SECRETARÍA DE EDUCACIÓN PÚBLICA
SEP
SUBSECRETARIA DE EDUCACION SUPERIOR

INSTITUTO TECNOLÓGICO
DE CD. MADERO

Sistema Nacional de Educación Superior Tecnológica
Dirección General de Educación Superior Tecnológica

DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN

"REDUCCIÓN DE INSTANCIAS DE GRAN ESCALA DEL PROBLEMA DE LA DISTRIBUCIÓN MEDIANTE MUESTREO PROGRESIVO"

PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:
I.S.C. JESÚS DAVID TERÁN VILLANUEVA

A S E S O R:
Dr. HÉCTOR JOAQUÍN FRAIRE HUACUJA

CD. MADERO, TAM. MEXICO DICIEMBRE DE 2005

SECRETARIA DE EDUCACIÓN PÚBLICA
SEP
SUBSECRETARÍA DE EDUCACIÓN SUPERIOR

Instituto Tecnológico de Cd. Madero

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

COORDINACIÓN DE POSGRADO EN COMPUTACIÓN
U5.367/05
AUTORIZACIÓN DE IMPRESIÓN
DE TESIS DE GRADO

2005-NOV-15

C. ING. JESÚS DAVID TERÁN VILLANUEVA
P r e s e n t e.

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su examen de grado de Maestro en Ciencias en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"REDUCCIÓN DE INSTANCIAS DE GRAN ESCALA DEL PROBLEMA DE LA DISTRIBUCIÓN MEDIANTE MUESTREO PROGRESIVO"

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
"POR MI PATRIA Y POR MI BIEN"

DRA. ANA MARÍA MENDOZA MARTÍNEZ
JEFA DE LA DIVISIÓN

ESTADOS UNIDOS MEXICANOS
S.E.P.
DIVISION DE ESTUDIOS DE POSGRADO E INVESTIGACION
I.T.C.M.

AMMM ' NECO ' cerc*

Av. 1° de Mayo 1501 Pte. Col. Los Mangos
Cd. Madero, Tam. C. P. 89440
Internet: www.itcm.edu.mx
Correo Electrónico: itcm@itcm.edu.mx

Teléfono: 3-57-48-20 al 29
Fax. 3-57-48-20 ext. 1002

SECRETARÍA DE EDUCACIÓN PÚBLICA | SEP

SUBSECRETARIA DE EDUCACION SUPERIOR

INSTITUTO TECNOLÓGICO DE CD. MADERO

Sistema Nacional de Educación Superior Tecnológica

Dirección General de Educación Superior Tecnológica

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

POR MI PATRIA Y POR MI BIEN

"REDUCCIÓN DE INSTANCIAS DE GRAN ESCALA DEL PROBLEMA DE LA DISTRIBUCIÓN MEDIANTE MUESTREO PROGRESIVO"

PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

I.S.C. JESÚS DAVID TERÁN VILLANUEVA

A S E S O R:

Dr. HÉCTOR JOAQUÍN FRAIRE HUACUJA

J U R A D O:

PRESIDENTE:	Dr. HÉCTOR JOAQUÍN FRAIRE HUACUJA.
SECRETARIO:	Dra. LAURA CRUZ REYES.
VOCAL:	MC. JOSÉ ANTONIO MARTÍNEZ FLORES
SUPLENTE:	MC. ARQUÍMEDES GODOY VINAJA

CD. MADERO, TAM. MEXICO DICIEMBRE DE 2005

Dedicatorias

A Dios por permitirme llegar a este momento, y dirigirme en esta ruta.

A mis padres por apoyarme y darme grandes enseñanzas de la vida, dado que soy un reflejo de lo que me han enseñado.

A mis maestros que me han dedicado tiempo para realizar este trabajo, apoyo y conocimiento que siempre es de gran utilidad como alumno y persona.

A mi novia que me ha apoyado todo este tiempo y me ha ayudado a tener una visión más grande.

A todos y cada uno de mis amigos y compañeros que siempre están conmigo y me han apoyado directa e indirectamente.

A todas estas personas que han sido un gran ejemplo y un motivo para continuar.

Resumen

En este proyecto de tesis se aborda el problema del diseño de la distribución de datos en bases de datos distribuidas, éste consiste en distribuir un conjunto de objetos (tuplas, imágenes, archivos) en diferentes servidores interconectados a través de Internet ubicados en diferentes lugares geográficos y que son requeridos por un conjunto de consultas realizadas por los usuarios. El problema consiste en determinar la distribución de los objetos en los sitios de tal forma que el costo generado por la atención de las consultas, el almacenamiento de los datos en los sitios y el de la migración de los mismos sea el menor posible.

Este problema ha sido abordado por varios investigadores, sin embargo prevalecen las dificultades para resolver instancias realistas. En un trabajo reciente se propone el uso de un método de reducción de las instancias mediante muestreo progresivo, el cual permite reducir la cantidad de recursos requeridos para resolver una instancia. El impacto en la escalabilidad de los métodos de solución depende críticamente de la métrica de similitud y del criterio de convergencia utilizados. En este trabajo se propone una nueva métrica y un nuevo criterio de convergencia.

Se realizaron pruebas comparativas entre las métricas de similitud y criterios de convergencia, para determinar su impacto en los niveles de reducción y calidad de la solución. Estas pruebas muestran que la métrica y criterio propuestos logran un mejor rendimiento en cuanto a la calidad de la solución de las instancias de prueba. En contraparte se observa una ligera disminución en los niveles de reducción.

Summary

In this thesis project is present the distribution problem on distributed data bases, having a quantity of objects (tuples, images, files) located in several servers interconnected through Internet on different places and a set of queries that access those objects. The problem is to distribute the objects on the sites, thus the cost generated by the queries, storage and migration will be the minor possible.

This theme have been investigated on previous documents, however there is not yet a method to solve realistic instances.

There is solution only to small instances with few objects, queries and the sites is not a problem since ten sites are good enough.

In this work it is exposed a method of reduction of instances through progressive sampling, which let us solve a smaller instance than the original and thus get a greater scalability solving this problem.

There has been fulfilled several tests with different similarity metrics and different convergence approaches to obtain a small instance representative of the original.

These tests demonstrate that it is possible to reduce the original instance by progressive sampling and get a solution like the ones obtained with the original instance, with a small error.

Índice general

Índice de tablas

Índice de figuras

CAPÍTULO 1

Introducción

En este capítulo se presentan las principales motivaciones de esta investigación, una introducción al problema de estudio, la definición del problema, el contexto de la investigación, objetivos, delimitaciones y una breve descripción de la organización de la tesis.

1.1 Introducción

Una de las consecuencias del creciente uso de Internet y el ebusiness en las empresas es la necesidad de aplicaciones de bases de datos distribuidas (BDD), que les permitan incorporar las ventajas de la conectividad a sus sistemas de información. Estas aplicaciones se realizan utilizando Sistemas Manejadores de Bases de Datos Distribuidas (SMBDD) [1]. A la fecha, existen productos comerciales y prototipos de investigación de manejadores de base de datos como Oracle, Sybase, SQL Server, etc., que ofrecen una gran funcionalidad para la administración de los datos. Algunas de las funciones más relevantes que se han incorporado en los últimos años a los prototipos y manejadores de bases de datos son: soporte a bases de datos distribuidas fragmentadas horizontalmente

(grupo de tuplas), fragmentadas verticalmente (grupo de atributos), fragmentadas de forma mixta (grupo de tuplas y atributos) y replicadas. Sin embargo, la función de administración del rendimiento de las bases de datos es una de las tareas que actualmente se sigue realizando en forma manual.

En [2, 3] se presenta el proceso del diseño de bases de datos distribuidas con un enfoque 'arriba a abajo'. Este proceso se lleva acabo como se muestra en la Figura 1.1.

Figura 1.1: Enfoque de arriba a abajo

En la Figura 1.1 se puede apreciar un esquema general del enfoque de distribución de datos. Este inicia con un análisis de requerimientos que define el ambiente del sistema y obtiene los datos y necesidades de procesamiento de todos los usuarios potenciales de la base de datos. Con esta información se genera un esquema global. Una vez que se tiene este esquema se procede a generar el proceso de distribución de datos, realizando fragmentación y ubicación de datos en los diversos sitios de la base de datos. Habiendo concluido con este diseño de distribución se procede a generar la base de datos de forma

física.

Para garantizar un nivel adecuado en el desempeño de las aplicaciones de base de datos distribuidas, el SMBDD debe de contar con mecanismos eficientes para ubicar los datos donde son requeridos por los usuarios en los diferentes sitios de la red.

Éste es un problema crítico, ya que si la información no esta distribuida de forma eficiente se vera deteriorado el desempeño de la aplicación. El número de formas alternativas del diseño de la base de datos, crece explosivamente con el número de atributos y de tuplas de las tablas de la base de datos [4]. Al igual que muchos otros problemas reales éste se convierte en un problema de optimización combinatoria NP-duro.

1.2 Problema de investigación

El problema del diseño de la distribución ha sido abordado por muchos investigadores, sin embargo prevalecen las dificultades para resolver instancias realistas. En un trabajo reciente se propone el uso de un método de reducción de las instancias mediante muestreo progresivo [5], el cual permite reducir la cantidad de recursos requeridos para resolver una instancia. El impacto en la escalabilidad de este método depende críticamente de la métrica de similitud y del criterio de convergencia utilizados, en este trabajo se pretende desarrollar una métrica de similitud y un criterio de convergencia que permitan obtener una muestra más representativa que la generada en el trabajo anterior.

Definición formal del problema del diseño de la distribución

Dados un conjunto de objetos $O = \{o_1, o_2, ..., o_{no}\}$, un conjunto de computadoras o sitios, interconectados por una red, $S = \{s_1, s_2, ..., s_{ns}\}$, en los que se ejecutan un conjunto de consultas $Q = \{q_1, q_2, ..., q_{nq}\}$, el esquema de la ubicación inicial de los objetos,

los objetos requeridos por cada consulta, la frecuencia con que se emite cada consulta en los diferentes sitios, la selectividad de las consultas sobre los objetos, la capacidad de almacenamiento de los sitios, los costos de comunicación entre los sitios y el costo de almacenamiento en los sitios, en un período dado de tiempo, el problema consiste en determinar un nuevo esquema de la ubicación replicada de los objetos que se adapte al nuevo patrón del uso de la base de datos y que minimice el costo del procesamiento de las consultas que incluye: costos de comunicación y de almacenamiento.

Un objeto de base de datos (o simplemente objeto) es cualquier entidad que requiere ser ubicada, y puede ser un atributo, un grupo de tuplas, una relación o un archivo. Los objetos son unidades independientes que deben ser ubicados en los sitios de la red.

1.3 Contexto de la investigación

En [6] se propone un enfoque general de solución para resolver el diseño de la distribución que incluye acciones que abarcan aspectos o dimensiones: elegir una transformación de compresión, seleccionar el algoritmo que mejor desempeño ha mostrado en la solución de instancias similares, y seleccionar un modelo del problema de bajo costo computacional. Una combinación específica de este tipo de aspectos, definen una estrategia particular de solución. Una definición formal se plantea como sigue:

Dados:

- π: El problema del diseño de la distribución
- I: Un conjunto de instancias de π
- R: Un conjunto finito de transformaciones de compresión de instancias de π
- M: Un conjunto finito de modelos de π
- A: Un conjunto finito de algoritmos de solución para los modelos de π

Entonces, se dice que s es una estrategia de solución de una instancia dada de π si y solo si, s es un elemento del producto cartesiano $R \times A \times M$ (Figura 1.2).

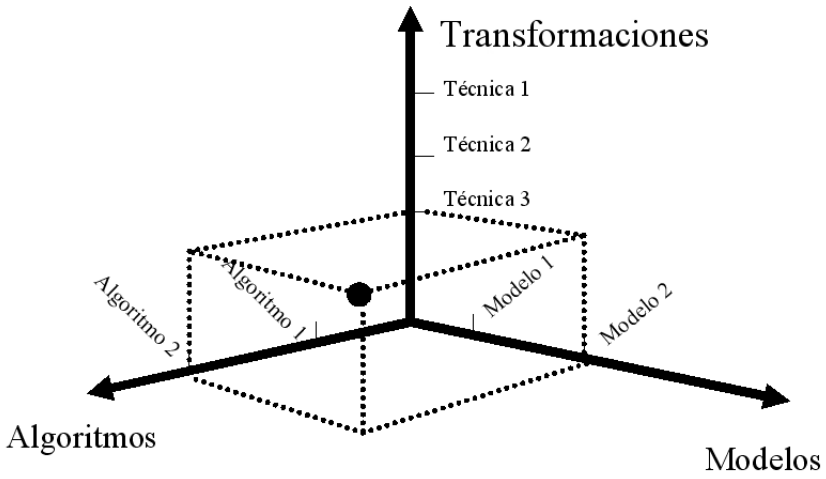

Figura 1.2: Alternativas de solución

Para una instancia dada $i \in I$, el enfoque es determinar automáticamente una estrategia $s = (r_j, a_k, m_l)$, donde $r_j \in R, a_k \in A$ y $m_l \in M$, la cual permita resolver la instancia i usando la transformación r_j, el algoritmo a_k y el modelo m_l.

En este trabajo se aborda la dimensión de transformaciones de instancias. Esta dimensión supone que, para una instancia dada, es posible diseñar una transformación que comprima la instancia de manera que la solución de la instancia transformada sea una solución de calidad adecuada de la instancia original. La instancia transformada debe

ser de menor tamaño que la instancia original, como se muestra en la Figura 1.3.

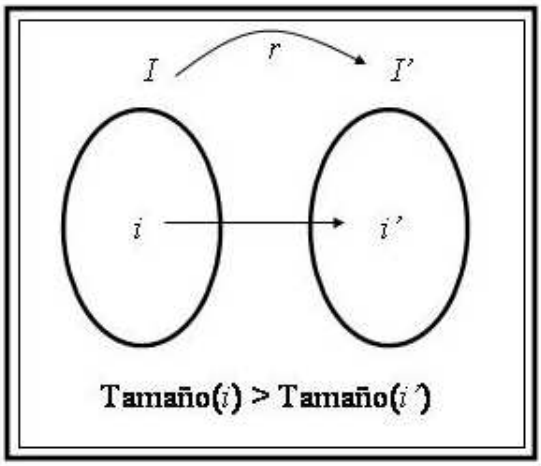

Figura 1.3: Transformación de instancias

La Figura muestra una transformación r, donde I es el conjunto de instancias del problema del diseño de la distribución, i e i' son instancias de I, e I' es un subconjunto de I.

La idea básica en este método es generar una muestra representativa de las operaciones de la instancia original i y resolver la instancia i', que se determina al considerar únicamente las operaciones de la muestra. Se observa que, una solución factible de la instancia transformada i' es también una solución factible de la instancia original i, pero la solución óptima de i depende de la representatividad de i'.

Para instancias muy grandes del problema, si el número de sitios es mucho menor que el número de operaciones, el patrón de acceso de las operaciones en los sitios aparecerá repetido en la matriz de frecuencia de acceso. En estas condiciones es razonable suponer que un proceso progresivo de muestreo pueda identificar el patrón general de acceso de las operaciones usando muestras relativamente pequeñas.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar una nueva métrica de similitud y un nuevo criterio de convergencia para el método de reducción de instancias del problema del diseño de la distribución, basado en muestreo progresivo.

1.4.2 Objetivos particulares

- Determinar el impacto de las métricas de similitud en el desempeño del método de reducción.
- Determinar el impacto de los criterios de convergencia en el desempeño del método de reducción.

1.5 Delimitaciones

- Sólo se considera el método de reducción reportado en [5].
- El modelo del problema que se utiliza es el reportado en [6]
- Para implementar el método se consideran instancias originales de tamaño tal que se puedan resolver en forma exacta.

1.6 Organización de la tesis

Capítulo 2: En este capítulo se describe el marco teórico en que se fundamenta este trabajo. Esto incluye el fundamento teórico del problema de la distribución, del proceso de muestreo y del método de ramificación y acotamiento. Además presenta la demostración de que la versión replicada del problema del diseño de la distribución es *NP-Difícil*. Se incluye también una sección que contiene los trabajos relacionados con el problema de la automatización del diseño de la distribución.

Capítulo 3: En este capítulo se presentan el modelo matemático y el método de compresión de instancias del problema del diseño de la distribución que son utilizados para describir la métrica de similitud y el criterio de convergencia utilizados en [5] y los propuestos en esta tesis.

Capítulo 4: En este capítulo se muestra la implementación descrita en el capítulo anterior, se presenta un diagrama general del proceso de experimentación. Una vez visto este diagrama se expondrán los detalles de cada paso del mismo.

Capítulo 5: En este capítulo se describen los experimentos realizados para analizar el impacto de las métricas de similitud y los criterios de convergencia.

Capítulo 6: En este capítulo se presentan las aportaciones realizadas por este trabajo de tesis, al igual que los resultados de las pruebas realizadas en términos generales.

CAPÍTULO 2

Marco teórico

En este capítulo se describe el marco teórico en que se fundamenta este trabajo. Esto incluye los fundamentos del problema de la distribución, de la técnica de muestreo progresivo, el método de muestreo secuencial aleatorio y del método de ramificación y acotamiento. Se incluye también una sección que contiene los trabajos relacionados con el problema de la automatización del problema del diseño de la distribución y del muestreo progresivo.

2.1 El problema de la distribución

2.1.1 Bases de datos distribuidas

Una *base de datos distribuida* es una colección de datos que pertenecen lógicamente al mismo sistema pero que están distribuidos en los diferentes sitios de una red de computadoras. Cada sitio de la red tiene capacidad de procesamiento autónoma y puede realizar aplicaciones locales. Además, cada sitio participa en la ejecución de al menos una aplicación global, la cual requiere acceder a datos desde distintos sitios utilizando

un subsistema de comunicación. Los problemas tecnológicos más importantes de las bases de datos distribuidas derivan de considerarlas el resultado de una "cooperación entre sitios autónomos" [2].

Dos aspectos importantes de una base de datos distribuida son los siguientes:

1. **Distribución:** Los datos, a diferencia de una base de datos centralizada, no residen en el mismo sitio.

2. **Correlación lógica:** Los datos tienen ciertas propiedades que los relacionan o mantienen juntos, lo cual diferencia una base de datos distribuida de un conjunto de bases de datos locales o archivos que residen en diferentes sitios de una red de computadoras.

La *transparencia de la distribución* se refiere a que los programas pueden ser escritos como si la base de datos no fuera distribuida. De esta manera la validez de los programas no es afectada por el movimiento de los datos de un sitio a otro; sin embargo la velocidad de ejecución sí es afectada [2].

En bases de datos distribuidas existen varias razones para considerar la *redundancia de los datos* como una característica deseable. La localidad de las aplicaciones puede incrementarse si los datos están replicados en todos los sitios donde las aplicaciones los necesitan. La disponibilidad del sistema se incrementa debido a que la falla de un sitio no detiene la ejecución de las aplicaciones en otros sitios donde los datos están replicados [2].

Un *sistema manejador de bases de datos distribuidas* da soporte a la creación y mantenimiento de bases de datos distribuidas. En las *bases de datos relacionales*, los datos están almacenados en tablas llamadas *relaciones*. Cada relación tiene cierta cantidad de columnas llamadas *atributos* y un número de filas llamadas *tuplas*. El número de atributos de una relación es llamado *grado* de la relación y la cantidad de tuplas se conoce

como su *cardinalidad*. El conjunto de posibles valores que puede tomar un atributo es llamado *dominio* [2].

Las *llaves* son subconjuntos de los atributos de un esquema de relación cuyos valores son únicos dentro de la relación y por lo tanto pueden ser utilizados para identificar de forma única las tuplas de la relación. Una *aplicación* es una secuencia de operaciones que pueden ser solicitadas por un usuario final (no un programador) con una activación de solicitud sencilla [2].

