INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

"OPTIMIZACIÓN DE LA UBICACIÓN DE FRAGMENTOS EN BASES DE DATOS DISTRIBUIDAS CONSIDERANDO TIEMPO DE RESPUESTA"

TESIS

QUE PARA OBTENER EL GRADO DE
DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

M. en C. GRACIELA VÁZQUEZ ÁLVAREZ

DIRECTORES DE TESIS:

Dr. RODOLFO ABRAHAM PAZOS RANGEL
Dr. GILBERTO LORENZO MARTÍNEZ LUNA

MÉXICO D.F., a 11 de abril de 2014.

SIP-14 (0)

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 16:00 horas del día 22 del mes de Octubre de 2013 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de la:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Optimización de la ubicación de fragmentos en bases de datos distribuidas considerando tiempo de respuesta"

Presentada por la alumna:

VÁZQUEZ	ÁLVAREZ	GRACIELA
Apellido paterno	Apellido materno	Nombre(s)

Con registro: A 9 7 0 5 6 8

aspirante de: **DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dr. Gilberto Lorenzo Martínez Luna

Dr. Rodolfo Abraham Pazos Rangel

Dr. Ramón Mendoza Guerra

Dr. Alexander Gelbukh

Dr. Joaquín Pérez Ortega

Dr. Marco Antonio Moreno Ibarra

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. José Alfonso Villa [illegible]

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
DIRECCIÓN

INSTITUTO POLITÉCNICO NACIONAL

SECRETARIA DE INVESTIGACIÓN Y POSGRADO

CARTA DE CESIÓN DE DERECHOS

En la Ciudad de México, Distrito Federal, el día 22 del mes de octubre del año 2013, la que suscribe **Graciela Vázquez Álvarez**, alumna del Programa de Doctorado en Ciencias de la Computación, con número de registro A970568 adscrito al **Centro de Investigación en Computación** manifiesta que es autor intelectual del presente Trabajo de Tesis bajo la dirección de los Doctores: **Rodolfo Abraham Pazos Rangel** y **Gilberto Lorenzo Martínez Luna** y cede los derechos del trabajo intitulado **"Optimización de la ubicación de fragmentos en bases de datos distribuidas considerando tiempo de respuesta"**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficos o datos del trabajo sin permiso expreso de la autora y/o directores del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección:

gravazquez@hotmail.com

Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Graciela Vázquez Alvarez

Resumen

En esta tesis doctoral se plantea el problema de la optimización de la distribución y ubicación de fragmentos verticales en bases de datos distribuidas considerando el tiempo de respuesta como criterio de optimización.

Para tal efecto se planteó la hipótesis de que es posible formular el problema de la optimización de la distribución y ubicación de fragmentos verticales en bases de datos distribuidas, considerando el tiempo de respuesta (FVU-TR) como criterio de optimización.

Esta hipótesis es relevante por dos razones: durante muchos años se ha reconocido la importancia de considerar el tiempo de respuesta en el modelado de Bases de Datos Distribuidas (BDDs) [Chakravarthy, 94], y una revisión de la literatura especializada ha revelado que el tiempo de respuesta no ha sido considerado por otros autores hasta antes de esta investigación (como se explica en la Sección 2.3). En contraste, los modelos tradicionales minimizan los costos de transmisión y de procesamiento de consultas para el diseño de una BDD con fragmentación vertical (FURD).

Con el fin de comprobar la hipótesis, se justificó la formulación del modelo del tiempo de respuesta, lo cual se logró mediante la simulación de los procesos aleatorios involucrados en la generación, transmisión y procesamiento de consultas, así como en la transmisión de las respuestas. Después, mediante la técnica de mínimos cuadrados, se ajustó la curva generada por el modelo de tiempo a los valores de tiempo obtenidos por la simulación. Los resultados experimentales muestran que la curva generada se ajusta bastante bien a la curva de los valores obtenidos por simulación, con lo cual se demuestra que la fórmula propuesta describe con buena precisión el comportamiento observado del tiempo de respuesta.

