se trata de una población mayor, deja de ser un tradicional cubo y se vuelven hipercubos (WEISSTEIN, n.d.).

2.5.1 Relación con la investigación

El método del cubo resulta útil en casos en los que el objetivo sea crear una muestra balanceada respecto a la población original, sin embargo, para el muestreo democrático en sí no resulta útil directamente (dado que las variantes actuales crean muestras balanceadas, no democráticas).

Sin embargo, para una futura investigación, resultaría interesante considerar si se puede realizar algo similar basado en una figura n-dimensional que no sea un hipercubo. Esta figura podría ser construida no con base en los individuos discretos, sino en los posibles valores que puedan tomar las distintas propiedades de la población. Asimismo, no tomaría valores de 0 o 1, sino que tomarían valores equivalentes al total de elementos dentro de la muestra que posean dicha propiedad.

Capitulo 3 Solución implementada

La solución propuesta consiste en un algoritmo voraz implementado en la plataforma .NET. Dado que se implementaron múltiples versiones y variantes del algoritmo, y que a su vez éstas debían ser comparadas contra el algoritmo SRS, los algoritmos están escritos sobre una plataforma descrita en la próxima sección.

3.1 Plataforma

3.1.1 Clases POCO

Se creó una sencilla clase POCO[2] para ser utilizada como uno de los individuos de la muestra. El algoritmo en sí es genérico y funciona con cualquier tipo de objeto, se incluye aquí esta clase simplemente a forma de ejemplo en la Figura 4.

```
public class TitanicSurvivor
   {
      public int Id { get; set; }
      public string Country { get; set; }
      public int PassengerClass { get; set; }
      public int AgeGroup{ get; set; }
      public bool Survived { get; set; }

   }
```

Figura 4 - Clase POCO de ejemplo

Representa uno de los pasajeros de la tragedia del titanic

3.1.2 Conjuntos Atómicos

Los conjuntos atómicos (implementados utilizando la clase POCO AtomicSampleSet) representan un conjunto de instancias de una clase, y contienen ciertas estadísticas sobre las mismas. Concretamente, estas estadísticas son los valores para todos los $|Sx_i v_j|$ dentro del conjunto.

Los conjuntos atómicos permiten 3 operaciones básicas. Agregar un elemento, remover un elemento y transferir un elemento hacia otra conjunto atómico. En cualquiera de las 3 operaciones, las estadísticas son actualizadas automáticamente.

```
public class AtomicSampleSet<T>
   {
      public LinkedList<T> Results { get; set; }
      public CollectionStatistics<T> Statistics { get; set; }
```

[2] POCO: acrónimo en inglés para Plain Old CLR Object. Es decir, una clase que representa simples objeto con Atributos y Métodos, implementada en algún lenguaje de la plataforma .NET.

```
    public void Add(T testResult)       { … }

     private void Remove(LinkedListNode<T> testResult) { …  }

     public int Transfer(AtomicSampleSet<T> target, IBCCommand<T> removeFunc)
     {// Implementation here }
  }
```

Figura 5 - Set Atómico

3.1.3 Muestreador

Principal encargado de realizar el muestreo. Posee 2 conjuntos atómicos. Uno representa los individuos que no están en la muestra, y otro los que sí están en ella. El primero inicia siendo la población. El segundo termina siendo la muestra.

El muestreador se encarga de mover elementos de un set a otro. Sin embargo, el muestreador no decide cuales elementos mover. Simplemente ejecuta una serie de comandos de transferencia que le son provistas por las estrategias de muestreo. Estos comandos básicamente indican qué elemento(s) mover y en qué dirección. De forma abreviada, ambas clases se muestra en las Figuras 6 y 7.

```
public class Sampler<T> // Muestreador
  {
     public AtomicSampleSet<T> NotInSample { get;private set; }
     public AtomicSampleSet<T> InSample { get; private set; }
     public IBCSampleStrategy<T> SampleStrategy { get; set; }
           public bool Sample()       { … }
  }
```

Figura 6 - Muestreador

```
public interface IBCCommand<T> // Comando
  {
     bool Evaluate(T testResult);
     bool IsDefaultDirection { get; set; }
  }
```

Figura 7 - Interfaz Comando

3.1.4 Estrategia de muestreo

Es a estas alturas, en que se realiza el trabajo más laborioso. Las estrategias de muestreo determinan una serie de comandos para mover elementos dentro o fuera de la muestra. Puede haber múltiples implementaciones.

Por ejemplo, una de ellas puede simplemente mover los primeros N elementos de la población hacia la muestra y terminar. Otra estrategia puede implementar el algoritmo SRS, o el muestreo estratificado o cualquier otro algoritmo similar.

En este caso, se implementaron el SRS y múltiples variantes de un algoritmo voraz. Estas estrategias deben cumplir con la interfaz presentada en la figura 8. (Nótese que la secuencia de comandos es una secuencia perezosa).

```
public interface IBCSampleStrategy<T>
{
   Sampler<T> Sampler { get; set; }
   IEnumerable<IBCCommand<T>> CommandSequence();
}
```

Figura 8 - Interfaz Estrategia De Muestreo

3.1.5 Resumen de la plataforma

Esta plataforma permite analizar y comparar el desempeño de distintos algoritmos de muestreo. La siguiente sección detallará las estrategias implementadas.

3.2 Estrategias de muestreo

Las 2 variantes básicas de las estrategias transfieren elementos desde la población hacia la muestra. Para establecer que elemento se transfiere, se utilizan 2 fórmulas matemáticas ligeramente distintas:

$$sea\ Rx_iv_j = \begin{cases} \dfrac{\dfrac{n}{|dom(x_i)|} - |Sx_iv_j|}{|Ux_iv_j|}, & si\ |Ux_iv_j| = |Sx_iv_j| \\ -\infty, si\ no \end{cases}$$

Fórmula 2 - Muestreo 1

$$sea\ Rx_iv_j = \begin{cases} \dfrac{\dfrac{n}{|dom(x_i)|} - |Sx_iv_j|}{|dom(x_i)|}, & si\ |Ux_iv_j| = |Sx_iv_j| \\ -\infty, si\ no \end{cases}$$

Fórmula 3 - Muestreo 2

Tanto para la fórmula de muestreo 1 como para la 2, el algoritmo es el siguiente:

Mientras el tamaño de la muestra sea inferior al tamaño de deseado, determinar para cual x_iv_j es mayor el Rx_iv_j y transferir un elemento que posea la propiedad x_iv_j hacia la muestra. El pseudocódigo sería similar al encontrado en la Figura 9

```
while (muestra.Tamaño < tamañoMetaDeLaMuestra)
{
   tuplasDeXiVjConCosto = listaDeXiVj.Map(xivj => new { xivj, Rxivj = calcularCosto(xivj) });

   tuplaConElMayorCosto = tuplasDeXiVjConCosto.ItemConMayorCosto();

   poblacion.TransferTo(muestra, tuplaConElMayorCosto);
}
```

Figura 9 - Pseudocódigo de la estrategia de muestrro

3.3 Variantes

3.3.1 Compensación

Hay escenarios en los cuales, por las características de la población, no haya suficientes individuos en la población para satisfacer una muestra perfectamente distribuida de tamaño n. Es decir, escenarios en los cuales $\exists v_j \mid \frac{n}{|dom(x_i)|} < \ vj$

En estos casos, utilizar $\frac{n}{|dom(x_i)|}$, tanto para la fórmula 1 como para la fórmula 2 puede no ser la mejor solución.