Una *transacción* es una unidad atómica de acceso a la base de datos. Está constituida por una secuencia de operaciones que son realizadas completamente o no son realizadas en absoluto. Nótese que es muy común que coincidan una transacción y una aplicación, es decir, que una función elemental solicitada por el usuario es además una unidad atómica de acceso a la base de datos. Un *programa* es la definición de un cálculo o cómputo que incluye accesos a la base de datos. También se considera un programa como una unidad que tiene su propio espacio de memoria y que se comunica con otros programas solamente a través de mensajes y primitivas de sincronización [2].

Una *consulta* es una expresión en un lenguaje compatible con la base de datos, que define una porción de los datos contenidos en la base de datos. Por lo tanto una consulta puede ser utilizada para definir la semántica de una aplicación o la función realizada por un programa que accesa a la base de datos [2]. Una *vista* es una relación derivada, cuyo valor es determinado como resultado de evaluar una expresión relacional y proporciona diferentes formas de ver a “los datos reales" [7].

Nótese que se utiliza el término “aplicación” para denotar una función solicitada por un usuario, el término “consulta” para denotar una solicitud a la base de datos y el término “programa” para denotar la implementación de una aplicación en un lenguaje de programación [2].

Un *servidor* es una computadora conectada por una red de comunicación y es capaz

de realizar trabajos de manera autónoma [2]. Normalmente utilizamos la palabra *sitio* para denotar una computadora host.

La red de comunicación esta constituida de enlaces de comunicación de varios tipos (líneas telefónicas, cables coaxiales, enlaces satelitales, etc.) e incluyen varias computadoras. Estas computadoras, dedicadas a funciones de comunicación, no son computadoras host y no se consideran de manera explícita en este estudio, son simplemente útiles para implementar la red de comunicación.

2.1.2 Fragmentación

La fragmentación es el proceso de dividir las relaciones de la bases de datos en subrelaciones. El propósito del proceso es colocar en un mismo fragmento los datos que son requeridos juntos por varias aplicaciones [3, 2].

Razones para Fragmentar

Una relación no es una unidad apropiada de distribución por varias razones [3]:

- Las vistas de aplicación son normalmente subconjunto de relaciones. Debido a esto la localidad de los accesos de las aplicaciones no está definida sobre una relación entera sino en sus subconjuntos. Por lo tanto es natural considerar los subconjuntos de las relaciones como unidades de distribución.
- Si las aplicaciones que tienen vistas definidas sobre alguna relación residen en sitios diferentes, se pueden seguir dos alternativas con la relación entera como unidad de distribución: la relación no está replicada y está ubicada en un sitio solamente o está replicada en todos o algunos sitios donde las aplicaciones residen.

La primera opción resulta en un gran volumen de accesos remotos innecesarios. Por otra parte, la segunda opción produce replicas innecesarias, lo que causa problemas al ejecutar las actualizaciones y puede no ser deseable si el almacenamiento es limitado.

- La descomposición de una relación en fragmentos, cada uno tratado como una unidad, permite a un número de transacciones ejecutarse de manera concurrente. Además, la fragmentación de relaciones normalmente resulta en la ejecución paralela de una consulta simple dividiéndola en un conjunto de subconsultas que operan con fragmentos. Por lo tanto la fragmentación incrementa el nivel de concurrencia y por consiguiente el desempeño.

También existen desventajas frente al problema de la fragmentación. Se destacan las siguientes [3]:

- Si las aplicaciones tienen requerimientos que evitan la descomposición de la relación en fragmentos mutuamente exclusivos, esas aplicaciones cuyas vistas se definen en más de un fragmento pueden sufrir una degradación en el desempeño. Puede ser necesario, por ejemplo, recuperar datos de dos fragmentos y entonces tomar su unión o junta, lo cual es costoso. Evitar esto es un problema fundamental de la fragmentación.

- El segundo problema está relacionada con el control semántico de datos, específicamente para la revisión de la integridad. Como resultado de la fragmentación, los atributos que participan en una dependencia pueden ser descompuestos en diferentes fragmentos que pueden ser ubicados en sitios diferentes. En este caso, incluso la tarea más simple de revisión de dependencias podría requerir de la búsqueda de datos en muchos sitios.

Alternativas de fragmentación

Las instancias de las relaciones son tablas esencialmente, así que el problema es encontrar formas alternativas para dividir una tabla en tablas más pequeñas. Existen claramente dos alternativas para esto: dividirla horizontalmente o verticalmente [3].

Tabla 2.1: Relación Proyectos.

NUM_PROY	PROYECTO	PRESUPUESTO	CIUDAD
P1	Instrumentación	150,000	Monterrey
P2	Desarrollo de BD	135,000	D.F.
P3	CAD / CAM	250,000	D.F.
P4	Mantenimiento	310,000	Tampico

En la *fragmentación horizontal*, la relación se divide en grupos de tuplas. Las tuplas que cumplen una condición dada son integradas al mismo fragmento [3]. Esto es útil en las bases de datos distribuidas donde cada grupo de atributos puede contener datos con propiedades geográficas comunes. Además, debe ser posible reconstruir la relación original realizando una operación de junta con todos los fragmentos [2].

La fragmentación horizontal reduce el acceso a disco requerido para ejecutar una aplicación, minimiza el número de objetos irrelevantes recuperados y permite un alto grado de paralelismo en la ejecución de las aplicaciones [8].

Tabla 2.2: Ejemplo No. 1 de fragmentación horizontal sobre la relación Proyectos.

NUM_PROY	PROYECTO	PRESUPUESTO	CIUDAD
P1	Instrumentación	150,000	Monterrey
P2	Desarrollo de BD	135,000	D.F.

Tabla 2.3: Ejemplo No. 2 de fragmentación horizontal sobre la relación Proyectos.

NUM_PROY	PROYECTO	PRESUPUESTO	CIUDAD
P3	CAD / CAM	250,000	D.F.
P4	Mantenimiento	310,000	Tampico

En la *fragmentación vertical*, la relación se divide proyectándola sobre algunos de sus atributos. Cada fragmento está constituido por un conjunto de atributos, entre los que

se incluye la llave primaria. La llave primaria es utilizada para reconstruir la relación mediante la operación de reunión aplicada a los fragmentos [3].

Tabla 2.4: Ejemplo No. 1 de fragmentación vertical sobre la relación Proyectos.

NUM_PROY	PRESUPUESTO
P1	150,000
P2	135,000
P3	250,000
P4	310,000

Tabla 2.5: Ejemplo No. 2 de fragmentación vertical sobre la relación Proyectos.

NUM_PROY	PROYECTO	CIUDAD
P1	Instrumentación	Monterrey
P2	Desarrollo de BD	D.F.
P3	CAD / CAM	D.F.
P4	Mantenimiento	Tampico

La fragmentación puede ser anidada. Si en diferentes niveles de anidación se tienen diferentes tipos de fragmentación, se genera una *fragmentación mixta*. Este tipo de fragmentación se puede realizar proyectando la relación y luego seleccionando las tuplas o de manera inversa [3].

Reglas de fragmentación

Estas son las reglas que aseguran que la base de datos no sufra cambios semánticos durante la fragmentación [3]:

1. **Completez:** Si una instancia de relación R es descompuesta en fragmentos R_1, R_2, ... R_n, cada unidad que puede ser encontrada en R también puede ser encontrado en uno o más de los fragmentos R_i's.

2. **Reconstrucción:** La propiedad de reconstrucción de la relación de sus fragmentos asegura que se preserven las restricciones definidas en los datos en forma de dependencias.

3. **Disjunción:** Si una relación R es descompuesta horizontalmente en fragmentos R_1, R_2, ... R_n, y un dato d_i está en R_j, no está en ningún otro fragmento R_k ($k \neq j$). Este criterio asegura que los fragmentos horizontales sean disjuntos. Si la relación R es es descompuesta verticalmente, sus atributos de llave primaria son repetidos en todos sus fragmentos. Por lo tanto, en el caso del particionamiento vertical, la disjuntividad es definida solamente en los atributos que no pertenecen a la llave primaria de la relación.

2.1.3 Ubicación

El problema de la ubicación de los datos consiste en determinar la mejor ubicación de los fragmentos, considerando un conjunto de restricciones dadas. Para la formulación de un modelo del problema se requiere definir qué factor se va a optimizar. Algunos factores utilizados en el análisis del problema son: el espacio de almacenamiento, el ancho de banda, el número de réplicas, y/o el costo total de comunicación [4].

Suponga que se tiene un conjunto de fragmentos $F = \{F_1, F_2, ..., F_i\}$ y una red que consiste de los sitios $S = \{S_1, S_2, ..., S_j\}$ donde se ejecuta un conjunto de aplicaciones $Q = \{q_1, q_2, ..., q_k\}$. El problema de ubicación consiste en encontrar la distribución "óptima" de F en S [3].

La optimalidad se puede definir con respecto a dos medidas [3]:

1. **Costo mínimo:** La función del costo consiste de los costos de almacenar cada F_i en cada sitio S_j, el costo de consultar F_i en el sitio S_j, el costo de actualizar F_i en todos los sitios donde esté almacenada, y el costo de la comunicación de los datos. El problema de la asignación trata de encontrar un esquema de ubicación que minimice una función de costo.

2. **Desempeño:** La estrategia de ubicación se diseña para mantener una métrica de desempeño. Dos bien conocidas son minimizar el tiempo de respuesta y maximizar el servicio del sistema en cada sitio.

2.1.4 Replicación

La replicación consiste en colocar copias de un mismo objeto en diferentes sitios. Las razones más importantes para replicar los datos son: asegurar un alto nivel de disponibilidad, mejorar la tolerancia contra fallas del sistema y mejorar el rendimiento del sistema [6].

Una base de datos no replicada (comúnmente llamada una base de datos *particionada*) contiene fragmentos que están ubicados en los sitios, y solamente existe una copia de cada fragmento en la red. En caso de replicación, la base de datos existe en su totalidad en cada sitio (base de datos *completamente replicada*) o en fragmentos distribuidos en los sitios de manera que las copias de un mismo fragmento existan en varios sitios (base de datos *parcialmente replicada*) [3].

En sistemas críticos la disponibilidad de los datos es un requisito de alta prioridad. Si un servidor falla, se puede recuperar el dato de otro servidor [3, 9]. La tolerancia a fallos, tiene que ver con el problema de garantizar la consistencia de los datos después de que ocurre una falla. La mejora en rendimiento se logra ubicando copias de los datos lo más cerca de sus usuarios [10]. Actualmente las más importantes compañías de administración de servicios de Internet, utilizan esquemas de replicación en espejo con este propósito [11, 12, 13].

2.2 Muestreo Progresivo

2.2.1 Conceptos preliminares

Antes de explicar el método de muestreo progresivo, es conveniente detallar un conjunto de conceptos que son la base, de esta técnica de muestreo. Los conceptos que se detallan a continuación son la minería de datos, el aprendizaje de máquinas y los algoritmos de inducción.

Minería de datos

La minería de datos está estrechamente relacionada con el descubrimiento de conocimiento en grandes bases de datos (DCBD). La minería de datos es un paso en el proceso de DCBD.

Fayyad [14] describe la minería de datos como la aplicación del análisis de datos y algoritmos de inducción que, bajo ciertas limitaciones de eficiencia computacional aceptables, produce una serie de patrones sobre los datos.

Hay que resaltar que el espacio de patrones es a menudo muy grande, y la enumeración de patrones requiere de una forma de búsqueda eficiente en este espacio. En este caso, las restricciones computacionales limitan el espacio de búsqueda de los algoritmos de inducción.

En ese mismo documento se describe el proceso de descubrimiento del conocimiento en los datos como el "proceso que utiliza la base de datos junto con cualquier selección requerida, preprocesamiento, submuestreo y transformación de ella; también aplica métodos de minería de datos para enumerar patrones, y evaluar éstos para obtener el conocimiento."

La Figura 2.1 muestra el proceso completo de DCBD, y el correspondiente lugar de la minería de datos en este proceso.

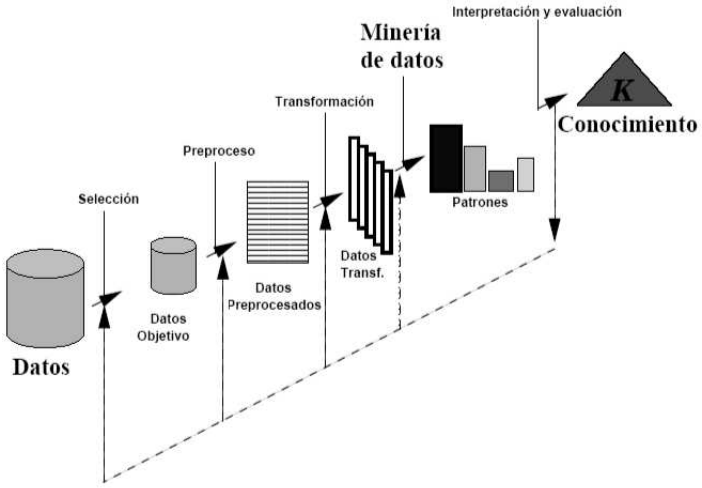

Figura 2.1: Proceso del DCBD

Aprendizaje en máquinas

El aprendizaje en máquinas es una área de la Inteligencia Artificial que involucra técnicas de desarrollo para permitir que las computadoras "aprendan". Más específicamente, el aprendizaje en máquinas es un método que crea programas de computadora mediante el análisis de conjuntos de datos, en lugar de la intuición. El aprendizaje en máquinas está estrechamente relacionado con estadísticas, dado que ambas áreas estudian el análisis de datos. Específicamente, el objetivo del aprendizaje en máquinas es desarrollar programas que mejoren su desempeño basándose en sus experiencias anteriores.

Los algoritmos de aprendizaje en máquinas están organizados en una taxonomía, basándose en el resultado deseado del algoritmo. Los tipos de algoritmo comunes incluyen:

- Aprendizaje dirigido, donde el algoritmo genera una función que relaciona las

entradas con los rendimientos deseados.

- Aprendizaje no supervisado, donde el algoritmo genera un modelo para un conjunto de entradas.
- Aprendizaje por refuerzo, donde el algoritmo aprende una política de cómo actuar dada una observación del mundo.
- Aprendiendo a aprender, donde el algoritmo aprende su propio proceso inductivo basado en la experiencia anterior.
- Aprendizaje por ejemplos, donde el algoritmo aprende a realizar una tarea con base en un conjunto de ejemplos.

El desempeño y el análisis computacional de los algoritmos de aprendizaje de máquinas son una rama de la estadística conocida como teoría del aprendizaje.

Algoritmos de inducción

La inducción es el proceso de ir más allá de lo que se conoce para alcanzar una conclusión que no está implicada de forma deductiva en el conocimiento [15]. Básicamente este tipo de algoritmos utiliza aprendizaje por ejemplos.

Los algoritmos de Inducción permiten el desarrollo de descripciones simbólicas de los datos, que puedan caracterizar uno o más grupos de conceptos [16, 17], diferenciar entre distintas clases, crear nuevas clases, crear una nueva clasificación conceptual, seleccionar los atributos más representativos, y ser capaces de predecir secuencias lógicas [18, 19]. Son esencialmente cualitativos.

2.2.2 Descripción del muestreo progresivo

En [20] se menciona el hecho de que tener grandes masas de información no necesariamente implica que los algoritmos de inducción deben utilizar toda la información. Las muestras pueden proveer la misma exactitud con una gran diferencia en costo computacional en cuanto al cálculo de la misma. Un método eficiente para obtener estas muestras es el *muestreo progresivo*.

Dada una población normalizada de tamaño N y media μ, el muestreo progresivo es el proceso que consiste en obtener una secuencia de M muestras aleatorias independientes de tamaño n, donde $n = k0, k0+1, k0+2, \ldots, N$. El valor de $k0$ se adapta para que la muestra inicial tenga un significado razonable; por ejemplo, no es conveniente que sea cero ya que no tiene sentido considerar una muestra de este tamaño. La media m de cada muestra estima la media de la población con el nivel de exactitud $A = 1 - |m - \mu|$.

Un componente central del muestreo progresivo es el programa de muestreo $S = \{n_0, n_1, n_2, \ldots, n_k\}$, donde $n_i < n_j$ para cada $i < j$. Cada n_i especifica el tamaño de la muestra que será probada en un algoritmo por inducción.

Si se extrae una muestra de tamaño N, su exactitud es 1. Para muestras de menor tamaño, A toma valores entre 0 y 1. Detalles adicionales de este desarrollo pueden ser revisados en [21].

El proceso del muestreo progresivo puede representarse como una curva de aprendizaje que establece la relación entre los valores de alguna métrica y el tamaño de la muestra. La Figura 2.2 exhibe esta relación.

El eje horizontal representa a n, que es el tamaño de la muestra, donde n varía entre 0 y N. El eje vertical representa los valores de la métrica calculada por el algoritmo de inducción para las diferentes muestras del programa.

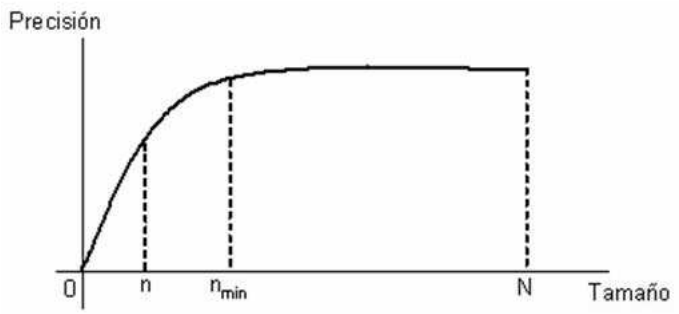

Figura 2.2: Curva de relación tamaño-precisión

Por lo general, este tipo de curvas constan de 3 partes: una parte inclinada, otra de media inclinación y una última que es una llanura. Esta última parte sucede cuando al incrementar el tamaño de las muestras no se presenta una mejora en la precisión. El punto de unión entre la parte media y la llanura se denomina n_{min}, y representa el tamaño óptimo de la muestra.

Para calcular el tamaño de las muestras, el muestreo progresivo hace uso de diferentes programas de muestreo. Dos programas muy utilizados son el programa aritmético y el geométrico.

El muestreo aleatorio permite extraer información de grandes volúmenes de datos, examinando únicamente una pequeña parte de ellos [22]. En aplicaciones computacionales, es una alternativa para minimizar el consumo de recursos requeridos para procesar un gran volumen de datos [20]. Sin embargo, para aplicar esta técnica se requiere estimar el tamaño de la muestra más adecuado para asegurar la eficiencia del proceso [21]. El muestreo progresivo propone examinar progresivamente muestras aleatorias de tamaño creciente, para no tomar instancias de un tamaño tal que no puedan ser resueltas [21, 20].

2.2.3 Programa de muestreo aritmético (PMA)

El PMA, propuesto por John y Langley [23], calcula el tamaño de las muestras como sigue:

$$PMA = n_\delta + (i \cdot n_\delta) = \{n_0, n_0 + n_\delta, n_0 + 2n_\delta, \ldots, N\}$$

donde n_0 es el tamaño inicial de la muestra, y n_δ es el incremento en el tamaño. Un ejemplo de este esquema es $\{100, 200, 300, \ldots, N\}$. Este método, aunque sencillo, tiene una seria desventaja: Considérese n_{min} como el tamaño mínimo para una muestra representativa. Si se conoce el valor de n_{min} y además es un múltiplo grande de n_δ, el programa requerirá muchas ejecuciones del algoritmo de inducción. Por ejemplo, si $n_{min} = 150,000$ y $n_0 = n_\delta = 100$, se requerirán 1500 ejecuciones del algoritmo.

2.2.4 Programa de muestreo geométrico (PMG)

El PMG es más rápido para encontrar el tamaño óptimo de n_{min} que el PMA. Este programa tiene la siguiente definición:

$$PMG = a_i \cdot n_0 = \{n_0, a \cdot n_0, a^2 \cdot n_0, \ldots, N\}$$

donde a es un factor de multiplicación del tamaño (usualmente con un valor de 2). Un ejemplo es $\{100, 200, 400, 800, 1600, 32000 \ldots, N\}$.