Dicha fórmula permitió desarrollar un nuevo modelo de programación matemática (llamado FVU-TR) para el problema del diseño de la fragmentación vertical en BDDs. Específicamente, la fórmula se usó para definir una nueva función objetivo con el fin de minimizar el tiempo de respuesta de ida y vuelta.

Dado que el problema del modelo FVU-TR es un problema NP-duro, se consideró conveniente resolver este problema mediante la implementación de dos algoritmos metaheurísticos: aceptación por umbral y búsqueda tabú. Pruebas experimentales realizadas con ambos algoritmos revelaron que, para el problema FVU-TR, aceptación por umbral tiene mejor desempeño que búsqueda tabú.

Finalmente, surge la pregunta: ¿qué tan diferentes son las soluciones óptimas obtenidas para un modelo que minimiza tiempo de respuesta (como FVU-TR) y un modelo que minimiza costos de transmisión y procesamiento (como FURD)? Para responder esta pregunta, se realizó un experimento comparativo de los modelos FVU-TR y FURD, utilizando un algoritmo exacto para resolver cada uno. Los resultados experimentales revelaron que los tiempos de respuesta de las soluciones óptimas obtenidas con el modelo

FURD son en promedio 30% más grandes que los tiempos de respuesta de las soluciones óptimas para FVU-TR.

Abstract

In this PhD dissertation the problem of optimizing the distribution and vertical fragment allocation in distributed databases is addressed considering response time as optimization criterion.

To this end the following hypothesis was considered: it is possible to formulate the problem of optimizing the distribution and vertical fragment allocation in distributed databases considering response time (VFA-RT) as optimization criterion.

This hypothesis is relevant for two reasons: for many years the importance of considering the response time in modeling Distributed Databases (DDBs) [Chakravarthy, 94] has been recognized, and a review of the specialized literature has revealed that response time has not been considered by other authors prior to this research (as explained in Section 2.3). In contrast, traditional models minimize the transmission and query processing costs of a DDB with vertical fragmentation (DFAR).

In order to validate the hypothesis, the formulation of the response time model was justified, which was carried out by simulating the random processes involved in the generation, transmission and processing of queries, as well as the transmission of query responses. Next, by using the least squares method, the curve generated by the time model was adjusted to the time values obtained by the simulation. The experimental results show that the generated curve fits quite well the curve of the values obtained by simulation, which shows that the proposed formula describes with good accuracy the observed behavior of response time.

This formula permitted developing a new mathematical programming model (called VFA-RT) for the vertical fragmentation design in DDBs. Specifically, the formula was used for defining a new objective function aiming at minimizing the roundtrip response time.

Since the problem of the VFA-RT model is NP-hard, it was considered convenient to solve this problem by implementing two metaheuristic algorithms: threshold accepting and tabu search. Experimental tests carried out with both algorithms revealed that, for the VFA-RT problem, threshold accepting outperforms tabu search.

Finally, a question arises: how different are the optimal solutions obtained for a model that minimizes response time (like VFA-RT) and a model that minimizes transmission and processing costs (like DFAR)? For answering this question a comparative experiment was conducted for the VFA-RT and DFAR models, using an exact algorithm for solving each one. The experimental results revealed that the response times of the optimal solutions obtained for the DFAR model are on average 30% larger than the response times of the optimal solutions for VFA-RT.

AGRADECIMIENTOS

Al finalizar un trabajo tan arduo y lleno de dificultades y contratiempos como fue el desarrollo de esta tesis doctoral es inevitable hacer un análisis el cual muestra que la magnitud de este aporte hubiese sido imposible sin la participación de personas e instituciones que facilitaron las cosas para que este trabajo llegara a feliz término.

Agradezco a Dios por haberme permitido vivir hasta este día, haberme guiado a lo largo de mi vida, por ser mi vida, mi luz y mi camino, por haberme dado la fortaleza de seguir adelante en aquellos momentos de debilidad.