La *compensación* intenta evitar estos escenarios compensando el valor ideal representado por $\frac{n}{|dom(x_i)|}$ y la realidad de la población, reemplazando el $\frac{n}{|dom(x_i)|}$ por min $(\frac{n}{|dom(x_i)|}, |Ux_i v_j|)$

Una característica positiva de la compensación es que, en casos en que no se presente este escenario, se comporta de forma idéntica a la fórmula anterior.

De esta forma, las fórmulas anteriores se ven transformada en las siguientes (Fórmula 4 y 5).

$$sea\ Rx_i v_j = \begin{cases} \dfrac{\min (\frac{n}{|dom(x_i)|}, |Ux_i v_j|) - \ |Sx_i v_j|}{|Ux_i v_j|}, & si\ |Ux_i v_j| = \ |Sx_i v_j| \\ -\infty, si\ no \end{cases}$$

Fórmula 4 - Muestreo 1 con compensación

$$sea\ Rx_i v_j = \begin{cases} \dfrac{\min\left(\dfrac{n}{|dom(x_i)|}, |Ux_i v_j|\right) - |Sx_i v_j|}{|dom(x_i)|}, & si\ |Ux_i v_j| = |Sx_i v_j| \\ -\infty, si\ no \end{cases}$$

Fórmula 5 - Muestreo 2 con compensación

3.3.2 Corrección

La *corrección* es una idea que surge basada en el procedimiento de Yate (YATES, A Review of Recent Statistical Developments in Sampling and Sampling Surveys, 1946). Básicamente, luego de que la muestra llega al tamaño deseado, se procede a cambiar elementos que tengan valores sobre-representados por valores que estén sub-representados. Para determinar cuáles elementos están sub-representados, se utiliza las fórmulas presentadas anteriormente. Para las sobre-presentadas, se utiliza una variante (la cual es básicamente la contraparte)

$$sea\ Cx_i v_j = \frac{|Sx_i v_j| - \frac{n}{|dom(x_i)|}}{|Ux_i v_j|}$$

Fórmula 6 - Muestreo 1 con correción

$$sea\ Cx_i v_j = \frac{|Sx_i v_j| - \frac{n}{|dom(x_i)|}}{|dom(x_i)|}$$

Fórmula 7 - Muestreo 2 con corrección

Básicamente se reemplaza el valor que posea el mayor $Cx_i v_j$ por el que posea el mayor $Rx_i v_j$

3.3.3 Corrección y compensación

La última variante es un algoritmo que combine tanto corrección como compensación. Es decir, se ejecuta una selección estándar con compensación y luego se ejecuta una fase de corrección. Las fórmulas son idénticas a las fórmulas 4,5,6 y 7.

3.3.4 Pesos

Para poder priorizar las propiedades a democratizar, se implementó una variante de las fórmulas. Estas variantes utilizan pesos para determinar cuál propiedad es más importante. Se presenta a continuación las fórmulas 1 y 2, con la variante del peso agregado. Las fórmulas con compensación y corrección siguen el mismo principio.

$$sea\ Rx_i v_j = \begin{cases} \dfrac{\dfrac{n}{|dom(x_i)|} - |Sx_i v_j|}{|Ux_i v_j|} \Big/ w_i, & si\ |Ux_i v_j| = |Sx_i v_j| \\ -\infty, si\ no \end{cases}$$

Fórmula 8 - Muestreo 1 con pesos

$$sea\ Rx_i v_j = \begin{cases} \dfrac{\dfrac{n}{|dom(x_i)|} - |Sx_i v_j|}{|dom(x_i)|} \Big/ w_i, & si\ |Ux_i v_j| = |Sx_i v_j| \\ -\infty, si\ no \end{cases}$$

Fórmula 9 - Muestreo 2 con pesos

Al dividir por un peso único, se altera intrínsecamente la prioridad de las propiedades. Para evitar que el valor de $Rx_i v_j$ se salga de [0,1], el valor de w_i debe encontrarse entre $[1, \infty]$. Entre más cercano a uno esté el peso de una propiedad, más prioridad tendrá.

3.4 Resumen

Utilizando la fórmula una y dos con sus respectivas combinaciones de corrección y composición se obtienen finalmente 8 variantes de la fórmula de minimización utilizada por el algoritmo voraz. Finalmente, se agrega la capacidad de priorizar propiedades utilizando un peso por cada una de ellas.

Capitulo 4 Experimentos

Con los datos ya estructurados y limpios, puede procederse con múltiples experimentos para comparar el desempeño de las 8 variantes del algoritmo contra el SRS clásico. La variante básica del experimento consiste sencillamente en partir de una población con N individuos, X propiedades y ejecutar el algoritmo sobre dicha población para obtener una muestra de tamaño M.

Todos los experimentos repetirán este proceso. Básicamente lo que variará serán los valores de estas 3 variables. M resulta poco interesante por sí sola, por lo que se variará en su lugar la variable M/N.

Todo experimento debe ser ejecutado tanto para el algoritmo SRS como para el algoritmo democrático.

Los resultados a tomar para cada experimento serán los siguientes:

- Valor de ejecutar la función objetivo al finalizar el algoritmo. Esto permite caracterizar la eficiencia de la democratización de los datos. Entre más cercano a 0 más efectivo fue. El algoritmo democrático debería estar por debajo del SRS.

- Tiempo total de ejecución.

4.1 Datos de prueba

Este algoritmo toma como entrada una población y genera una muestra determinística. Por lo tanto, hace falta un gran número de poblaciones, cada una de ellas, muy diversas para probarlo.

Por lo tanto, se creó un generador de poblaciones. Este básicamente genera una población de N elementos de la clase TitanicSurvivor (descrita en la sección anterior). Las poblaciones generadas utilizan generadores aleatorios para asignar el valor de cada una de las propiedades de los individuos.

Para evitar que sigan una distribución uniforme, el generador aleatorio sigue una distribución binomial. Asimismo, cada propiedad tiene su propio generador, para el cual el valor P es seleccionado aleatoriamente.