Sin embargo, suponiendo que $n_{min} = 150,000$ se requieren 12 iteraciones, con la desventaja de que no se obtiene una $n_{min} = 150,000$ sino una $n = 204,800$ que es un 36.53 % más grande que la n_{min}.

2.3 Muestreo secuencial aleatorio

Esta técnica se aplica principalmente en la extracción de muestras aleatorias en archivos secuenciales muy grandes. Consiste en extraer n elementos sin reemplazo de un archivo de N elementos; estos elementos deben aparecer en el mismo orden en que aparecen en el archivo y el tamaño de n se calcula por algún programa de muestreo.

Una implementación simple para lograr extraer estos elementos es el método S [24], el cual consiste en generar un número aleatorio uniforme para determinar si el elemento deber ser elegido o no para la muestra. Si m elementos ya han sido elegidos de entre los primeros t elementos, el elemento $(t + 1)$ se elegirá con una probabilidad de $(n - m)/(N - t)$. De esta manera, el método S requiere de N números aleatorios y se ejecuta con un tiempo de $O(N)$.

Una variante del método S es conocida como método A [25]. Este método acelera de manera significativa al método propuesto por Knuth [24], ya que se enfoca en determinar cuántos elementos deben ser *saltados* antes de elegir el siguiente elemento para la muestra.

Para explicar este método, se denota como $S(n, N)$ al número de registros a saltar (*distancia de salto*). Sean n y N el número de registros a ser escogidos para la muestra y el número de registros que no han sido procesados, respectivamente. El algoritmo de este método se muestra en la Figura 2.3.

{ Algoritmo A }
$top := N - n$,
$Nreal := N$;
mientras $n \geq 2$ **hacer**
    $V := RANDOM()$; $S := 0$; $quot := top/Nreal$;
    mientras $quot > V$ **hacer**
        $S := S + 1$; $top := top - 1.0$; $Nreal := Nreal - 1.0$;
        $quot := (quot \times top)/Nreal$;
    fin mientras
    Saltar los siguientes S registros y entonces elegir el siguiente registro para la muestra;
    $Nreal := Nreal - 1.0$; $n := n - 1$;
fin mientras
{Caso especial $n = 1$ }
$S := TRUNC(ROUND(Nreal) \times RANDOM())$;
Saltar los siguientes S registros y entonces elegir el siguiente registro para la muestra;

Figura 2.3: Algoritmo del método A

En este trabajo se propone el uso del muestreo secuencial aleatorio debido a su corto tiempo de ejecución para extraer muestras de grandes bases de datos, específicamente el método A propuesto en [25]. Los tamaños de las muestras que se extraen se controlan mediante un programa aritmético. Se utilizan dos criterios de convergencia: regresión lineal y similitud entre las muestras. Todos estos elementos se combinan para integrar una metodología de solución de instancias de gran escala del problema de la distribución de fragmentos.

2.4 Método de ramificación y acotamiento

El método de solución utilizado para las instancias del problema de la distribución en este trabajo es el de **Ramificación y Acotamiento** (*Branch & Bound*). Este método es un enfoque casi enumerativo que ha sido aplicado a una gran variedad de problemas combinatorios, además se ha utilizado con éxito en el caso de este problema [26].

La idea básica del método de ramificación y acotamiento es dividir un problema en subproblemas. Este proceso de división es llamado normalmente ramificación y su propósito es establecer los subproblemas que serán más fáciles de resolver que el problema original debido a su menor tamaño o a su estructura. La ramificación es representada generalmente en forma de estructura de árbol, como el la Figura 2.4 donde cada nodo i del árbol de búsqueda representa un subproblema P_i.

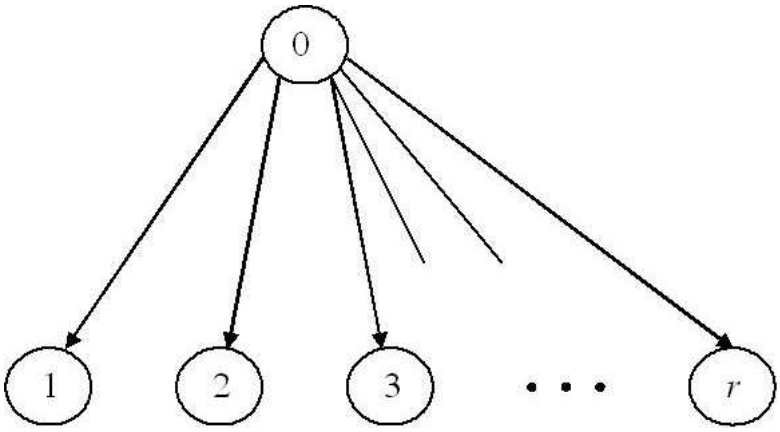

Figura 2.4: **Ramificación.**

Este árbol de búsqueda puede tener muchos niveles, con los nodos al fondo de la ramificación conocidos como nodos pendientes. El proceso de solución involucra una evaluación sistemática de los nodos pendientes del árbol de búsqueda. El proceso de evaluación consiste de tres componentes clave: ramificar, calcular cotas y explorar.

Ramificación

Claramente, si se pretende derivar la solución de un problema dado P_0, el conjunto de subproblemas de P_0 debe representar completamente a P_0. Para mayor claridad, sea $\{P_i\}$ el conjunto de soluciones enteras factibles del problema P_i. Entonces, si P_0 se divide en $P_1, P_2, ..., P_r$, se debe cumplir que:

$$\{P_0\} = \{P_1\} \cup \{P_2\} \cup \cdots \cup \{P_r\} \tag{2.1}$$

Además, en general, es más eficiente escoger los subproblemas $P_1, P_2, ..., P_r$ de manera que:

$$\{P_i\} \cap \{P_j\} = \emptyset \quad \text{para toda } i \neq j \tag{2.2}$$

Para comprender mejor el proceso de ramificación, considérese un problema P_0 con n variables, y suponga que una variable en particular, digamos x_k, puede tomar los valores enteros 0, 1, 2 o 3. Las dos ramificaciones que satisfacen las condiciones 2.1 y 2.2 se ilustran en la Figura 2.5.

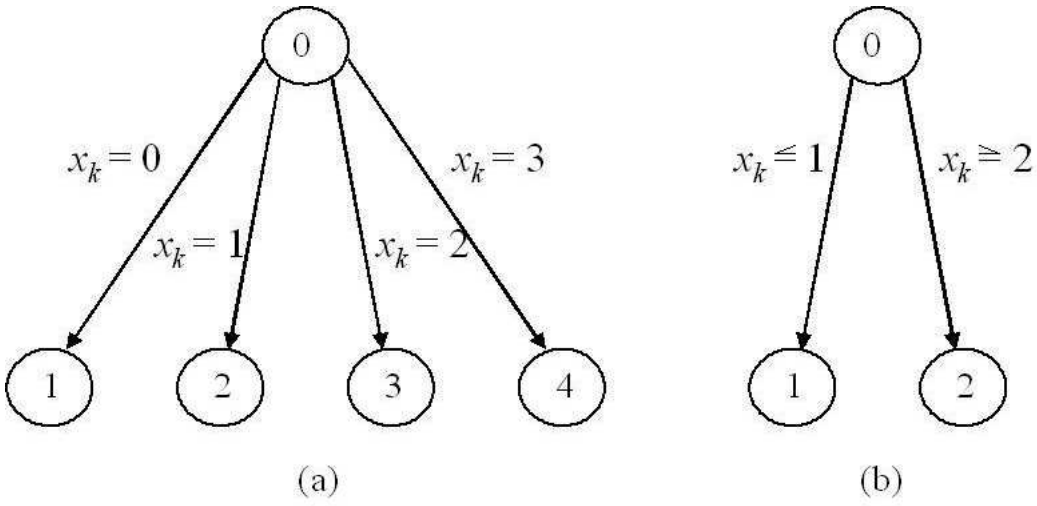

Figura 2.5: Diferentes estrategias de ramificación.

Por supuesto, estas son solo dos posibilidades de muchas más. Nótese que en la Figura 2.5a, se crean cuatro subproblemas, cada uno corresponde a fijar la variable x_k a uno de sus valores posibles. Debido a que x_k tiene ahora un valor fijo, cada uno de los subproblemas involucra $n-1$ variables.

Otra posibilidad de ramificación que satisface las condiciones 2.1 y 2.2 se encuentra en la Figura 2.5b. En este caso las restricciones adicionales $x_k \leq 1$ y $x_k \geq 2$ se utilizan para crear una división disjunta. En efecto estas condiciones especifican $x_k = 0, 1$ en el subproblema P_1 y a $x_k = 2, 3$ en el subproblema P_2.

Durante el proceso de ramificación, en esencia, estamos añadiendo restricciones a un problema particular para construir los subproblemas resultantes. Por consiguiente, la región factible de un subproblema es un subespacio de la región factible del problema padre.

Cálculo de cotas

Suponga que se conoce una solución entera factible de un problema de maximización entero. El valor objetivo proporcionado por esta solución es una cota inferior del valor objetivo del problema. Es decir, debido a que conocemos el valor de una solución factible en particular, estas seguros de obtener un valor óptimo al menos tan bueno como tal. Designamos esta cota inferior como z_L. Si se conocen varias soluciones factibles, entonces, z_L corresponde al valor más grande conocido. Esto es, z_L es la mayor cota inferior y la solución entera correspondiente a este valor es la solución actual, debido a que es la mejor solución entera conocida.

El propósito del cálculo de cotas superiores (en un problema de maximización) es determinar que tan buena puede ser la solución óptima en un nodo sin resolver el problema hasta ese nodo.

Exploración

Durante el proceso de ramificación y acotamiento, se intenta resolver cada uno de los subproblemas correspondientes a los nodos pendientes del árbol de búsqueda. Una vez que todos los subproblemas han sido resueltos, el problema se ha solucionado. Un subproblema puede ser eliminado de futura consideración en una de tres formas:

1. El subproblema produce una solución entera óptima. En este caso, se actualiza z_L y la solución actual si es necesario, y continua con el proceso de selección de nodos.
2. Se puede mostrar que el valor óptimo del subproblema no es mejor que la mejor solución encontrada hasta ese punto, Esto es hecho al calcular una cota del valor óptimo del nodo. Esta cota se compara con el valor de la solución actual.
3. El subproblema no es factible.

Se dice que un nodo (subproblema) que ha sido eliminado de toda futura consideración en una de estas formas es un nodo explorado y no se requiere más ramificación en ese nodo. Sin embargo, si no es posible explorar un nodo pendiente, entonces el subproblema asociado con ese nodo es dividido en subproblemas más pequeños mediante ramificación. El proceso se repite hasta que todos los nodos pendientes se han explorado.

2.5 Trabajos relacionados

2.5.1 Automatización del diseño de bases de datos

Existen dos tipos de estudios del problema de la automatización del diseño de bases de datos. Los trabajos tradicionales se basan en la simulación del manejador de bases de

datos [2]. Trabajos muy recientes realizan el estudio en el contexto de la implementación de herramientas para el diseño automático de la base de datos que interactúan directamente con el manejador de base de datos [27, 28]. En esta sección se describen los trabajos más relevantes basados en simulación y basados en aplicaciones reales.

Estudios basados en modelación

Ceri, **Navathe** y **Wiederhold** abordan el problema de determinar la ubicación óptima de un conjunto de fragmentos de relaciones en [29, 2]. Su enfoque de solución divide el problema en dos fases seriadas: fragmentación y ubicación. La idea central del enfoque serial es que esto permite trabajar con instancias menos duras que los generados con un enfoque integral. Sin embargo el enfoque serial no considera que muchas de las entradas de ambos procesos son las mismas [3]. En [30], se saca ventaja de este hecho y se demuestra que un enfoque integral es mucho más adecuado que el serial.

Como se señaló, **Pérez** presenta en [30] un nuevo enfoque en la solución del problema. Desarrolla un modelo de programación lineal entera, al que denomina **FURD** (Fragmentación, Ubicación, Reubicación y Distribución de Datos), en el que integra la fragmentación vertical y la ubicación no replicada de los fragmentos.

La función objetivo del modelo incorpora costos de las operaciones de lectura, costos de reunión de fragmentos, costos de migración y costos de almacenamiento. Reporta la solución de ejemplares de hasta 500 fragmentos, 500 sitios y 500 consultas, superando significativamente los resultados reportados con el enfoque serial.

Prueba que esta versión es un problema NP-Duro y propone una solución heurística utilizando Aceptación por Umbral [31, 32]. En [33], propone un método en línea para configurar los parámetros de control de un algoritmo de Aceptación por Umbral.

En [34, 35] reporta varias ampliaciones realizadas al modelo, entre las que incluye una

versión replicada del problema, y reporta la solución de instancias replicadas de hasta 100 atributos, 100 sitios y 100 consultas. Esta disminución, en la escala de las soluciones, muestra que es más difícil resolver las instancias de la versión replicada que las de la no replicada. En [36], reporta una solución del problema usando redes neuronales.

En [37], presenta un mecanismo para ajustar automáticamente los parámetros de control de un algoritmo genético de solución del problema del diseño de la distribución. Este trabajo de tesis se desarrolla en el contexto de las investigaciones de **Pérez Ortega** y el modelo FURD. Un resumen de estos trabajos se presenta en la Tabla 2.6.

Tabla 2.6: Trabajos relacionados con el modelo FURD.

Algoritmos	[26]	[38]	[39]	[40]	[41]	[42]	[43]	[44]	[45]	[46]	[47]	[48]
Ramificación y Acotamiento	✓									✓		
Aceptación por Umbral		✓		✓	✓							
Recocido Simulado			✓		✓							
Algoritmos Genéticos						✓						
Búsqueda Tabú							✓					✓
Aprendizaje Reforzado								✓				
SARSA									✓			
Redes Neuronales											✓	

Johansson aborda el problema de la ubicación replicada de datos para un sistema de bases de datos distribuido en una red alta velocidad [10]. Propone un modelo del tiempo de latencia de la red, para incorporarlo como uno de los componentes del tiempo de respuesta del sistema. Formula un modelo, con parámetros realistas, para optimizar el tiempo de respuesta de un sistema distribuido de transacciones paralelizadas que operan sobre datos replicados.

Reporta una simulación para evaluar el nivel de la reducción que se obtiene con transacciones paralelizadas, con un ejemplar con 1 dato, 3 sitios y una consulta. Caracteriza un escenario posible con una docena de sitios, cientos de tablas y miles de consultas.

Huang aborda el problema de la ubicación replicada de fragmentos en una red amplia [49]. Propone un modelo, que incorpora los costos de recuperación de los fragmentos, el costo de operaciones de lectura y de escritura y el costo del protocolo de conexión; y dos soluciones heurísticas de propósito específico. Reporta la solución de ejemplares aleatorios de 3 fragmentos, 4 sitios y 3 consultas y de 5 fragmentos, 4 sitios y 3 consultas. Compara sus resultados con los de **Lin** y **Orlowska** de [50].

Visinescu, aborda el problema de la ubicación dinámica de fragmentos replicados, en ambientes de Internet [1]. Propone un modelo que simula un sistema físico y una heurística exacta para su solución. El modelo minimiza el costo total del procesamiento de las consultas, considerando el costo de la recuperación, de la reunión y de la migración de los fragmentos. Considera un máximo de 4 réplicas y los parámetros del modelo permiten especificar ejemplares que simulan cargas típicas de Internet. La mayor instancia que reporta es de 100 fragmentos, 1000 sitios y 50,000 operaciones.

La cantidad de sitios reportados resulta ser excedente, debido a que la compañía Amazon Inc. utiliza una base de datos distribuida en 6 servidores solamente, un catalogo de 10 millones de productos y 4000 operaciones diarias [51]

Baiao, propone una metodología para la fragmentación de bases de datos distribuidas [8]. La metodología incluye técnicas para generar fragmentos horizontales, verticales y mixtos. Evalúa los esquemas de fragmentación que se producen, considerando el nivel de concurrencia en la ejecución de las operaciones. La calidad de las soluciones que se producen con el enfoque de simulación, depende de la fidelidad con que el modelo simule los procesos del manejador y depende también de la configuración realista de los parámetros. Tiene la ventaja de que permite aplicar criterios formales de optimalidad.

Estudios basados en aplicaciones industriales

En estos estudios se usa el procesador de operaciones del sistema para evaluar el costo de las alternativas de diseño [27, 28]. Estos trabajos se surgen como consecuencia de la estrategia, para desarrollar software autónomo, auto administrable y auto configurable, para las compañías desarrolladoras de SABDD's [52]. Realizan contribuciones importantes al estudio del problema de la automatización del diseño de bases de datos tales como la caracterización de nuevos objetos (tablas multidimensionales) y evaluaciones con pruebas estándar [53]. A continuación se describen algunos de los estudios basados en aplicaciones industriales más recientes.

Papadomanolakis, aborda el problema de la fragmentación de una base de datos de escala muy grande [54]. Presenta un algoritmo denominado **AutoPart**, que automáticamente genera el esquema de la fragmentación física de la base de datos, para una carga representativa de transacciones. Presenta resultados experimentales con una base de datos real, implementada en **Microsoft SQL Server 2000**. Los objetos de la base de datos que se fragmentan son tablas e índices.

Agrawal reporta una herramienta automática para el diseño físico de una base de datos [27]. Dada una carga de consultas, determina automáticamente los esquemas de fragmentación y ubicación física, dadas una carga de lecturas y actualizaciones y una fragmentación de la base de datos lógica. Actualmente sólo realiza el diseño de un nodo y están considerando extenderlo a múltiples nodos. Presentan resultados experimentales, sobre SQL Server, realizados a partir de pruebas estándar [53].

Zilio reporta una herramienta automática para el diseño de la distribución física, sobre múltiples servidores, de índices, vistas materializadas y tablas multidimensionales, para una carga de consultas dada [28]. La herramienta incorpora un módulo independiente para la compresión de la carga, con el propósito de incrementar su escalabilidad. Como el método de agrupación que utilizan es de baja escalabilidad, sólo lo aplican a la

compresión de una fracción de la carga definida por el usuario [55]. Para evaluar la calidad de los diseños utilizan la base de datos universal de DB2 y pruebas estándar [53].

La principal ventaja de estos enfoques es que operan directamente sobre el manejador de bases de datos, evitando los problemas derivados de la simulación del sistema. Su principal limitación es que, como no pueden aplicar técnicas de optimización, la evaluación de la calidad de la solución sólo se puede realizar mediante pruebas comparativas. Un aspecto relevante del trabajo de **Zilio** es que, identifica el tamaño de la carga como un factor clave que afecta la escalabilidad de las herramientas para el diseño automático de las bases de datos.

Análisis comparativo

La Tabla 2.7 describe las características más relevantes de los trabajos relacionados con el problema de la automatización del diseño de bases de datos. En esta comparación n está expresada por las operaciones.

Tabla 2.7: Trabajos relacionados con la automatización del diseño de la distribución.

Trabajo	**Fragmentación**	**Ubicación Replicada**	**Optimalidad Formal**	**Compresión de Instancias**	**Eficiencia de la Compresión**
Perez1999	✓	✓	✓		
Johansson2000		✓	✓		
Huang2001		✓	✓		
Visinescu2003		✓	✓		
Baiao2004	✓		✓		
Papadimitriou1998	✓				
Agrawal2004	✓				
Zilio2004	✓	✓		✓	$O(n^2)$

Se puede observar que las propuestas **Pérez** [30] y de **Zilio** [28] son las que abordan el problema de una manera más completa. Sin embargo, el enfoque de **Pérez** es insuficiente

ya que el tamaño de la mayor instancia soluble baja de 500 a 100 al incorporar la replicación a su modelo. En su trabajo, como todos los revisados en este contexto, *tiene la limitación de que no toma en cuenta que el tamaño de la carga es un factor determinante del rendimiento del proceso de automatización del diseño*.

En contraste **Zilio**, reconoce la relevancia de la compresión de las cargas, pero *no considera el impacto de la compresión en la calidad de las soluciones*. Por lo tanto, los métodos de compresión que propone, no garantizan la escalabilidad de las herramientas de diseño.