Un agradecimiento muy especial merece la comprensión, paciencia y el ánimo recibidos de mi familia Mi querida mamá, hermanas y sobrinos).

Al Instituto Politécnico Nacional por abrirme las puertas y darme la oportunidad de desarrollarme profesionalmente, por prepararme para obtener el grado de doctor y sobre todo darme a mis mejores amigos, aunque algunos de ellos ya no estén en este plano.

Me gustaría que estas líneas sirvieran para expresar mi más profundo y sincero agradecimiento muy especial al Dr. Rodolfo Abraham Pazos Rangel, director de esta investigación, por la orientación, el seguimiento y la supervisión continúa de la misma, pero sobre todo por la motivación y el apoyo recibido a lo largo de estos años.

A mis grandes amigos: Tomas Ignacio Asían Olivares, Carlos Islas Moreno y Fernando Garduño Taboada por la confianza en mí depositada y por creer siempre en mí.

Gracias a mis amigas Sandra Dinora Orantes Jiménez, Concepción Silvana Sotelo Nieto y Soledad Asunción Arroyo Amador, compañeras de trabajo, viajes y estudio, además de compartir de corazón a sus hermosas familias que me adoptaron como un miembro más.

Otro especial reconocimiento merecen por el tiempo y apoyo para realizar la simulación de mi trabajo el M.en C. Juan Moncada Bolón y las correcciones recibidas del Dr. José Antonio Martínez, con quienes me encuentro en deuda por el ánimo infundido y la ayuda desinteresada recibida.

Quisiera hacer extensiva mi gratitud a mis compañeros del doctorado y, especialmente, al Dr. Máximo López Sánchez por su amistad y confianza al haber compartido momentos difíciles.

A todos mis alumnos de posgrado (CIC y ESIME-Zacatenco) con quienes he vivido experiencias gratificantes.

Quiero agradecer a todos mis compañeros de ESIME y excompañeros del CIC por su amistad y colaboración, así como a todos mis compañeros y amigos de la chispa que hoy se encuentra extinta, por su compañerismo y franca amistad.

Finalmente agradecer a todos los sínodos revisores de este trabajo: Dres. Gilberto Lorenzo Luna Martínez quien también participó como Director de esta tesis, Joaquín Pérez Ortega, Sergio Suarez Guerra, Alexander Gelbukh y Marco Antonio Moreno Ibarra por sus valiosos comentarios para enriquecer este trabajo.

Gracias a esas personas importantes en mi vida, que siempre estuvieron listas para brindarme su ayuda.

Sinceramente,

Graciela Vázquez Álvarez

Contenido de la Tesis

Lista de figuras

figura **título** **página**

Lista de tablas

tabla **título** **página**

Capítulo 1

Introducción

Capítulo 1. Introducción

El uso generalizado de Internet ha facilitado en los últimos años la implementación de sistemas de bases de datos distribuidas (SBDDs), ya que cada vez es más común encontrarlos ahí. Aunque existen modelos para optimizar el diseño de la distribución de datos (como se puede ver en la parte del estado del arte más adelante), los cuales buscan optimizar los costos de transmisión y procesamiento de consultas, desafortunadamente pasan por alto el retardo ocasionado por su transmisión (tanto al transmitir y procesar la consulta como al obtener la respuesta) y su procesamiento.

El explosivo aumento de popularidad de las computadoras personales desde el siglo pasado, ha hecho que millones de personas las utilicen, y conforme las computadoras y las redes de computadoras se extienden a través de las organizaciones, los datos ya no se encuentran en un único sistema bajo el control de un único sistema de administración de bases de datos (SABD). En vez de ello, los datos se han extendido a través de diferentes sistemas, cada uno con su propio manejador de bases de datos.