Utilizando este generador, se termina obteniendo poblaciones en las cuales la distribución de cada factor es muy distinta a la de otras poblaciones, lo cual es realmente lo relevante en este ejercicio.

4.2 Experimentos concretos

El conjunto de pruebas aquí discreto está destinado a proveer un panorama general de la eficacia del algoritmo. Están basadas en alterar una a una las variables, manteniendo las otras 2 en condiciones *ceteris paribus.*

La lista de los experimentos ejecutados es la siguiente:

Experimento	**Descripción**
1	*Comparar las 8 versiones implementadas del algoritmo*
2	Caracterizar el desempeño del algoritmo para distintos N, *ceteris paribus* el resto de variables
3	Caracterizar el desempeño para distintos valores de X, *ceteris paribus* el resto de variables
4	Caracterizar el desempeño para distintos valores de M/N, *ceteris paribus* el resto de variables
5	Caracterizar la efectividad de los pesos para priorizar las prioridades.

Tabla 3 – Lista de experimentos

4.2.1 Experimento 1

Descripción: Comparar las 8 versiones implementadas del algoritmo

En este experimento, se ejecutaron las 8 versiones del algoritmo 10000 veces. Cada una de estas 10000 veces, se utilizó una población distinta, generada como se describió anteriormente. Luego se sacó el promedio simple del índice democrático de cada uno de los algoritmos. Los 2 más bajos se muestran resaltados a continuación. (Al usar una muestra de 10000, existe un 99% de probabilidad de que el valor real de este promedio sea cierto para cada uno de los algoritmos)

Valores	
Promedio de Fórmula 1 - NoCompensación-NoCorrección	217.2431829

Promedio de Fórmula 1 - NoCompensación-Corrección	217.2948055
Promedio de Fórmula 1 - Compensación-NoCorrección	212.0037933
Promedio de Fórmula 1 - Compensación-Corrección	212.7291088
Promedio de Fórmula 2 - NoCompensación-NoCorrección	367.7436243
Promedio de Fórmula 2 - NoCompensación-Corrección	369.2193255
Promedio de Fórmula 2 - Compensación-Corrección	217.191881
Promedio de Fórmula 2 - Compensación-NoCorrección	217.2947029
Promedio de Fórmula SRS	370.8589448

Tabla 4 - Resultados experimento 1

4.2.2 Experimento 2

Descripción: Caracterizar el desempeño del algoritmo para distintos N, *ceteris paribus* el resto de variables.

- N tiene dentro de [1000, 3500, 6000, 8500, … 51000]
- X se encuentra fijo en 5.
- M se encuentra fijo en 512