En [5] se considera que *ambos factores son críticos para la solución práctica del problema*, y se propone un método de compresión de instancias para incrementar la escalabilidad de las soluciones de instancias replicadas, en al menos un orden de magnitud, usando un modelo. Las principales limitaciones de este trabajo son que la métrica de similitud conjunta produce valores muy próximos a cero, lo que dificulta la detección de la convergencia. En este trabajo se exploran otras posibilidades de definir dicha métrica y un nuevo criterio de convergencia.

2.5.2 Muestreo progresivo

Los siguientes trabajos reportan los trabajos de investigación más relevantes que utilizan muestreo progresivo en diversas áreas.

Eldar reporta una aplicación de muestreo progresivo en el contexto de la transmisión de imágenes digitales [56]. Propone la transmisión de muestras aleatorias de tamaño creciente, extraídas de la imagen original, hasta lograr un nivel adecuado de definición de la imagen transmitida.

Stamatopoulos fundamenta estadísticamente el enfoque de muestreo progresivo considerando poblaciones finitas y muestras aleatorias independientes extraídas sin reem-

plazo [21]. El método propuesto se basa en la definición de un índice de proximidad entre las medias de las muestras y la media de la población.

Provost propone un modelo de aprendizaje automático basado en el enfoque de muestreo progresivo [20]. Parthasarthy aplica el modelo propuesto por Provost al problema de la identificación de reglas de asociación [57]. Baoua, con base en el modelo propuesto por Provost, aborda el problema de determinar el tamaño de la muestra inicial de los programas de muestreo [58].

Un aspecto esencial en el muestreo progresivo es la detección del punto de convergencia; sin éste, no se sabría hasta qué punto la muestra sería representativa sin utilizar la mayor parte de los datos del problema original.

John y Langley [23] utiliza un método de convergencia denominado PCE (*Probably Close Enough*, probablemente cercano). El método de PCE establece que la convergencia se alcanza cuando:

$$Pr((acc(N) - acc(n)) > \epsilon) \leq \delta$$

donde $acc(x)$ es la precisión del modelo que produce un algoritmo después de evaluar x instancias, ϵ es el decremento máximo aceptable en la precisión, y δ es la probabilidad de que la máxima diferencia de precisión sea excedida por cualquier ejecución individual.

Provost [20] utiliza el método de LRLS (*Linear Regression with Local Sampling*, Regresión lineal con muestreo local). Este método comienza en la última muestra calculada n_i, y muestrea r puntos adicionales en la vecindad de n_i. Estos puntos son posteriormente usados para estimar una línea de regresión lineal, cuya pendiente se compara con cero. La convergencia es alcanzada si la pendiente es suficientemente cercana a cero.

Este método es eficiente cuando se tienen comportamientos como el de la Figura 2.2, sin embargo puede ocurrir que la curva se describa como se muestra en la Figura 2.6,

en cuyo caso la muestra se quedara estancada antes de llegar al verdadero valor de la muestra mínima representativa.

Otro método es el propuesto por Stamatopoulos [21], que se basa en el uso de la similitud entre muestras. La similitud se define como sigue:
$A = 1 - |m - \mu|$
donde m y μ son las medias de la población y la muestra, respectivamente.

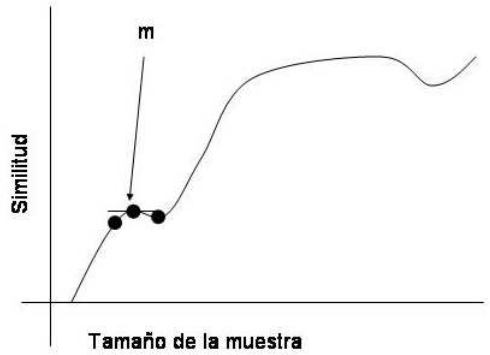

Figura 2.6: Curva de relación tamaño-precisión alterna

Este método toma ventaja del comportamiento de la curva de la Figura 2.6, debido a que se evalúa solamente la similitud, se obtiene la diferencia entre similitudes y no una pendiente de cero como se muestra en la Figura 2.7.

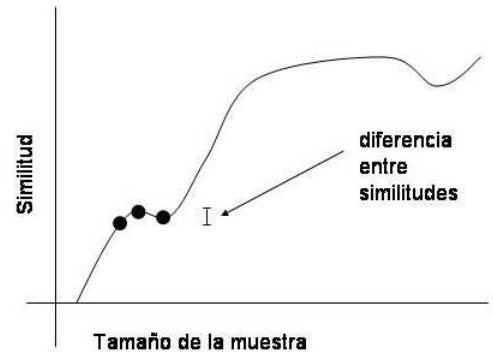

Figura 2.7: Curva de similitud

Un elemento crítico para la aplicación del muestreo progresivo, es la definición de la métrica de similitud entre muestras. La aplicación del muestreo progresivo a la determinación de patrones de consulta es una aplicación novedosa de esta técnica. Por otra parte, en el estudio del muestreo progresivo existen aún muchas preguntas abiertas, relacionadas con la eficiencia de los programas de muestreo, métricas de similitud, criterios de convergencia y el tamaño de la muestra inicial.

CAPÍTULO 3

Propuesta de solución

En este capítulo se presentan el modelo matemático y el método de compresión de instancias del problema del diseño de la distribución que son utilizados para describir la métrica de similitud y el criterio de convergencia utilizados en [5] y los propuestos en esta tesis.

3.1 Modelo matemático

Para describir el método de compresión, se utiliza el modelo del problema del diseño de la distribución propuesto en [6]. La función objetivo del modelo incluye cuatro términos: el primero modela el costo del procesamiento de las consultas de sólo lectura, el segundo el costo de las consultas de escritura, el tercero los costos de la replicación de los objetos y el cuarto son los costos de almacenamiento que se aplican a los objetos por permanecer en los sitios.

$$minz = \sum_k \sum_j f_{kj} \sum_m \sum_i q_{km} l_{km} c_{ji} w_{jmi} + \sum_k \sum_j f'_{kj} \sum_m \sum_i q'_{km} l'_{km} c_{ji} x_{mi} + \sum_j \sum_m \sum_i c'_{ji} d_{mi} w'_{jmi} + \sum_m \sum_i CA_i b_m x_{mi}$$

El problema es modelado utilizando programación lineal entera. En la Tabla 3.1 se describen los elementos utilizados en la formulación del modelo.

Tabla 3.1: Elementos del modelo de programación lineal

Elemento	Significado
no	Número de objetos a distribuir.
ns	Número de sitios en la red.
nq	Número de operaciones.
b_m	Tamaño en bytes del objeto m.
s_{km}	Selectividad de la operación de lectura k sobre el objeto m.
PA	Tamaño en bytes del paquete de comunicación.
F_{mi}	Frecuencia de uso del objeto m en el sitio i
f_{ki}	Matriz que describe la frecuencia de emisión de la operación de lectura k desde el sitio i, en un intervalo de tiempo dado.
q_{km}	Matriz de uso que indica que objetos utilizan las diferentes operaciones de lectura: $q_{km} = 1$ si la operación de la lectura k usa el objeto m, $q_{km} = 0$ en otro caso.
l_{km}	Paquetes de comunicación requeridos para transmitir la parte del objeto m, requerida para satisfacer la operación de lectura. k, $l_{km} = (b_m \times s_{km})/PA$.
f'_{ki}	Matriz que describe la frecuencia de emisión de la operación de escritura k desde el sitio i, en un intervalo de tiempo dado.
q'_{km}	Matriz de uso que indica que objetos utilizan las diferentes operaciones de escritura: $q'_{km} = 1$ si la operación de escritura k usa el objeto m, $q'_{km} = 0$ en otro caso.
P_k	Tamaño en bytes de la instrucción de escritura asociada a la operación de escritura k.
l'_k	Paquetes de comunicación requeridos para transmitir una instrucción de estructura $l'_k = p_k/PA$.

Elemento	Significado
d_{mi}	Paquetes de comunicación requeridos para migrar una réplica del objeto m, desde su ubicación actual hasta el sitio i.
CA_i	Costo de almacenamiento por byte en el sitio i.
CS_i	Capacidad de almacenamiento en bytes del sitio i.
c_{ij}	Matriz que contiene los costos de transmisión entre los sitios i y j.
A_{mi}	Ubicación original de los objetos a distribuir.
X_{mi}	Variable binaria que indica si el objeto m esta localizado en el sitio $i : x_{mi} = 1$ si el objeto m esta en el sitio i, en otro caso $x_{mi} = 0$.
W_{jmi}	Variable binaria que indica si el objeto m, localizado en el sitio i es requerido por una lectura ubicada en el sitio j, en otro caso $W_{jmi} = 0$.
W'_{jmi}	Variable binaria que indica que el objeto m, que debe ser ubicado en el sitio i, esta actualmente localizado en el sitio j. $W'_{jmi} = 1$ si el objeto m que debe ser ubicado en el sitio i, esta actualmente en el sitio j, $(j \neq i)$. En otro caso $W'_{jmi} = 0$.

Una solución del modelo debe satisfacer un conjunto de restricciones que especifican la posible replicación de los objetos, ubicación, la política de acceso aplicada en las operaciones de lectura y escritura, las condiciones para la migración de los objetos, y la capacidad de almacenamiento de los sitios. En la Tabla 3.2 se describen todas las restricciones del modelo.

Tabla 3.2: Restricciones del modelo de programación lineal.

Restricción	Significado
$\sum_i x_{mi} > 1 \forall m$	Cada objeto debe ser ubicado en al menos un sitio.
$x_{mi} \leq \sum_k q_{km}\theta_{ki}$ $\forall m, i$	Los objetos deben ser ubicados en un sitio i que ejecute al menos una operación que utilice al objeto. $\theta = 1 si \sum_k f_{ki}q_{km} > 0$, $\theta = 0 si \sum_k f_{ki}q_{km} = 0$.
$nsx_{mi} - \sum_j w_{jmi} \geq 0$	Solo es posible requerir desde otros sitios, los objetos ubicados en un sitio dado. Si un objeto no está en un sitio dado, este objeto no puede ser solicitado a dicho sitio desde ningún otro sitio.
$\sum_i w_{jmi} - \theta = 0$	Cuando un objeto replicado es requerido desde un sitio dado, solo una de sus réplicas debe ser usada para satisfacer la operación. Donde $\theta = 1$ si $\sum_k f_{kj}q_{km} > 0$ y $\theta = 0$ en cualquier otro caso.
$\sum_m x_{mi}b_m \leq CS_i$ $\forall i$	El espacio total usado para almacenar los objetos ubicados en un sitio, no debe rebasar la capacidad de almacenamiento del sitio.
$w'_{jmi} - A_{mi} \leq 0$ $\forall j, m, i$	Una réplica de un objeto debe ser generada a partir de una réplica del objeto existente en el esquema de ubicación previo.
$\sum_j w'_{jmi} -$ $(1 - A_{mi})x_{mi} = 0$ $\forall m, i$	Una nueva réplica de un objeto debe ser generada a partir de solamente una de las réplicas existentes en el esquema de ubicación previo.

3.2 Definición de matrices

Las matrices de uso de objetos por consultas y frecuencia de uso de consultas por sitios están definidas como:

$q_{km} = q$[CONSULTAS][OBJETOS]

$f_{kj} = f$[CONSULTAS][SITIOS]

q_{km}	t_1	t_2	...	t_{no}
q_1				
q_2				
$\vdots$				
q_{nq}				

f_{kj}	s_1	s_2	...	s_{ns}
q_1				
q_2				
$\vdots$				
q_{nq}				

La matriz q_{km} indica que consultas accesan a que objetos, el número de objetos puede llegar a ser superior a 1´000,000. El volumen máximo estimado de consultas es de este mismo rango.

Por otra parte la matriz f_{kj} contiene las frecuencias de emisión de las consultas desde los diferentes sitios. Un número máximo de 100 sitios es suficiente para representar casos reales.

3.3 Método de reducción de instancias

Este trabajo se ubica en el contexto de un método de reducción del tamaño de las instancias del problema del diseño de la distribución mediante muestreo de las consultas [5]. Este método consiste en obtener una muestra de tamaño apropiado para lograr que la solución de la instancia reducida produzca una buena aproximación de la solución óptima de la original. El resultado obtenido de la instancia reducida es diferente que el obtenido de la instancia original, debido a que la muestra no incluye todas las consultas y ni todos los costos. Sin embargo, se puede considerar que proporciona un resultado aproximado al óptimo con una disminución del costo del proceso. El grado de la aproximación depende de las características de los elementos estructurales del método.

Para utilizar esta técnica se definen los siguientes parámetros:

- *Probabilidad de acceso a un objeto.* Es la razón de el tamaño del objeto entre el tamaño total de los objetos.

$$p_{o_i} = \frac{Tam(o_i)}{\sum_{j}^{no} Tam(o_j)} \qquad \text{donde:} \quad \begin{array}{lll} Tam(o_i) & \rightarrow & \text{Tamaño del objeto} \\ nt & \rightarrow & \text{Número de objetos} \end{array} \tag{3.1}$$

- *Probabilidad de acceso de una consulta.* Razón de el número de accesos a objetos de una consulta entre el número de accesos totales a los objetos.

$$p_{c_i} = \frac{Acc(c_i)}{\sum_{j}^{nc} Acc(c_j)} \qquad \text{donde:} \quad \begin{array}{lll} Acc(c_i) & \rightarrow & \text{Num. accesos a la consulta } c_i \\ nc & \rightarrow & \text{Número de consultas} \end{array} \tag{3.2}$$

- *Probabilidad conjunta de una consulta de una muestra.* Es el producto de la suma de las probabilidades de acceso a los objetos que utiliza, por la probabilidad de acceso de la consulta.

$$P_{co_i} = p_{c_i} \cdot \sum_{j} P_{o_j} \qquad \forall f_j \in c_i \tag{3.3}$$

- *Consultas relevantes de una muestra.* Son aquellas cuya probabilidad conjunta es mayor o igual a $\lambda = p_{min} + \alpha(p_{max} - p_{min})$. Donde p_{min} y p_{max} son las probabilidades conjuntas mínima y máxima de las consultas de la muestra y $\alpha \in (0, 1)$ es un factor que permite controlar la frontera de la relevancia.

$$CR = \{c_i | P_{c_i} \geq \lambda\} \qquad \lambda = p_{min} + \alpha(p_{max} - p_{min}) \tag{3.4}$$

- *Representatividad de una muestra m.* Es la media de las probabilidades conjuntas de las consultas relevantes.

$$rm_i = \bar{X}(CR) \qquad \text{donde} \qquad \bar{X}(CR) = \frac{\sum_{c_i \in CR} P_{co_i}(c_i)}{\|CR\|} \tag{3.5}$$

- *Similitud entre dos muestras.* Se define como el recíproco del valor absoluto de la diferencia de sus representatividades.

$$y_a = sim(rm_a, rm_b) = 1 - |rm_a - rm_b| \tag{3.6}$$

- *Pendiente de la curva de aprendizaje.* Valor de la pendiente de la curva de aprendizaje del tamaño óptimo de la MMR. Se utiliza la estimación lineal para el cálculo de esta pendiente, utilizando los datos del tamaño de la muestra y su representatividad.

$$pen(y_a, y_{a+1}, y_{a+2}) = \frac{n(\sum_a^n (tam_a \cdot y_a)) - (\sum_a^n tam_a)(\sum_a^n y_a)}{n(\sum_a^n tam_a^2) - (\sum_a^n tam_a)^2} \tag{3.7}$$

  donde y_a, y_{a+1} y y_{a+2} son las últimas 3 similitudes calculadas, mediante la Ecuación 3.6 y tam_a es el tamaño de la muestra obtenida relativa a las similitudes

- *Factor de ajuste de la pendiente.* Valor de ajuste de la pendiente, denominado como ε. Se calcula mediante el logaritmo base 10 de la pendiente estimada.

$$\varepsilon = 10^{factor} \qquad factor = \lfloor \log(pen(y_j, y_{j+1}, y_{j+2})) \rfloor \tag{3.8}$$

- *Pendiente ajustada.* Valor de la pendiente calculada mediante la Ecuación 3.7, a la que se le aplica el factor de ajuste de la pendiente (Ecuación 3.8).

$$pen'(y_a, y_{a+1}, y_{a+2}) = \frac{n(\sum_a^n (tam_a \cdot y_a)) - (\sum_a^n tam_a)(\sum_a^n y_a)}{n(\sum_a^n tam_a^2) - (\sum_a^n tam_a)^2} \cdot \varepsilon \tag{3.9}$$

3.4 Proceso de obtención de la muestra mínima representativa (MMR)

3.4.1 Métricas de similitud

La parte medular de este trabajo de tesis se enfoca al análisis del proceso para determinar la muestra mínima representativa. Este proceso utiliza los parámetros definidos

anteriormente, con el fin de determinar la curva de aprendizaje, y posteriormente el tamaño óptimo de la MMR. Uno de los elementos estructurales más importantes de este método es la métrica de similitud entre muestras, la cual es utilizada para especificar el criterio de parada del proceso.

El algoritmo utiliza el programa de muestreo 'aritmético' para calcular el siguiente tamaño de muestra, la métrica de similitud de probabilidad conjunta y el criterio de convergencia por pendiente de la curva de similitud. El esquema general se muestra en la Figura 3.1.

Este esquema es modificado para obtener una comparación con un algoritmo con otra métrica de similitud. En términos generales la siguiente métrica se le llama de probabilidad de consulta y lo que intenta principalmente es eliminar la probabilidad del objeto P_o al igual que la probabilidad conjunta P_{co} y solo tomar en cuenta la probabilidad de acceso de la consulta P_c. El algoritmo es completamente igual al anterior solamente se eliminan las probabilidades aquí mencionadas y por lo tanto no son evaluadas dichas probabilidades.

3.4.2 Criterios de convergencia

Otro elemento fundamental de la estructura del método es el criterio de convergencia que se aplica. El algoritmo anterior utiliza un criterio de convergencia basado en la pendiente de la curva de similitud (curva de aprendizaje). Este criterio consiste en calcular la pendiente de la curva de similitud que va cambiando conforme se extraen las muestras programadas. El proceso se detiene cuando el cambio en la similitud es mínimo, es decir, cuando la pendiente de la curva tiende a cero[5]. A fin de calcular la pendiente de la curva de simlilitud en un punto, se utiliza estimación lineal considerando las últimas tres muestras.

El método anterior tiene la limitación de que puede obtener una pendiente menor o

Entrada: Instancia a ser reducida mediante muestreo.
Salida: Tamaño de la muestra mínima representativa (MMR).
Algoritmo con métrica de similitud de probabilidad conjunta y criterio de convergencia de pendiente de la curva de similitud:

1. **Mientras** el número de muestras sea menor de 3 **hacer**
   1.1. Obtener un muestra aleatoria de tamaño n
   1.2. Calcular los parámetros P_o, P_c y P_{co} para la muestra
   1.3. Determinar las consultas que formarán parte del conjunto relevante de la muestra, verificando que igualen o superen el umbral de relevancia (ver Ecuación 3.4)
   1.4. Obtener la representatividad mediante las probabilidades conjuntas de las consultas relevantes
   1.5. $n = 20 + n$
2. **Fin mientras**
3. Calcular las similitudes entre las muestras 1 y 2, y 2 y 3.
4. Estimar linealmente la pendiente de las similitudes del paso 2, mediante la Ecuación 3.7
5. Estimar la tolerancia de la pendiente ε mediante la Ecuación 3.8
6. Ajustar la pendiente por medio de ε.
7. **Mientras** pendiente >1 y Tamaño_Muestra <(Consultas_Totales / 2) **hacer**
   7.1. Seguir los pasos 1.1 hasta 1.4
   7.2. Calcular la similitud entre la muestra obtenida y la anterior
   7.3. Estimar linealmente la pendiente, mediante la Ecuación 3.7
   7.4. $n = 20 + n$
8. **Fin mientras**

Figura 3.1: Algoritmo de obtención del tamaño de la MMR

igual a cero de forma prematura. Esto se produce debido que los valores de similitud tienden a ser muy pequeños. En esta sección se describe el uso de un nuevo criterio de

convergencia al que se le denomina criterio de similitud.