Las compañías que se expanden, lo hacen con el fin de ofrecer bienes o servicios de una forma más rápida a sus clientes. La expansión consiste en la instalación de nuevas sucursales en distintas partes de su país o incluso del mundo. También hay empresas que planean establecer vínculos de cooperación entre sí, con el fin de iniciar su fusión y dar paso al nacimiento de una sola empresa que ofrezca bienes o servicios que constituyan una mejor alternativa en el mercado.

Dicha expansión crea vínculos de cooperación entre empresas que en el pasado era impensable que se dieran, debido a que eran adversarias comerciales que competían en el mercado. Ahora han formado consorcios, bajo un mismo nombre y con un objetivo común.

Con frecuencia los diferentes sistemas informáticos y los diferentes SABDs proceden de diferentes fabricantes.

Actualmente la industria informática y la comunidad de administración de datos enfocan su atención en los problemas de administración de bases de datos distribuidas en Internet.

La investigación y desarrollo en la tecnología de SBDDs, ha sido fomentada por dos factores separados. Uno es la evolución de las organizaciones hacia la descentralización, donde cada unidad descentralizada cuenta con el soporte de computadoras. El otro son los avances en la tecnología de las telecomunicaciones que han hecho posible el enlace electrónico entre computadoras que se encuentran en lugares geográficamente dispersos.

Una *base de datos distribuida* (BDD) es aquélla en la que sus datos se encuentran repartidos en varias computadoras que están interconectadas por una red de comunicaciones [Ceri, 84]. Bajo este concepto las BDDs han sido desarrolladas para dar

respuesta a las necesidades de almacenamiento, recuperación y procesamiento de información, la cual debe ser compartida entre diversas entidades bajo esquemas que garanticen un buen manejo de este recurso. Estos nuevos requerimientos funcionales han aparecido en nuestra sociedad al mismo tiempo que las grandes empresas han evolucionado, desde un esquema de funcionamiento eminentemente centralizado a esquemas corporativos distribuidos.

La solución al problema de sistemas de información que operan en regiones amplias, se da mediante la comprensión del uso de los sistemas distribuidos, dando como resultado una explotación máxima de los recursos.

En este sentido se investiga el diseño de BDDs, el cual presenta nuevos problemas de alta complejidad. Para ello, la intuición no basta, ni considerarlo o reducirlo a un problema NP-duro, sino que se requiere de una metodología especial que describa la información pertinente y los procedimientos en forma sistemática [Sagols, 92].

Se han realizado diferentes trabajos, como se explicará más adelante, todos ellos enfocados al diseño de las BDDs, probando que es, como dijo Eswaran [Eswaran, 74], un problema NP-duro.

1.1. Descripción del problema

Dentro de un contexto general, el problema de diseño de BDDs consiste en tomar decisiones acerca de la ubicación de los datos y los programas en los sitios de una red de computadoras, así como también posiblemente el diseño de la propia red.

Para entender el problema de distribución en las bases de datos distribuidas, se supone que existen un conjunto de fragmentos $F = \{f_1, f_2, ..., f_n\}$ y una red formada por sitios $S = \{s_1, s_2, ..., s_m\}$ y en ellos un conjunto de aplicaciones ejecutándose $Q = \{q_1, q_2, ..., q_q\}$. El problema de distribución consiste en encontrar la distribución "óptima" de F en S.

En [Dowdy, 82] se indican dos aspectos importantes para definir la optimalidad:

1. Costo mínimo. En el cual, la función de costo consiste en considerar: el costo de almacenar cada fragmento f_i en un sitio s_j, el costo de consultar el fragmento f_j en el sitio s_j, el costo de actualizar f_i en todos los sitios donde estén almacenados, y el costo de comunicación de datos. El problema de distribución consiste en encontrar un esquema de distribución que minimice una combinación de la función de costos.

2. Desempeño. Para esto se sigue la estrategia de distribución que consiste en mantener una medida de desempeño, la cual se logra al minimizar el tiempo de respuesta y maximizar la tasa neta de transmisión en cada sitio.