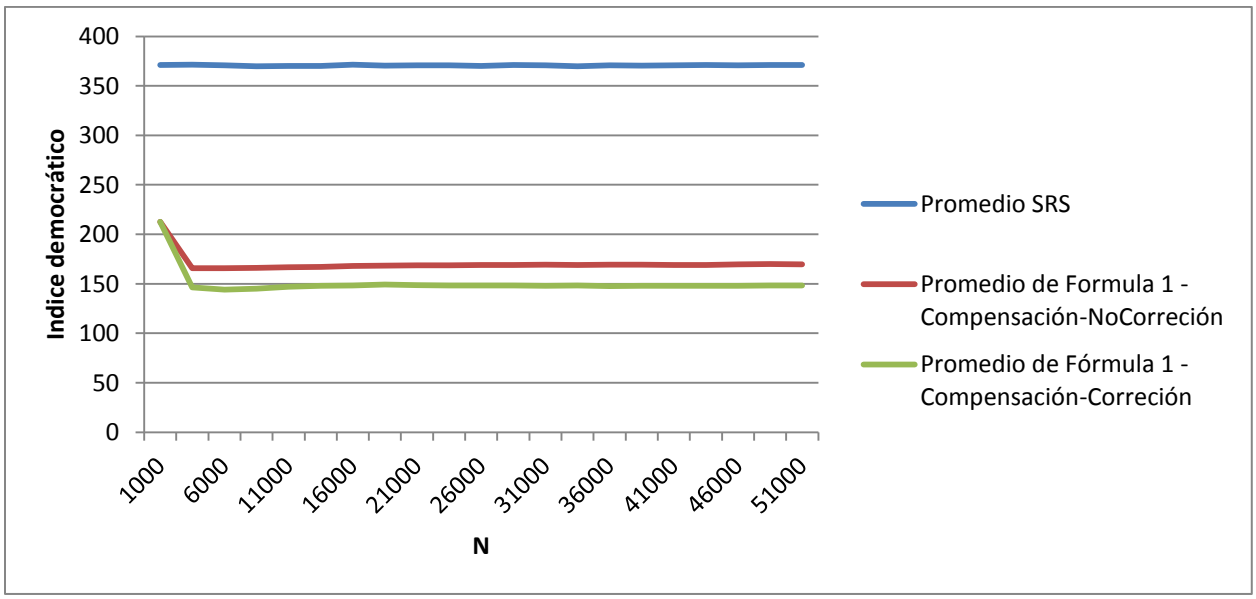

Figura 10 - Resultado Experimento 2 - índice democrático

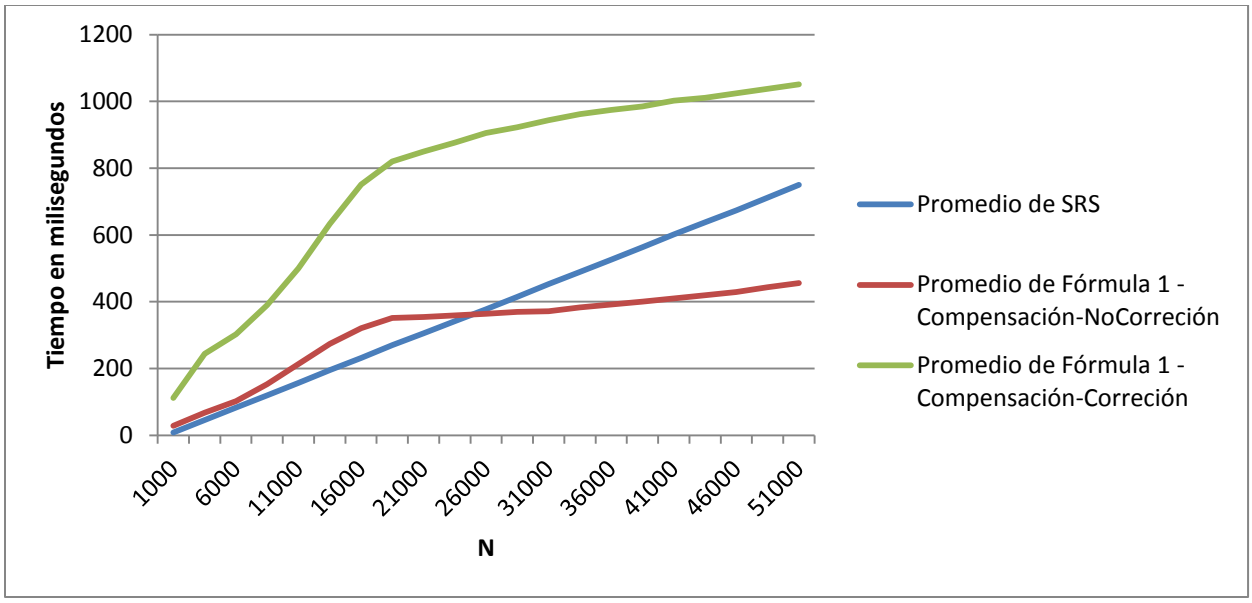

Figura 11 - Resultado Experimento 2 - tiempo en milisegundos

4.2.3 Experimento 3

Descripción: Caracterizar el desempeño para distintos valores de X, *ceteris paribus* el resto de variables.

- X posee valores dentro de [1, 2, 3, 4]
- M tiene un valor fijo de 512
- N tiene un valor fijo de 1000