Esta no es la única forma de obtener una pendiente menor o igual a cero, se puede dar el caso que se muestra en la Figura 3.2.

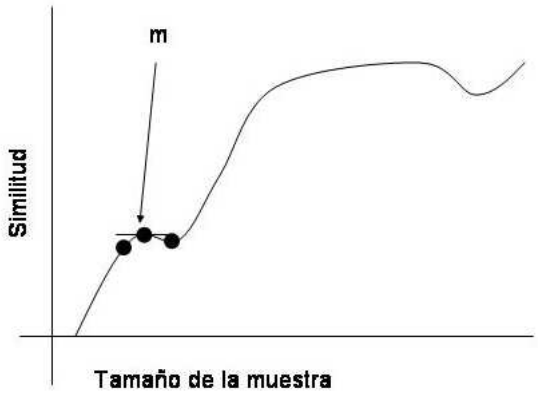

Figura 3.2: Pendiente de la curva de similitud.

Debido a lo anterior se diseño e implementó un criterio de convergencia que pudiera dar más precisión en la determinación de la similitud entre las muestras, por lo cual se propone utilizar comparaciones según la variación en la similitud entre las muestras, véase la Figura 3.3.

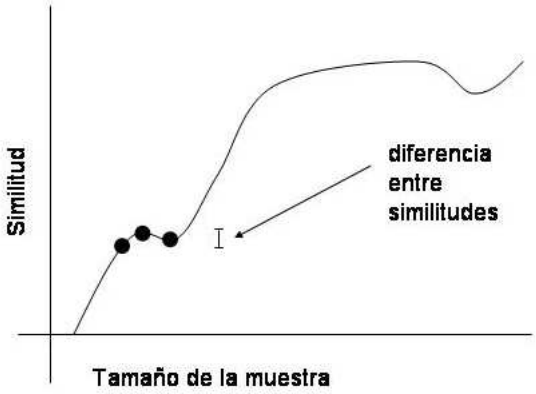

Figura 3.3: Similitud.

Se generan tres similitudes continuas y se evalúa que el valor mayor de las similitudes menos el valor menor multiplicado por cien sea menor a un valor de tolerancia β.

De tal forma que lo importante es que las similitudes sean próximas unas a otras, y en cuanto se observa que el cambio entre similitudes es pequeño se detiene el incremento de la muestra. Como se puede observar en la Figura 3.4.

Algoritmo con probabilidades conjuntas y criterio de convergencia de similitud:

1. **Mientras** el número de muestras sea menor de 3 **hacer**
   1.1. Obtener un muestra aleatoria de tamaño n
   1.2. Calcular los parámetros P_o, P_c y P_{fc} para la muestra
   1.3. Determinar las consultas que formarán parte del conjunto relevante de la muestra, verificando que igualen o superen el umbral de relevancia (ver Ecuación 3.4)
   1.4. Obtener la representatividad mediante las probabilidades conjuntas de las consultas relevantes
   1.5. $n = 20 + n$
2. **Fin mientras**
3. Calcular las similitudes entre las muestras 1 y 2, y 2 y 3.
4. Estimar la diferencia máxima entre las similitudes $|S_{max} - S_{min}| * 100$
5. **Mientras** la diferencia máxima $>beta$ y Tamaño_Muestra <(Consultas_Totales / 2) **hacer**
   5.1. Seguir los pasos 1.1 hasta 1.4
   5.2. Calcular la similitud entre la muestra obtenida y la anterior
   5.3. Estimar la diferencia máxima entre las similitudes $|S_{max} - S_{min}| * 100$
   5.4. $n = 20 + n$
6. **Fin mientras**

Figura 3.4: Algoritmo de obtención del tamaño de la MMR

3.5 Ejemplo: Cálculo del tamaño de la MMR con probabilidad conjunta y criterio de convergencia de pendiente de la curva de similitud.

A partir de una instancia de 20 objetos, 10 sitios y 400 consultas. Con un tamaño inicial de muestra de 10 consultas se realiza una muestra aleatoria secuencial inicial. Dicha muestra incrementa en tamaño de 20 en 20 hasta encontrar el tamaño óptimo de la muestra también llamada muestra mínima representativa (MMR).

Muestra inicial:

Consultas	20	31	111	145	169	215	236	258	318	323

3.5.1 Cálculo de la representatividad de la muestra

Para esta muestra se calcula la probabilidad de acceso a las consultas, la probabilidad de acceso a los objetos y la probabilidad conjunta.

Se calcula la probabilidad de acceso a cada consulta, esta se obtiene dividiendo el total de accesos de esa consulta entre el total de accesos de todas las consultas, como muestra es la siguiente ecuación, dando el resultado de la primera consulta.

$$P_{c20} = \frac{Acc(c20)}{\sum_{j}^{nc} Acc(c_j)} = \frac{2}{1527} = 0.001310$$

La consulta C20 se realiza 2 veces en todos los sitios y el total de consultas realizadas es de 1527. Dado que el tamaño de la muestra es de 10 se tiene que calcular diez veces esta ecuación. En la Tabla 3.3 se encuentran las demás probabilidades de la consulta.

Consulta	Probabilidad de acceso a las consultas
20	0.001310
31	0.001310
111	0.001310
145	0.000655
169	0.001310
215	0.002620
236	0.000655
258	0.001965
318	0.001965
323	0.001310

Tabla 3.3: Cálculos de la probabilidad de acceso de las consultas de la muestra

Ahora se calcula la probabilidad de cada objeto. Asumiendo que todos los objetos son de tamaño 1, se puede calcular que la probabilidad es como sigue para todos los objetos:

$$P_{o_i} = \frac{Tam(o_i)}{\sum_{j}^{no} Tam(o_j)} = \frac{1}{20} = 0.0500$$

Posterior a esto se obtiene la probabilidad conjunta, el número de probabilidades estimadas depende del tamaño de la muestra, una por consulta seleccionada.

Tomamos el primer elemento de la muestra para realizar esta operación. Esta consulta utiliza siete objetos por lo tanto se multiplica el valor de la probabilidad de objeto por 7. La Tabla 3.4 muestra los cálculos para el resto de las consultas.

$$P_{cf_{20}} = P_{c_{20}} \cdot \sum_{i} P_{o_i} = 0.001310 \cdot (7)(0.0500) = 0.001310 \cdot 0.35 = 0.000458$$

En este punto existe un problema el cual podemos ver claramente en las probabilidades conjuntas. Al multiplicar una probabilidad por otra se obtiene un valor menor a las probabilidades originales. Esto produce que al calcular la pendiente, esta sea mas próxima a a cero. Lo cual fue uno de los motivos para generar otra métrica de similitud, al igual que otro criterio de convergencia.

Consulta	Probabilidad conjunta
20	0.000458
13	0.000262
111	0.000196
145	0.000033
169	0.000196
215	0.000393
236	0.000131
258	0.000196
318	0.000786
323	0.000262

Tabla 3.4: Cálculos de la probabilidad conjunta de las consultas de la muestra

A continuación se determinan las consultas que serán relevantes en la muestra. Como primer paso se determinan la probabilidad máxima y la probabilidad mínima de las consultas de la muestra. En el ejemplo, estas probabilidades pertenecen a c_{318} (0.000786) y c_{145} (0.000033). Como se dijo antes, se tiene un valor de α que tomará un valor entre 0 y 1 para determinar el comportamiento del conjunto relevante. Si el valor se aproxima a 0 entonces la cantidad de consultas tomadas como relevantes será muy pequeña dejando de lado algunas consultas importantes, por otra parte si es más próximo el valor a 1 se tomarán demasiadas consultas incluyendo consultas que no son relevantes. Para este caso se tomó un valor de α de 0.5. El valor del umbral de relevancia será:

$$u_r = p_{min} + \alpha(p_{max} - p_{min}) = 0.000033 + 0.5(0.000786 - 0.000033) = 0.000409$$

Se obtiene un pequeño conjunto de las consultas de la muestra que se le define como 'Conjunto Relevante'. Pertenecen a este conjunto aquellas consultas que igualen o sobrepasen al umbral, siendo estas las que contienen más consultas realizadas de diferentes sitios; en la siguiente tabla se muestran las consultas que se consideran relevantes:

Ahora se calcula la representatividad de la muestra, se realiza un promedio con los

Consulta	Probabilidad
20	0.000458
318	0.000786

Tabla 3.5: Consultas que forman parte del conjunto relevante

valores del Conjunto Relevante y se obtiene:

$$r_{m_1} = \frac{0.000458 + 0.000786}{2} = 0.000622$$

Este valor es indispensable dado que con este se calcula la pendiente y esta sirve para determinar la Muestra Mínima Representativa (MMR).

3.5.2 Obtención de la MMR

Como ya se mencionó, el valor de la MMR se determina a través de una curva de la pendiente que requiere dos valores de similitud.

Por lo tanto, deben extraerse al menos 3 muestras de tamaños diferentes incrementando el tamaño de la muestra de 20 en 20, esto para poder obtener dos similitudes y obtener la MMR.

Al final tenemos una tabla con los tamaños de muestra y sus representatividades como sigue:

Tamaño	Representatividad
10	0.000622
30	0.000655
50	0.000533

Tabla 3.6: Representatividad de las muestras

Cálculo de la similitud

Una vez que se tienen las representatividades de las 3 primeras muestras, obtener dos similitudes, una con la representatividad 1 y 2 y otra con la representatividad 2 y 3.

El proceso es el siguiente:

$$y_1 = sim(m_1, m_2) = 1 - |r_{m_1} - r_{m_2}| = 1 - |0.000622 - 0.000655| = 0.999967$$
$$y_2 = sim(m_2, m_3) = 1 - |r_{m_2} - r_{m_3}| = 1 - |0.000655 - 0.000533| = 0.999878$$

Una vez obtenida la similitud se procede a calcular la pendiente, esta se estima linealmente mediante el método de mínimos cuadrados (definido por la ecuación 3.7).

$$\begin{aligned}
pen(y_j, y_{j+1}) &= \frac{n(\sum_i^n (tam_i \cdot y_i)) - (\sum_i^n tam_i)(\sum_i^n y_i)}{n(\sum_i^n tam_i^2) - (\sum_i^n tam_i)^2} \\
pen(y_1, y_2) &= \frac{2[(10(0.999967) + 30(0.999878)] - [(10+30)(0.999967 + 0.999878)]}{2(10^2 + 30^2) - (10+30)^2} \\
&= \frac{2(9.99967 + 29.99634) - (40)(1.98372)}{2(100+900) - 1600} \\
&= \frac{2(39.99601) - (79.3488)}{2(1000) - 1600} \\
&= \frac{0.64322}{400} \\
&= 0.00160805
\end{aligned}$$

Dado que el valor de la pendiente es muy pequeño, es conveniente realizar un ajuste para incrementar el valor de la misma. A continuación se muestra el ajuste de la pendiente:

$$factor = \lfloor \log m \rfloor$$
$$\varepsilon = 10^{factor}$$
$$factor = \lfloor \log(0.00160805) \rfloor = \lfloor -2.7937004 \rfloor = -3$$
$$\varepsilon = 10^{-3}$$
$$\varepsilon = .001$$

Se divide la pendiente obtenida entre el valor de ajuste y se obtiene un valor de pendiente igual a 1.60805. El cual es mayor a 1 y por lo tanto se continúa con el muestreo. Ahora se calculan más muestras y sus pendientes para obtener 3 similitudes y así generar las nuevas pendientes con más datos de similitud.

La siguiente muestra es de 70 elementos y obtenemos como resultado una representatividad de .000566. Una vez con este dato se calcula la similitud con la representatividad de la muestra anterior

$$r_{m_4} = 0.000566$$
$$y_3 = sim(m_3, m_4) = 1 - |r_{m_3} - r_{m_4}| = 1 - |0.000533 - .000566| = 0.999967$$

y ahora se calcula la pendiente con tres similitudes

$$\begin{aligned}
&pen(y_1, y_2, y_3) \\
&= \frac{3[(10(0.999967)+30(0.999878)+50(0.999967)]-[(10+30+50)(0.999967+0.999878+0.999967)]}{3(10^2+30^2+50^2)-(10+30+50)^2} \\
&= \frac{3(9.99967 + 29.99634 + 49.99835) - (90)(2.999812)}{3(100 + 900 + 2500) - 8100} \\
&= \frac{3(89.99436) - (269.98308)}{3(3500) - 8100} \\
&= \frac{269.98309 - 269.98308}{2400} \\
&= \frac{0.00001}{2400} \\
&= 0.00000000416
\end{aligned}$$

Aplicando el factor de ajuste obtenemos que es una pendiente de 0.00000416 lo cual es menor que 1. Por lo tanto detenemos el incremento de la muestra y tomamos por muestra representativa la que se encuentra en medio de los 3 tamaños. Los tamaños de muestra son 10, 30, 50 por lo tanto tomamos el tamaño de muestra de 30 consultas.

3.6 Ejemplo: Cálculo del tamaño de la MMR con probabilidad de consulta y criterio de convergencia de similitud.

Al igual que el ejemplo anterior, se utilizará la misma instancia con los mismos datos, 20 objetos, 10 sitios y 400 consultas.

Se genera la muestra inicial de 10 elementos:

Consultas	20	31	111	145	169	215	236	258	318	323

3.6.1 Cálculo de la representatividad de la muestra.

En este caso solo se calcula la probabilidad de acceso a las consultas.

Este cálculo se realiza como lo muestra la eq. 3.2, se divide el total de accesos de esa consulta entre el total de accesos de todas las consultas.

Esto se puede apreciar con el siguiente ejemplo:

$$P_{c20} = \frac{Acc(c20)}{\sum_{j}^{nc} Acc(c_j)} = \frac{2}{1527} = 0.001310$$

En la siguiente tabla se muestra el resto de las probabilidades de acceso a las consultas.

Como siguiente punto, se determinan las consultas pertenecientes al 'conjunto relevante', para esto se requiere de la probabilidad mínima y la probabilidad máxima.

En este ejemplo estas probabilidades estan dadas por c_{150} y c_{239}.

El valor de α es igual a 0.5, por lo tanto el valor del umbral de relevancia será.

Consulta	Probabilidad de acceso a las consultas
12	0.001310
24	0.001310
96	0.001310
150	0.000655
155	0.001310
239	0.002620
326	0.000655
330	0.001965
370	0.001965
375	0.001310

Tabla 3.7: Cálculos de la probabilidad de acceso de las consultas de la muestra

$$Ur = P_{min} + \alpha(P_{max} - P_{min})$$

$$Ur = 0.000655 + 0.5(0.002620 - 0.000655)$$

$$Ur = .0016375$$

El conjunto relevante queda como se muestra a continuación

Consulta	Probabilidad
239	0.002620
330	0.001965
370	0.001965

Tabla 3.8: Consultas que forman parte del conjunto relevante

En este momento se calcula la representatividad de la muestra mediante un promedio de las probabilidades del conjunto relevante.

$$r_{m_1} = \frac{0.002620 + 0.001965 + 0.001965}{3} = 0.0021833$$

3.6.2 Obtención de la MMR

En este caso no se realiza ningún cálculo con la curva de la pendiente. Sin embargo se deben de obtener 4 representatividades para obtener 3 similitudes.

En el primer ciclo del algoritmo solo se tienen 3 representatividades las cuales generan 2 similitudes. Esto indica que la tercera similitud tiene un valor de 0. La diferencia máxima de esta similitud con cualquiera de las anteriores no mejora el valor de β, este valor es propuesto para detener el incremento de la muestra.

Por lo tanto se genera una cuarta representatividad y por lo tanto una tercera similitud.

Los valores de las representatividades son los siguientes.

Tamaño	Representatividad
10	0.0021833
30	0.0035270
50	0.0014829
70	0.0038592

Tabla 3.9: Representatividad de las muestras

La similitud se calcula como se muestra en la eq.3.6.

Tamaño	Similitud
10	0.9986563
30	0.9979559
50	0.9976237

Tabla 3.10: Similitud entre las muestras

Una vez teniendo estos datos se calcula la diferencia máxima entre las 3 primeras similitudes.

$$DifMax = |y_{max} - y_{min}| * 100$$

$$DifMax = |0.9986563 - 0.9976237| * 100$$

$$DifMax = 0.10326$$

Si esta diferencia máxima es mayor que el valor de β propuesto, se genera otra representatividad y otra similitud. Se calcula nuevamente esta diferencia máxima con los últimos 3 valores de similitud hasta que esta diferencia sea menor o igual al valor β propuesto, que en este caso es 0.1.

Tamaño	Representatividad
90	0.0035486

Tabla 3.11: Nueva representatividad de la nueva muestra

Tamaño	Similitud
30	0.9979559
50	0.9976237
70	0.9996894

Tabla 3.12: Similitud entre las muestras

$$DifMax = |y_{max} - y_{min}| * 100$$

$$DifMax = |0.9996894 - 0.9976237| * 100$$

$$DifMax = 0.20657$$

Se repite el proceso hasta llegar a:

Tamaño	Representatividad
90	0.0035486
110	0.0036482
130	0.00396741

Tabla 3.13: Representatividades siguientes

Tamaño	Similitud
70	0.9996894
90	0.9999004
110	0.99968079

Tabla 3.14: Similitud entre las muestras

$$DifMax = |y_{max} - y_{min}| * 100$$
$$DifMax = |0.9999004 - 0.99968079| * 100$$
$$DifMax = 0.021961$$

Este método genera un tamaño de muestra mayor al anterior, sin embargo más exacto y esto puede ser modificado dependiendo del valor designado a β.

CAPÍTULO 4

Implementación

En este capítulo se describe la implementación de las propuestas presentadas en el capítulo anterior. Se presenta un diagrama general del proceso de experimentación y los detalles del mismo.

4.1 Diagrama general

A continuación se presentan los pasos realizados en el proceso de la experimentación. La Figura 4.1 muestra la organización general del proceso.

Se trabaja con tres algoritmos diferentes, la metodología utilizada para realizar las pruebas es la misma para cada uno de ellos y los archivos de entrada y salida tienen el mismo formato.

Como se muestra en la Figura 4.1, primero se genera una instancia con ciertos parámetros que son introducidos en un archivo guión.

Un ejemplo de archivo guión se muestra en la tabla 4.1:

Los índices i y j hacen referencia a sitios, mientras que m son objetos y k son consultas

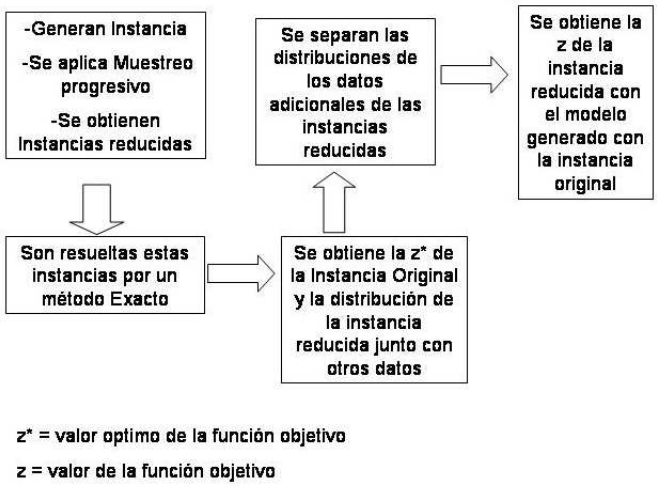

Figura 4.1: Diagrama General

// nomArch, tamInicial, TAM, CU, LEC, PROB, SEL, PK, TRAS, BLOK, CAP, COSTO, CANTIDAD
d_20_3_100,10,1,1,1,20,1,25,8,76,1,0,30,
d_20_5_200,10,1,1,1,20,1,25,8,76,1,0,30,
d_30_7_300,10,1,1,1,20,1,25,8,76,1,0,30,
d_50_10_1000,10,1,1,1,20,1,25,8,76,1,0,30,
d_20_3_1000,10,1,1,1,20,1,25,8,76,1,0,30,
d_40_6_2000,10,1,1,1,20,1,25,8,76,1,0,30,

Tabla 4.1: Ejemplo de archivo guión

- nomArch, tamInicial, TAM, CU, LEC, PROB, SEL, PK, TRAS, BLOK, CAP, COSTO, CANTIDAD
- nomArch = 'd_m_j_k' (m objetos, j sitios y k consultas).
- tamInicial = Tamaño inicial de la muestra.
- TAM = Tamaño en kilo bytes del objeto.
- CU = Costo de actualización.
- LEC = Numero a colocar en F_{kj} como frecuencia de consultas realizadas por un sitio.