En este punto conviene destacar que todos los trabajos sobre optimización del diseño de BDDs con fragmentación que se han publicado hasta antes de iniciar esta investigación habían adoptado el primer criterio de optimalidad; sin embargo, en este proyecto por primera vez se ha considerado la optimización del tiempo de respuesta.

1.2. Justificación del proyecto

Las grandes organizaciones requieren tanto de software como de hardware para realizar sus actividades; sin embargo, como consecuencia del crecimiento en la tecnología de comunicaciones, de sistemas administradores de bases de datos distribuidas (SABDDs) y el decremento en los costos de las computadoras, dichas organizaciones han optado por utilizar redes de comunicación para computadoras incluyendo Internet, logrando así que sea más factible la implementación de SBDDs.

Sin embargo, los administradores de los sistemas son los encargados de realizar la fragmentación, ubicación, replicación y monitoreo de la información en los nodos de la red. Dicha actividad es efectuada de manera estática, se determina en el momento del diseño de la BDD y permanece sin cambios (diseño estático) hasta que el administrador considera un nuevo diseño. En la mayoría de los SBDDs, el diseño estático afecta de manera importante el desempeño total del sistema, esto se debe principalmente a que el tiempo requerido para la formulación de las consultas, el procesamiento de las consultas y las respuestas de dichas consultas, dependen en gran parte del lugar donde residen los datos, de las frecuencias de accesos y de la velocidad de trasmisión de las consultas.

Un defecto poco conocido que tienen los SBDDs es que su desempeño puede ser insatisfactorio si el diseño de la distribución no es óptimo. De aquí la necesidad de contar con herramientas que permitan obtener un diseño de la distribución que optimice el diseño de los SBDDs.

Se ha reconocido durante muchos años la importancia de considerar el tiempo de respuesta en el modelado de BDDs [Chakravarthy, 94]. Desafortunadamente, una revisión de la literatura especializada ha revelado que el tiempo de respuesta no ha sido considerado (como se explica en la Sección 2.3). Por lo tanto, los modelos previamente publicados para la optimización del diseño de BDDs sólo se han propuesto minimizar los costos de transmisión y procesamiento de las consultas, ignorando los retardos incurridos por los tiempos de transmisión y procesamiento (incluyendo tiempos de espera), los cuales son una de las principales preocupaciones en sistemas que operan en Internet. En contraste, en este proyecto por primera vez se ha considerado tomar en cuenta el criterio de desempeño, en particular, el tiempo de respuesta.

En este punto resulta oportuno destacar que una de las principales razones por la que en los trabajos anteriores a éste no se haya considerado el tiempo de respuesta, es que es muy complejo desarrollar una expresión para modelar éste, como queda de manifiesto en la expresión (4.20) para la función del objetivo que se presenta en la Sección 4.2.

El objetivo de este trabajo es desarrollar un modelo que permita optimizar el diseño de la fragmentación vertical de una BDD considerando el tiempo de respuesta.

1.3. Complejidad del problema

El problema de diseño de la distribución de datos, en las bases de datos distribuidas, es un problema de optimización muy complejo, por el número de variables que intervienen y las interrelaciones que existen entre éstas. Existen numerosas publicaciones que demuestran la complejidad del problema. Los primeros trabajos son los relacionados con la ubicación de archivos en los sitios de una red de computadoras, teniendo como finalidad que los programas de aplicación accedan a esos archivos, optimizando tiempos de respuesta y/o costos.

De manera generalizada, los investigadores en el área de diseño de bases de datos distribuidas dan como un hecho que el problema es un problema NP-duro. Un razonamiento de porqué el problema es NP-duro se encuentra en [Ozsu, 10], donde se afirma: "Al paso de los años se ha demostrado que varias formulaciones del problema son igualmente difíciles. La implicación, desde luego, es que obtener soluciones óptimas probablemente no es computacionalmente factible. Muchas investigaciones se han dedicado a encontrar mejores heurísticas, que proporcionen soluciones subóptimas". Una descripción de las variables que intervienen en el problema y del grado de complejidad del mismo puede verse en el trabajo presentado por March en [March, 95].