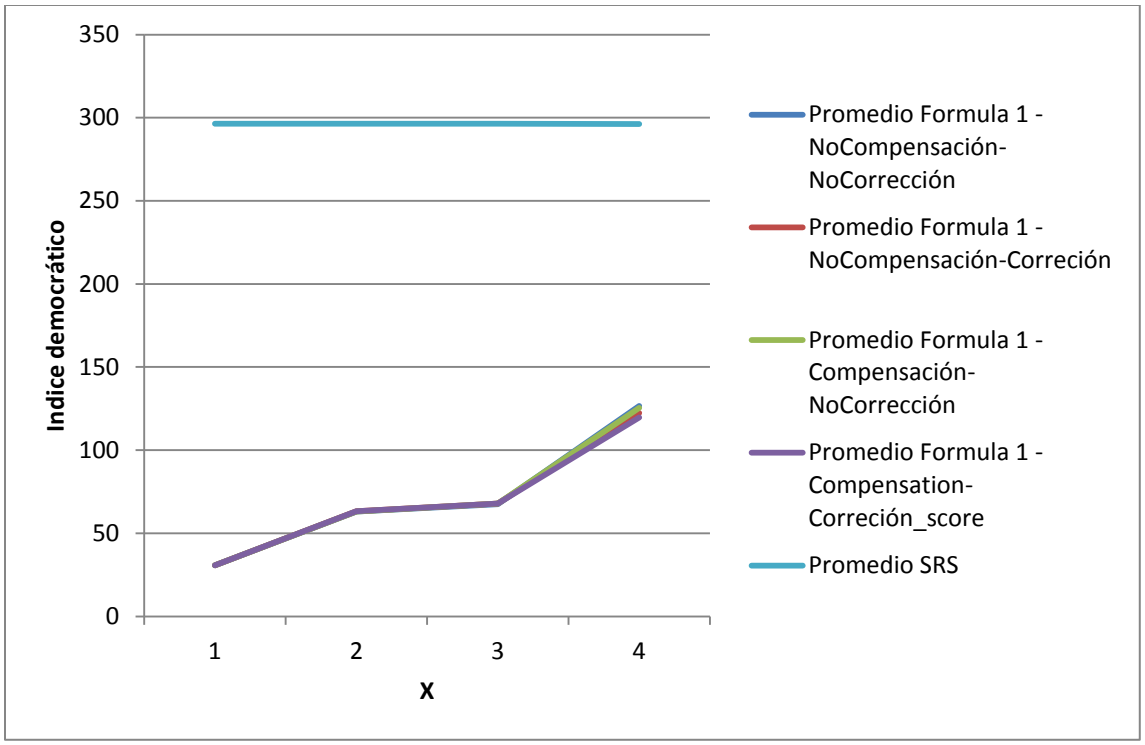

Figura 12 - Resultado Experimento 3 - índice democrático

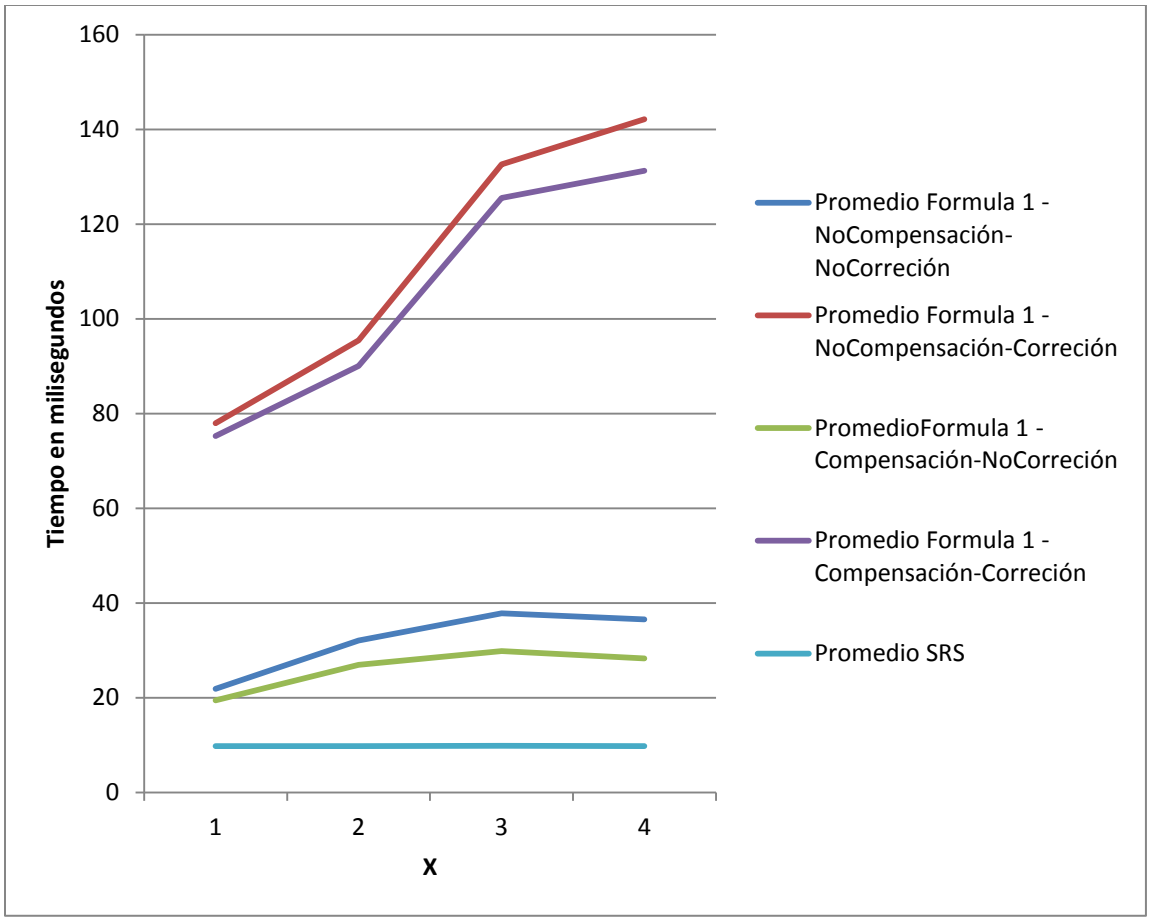

Figura 13 - Resultado Experimento 3 - tiempo en milisegundos

4.2.4 Experimento 4

Descripción: Caracterizar el desempeño para distintos valores de M/N, *ceteris paribus* el resto de variables.

- M/N tiene valores de [1%, 11%, 21% … 91%]
- M se encuentra fijo en 10000
- X se encuentra fijo en 4