- PROB = La probabilidad en Q_{km} de que una consulta accesa a los objetos y en F_{kj} de que una consulta sea realizada por los sitios.
- SEL = Selectividad de la consulta K_j.
- PK = Tamaño en kilo bytes de la instrucción de actualización.
- TRAS = Costo de Transmisión entre sitios.
- BLOK = Tamaño del bloque de comunicación en kilo bytes.
- CAP = Capacidad del sitio en mega bytes.
- COSTO = Costo de almacenamiento por kilo byte en cada sitio.
- CANTIDAD = La cantidad de instancias que se generarán.

4.1.1 Datos del programa de muestreo

A partir de los datos especificados en el guión, se procede a crear una instancia que cumpla con las características mencionadas. Esta instancia se utiliza como la instancia original de la que se extraen las muestras.

El programa que genera las instancias produce tres resultados diferentes:

El primero es un conjunto de instancias generadas a partir de las especificaciones del guión. Por ejemplo, si se tiene un guión como el que se muestra en la Tabla 4.2.

// nomArch, tamInicial, TAM, CU, LEC, PROB, SEL, PK, TRAS, BLOK, CAP, COSTO, CANTIDAD
d_20_3_1000,10,1,1,1,20,1,25,8,76,1,0,3,
d_40_6_2000,10,1,1,1,20,1,25,8,76,1,0,2,

Tabla 4.2: Ejemplo de archivo guión que producirá 3 instancias del primer tipo y 2 del segundo

d1_20_3_1000.txt
d2_20_3_1000.txt
d3_20_3_1000.txt
d1_40_6_2000.txt
d2_40_6_2000.txt

Tabla 4.3: Instancias producidas por el programa de muestreo

Se generan los archivos de instancias completas mostrados en la Tabla 4.3, dichas instancias son generadas con las características indicadas por los parámetros del archivo guión.

También se obtiene un conjunto de muestras de las instancias anteriores (una por cada instancia) tal como se muestra en la Tabla 4.4.

rd1_20_3_1000.txt
rd2_20_3_1000.txt
rd3_20_3_1000.txt
rd1_40_6_2000.txt
rd2_40_6_2000.txt

Tabla 4.4: Muestras de las instancias generadas

El último producto de este programa es un archivo que servirá de guión para la entrada del programa que resolverá las instancias originales y las muestras de las mismas. El archivo tiene la estructura mostrada en la Tabla 4.5.

d1_20_3_1000.txt
rd1_20_3_1000.txt
d2_20_3_1000.txt
rd2_20_3_1000.txt
d3_20_3_1000.txt
rd3_20_3_1000.txt
d1_40_6_2000.txt
rd1_40_6_2000.txt
d2_40_6_2000.txt
rd2_40_6_2000.txt

Tabla 4.5: Muestras de las instancias generadas

4.1.2 Esquema del programa de muestreo

Este programa realiza los siguientes pasos:

1. Lee el archivo guión que contiene los parámetros para crear las instancias.
2. Genera una instancia según los parámetros del guión.
3. Se le aplica muestreo secuencial aleatorio a esta instancia original con un tamaño predefinido. Posteriormente se genera otra muestra con un tamaño mayor (muestreo progresivo) y según las condiciones de parada se detiene el crecimiento de la muestra y se obtiene una MMR.
4. Se genera una muestra con el tamaño de la MMR para la instancia analizada.
5. Se escriben los nombres de los archivos generados en el archivo de guión de salida.
6. Se repite el proceso desde el punto 1 hasta 5 hasta que se termina el archivo guión y se generan todas las instancias.

4.1.3 Detalle del esquema del programa de muestreo

Paso 1 El programa lee el guión, el cual tiene la estructura siguiente:

```
// nomArch, tamInicial, TAM, CU, LEC, PROB, SEL, PK, TRAS, BLOK, CAP, COSTO,-
 CANTIDAD
d_5_3_10,10,0,1,1,20,1,25,8,76,1,0,1,
```

Paso 2. El programa lee el guión y según las características construye una instancia.
A continuación se muestra un ejemplo de instancia generada:

```
// Tuplas, Sitios, Consultas
5,3,10,
// Tamaño en kbytes del fragmento
126,203,215,183,151,
// Costo de actualizacion CU
1,
//Matriz de uso Qkm[CONS][TUPLAS]
0,0,0,1,0,
1,0,0,0,0,
0,0,0,1,0,
0,1,1,0,1,
1,0,0,0,0,
0,0,0,0,1,
0,0,0,0,1,
0,0,1,0,0,
1,0,0,1,0,
0,0,0,0,1,
// Uso de atributos de escritura Q'km
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
//Matriz de frecuencia de CONSULTAS Fki[CONS][SIT]
0,0,1,
1,0,0,
0,0,1,
1,0,0,
1,0,0,
1,0,0,
0,0,1,
0,0,1,
0,0,1,
1,0,0,
// Frecuencia de la escritura k en el sitio j ( f'kj )
0,0,0,
0,0,0,
0,0,0,
```

```
0,0,0,
0,0,0,
0,0,0,
0,0,0,
0,0,0,
0,0,0,
0,0,0,
// Selectividad de la consulta k S(k), 0 - 100
1,1,1,1,1,1,1,1,1,1,
// Tamaño en bytes de la instruccion de actualización. 0 - 100 de un kilo byte
25,25,25,25,25,25,25,25,25,25,
// Costo de transmisión entre sitios
0,8,8,
8,0,8,
8,8,0,
// Tamaño del bloque de comunicacion en kbytes 0 - 100
76,
// Capacidad del sitio en kbytes (0 - 1024) kbytes
1024,1024,1024,
// Ubicación de inicio
1,0,0,
1,0,0,
1,0,0,
1,0,0,
1,0,0,
// Costo de almacenamiento por kb en cada sitio (0 - 5)
0,0,0,
```

Paso 3. En este punto se obtiene una muestra de la instancia generada.

Paso 4. Un ejemplo de instancia producida por muestreo es la siguiente:

```
// Tuplas, Sitios, Consultas
5,3,5,
// Tamaño en kbytes del fragmento
126,203,215,183,151,
// Costo de actualizacion CU
1,
//Matriz de uso Qkm[CONS][TUPLAS]
```

```
0,0,0,1,0,
0,0,0,1,0,
1,0,0,0,0,
0,0,0,0,1,
1,0,0,1,0,
// Uso de atributos de escritura Q'km
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
//Matriz de frecuencia de CONSULTAS Fki[CONS][SIT]
0,0,1,
0,0,1,
1,0,0,
0,0,1,
0,0,1,
// Frecuencia de la escritura k en el sitio j ( f'kj )
0,0,0,
0,0,0,
0,0,0,
0,0,0,
0,0,0,
// Selectividad de la consulta k S(k), 0 - 100
1,1,1,1,1,
// Tamaño en bytes de la instruccion de actualización. 0 - 100 de un kilo byte
25,25,25,25,25,
// Costo de transmisión entre sitios
0,8,8,
8,0,8,
8,8,0,
// Tamaño del bloque de comunicacion en kbytes 0 - 100
76,
// Capacidad del sitio en kbytes (0 - 1024) kbytes
1024,1024,1024,
// Ubicación de inicio
1,0,0,
1,0,0,
1,0,0,
1,0,0,
1,0,0,
// Costo de almacenamiento por kb en cada sitio (0 - 5)
0,0,0,
```

Paso 5 Se genera un archivo de salida con los nombres de las instancias producidas, como el siguiente:

```
d1\_5\_3\_10.txt
rd1\_5\_3\_10.txt
```

4.1.4 Muestreo progresivo

En este proceso se obtiene la muestra mínima representativa (MMR) de una instancia dada utilizando la técnica de muestreo progresivo.

El muestreo progresivo consiste en la generación de muestras progresivamente de mayor tamaño. Se evalúa la representatividad de cada una de las muestras y luego se obtienen un valor de similitud entre la representatividad de cada dos muestras.

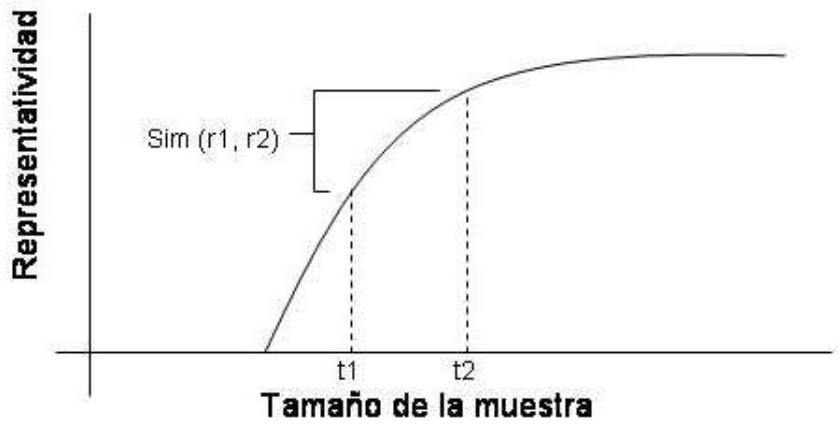

Figura 4.2: Curva de la representatividad

Este proceso es detenido de diferente forma según el criterio de convergencia. Este criterio es el que indica que una muestra tiene un tamaño apropiado para considerarla representativa.

4.1.5 Muestreo secuencial aleatorio

La eficiencia del método que se describe depende críticamente de la eficiencia del método que se utilice para extraer las muestras. El método de muestreo secuencial aleatorio propuesto en [25], el cual permite extraer una muestra aleatoria de la instancia original en un tiempo de a lo más n con respecto de las consultas de la instancia.

En el muestreo aleatorio secuencial se generan de forma ascendente los valores de las consultas que serán tomadas en cuenta para la muestra. Esta selección de renglones se realiza mediante el método A descrito en [25], el cual se muestra en la Figura 2.3.

Una vez seleccionados los renglones se realiza una lectura de la instancia y se toman solamente los renglones seleccionados, véase la Figura 4.3.

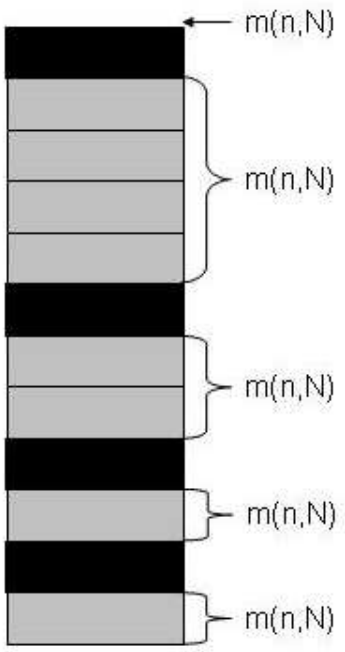

Figura 4.3: Muestreo aleatorio secuencial

4.1.6 Datos del programa de solución

Una vez terminado el proceso anterior se resuelven las instancias originales y las instancias correspondientes a las muestras extraídas.

Para resolver las instancias se usa un programa que invoca al método exacto de ramificación y acotamiento del motor de solución de LINDO 2.0. El cual recibe como entrada las instancias y muestras generadas al igual que un archivo guión con los nombres de los mismos.

Este programa regresa un archivo con el nombre de la instancia, el valor de Z óptimo, tamaño del original en Kb, tiempo de solución del original, tamaño de la muestra y tiempo de solución de la muestra. No se genera el valor de z de la muestra debido a que no tiene significado con respecto a la instancia original. La instancia reducida es diferente a la original, y le corresponde un modelo matemático diferente cuyas soluciones no son comparables.

A continuación se presenta un ejemplo de archivo de resultados de el programa que resuelve las instancias con el método exacto.

```
Nombre Zoptima Tamaño Orig Tiempo Orig Tamaño Muestra Tiempo Muestra
d1_3_3_60.txt 57026.312500 3504 0.010000 704 0.010000
d2_3_3_60.txt 52223.683594 3504 0.020000 704 0.010000
d1_3_3_90.txt 17276.314453 5184 0.010000 984 0.020000
d2_3_3_90.txt 22223.683594 5184 0.010000 704 0.010000
d3_3_3_90.txt 19131.578125 5184 0.010000 1264 0.010000
```

Además de reportar estos resultados este programa tiene la capacidad de proporcionar la distribución resultante de las instancias muestreadas. El resultado es un archivo como el que se muestra a continuación.

```
PROBLEM NAME     salidas\md1_3_3_20.txt
```

```
MILP OPTIMUM FOUND

ITERATIONS BY SIMPLEX METHOD    =          0
ITERATIONS BY BARRIER METHOD    =          0
ITERATIONS BY NLP METHOD        =          0
TOTAL BRANCHES CREATED          =          0
TOTAL NUMBER OF LPs SOLVED      =          0
NUMBER OF CONTRA CUTS           =          0
NUMBER OF OBJECTIVE CUTS        =          0
NUMBER OF GUB CUTS              =          0
NUMBER OF LIFTING CUTS          =          0
NUMBER OF FLOW COVER CUTS       =          0
NUMBER OF GOMORY CUTS           =          0
NUMBER OF GCD CUTS              =          0
NUMBER OF CLIQUE CUTS           =          0
NUMBER OF DISAGGREGATION CUTS   =          0
NUMBER OF PLANT LOCATION CUTS   =          0
NUMBER OF LATTICE CUTS          =          0
NUMBER OF COEFF. REDUCTION CUTS =          0
TOTAL NUMBER OF CUTS GENERATED  =          0

OBJECTIVE FUNCTION VALUE

 1)                0.200001000

VARIABLE                VALUE                REDUCED COST

 w1_1_2                   0.000000000             0.000000000
 w1_1_3                   0.000000000             0.000000000
 w1_2_2                   0.000000000             0.000000000
 w1_2_3                   0.000000000             0.000000000
 w1_3_2                   0.000000000             0.000000000
 w1_3_3                   0.000000000             0.000000000
 w2_1_1                   1.000000000             0.000000000
 w2_1_3                   0.000000000             0.000000000
 w2_2_1                   1.000000000             0.000000000
 w2_2_3                   0.000000000             0.000000000
 w2_3_1                   1.000000000             0.000000000
 w2_3_3                   0.000000000             0.000000000
 w3_1_1                   1.000000000             0.000000000
 w3_1_2                   0.000000000             0.000000000
 w3_2_1                   1.000000000             0.000000000
 w3_2_2                   0.000000000             0.000000000
```

```
  w3_3_1                1.000000000              0.000000000
  w3_3_2                0.000000000              0.000000000
  wp1_1_2               0.000000000              0.000000000
  wp1_1_3               0.000000000              0.000000000
  wp1_2_2               0.000000000              0.000000000
  wp1_2_3               0.000000000              0.000000000
  wp1_3_2               0.000000000              0.000000000
  wp1_3_3               0.000000000              0.000000000
  x1_1                  1.000000000              0.000000000
  x1_2                  0.000000000              0.000000000
  x1_3                  0.000000000              0.000000000
  x2_1                  1.000000000              0.000000000
  x2_2                  0.000000000              0.000000000
  x2_3                  0.000000000              0.000000000
  x3_1                  1.000000000              0.000000000
  x3_2                  0.000000000              0.000000000
  x3_3                  0.000000000              0.000000000
  w1_1_1                1.000000000              0.000000000
  w2_1_2                0.000000000              0.000000000
  w3_1_3                0.000000000              0.000000000
  w1_2_1                1.000000000              0.000000000
  w2_2_2                0.000000000              0.000000000
  w3_2_3                0.000000000              0.000000000
  w1_3_1                1.000000000              0.000000000
  w2_3_2                0.000000000              0.000000000
  w3_3_3                0.000000000              0.000000000

 ROW              SLACK OR SURPLUS          DUAL PRICES

  2)                   0.000000000              0.000000000
  3)                   0.000000000              0.000000000
  4)                   0.000000000              0.000000000
  5)                   0.000000000              0.000000000
  6)                   0.000000000              0.000000000
  7)                   0.000000000              0.000000000
  8)                   0.000000000              0.000000000
  9)                   0.000000000              0.000000000
  10)                  0.000000000              0.000000000
  11)                  0.000000000              0.000000000
  12)                  0.000000000              0.000000000
  13)                  0.000000000              0.000000000
  14)                  0.000000000              0.000000000
  15)                  0.000000000              0.000000000
```

```
  16)                0.000000000                0.000000000
  17)                0.000000000                0.000000000
  18)                0.000000000                0.000000000
  19)                0.000000000                0.000000000
  20)                0.000000000                0.000000000
  21)                0.000000000                0.000000000
  22)                0.000000000                0.000000000
  23)              247.000000000                0.000000000
  24)              250.000000000                0.000000000
  25)              250.000000000                0.000000000
  26)                0.000000000                0.000000000
  27)                0.000000000                0.000000000
  28)                0.000000000                0.000000000
  29)                0.000000000                0.000000000
  30)                0.000000000                0.000000000
  31)                0.000000000                0.000000000
  32)                1.000000000                0.000000000
  33)                1.000000000                0.000000000
  34)                1.000000000                0.000000000

END OF REPORT
```

Por lo tanto una vez que se genera este archivo se requiere obtener la distribución de los objetos en un archivo independiente, para determinar el costo de esta distribución con el modelo matemático de la instancia original.

Una vez pasado este archivo por el programa que genera la distribución, arroja otro archivo con el siguiente resultado.

```
//comentario
3 3
1,0,0,
1,0,0,
1,0,0,
```

4.1.7 Obtención del costo de la distribución de la muestra

El programa que calcula este valor se le llama *Calcula z* y obtiene el costo de la distribución generada a partir de la instancia reducida, con el modelo matemático de la instancia original.

Este programa requiere de los siguientes datos, el archivo con la instancia original y el archivo con la matriz de distribución de objetos en sitios, generado con la instancia reducida.

Este programa arroja un archivo como el siguiente con los resultados de la función objetivo de la instancia reducida.

```
Instancia          Costo
rd1_3_3_60.txt      37486.842041
rd2_3_3_60.txt      55697.364990
rd1_3_3_90.txt      50013.156616
rd2_3_3_90.txt      37236.841675
rd3_3_3_90.txt      38578.946167
```

Finalmente se determina la diferencia del costo de las distribuciones generadas a partir de la instancia original y la reducida.

CAPÍTULO 5

Validación de la propuesta

En este capítulo se describen los experimentos realizados para analizar el impacto de las métricas de similitud y los criterios de convergencia en el rendimiento del método de compresión de instancias basado en muestreo progresivo.

En cada uno de los casos de prueba utilizados, se resolvieron instancias de diferentes tamaños con diferentes probabilidades de acceso de las consultas a los objetos. Para cada caso de prueba, se evalúan el error promedio que se genera y el porcentaje de reducción con respecto a la instancia original.

	Características				
Caso	Objetos (O)	Sitios (S)	Operaciones (Q)	Tamaño en bytes	Q/S
C_1	20	3	100	19,820	33
C_2	20	5	200	42,620	40
C_3	30	7	300	93,252	42
C_4	50	10	1000	492,680	100
C_5	20	3	1000	192,620	333
C_6	40	6	2000	754,252	333

Tabla 5.1: Casos para evaluar la transformación por muestreo progresivo.