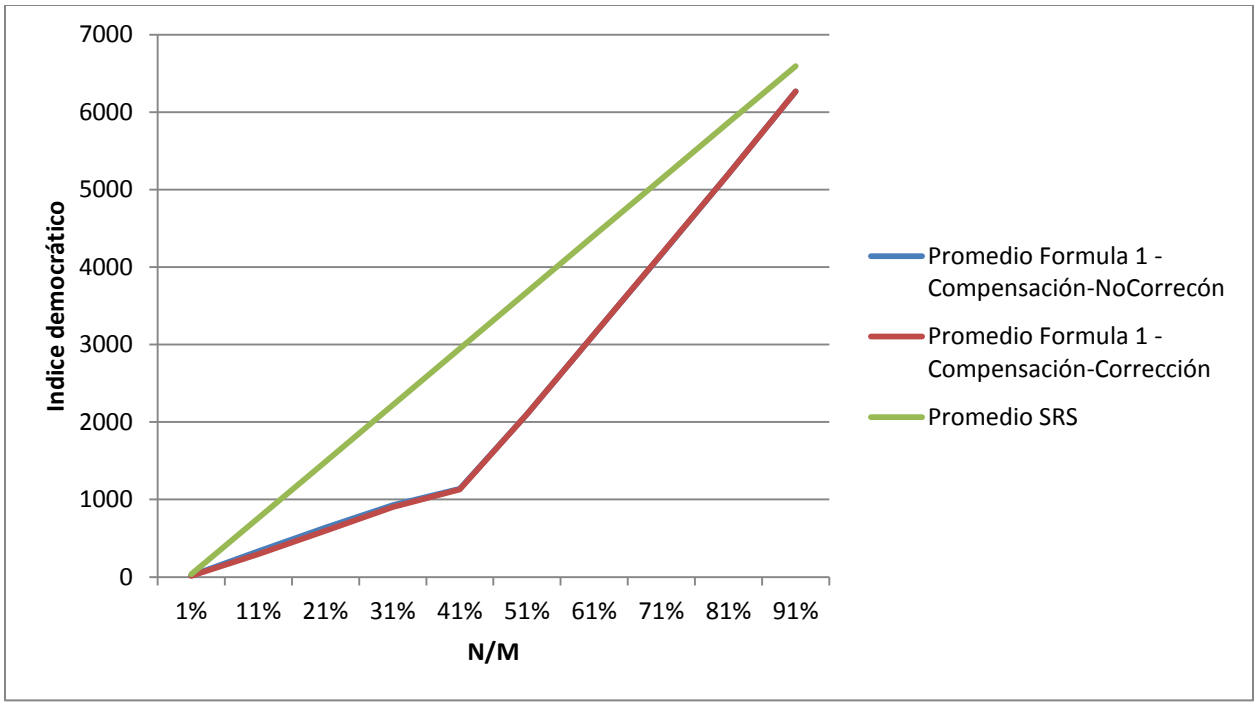

Figura 14 - Resultado Experimento 4 - índice democrático

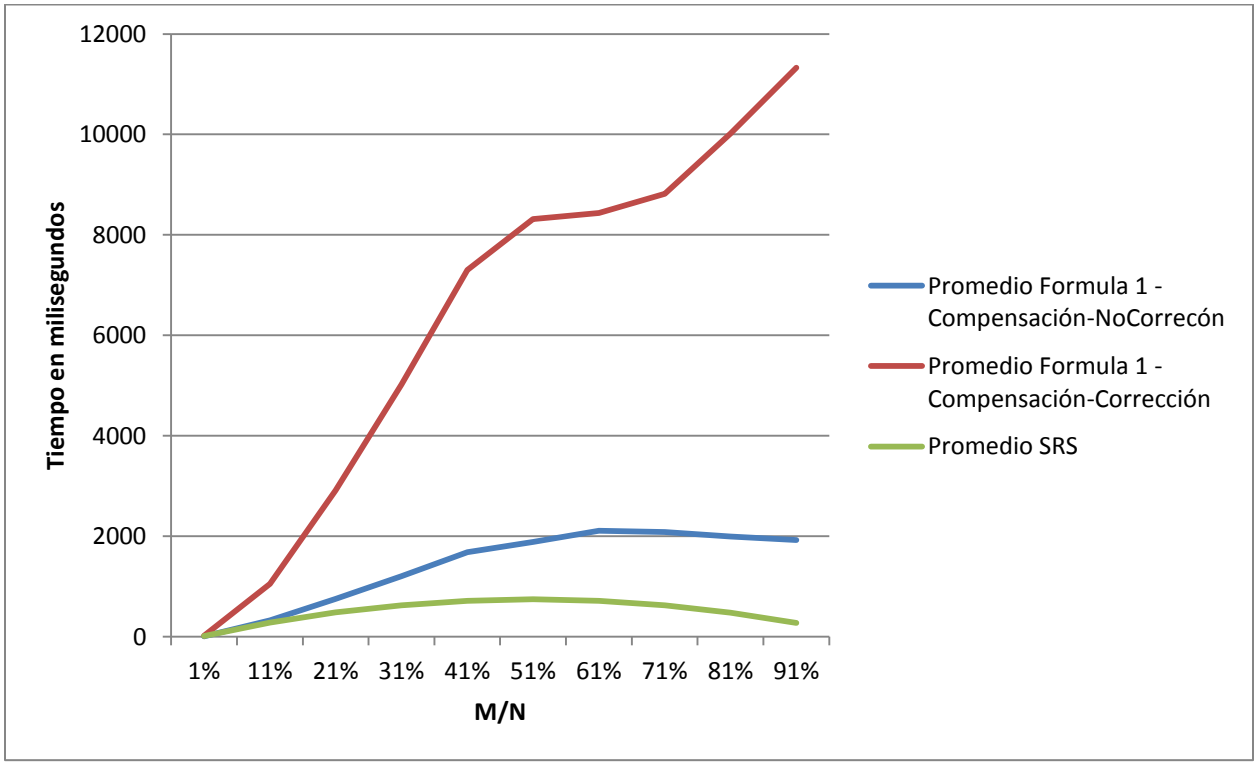

Figura 15 - Resultado Experimento 4 - tiempo en milisegundos

4.2.5 Experimento 5

Descripción: Caracterizar la efectividad de los pesos para priorizar las prioridades. Se asignó un peso de 1 a todas las variables, excepto a uno. El peso de esta última variable varía. Finalmente, se calcula el índice democrático pero no sobre todos los factores, sino solo sobre la variable en la cual el peso fluctua.

- M se encuentra fijo en 512
- N se encuentra fijo en 10000
- X se encuentra fijo en 4
- W1 toma valores dentro de [0.1,0.3,0.5,… 2.9]
- W2,W3, W4 se encuentran fijos en 1

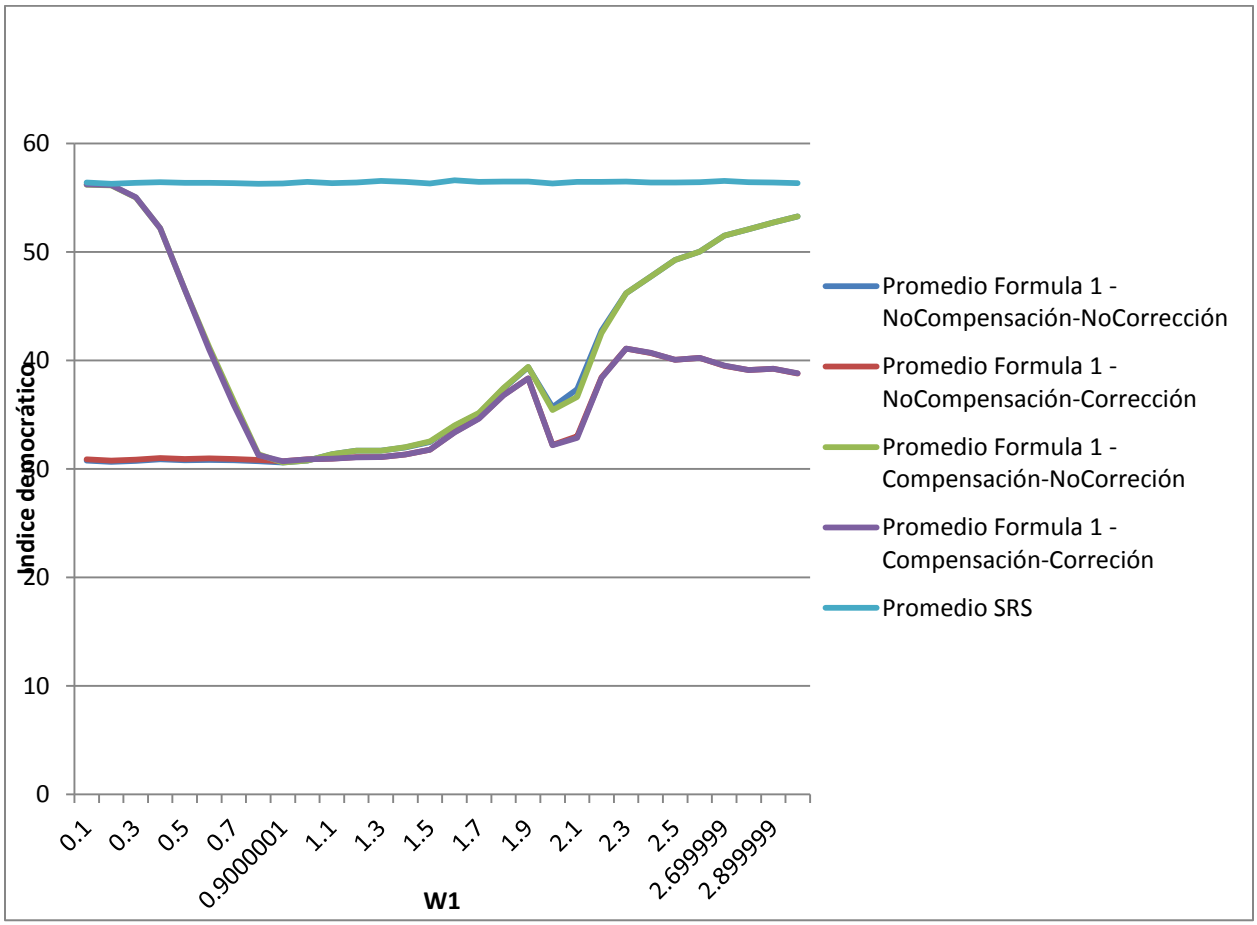

Figura 16 - Resultado Experimento 5 - índice democrático

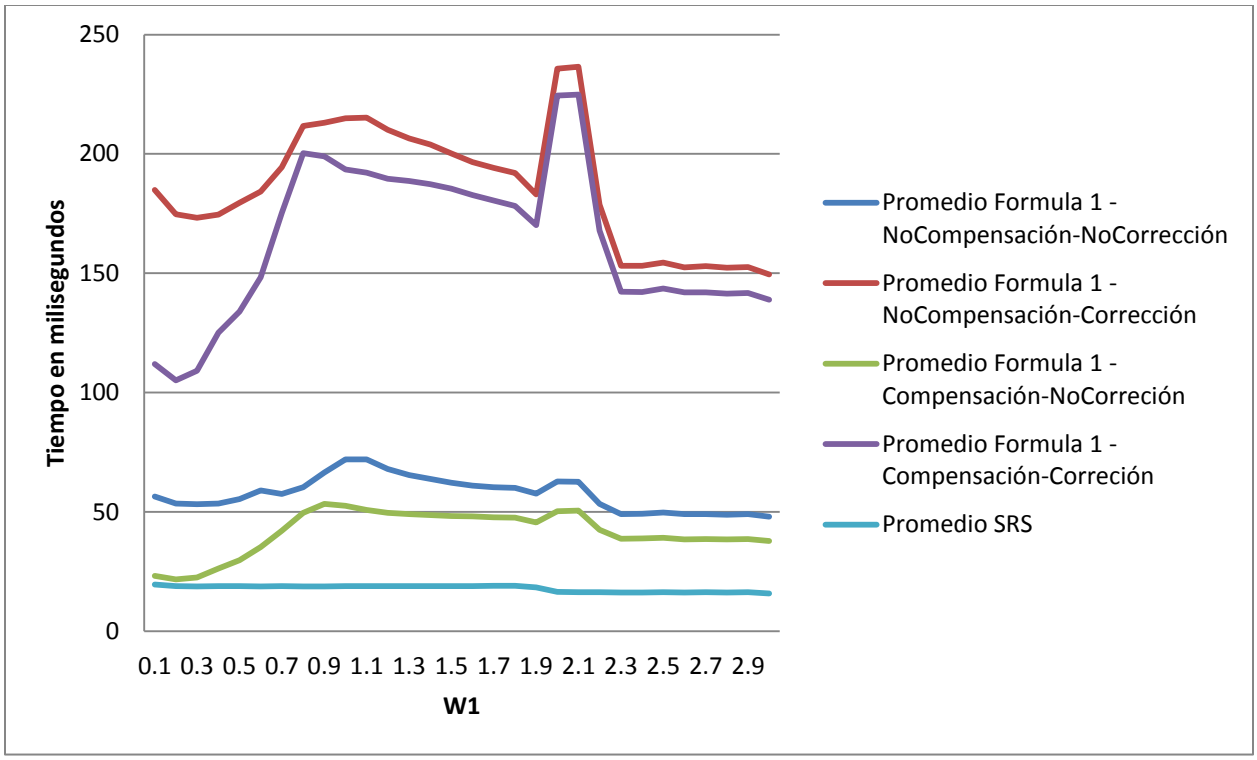

Figura 17 - Resultado Experimento 5 - tiempo en milisegundos

Capitulo 5 Análisis de resultados

5.1 Variantes de las fórmulas implementadas

Se concluye a partir del experimento 1 que las variantes de la fórmula 1 son superiores a sus equivalentes de la fórmula 2, y que al menos 6 de las variantes del algoritmo son significativamente mejores que el SRS tradicional.

La fórmula 1 con compensación y corrección consistentemente brindo el menor índice democrático a lo largo de todos los experimentos excepto el primero. Sin embargo, en el primero se encuentra de segunda, por un margen trivial. Por lo tanto, se considera que está es la mejor variantes de las fórmulas planteadas a la hora de minimizar el índice democrático.

5.2 Comportamiento al crecer el tamaño de la población

Se concluye, gracias al experimento 2, que el algoritmo propuesto es consistentemente mejor que el SRS a lo largo de una amplitud de tamaños de la población. Se puede notar en la Figura 10 que tanto la Formula 1 como la 2 retornan un índice democrático más bajo conforme crece el tamaño de la población, hasta que llagan a un punto mínimo. Posterior a este punto, el índice democrático se mantiene constante.

Esto probablemente se debe a que, dada la distribución de las propiedades de la población, después de cierto tamaño existen en ella una alta diversidad de individuos en suficientes cantidades como para garantizar cierto índice democrático. El reducir este índice aún más requeriría poblaciones extremadamente grandes, en la que individuos muy inusuales estén presentes. Sin embargo, conforme N tiende a infinito, el índice democrático también lo hará. Esto constituye una hipótesis sin comprobar.

5.3 Comportamiento respecto a la proporción del tamaño de la muestra respecto al de la población.

En el experimento 4, Figura 14 se observa que el índice democrático crece a un ritmo moderado cuando el tamaño de la muestra es pequeño comparado con el de la población. Luego, cuando la proporción entre ambos llega a un punto, comienza a crecer mucho más rápido y finalmente converge con el índice democrático del SRS cuando ambos tamaños son iguales.

Esto parece deberse a que, después de cierta proporción, los individuos que hubieran contribuido a aumentar la diversidad de la muestra ya están dentro de ella. Por lo tanto, los individuos que se terminan agregando consistentemente reducen esta diversidad. Por lo tanto, el índice democrático crece a un ritmo más rápido.

5.4 Desempeño en milisegundos

El algoritmo propuesto es consistentemente más lento que el algoritmo SRS en todos los experimentos. Solo hubo un escenario donde esto no se cumplió. Sin embardo, dado que la versión implementada del SRS es O(n) y existen versiones O(Log n) se puede concluir que el algoritmo SRS es más rápido. De esta forma, se confirma la hipótesis 5 (el SRS es más rápido).

5.5 Efecto de los pesos

En el experimento 5 se puede notar que conforme el peso de una variable aumenta, su índice democrático también lo hace.

La hipótesis 4 plantea que el mecanismo de los pesos es suficiente para lograr priorizar las propiedades. Los resultados actuales la respaldan, dado que son lo que se esperaría que son cierta. Sin embargo, no son suficientes para validarla y hacen falta experimentos que perfilen el comportamiento del índice democrático en múltiples propiedades simultáneamente para poder validarla.

Capitulo 6 Conclusiones

6.1 Resultado de los objetivos

El algoritmo descrito en el capítulo 3 satisface el objetivo 1: *Crear un algoritmo que minimice el valor de las funciones objetivo.* El código fuente de este algoritmo se encuentra adjunto a esta tesis bajo licencia MIT, lo que cumple con el objetivo 3 de brindar *una librería de código libre sobre una plataforma altamente utilizada (.NET/CLR).*

El objetivo 2, consiste en *Extender al menos una de las funciones objetivo para soportar un mecanismo de priorización de propiedades.* Fue llevado a cabo implementando una variante de las fórmulas que toma en cuenta un peso numérico por propiedad para priorizarlas.

Finalmente, el objetivo 4 de *comparar el de selección democrática contra el SRS* fue llevado a cabo con éxito con los 5 experimentos realizado. La comparación incluyo tanto el de rendimiento (en tiempo) como la capacidad para minimizar la varianza de las propiedades

6.2 Veracidad de las Hipótesis

La hipótesis 1, *un algoritmo voraz es suficiente para resolver el problema* fue comprobada en el experimento 1. Seis de las ocho versiones implementadas del algoritmo son significativamente mejores al algoritmo SRS.

De formar similar, la hipótesis 2 (*minimizar el valor de las funciones objetivo o debe ser la meta del algoritmo voraz)* fue comprobada experimentalmente. La fórmula objetivo 1 (basada en la desviación absoluta) es superior a la fórmula 2 (basada en la varianza). Sin embargo, la segunda también fue efectiva. Este resultado fue observado consistentemente a lo largo de los 5 experimentos. Queda abierto a la investigación determinar si hay mejores fórmulas.