Para realizar los experimentos se generaron 6 casos de prueba. Cada uno incluye 30 instancias aleatorias con las mismas características. La Tabla 5.1 describe las características utilizadas en cada uno de los casos e incluye el identificador del caso, el número

de objetos, sitios y operaciones, el tamaño de las instancias y la razón de operaciones a sitios.

Para realizar las pruebas se utilizó una computadora con procesador AMD Athlon XP 1.5GHz, 192MB de RAM, y 30Gb de disco duro. Para la solución exacta de las instancias se utilizó la API de LINDO 2.0, cuyas funciones se invocan desde un programa escrito en Visual C++ 6.0.

5.1 Experimento: Métrica de similitud de probabilidad conjunta

5.1.1 Objetivo

Evaluar el impacto de la métrica de de similitud basada en la probabilidad conjunta, en el desempeño del método de compresión. El criterio de convergencia utilizado es el de pendiente de la curva de similitud.

5.1.2 Análisis de resultados

La Tabla 5.2 muestra los resultados del experimento. La primera columna indica el caso que se resolvió (C). Los resultados se agrupan de acuerdo a la probabilidad de que una operación sea realizada en los sitios. Estas probabilidades se muestran en la parte superior de las columnas de resultados y debajo de cada una se presenta el porcentaje de error promedio (%E) al igual que el porcentaje de reducción promedio (%R) producido al transformar la instancia.

La Figura 5.1 muestra que las instancias pequeñas no tienen suficiente representatividad. Debido a que hay pocas consultas, los patrones de acceso prácticamente no se repiten. Por ejemplo: al entrevistar a 5 personas acerca de las preferencias de refresco

	Probabilidad 10 %		Probabilidad 20 %		Probabilidad 20 % TFA		Probabilidad 30 %	
EC	%E	%R	%E	%R	%E	%R	%E	%R
T20.3.100	297.8	71.0	126.2	72.3	122.9	74	17.1	70.7
T20.5.200	277.7	83.5	67.9	85.4	70.8	84.9	11.8	83.5
T30.7.300	229.8	88.9	43.9	88.9	44.0	88.9	5.9	88.5
T50.10.1000	66.6	96.0	10.7	96.1	3.1	95.4	1.8	96.3
T20.3.1000	60.5	96.7	8.7	96.7	12.0	97.1	21.4	95.6
T40.6.2000	38.0	98.3	7.7	98.4	1.5	97.7	82.2	97.8

	Probabilidad 40 %		Probabilidad 50 %	
EC	%E	%R	%E	%R
T20.3.100	5.2	71.7	0.3	72.3
T20.5.200	2.1	83.9	0.0	82.6
T30.7.300	0.6	88.7	0.0	88.5
T50.10.1000	52.6	94.9	140.3	95.5
T20.3.1000	82.0	96.0	161.8	96.1
T40.6.2000	210.6	98.1	384.4	97.1

Tabla 5.2: Resultados con Probabilidad conjunta

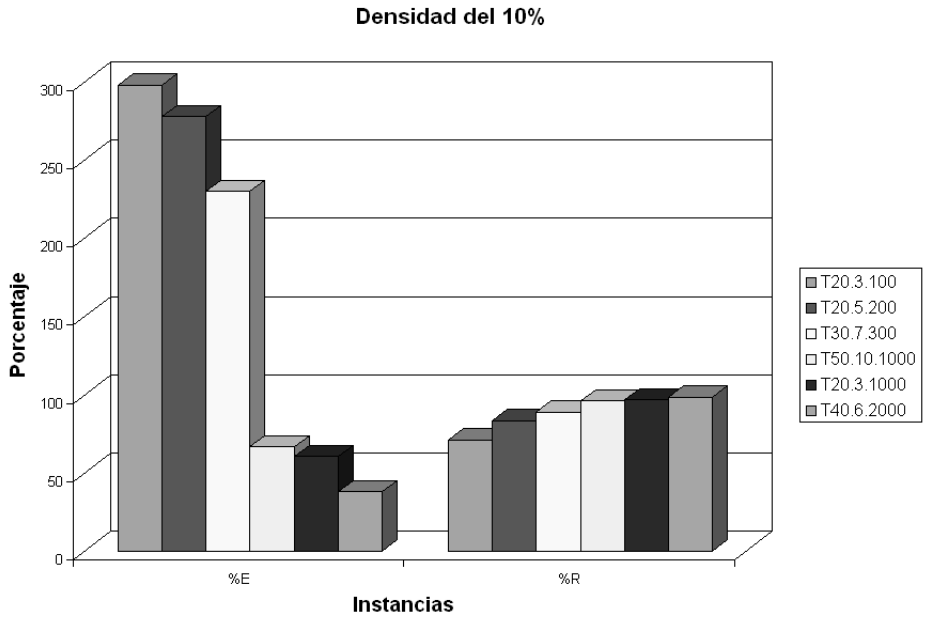

Figura 5.1: 10 % probabilidad de acceso

de cola, 4 personas prefieren coca cola y 1 persona prefiere pepsi cola, de esta entrevista sacamos una muestra y quitamos a uno de cada uno. Con estos datos se tiene que 3 personas prefieren coca cola y 0 pepsi cola. Se Puede calcular un porcentaje del $(3 * 100)/3 = 100\,\%$ que les gusta la coca cola, siendo que en realidad se obtiene que $(4 * 100)/5 = 80\,\%$ de la población prefiere coca cola.

Al aplicar este método a instancias con poca densidad, se obtiene un error de 38 % y una reducción de 98.3 %.

En la Figura 5.2 se presentan resultados de un 20 % de densidad con el tamaño del objeto aleatorio. Por lo tanto existen objetos que tendrán costos diferentes ya sea en migración replicación o las operaciones que se les apliquen.

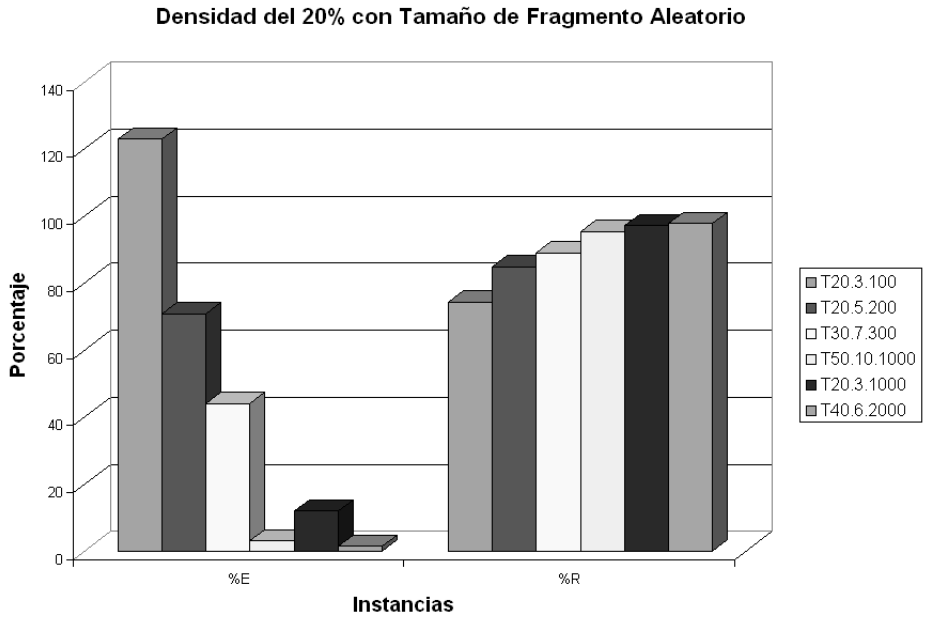

Figura 5.2: 20 % probabilidad de acceso con Tamaño de Fragmento Aleatorio

En este caso se puede observar una mejora, sin tener que alterar las probabilidades de las instancias. Esto debido a que se toma en cuenta el tamaño del fragmento en el modelo matemático, y no todas las consultas poseen el mismo costo, por lo tanto se puede prescindir de algunas consultas que aporten un costo mínimo dependiendo de su

tamaño de fragmento.

La Figura 5.3 muestra el conjunto de instancias con una probabilidad de que las consultas accedan a los objetos de 30 %.

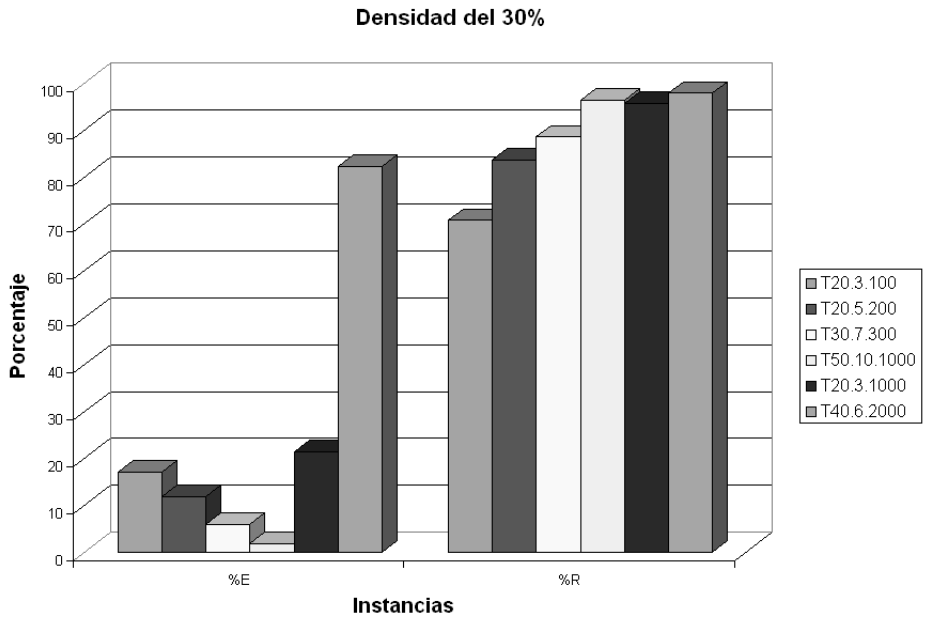

Figura 5.3: 30 % probabilidad de acceso

En este conjunto de instancias no se tiene un tamaño de objeto aleatorio. Tiene un comportamiento aceptable, a excepción de las las instancias grandes. Esto debido a un fenómeno que se ve más claro en la siguiente gráfica. En las instancias pequeñas se tiene un buen comportamiento. Esto se explica debido a que existe una distribución de datos mayor, no hay un sitio que sea dominante para algún objeto. De tal forma que en algunos casos al resolver la instancia no se migra ningún objeto y en su defecto migra pocos objetos a algún otro sitio.

Ahora se presentan los resultados obtenidos con instancias que poseen un 50 por ciento de densidad de distribución de objetos en consultas.

En la Figura 5.4 se puede ver más claramente lo que se mencionaba con anterioridad, las

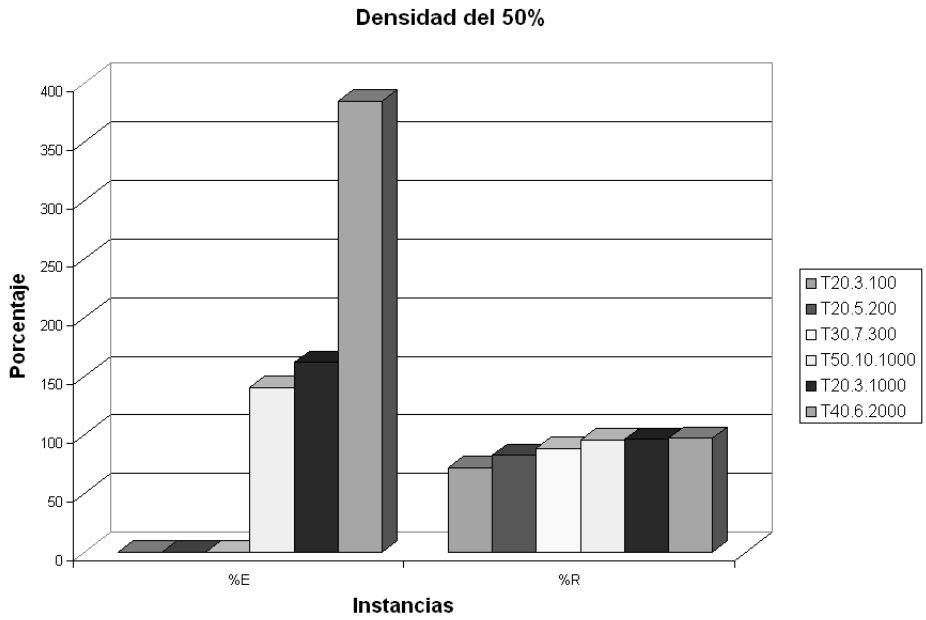

Figura 5.4: 50 % probabilidad de acceso

instancias pequeñas tienden a tener un error menor, debido a que no migran a ningún lugar los objetos. Sin embargo las instancias de tamaño mayor tienden a generar un error mayor entre más grandes sean, esto se explica de la siguiente forma.

Se tienen tantas consultas con tantos accesos a los objetos realizadas desde todos los sitios, que prácticamente todas las consultas accesan a casi todos los objetos. Al ser un numero elevado de consultas se tiene que todos los objetos son replicados a todos los sitios. Esto debido a que resulta más costoso realizar una consulta de muchos sitios que replicar los datos de un sitio a todos los demás. Debido a que la muestra es de tamaño pequeño, al no tener todos los costos de todas las consultas no replica a ningún sitio (se resuelve como los casos pequeños). Por lo tanto se produce una diferencia importante con respecto al valor óptimo, como consecuencia de que se tiende a replicar todos los objetos a todos los sitios.

5.2 Experimento: Métrica de similitud de probabilidad de acceso de la consulta

5.2.1 Objetivo

Evaluar el impacto de la métrica de de similitud basado en la probabilidad de acceso de la consulta, en el desempeño del método de compresión. El criterio de convergencia utilizado es el de pendiente de la curva de similitud.

5.2.2 Análisis de resultados

Los siguientes resultados son muy similares a los anteriores, debido a que se sigue utilizando la pendiente de la curva de similitud como criterio de convergencia para obtener una MMR.

La Tabla 5.3 muestra resultados completos de esta experimentación.

	Probabilidad 10 %		Probabilidad 20 %		Probabilidad 20 % TFA		Probabilidad 30 %	
EC	%E	%R	%E	%R	%E	%R	%E	%R
T20.3.100	369.0	71.4	142.2	72.3	93.9	73.5	21.9	72.3
T20.5.200	262.7	83.0	53.1	83.0	63.4	84.4	9.3	83.6
T30.7.300	219.8	88.0	31.9	88.3	48.3	89.9	6.1	88.9
T50.10.1000	62.5	95.7	9.5	95.3	4.8	96.0	1.5	95.9
T20.3.1000	39.1	95.9	10.2	96.2	11.0	96.2	20.1	95.7
T40.6.2000	33.2	98.3	6.8	97.6	1.4	98.2	82.7	98.0

En la Figura 5.5 hay una diferencia importante con respecto a las tablas anteriores de probabilidad conjunta, aquí se debe de tomar en cuenta que el tamaño del fragmento es aleatorio. La métrica de de similitud de probabilidad conjunta debe arrojar un mejor resultado, debido a que se toma en cuenta el tamaño del fragmento. Sin embargo esto no es así, ya que para instancias pequeñas, la métrica de similitud de probabili-

	Probabilidad 40 %		Probabilidad 50 %	
EC	%E	%R	%E	%R
T20.3.100	2.1	72.7	0.0	72.0
T20.5.200	2.2	84.4	0.3	84.3
T30.7.300	0.7	88.8	0.0	87.4
T50.10.1000	53.4	95.4	140.8	96.0
T20.3.1000	82.3	96.0	162.4	96.1
T40.6.2000	212.5	98.2	383.1	97.7

Tabla 5.3: Resultados con acceso de la consulta

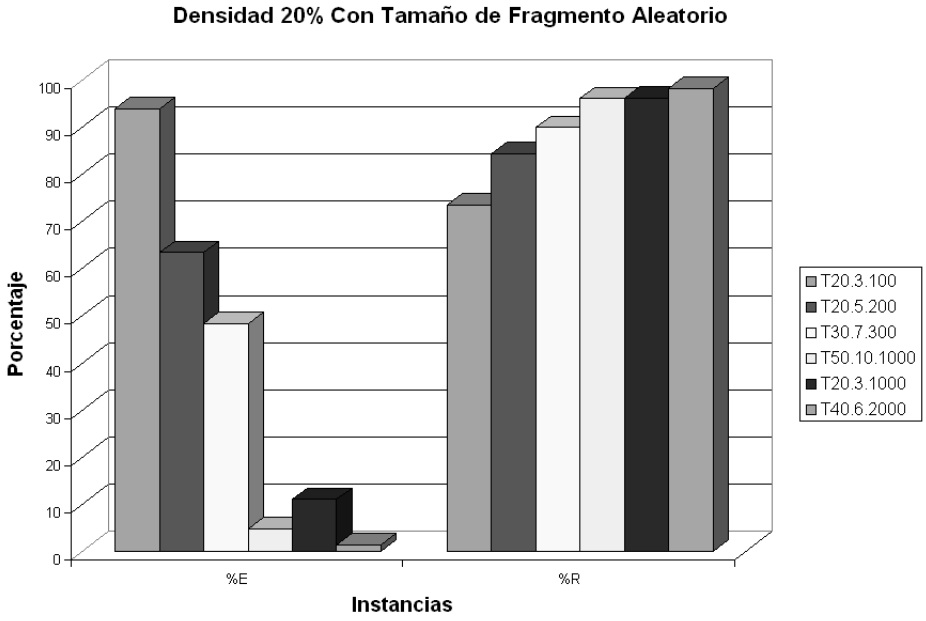

Figura 5.5: 20 % probabilidad de acceso con Tamaño de Fragmento Aleatorio

dad de consulta produce mejores resultados. En instancias grandes no hay un cambio significativo.

Esto se debe a que en la métrica de similitud de probabilidad conjunta se multiplican varias veces las probabilidades del tamaño del fragmento. Se obtienen valores menores que generan una representatividad menor numéricamente, por lo tanto una similitud menor, la cual tiende detener el proceso en forma prematura.

En el resto de los resultados existe una mejora en términos generales.

5.3 Prueba de similitud

5.4 Experimento: Criterio de convergencia de similitud

5.4.1 Objetivo

Evaluar el impacto del criterio de convergencia de similitud, en el desempeño del método de compresión, con la métrica de similitud de probabilidad de acceso de la consulta.

5.4.2 Análisis de resultados

El criterio de convergencia de similitud tiene una ventaja sobre el criterio de pendiente de la curva de similitud. Esta deriva de que al aplicar este criterio se cuenta con un parámetro para ajustar la tolerancia en la similitud entre muestras. De esta forma se evita el problema de la terminación anticipada y se producen muestras más representativas que las obtenidas con el criterio de pendiente de la curva de similitud.

Todas las pruebas que se presentan en esta experimentación fueron realizadas con un valor de tolerancia de 0.1. En la Tabla 5.4 se muestran los resultados de esta experimentación.

	Probabilidad 10 %		Probabilidad 20 %		Probabilidad 20 % TFA		Probabilidad 30 %	
EC	%E	%R	%E	%R	%E	%R	%E	%R
T20.3.100	178.5	48.4	12.6	48.4	16.2	49.0	0.0	48.4
T20.5.200	49.1	44.6	1.2	38.1	3.6	45.0	0.4	45.5
T30.7.300	27.8	42.8	1.9	53.7	4.3	51.1	0.1	49.8
T50.10.1000	36.2	89.0	2.4	91.3	2.1	91.2	0.3	91.0
T20.3.1000	15.2	91.8	1.4	91.4	0.7	92.9	19.4	92.9
T40.6.2000	22.2	96.7	2.6	97.0	2.8	97.1	80.8	97.1

	Probabilidad 40 %		Probabilidad 50 %	
EC	%E	%R	%E	%R
T20.3.100	0.0	48.4	0.0	48.4
T20.5.200	0.0	43.3	0.0	48.2
T30.7.300	0.0	56.5	0.0	52.6
T50.10.1000	52.8	92.3	140.2	92.3
T20.3.1000	82.2	93.8	162.9	94.4
T40.6.2000	212.5	97.2	383.4	97.2

Tabla 5.4: Resultados de similitud

En esta tabla se puede observar que los valores de error porcentual son menores a los reportados en los experimentos anteriores.