La hipótesis 3, *asignar pesos a cada propiedad, combinado con las métricas mencionadas anteriormente, es suficiente para permitir dar prioridad a la minimización de la diferencia de ciertas propiedades con respecto a otras,* parece ser cierta. En el experimento 5, se puede notar que conforme el peso de una variable aumenta, su índice democrático también lo hace. Sin embargo, los resultados no son completamente concluyentes y requieren más investigación.

El algoritmo SRS fue consistentemente más rápido que el algoritmo democrático, lo cual es consistente con la hipótesis 4 (*El algoritmo SRS será significativamente más rápido que el algoritmo voraz propuesto).* Esto se pudo observar en los 5 experimentos ejecutados. La única excepción fue con valores muy grandes de N. Sin embargo, la versión implementada aquí del SRS es O(n) y existen versiones O(Log n), por lo cual la hipótesis se mantiene cierta.

6.3 Comentarios finales y futuras áreas de investigación

Este estudio surgió por una necesidad muy concreta: ¿cómo crear una muestra lo más diversa posible a partir de una población? Muchos estudios han atacado este problema de una u otra forma (el muestreo estratificado desbalanceado siendo uno de los mejores ejemplos), sin embargo, las técnica disponibles no escalan cuando se desea crear una muestra diversa a partir de datos que cuentan con muchas propiedades (por ejemplo, genero, estado civil, etnia, etc.). Como sobrepasar esta limitación es la razón de ser de este documento.

La hipótesis clave del estudio es que un algoritmo voraz que transfiera secuencialmente individuos de la población hacia la muestra, hasta llegar al tamaño de la muestra deseado, resulto cierta y fue comprobada experimentalmente. Seis de las ocho versiones implementadas del algoritmo resultaron consistentemente superiores al algoritmo de muestreo aleatorio en maximizar la diversidad. Concretamente, la variante que calcula un costo para decidir cuál elemento transferir utilizando la *fórmula de muestreo 1 con compensación, seguida por una fase de corrección fue consistentemente la mejor*.

$$sea\ Rx_i v_j = \begin{cases} \dfrac{\frac{n}{|dom(x_i)|} - \left|Sx_i v_j\right|}{\left|Ux_i v_j\right|}, & si\ \left|Ux_i v_j\right| = \left|Sx_i v_j\right| \\ -\infty, si\ no \end{cases}$$

Fórmula 10 – Formula Muestreo 1 con compensación, seguido por fase de corrección

Un resultado inesperado es una hipótesis que surge de este trabajo: el algoritmo propuesto es una generalización del muestreo estratificado. Al ejecutarlo brinda el mismo resultado, siempre y cuando el muestreo sea tomando en cuanta una sola de las propiedades de los individuos. Sin embargo, el algoritmo propuesto soporta escenarios más amplios, por lo cual se vuelve una generalización. Realizar una prueba formal de esta hipótesis resulta un interesante tema de investigación que no debería ser complejo.

Permanecen otros puntos que pueden ser explorados en futuras investigaciones. De las hipótesis originales, la hipótesis 3 planteaba que un mecanismo de pesos es suficiente para priorizar la diversidad de una propiedad sobre otra. Asimismo, de los experimentos se desprende la hipótesis de que al tender el tamaño de la población a infinito la diversidad de la muestra, medida a través del índice democrático, tiende a crecer. Ambas hipótesis parecen ser respaldadas por los experimentos, pero se vuelven necesarios más experimentos para tener resultados concluyentes.

La explosión de datos que ha surgido a consecuencia de las tecnologías de información en años recientes nos presenta la oportunidad de realizar análisis que hace una década hubieran resultado imposibles. Sin embargo, al mismo tiempo nos presenta con dificultades que requieren

sobrepasar los límites técnicos usuales. El algoritmo de selección democrática de individuos aquí presentado representa una herramienta de muchas que nos puede ayudar a asumir este reto.

Bibliografía

BENTLEY, J. (1987). A sample of brilliance. *Communications of the ACM, 30*(9 (754-757)).

CORNELIUS, T. (05 de September de 2012). *About CodePlex*. (Microsoft) Recuperado el 27 de July de 2013, de Codeplex: http://codeplex.codeplex.com/

DAVIES, J. (6 de 2012). *Parallel Sets: titanic survivors*. Recuperado el 23 de 6 de 2013, de http://www.jasondavies.com/parallel-sets/

DAWSON, R. (1995). The "Unusual Episode" Data Revisited. *Journal of Statistics Education, 3*(3).

DEVILLE, J. (March 1998). Unequal probability sampling withouth replacement through splitting method. *Biometrika, 85*(1 (89-101)).

GROTHAUS, G. (8 de 10 de 2007). *Reservoir Sampling*. Recuperado el 16 de 7 de 2013, de Sampling from a stream of elements: http://gregable.com/2007/10/reservoir-sampling.html

NEYMAN, J. (1938). Contribution to the Theory of Sampling Human Populations. *Journal of the American Statistical Association, 33*(201 (101-116)).

PATHAK, P. (1988). Simple random sampling. En P. Krishnaiah, & C. Rao, *Handbook of Statistics Volume 6: Sampling.* Amsterdam: Elsevier.

REVELIOTIS, S. (20 de 06 de 1997). *Georgia Tech University*. Recuperado el 15 de 07 de 2013, de An Introduction to Linear Programming and the Simplex Algorithm: http://www2.isye.gatech.edu/~spyros/LP/LP.html

TILLE, Y. (2006). *Sampling Algorithms.* Neuchâtel: Switzerland.

TILLE, Y. (Noviembre de 2010). *Balanced sampling by means of cube method.* Recuperado el 07 de 07 de 2013, de University of Neuchatel: http://www.eustat.es/productosServicios/52.2_balanced_sampling.pdf

USDA. (17 de 5 de 2013). *USDA nutrient database.* Recuperado el 2013 de 7 de 28, de USDA: http://www.ars.usda.gov/SP2UserFiles/Place/12354500/Data/SR25/dnload/sr25db.zip

VITTER, J. (March de 1985). Random Sampling with Reservoir. *ACM transactions on Mathematical Software, 11*(1 (37-57)). Obtenido de University of Maryland.

WEISSTEIN, E. W. (s.f.). *Hypercube*. Recuperado el 23 de 7 de 2013, de MathWorld --A Wolfram Web Resource: http://mathworld.wolfram.com/Hypercube.html

WYNN, H. (1977). Convex sets of finite population plans. *Annals of Statistics, 5*(2 (414-418)).

YATES, F. (1946). A Review of Recent Statistical Developments in Sampling and Sampling Surveys. *Journal of the Royal Statistical Society, 109*(1 (12-43)).

YATES, F., & Grundy, P. (1953). Selection without replacement from within strata with probability proportional to size. *Journal of Royal Statistical Society, 15*((235-261)).