5.5 Comparaciones Globales

En esta sección se realizaron diversas comparaciones. Se pretende que se pueda ver de forma global el impacto de cada uno de los experimentos realizados.

Todas las instancias tienen un 20 % de densidad. En caso de evaluar métricas de similitud el criterio de convergencia es el mismo (pendiente de la curva de similitud). Al igual, si se comparan criterios de convergencia la métrica de similitud es la misma (probabilidad de acceso a las consultas).

En la gráfica 5.6 se presenta una comparación en el 'porcentaje de error' de la métrica de similitud: probabilidad conjunta y probabilidad de consulta.

En la gráfica 5.7 se presenta una comparación en el 'porcentaje de error' de el criterio de convergencia: pendiente de la curva de similitud y similitud.

En la gráfica 5.8 se presenta una comparación de el 'porcentaje de reducción' de la métrica de similitud: probabilidad conjunta y probabilidad de consulta.

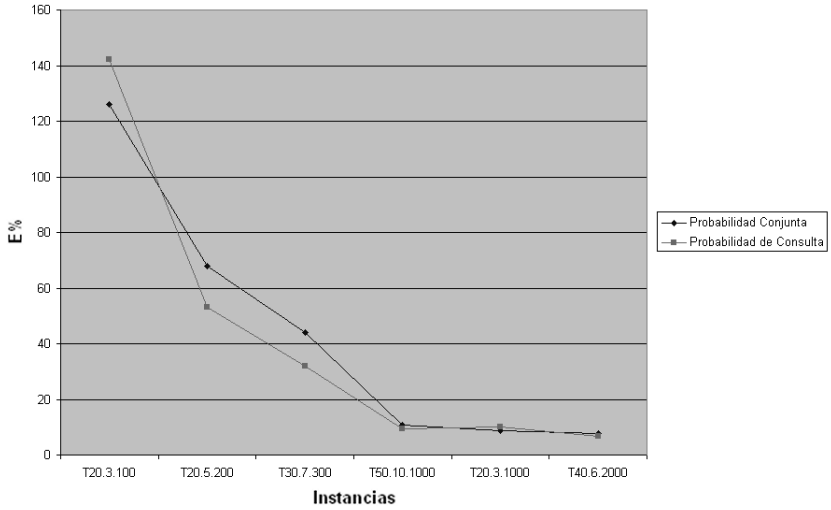

Figura 5.6: Error porcentual de métricas de similitud

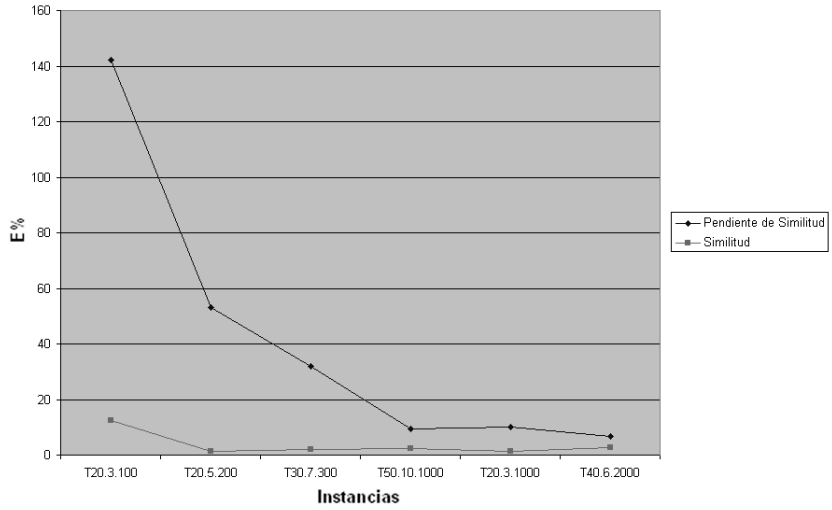

Figura 5.7: Error porcentual de criterios de convergencia

En la gráfica 5.9 se presenta una comparación de el 'porcentaje de reducción' de el criterio de convergencia: pendiente de la curva de similitud y similitud.

En estas pruebas se puede apreciar que las métricas de similitud no difieren mucho una de la otra, siendo mejor la métrica de similitud de probabilidad de acceso de la consulta. Mientras que en los criterios de convergencia existe una diferencia significativa en la calidad de las soluciones, siendo mejor el criterio de convergencia de similitud.

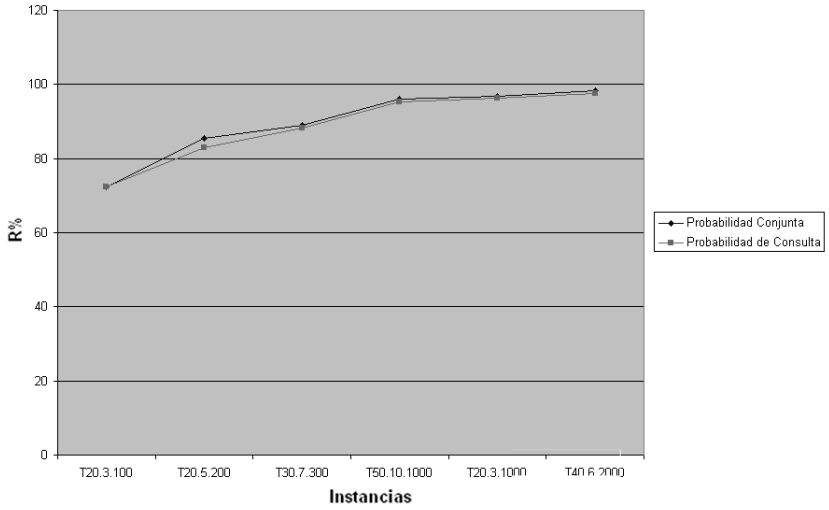

Figura 5.8: Porcentaje de reducción de métricas de similitud

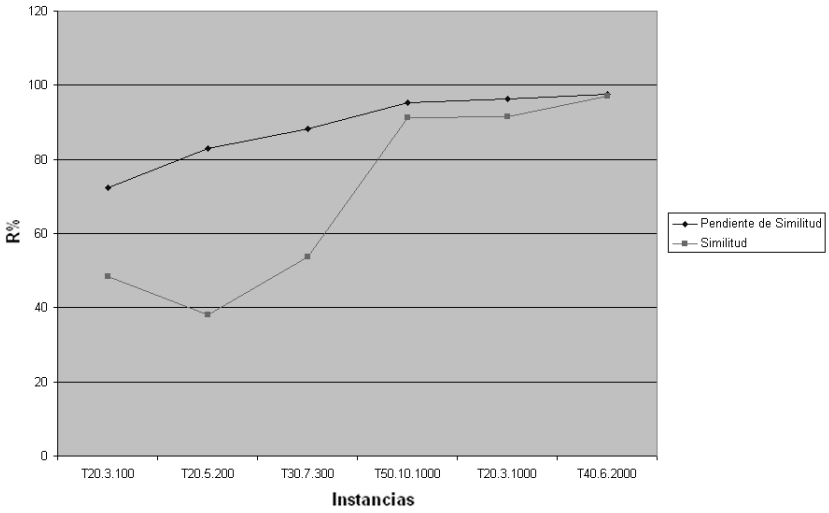

Figura 5.9: Porcentaje de reducción de criterios de convergencia

CAPÍTULO 6

Conclusiones y trabajos futuros

En este capítulo se presentan las aportaciones más relevantes de este trabajo de tesis y algunas posibles líneas de investigación que se pueden explorar en trabajos futuros.

6.1 Conclusiones

Las principales aportaciones de este trabajo son:

- Se aporta una nueva métrica de similitud basada en la probabilidad de acceso a las consultas. Esta tiene la ventaja de que implica menos trabajo computacional en el cálculo de la similitud de las muestras. La descripción detallada de esta métrica se ubica en la sección 3.4.1. Los resultados de los experimentos realizados con esta métrica son mejores que los alcanzados con la métrica basada en probabilidades conjuntas [5]. Los niveles de reducción y calidad son similares en ambas métricas.
- Se aporta un nuevo criterio de convergencia basado directamente en los valores de la similitud de las muestras. Este criterio, a diferencia del basado en la pendiente de la curva de similitud [5], permite evitar la terminación prematura del proceso y en consecuencia muestras con un mejor grado de representatividad. Los detalles

de este criterio se pueden encontrar en la sección 3.4.2. Tiende a generar muestras de mayor tamaño que las que se obtienen con el criterio de pendiente de la curva de similitud.

6.2 Trabajos futuros

Algunos de los trabajos futuros más relevantes relacionados con el enfoque aplicado en este proyecto son:

- *Desarrollo de nuevas métricas de similitud.* El desempeño del método de compresión basado en muestreo progresivo, depende críticamente de la métrica de similitud que se utilice. La métrica que se propone en este trabajo mejora la eficiencia del proceso, pero no genera una mejora significativa en el rendimiento general del método. Una posible línea de desarrollo es explorar la posibilidad de definir una nueva métrica que además de la frecuencia de las consultas, incorpore el patrón de acceso de las mismas.

- *Validación de la técnica con instancias reales.* Dado que las instancias aquí generadas son aleatorias, tienden a replicar todos los objetos a todos los sitios cuando hay una alta densidad de objetos solicitados por las consultas. Sin embargo en instancias reales no existan ese tipo de patrones de acceso, quizás sea prudente diseñar un programa para generar instancias reales que contengan patrones de acceso repetibles.

Bibliografía

[1] Visinescu, C.: Incremental data distibution on internet-based distributed systems: A spring system approach. Master's thesis, University of Waterloo, Ontario, Canada (2003)

[2] Ceri, S., Pelagatti, G.: Distributed databases principles & systems. McGraw Hill (1984)

[3] Oszu, M., Valduries, P.: Principles of Distributed Database Systems. Prentice-Hall, Englewood Cliffs, N. J. (1999)

[4] Agrawal, S., Narasayya, V., Yang, B.: Integrating vertical and horizontal partitioning into automated physical database design. In: SIGMOD Conference. (2004)

[5] Galván, E.: Diseño e implementación de algoritmos heurísticos para la solución de instancias de gran escala. Master's thesis, ITESM (2004)

[6] Fraire Huacuja, H.J.: Una Metodología para el Diseño de la Fragmentación y Ubicación en Grandes Bases de Datos Distribuidas. PhD thesis, CENIDET, Cuernavaca, Morelos, México (2005)

[7] Date, C.: Introducción a los sistemas de bases de datos. 7 edn. Pearson Education. Prentice Hall (2001)

[8] Baiao, F., Mattoso, M., Zaverucha, G.: A distribution design metodology for objects dbms. Distributed and Parallel Databases. Kluwer Academic Publishers **16** (2004) 45–90

[9] Tanenbaum, A., Oteen, M.: Distributed Systems: Principles and Paradigms. 1 edn. Prentice-Hall (2002)

[10] Johansson, J., March, S., Naumann, J.: The effects of parallel processing on update response time in distributed database design. In: Proceedings of the 21st International Conference On Information Systems. (2000) 187–196

[11] Akamai: Akamai technologies. http:///www.akamai.com (2004)

[12] Mirror: Mirror image internet. http:///mirror-image.com (2004)

[13] SinoCDN: Sinocdn. http:///www.sinocdn.com (2004)

[14] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: Knowledge discovery and data mining: Towards a unifying framework. In: Knowledge Discovery and Data Mining. (1996) 82–88

[15] Smith, B.D.: Induction as Knowloedge Integration. Phd thesis, University of Southern California (1995)

[16] García Martínez, R., Fritz, W., Blanqué, J.: Un algoritmo de aprendizaje de conceptos para sistemas inteligentes. In: V Congreso Nacional de Informática y Teleinformática, Buenos Aires, Argentina (1983) 91–96

[17] Mitchell, T.M.: Machine Learning. McGraw-Hill (1996)

[18] Michalski, R.S., Carbonell, J.G., Mitchell, T.M., eds.: Machine Learning: An Artificial Intelligence Approach. Volume 2. Morgan-Kauffman (1986)

[19] Michalski, R.S., Tecuci, G., eds.: Machine Learning: A Multistrategy Approach. Volume 4. Morgan Kauffman (1994)

[20] Provost, F.J., Jensen, D., Oates, T.: Efficient progressive sampling. In: Knowledge Discovery and Data Mining. (1999) 23–32

[21] Stamatopoulos, C.: Observations on the geometrical propeties of accuracy growth in sampling with finite populations. fao fisheries technical paper 388. Food and Agricultura Organization of the United Nations, Rome (1999)

[22] Ross, S.: Simulación. Prentice Hall (1999)

[23] John, G.H., Langley, P.: Static versus dynamic sampling for data mining. In Simoudis, E., Han, J., Fayyad, U.M., eds.: Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining, KDD, AAAI Press (1996) 367–370

[24] Knuth, D.: Seminumerical Algorithms. In: The Art of Computer Programming. 2 edn. Volume 2. Addison-Wesley, Reading,MA (1981)

[25] Vitter, J.S.: An efficient algorithm for sequential random sampling. ACM Trans. Math. Softw. **13** (1987) 58–67

[26] Velez, A.: Ubicación Óptima de datos en aplicaciones de bdds. Master's thesis, CENIDET, Cuernavaca, Morelos (1997)

[27] Agrawal, S., Surajit, C., Lubor, K., Arunprasad, M., Vivek, N., Manoj, S.: Database tuning advisor for microsoft sql server 2005. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases 2004, Toronto, Canada (2004) 1110–1121

[28] Zilio, D., Rao, J., Lightstone, S., Lohman, G., Storm, A., Garcia Arellano, C., Fadden, S.: Db2 design advisor: Integrated automatic physical database design. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases 2004, Toronto, Canada (2004) 1087–1097

[29] Ceri, S., Navathe, S., Wiederhold, G.: Distribution design of logical database schemes. In: IEEE Transactions on Software Engineering. Volume 9(4). (1983) 487–503

[30] Ortega, J.P.: Integración de la Fragmentación Vertical y Ubicación en el Diseño Adaptivo de Bases de Datos Distribuidas. Tesis doctoral, ITESM Campus Morelos (1999)

[31] Pérez, J., Pazos, R., Romero, D., L, C.: Análisis de complejidad del problema de la fragmentación vertical y reubicación dinámica en bases de datos distribuidas. In: 7mo Congreso Internacional de Investigación en Ciencias Computacionales, Ciudad Madero, Tamaulipas (2000) 63–70

[32] Pérez, J., Pazos, R., Frausto, J., Romero, D., Cruz, L.: Vertical fragmentation and allocation in distributed databases with site capacity restrictions using the threshold accepting algorithm. Lectures Notes in Computer Science. Springer-Verlag **1793** (2000) 75–81

[33] Pérez, J., Pazos, R., Vélez, L., Rodríguez, G.: Automatic generation of control parameters for the threshold accepting algorithm. Lectures Notes in Computer Science. Springer Verlag, Berlin Heidelberg New York **2313** (2002) 119–127

[34] Pérez, J., Pazos, R., Frausto, J., Romero, D., Cruz, L.: Data-object réplication, distribution and mobility in network environment. Lectures Notes in Computer Science. Springer Verlag, Berlin Heidelberg New York **2890** (2003) 539–545

[35] Perez, J., Pazos, R., Rodriguez, G., Fausto, J., Cruz, L., Fraire, H.: Replication and allocation management of data-objects in network environments. In: Proceeding of the IASTED International Conference on Computer Science and Technology, Cancún, México (2003) 388–392

[36] Pérez, J., Pazos, R., Fraire, H., Cruz, L., Pecero, J.: Adaptive allocation of data-object in the web using neural networks. Lectures Notes in Computer Science. Springer-Verlag **2829** (2003) 154–164

[37] Pérez, J., Pazos, R., Frausto, J., Rodríguez, G., Cruz, L., Mora, G., Fraire, H.: Self-tuning mechanism for genetic algorithms parameters, an application to data-

object allocation in the web. Lectures Notes in Computer Science. Springer Verlag, Berlin Heidelberg New York **3046** (2004) 77–86

[38] Pérez, J., Romero, D., Fausto, J., Pazos, R., Rodríguez, G., Reyes, F.: Dynamic allocation of vertical fragments in distributed databases using the threshold accepting algorithm. In: Proceeding of the 10th IASTED International Conference on Parallel and Distributed Computing and System, Las Vegas, Nevada (1998) 210–213

[39] Cruz, L.: Automatización del diseño de la fragmentación vertical y ubicación en bases de datos distribuidas usando métodos heurísticos. Master's thesis, ITESM (1999)

[40] Velez, L.: Esquema de enfriamiento adaptativo para el algoritmo de aceptación por umbral aplicado al diseño de bases de datos distribuidas. Master's thesis, Instituto Tecnológico de Leon, Leon, Guanajuato (2000)

[41] Barba, B.: Análisis comparativo de las técnicas de recocido simulado vs aceptación por umbral aplicadas a la solución del modelo furd. Master's thesis, ITESM, Campus Morelos (2000)

[42] Hernández, I., López, R., Nieto, A.: Automatización de la evaluación de algoritmos genéticos. Master's thesis, Instituto Tecnológico de Ciudad Madero (2000)

[43] Laurence, C.: Evaluación del modelo furd usando búsqueda tabú. Technical report, Instituto Tecnológico de Ciudad Madero (2002)

[44] González, J., Fraire, H., Moreno, A., Muñoz, L., Ramirez, M.: Agente inteligente de aprendizaje reforzado aplicado al diseño de bases de datos distribuidas. In: CIECE. (2001)

[45] Colunga, A., Muñoz, L., Ramirez, M.: Agente inteligente de aprendizaje reforzado aplicado al diseño de dases de datos distribuidas. Master's thesis, Instituto Tecnológico de Ciudad Madero (2001)

[46] Hernández, I.: Evaluación del desempeño de algoritmos de retroceso aplicados a la solución del modelo furd. Master's thesis, Instituto Tecnológico de Ciudad Madero (2002)

[47] Pecero, J.: Un método estadístico que permita caracterizar y comparar el desempeño de algoritmos heurísticos, y mediante el cual sea posible estimar el mejor algoritmo para solucionar casos específicos del problema del diseño de la distribución. Master's thesis, Instituto Tecnológico de Ciudad Madero (2002)

[48] Vega, R.: Método estadístico para caracterizar y comparar algoritmos heurísticos utilizados para resolver el modelo furd. Master's thesis, Instituto Tecnológico de Ciudad Madero (2002)

[49] Huang, Y., Chen, J.: Fragment allocation in distributed database design. In: Journal of Information Science and Engineering. Volume 17. (2001) 491–506

[50] Lin, X., Orlowska, M.: An integer linear programming approach to data allocation with the minimum total communication cost in distributed database systems. Information Sciences (1995) 1–10

[51] : (2005)

[52] [Autonomic. http://researchweb.watson.ibm.com/autonomic/manifiesto (2000)

[53] : Tpc benchmarks. http://www.tpc.org (2005)

[54] Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications (1998)

[55] Chaudhuri, S., Gupta, A., Narasayya, V.: Compressing sql workloads. In: IGMOD Conference. (2002) 488–499

[56] Eldar, Y., Lindenbaum, M., Porat, M., Zeevi, Y.: The farthest point strategy for progressive image sampling. In: IEEE Trans. On Image Processing. Volume 6(9). (1997) 1305–1315

[57] Computer, S.P.: Efficient progressive sampling for association rules (2002)

[58] Baoua, G., B.L.F.H.H.L.: Efficiently determine the starting sample size for progressive sampling. Lectures Notes in Computer Science, Springer-Verlag **Vol. 2167.** (2001) 192–202