Muchos modelos propuestos hacen uso de estos aspectos para la optimización; sin embargo, se desearía que en un esquema de distribución se responda a las consultas del usuario en un tiempo mínimo mientras se mantiene un costo mínimo de procesamiento. Similarmente se puede buscar un esquema de distribución para maximizar el rendimiento. Cuando uno se pregunta ¿por qué dicho modelo no ha sido desarrollado?, la respuesta es bastante simple: por complejidad.

El problema de ubicación de datos en un SBDD se define de la siguiente manera [Ozsu y Valduriez, 10]: se supone que existen un conjunto de fragmentos $F = \{f_1, f_2, ..., f_n\}$ y una red de comunicaciones para computadoras que consiste de sitios $S = \{s_1, s_2, ..., s_m\}$, en donde se ejecutan una serie de aplicaciones $Q = \{q_1, q_2, ..., q_p\}$. El problema de la ubicación consiste en encontrar la ubicación óptima de F en S de manera que el procesamiento de las consultas Q sea el más eficiente. En el resto de esta sección se presenta una de muchas posibles formulaciones del problema que ilustra la dificultad de éste.

Se considera una simplificación para la formulación del problema, la cual consiste en considerar sólo un fragmento simple f_k a la vez. Además, se harán un número de suposiciones y definiciones que se incorporarán en este modelo del problema de distribución.

1. Se supone que Q puede modificarse, de tal manera que sea posible identificar las solicitudes de actualización de consulta, y se define lo siguiente para ese fragmento simple f_k:

   $T = \{t_1, t_2, ..., t_m\}$, donde t_i es el tráfico de consulta generado en el sitio s_i para f_k, y

   $U = \{u_1, u_2, ..., u_m\}$, donde u_i es el tráfico de actualización generado en el sitio s_i para f_k.

2. Se supondrá que el costo de comunicación entre cualesquiera dos pares de sitios s_i y s_j, es una unidad de transmisión. Además, se supondrá que éste es diferente para actualizaciones y para consultas, por lo que se define:

   $C(T) = \{c_{12}, c_{13}, ..., c_{1m}, ..., c_{m-1, m}\}$ para consultar

   $C'(U) = \{c'_{12}, c'_{13}, ..., c'_{1m}, ..., c'_{m-1, m}\}$ para actualizar

donde c_{ij} es el costo unitario de comunicación para las peticiones de consulta entre los sitios s_i y s_j, y c'_{ij} es el costo unitario de comunicación para operaciones de actualización entre los sitios s_i y s_j.

3. Sea d_i el costo de almacenar el fragmento en el sitio s_i, por lo cual se define $D = \{d_1, d_2, ..., d_m\}$, como el costo de almacenar el fragmento f_k en todos los sitios.

4. Se considerará que no hay limitaciones de capacidad en los sitios o enlaces de comunicación.

Entonces el problema de distribución se puede definir como un problema de minimización de costos, el cual consiste en encontrar el conjunto $I \subseteq S$ que especifique dónde se almacenará la copia de los fragmentos. En adelante se denotará con x_j la variable de decisión para la asignación tal que:

$$x_j = \begin{cases} 1 \text{ si el fragmento } f_k \text{ es asignado al sitio } s_j \\ 0 \text{ en caso contrario} \end{cases}$$

La especificación precisa de la función objetivo del problema es como sigue:

$$\min\left[\sum_{i=1}^{m}\left(\sum_{j|S_j \in I} x_j u_j c'_{ij} + t_j \min_{j|S_j \in I} c_{ij}\right) + \sum_{j|S_j \in I} x_j d_j\right]$$

sujeto a $x_j = 0$ o 1.

El primer término corresponde al costo de comunicación para las operaciones de actualización de datos a todos los sitios que contengan réplicas de los fragmentos, más el costo de comunicación para las operaciones de consultas en los sitios, resultando así un costo mínimo de la transmisión de datos.

El segundo término de la función objetivo calcula el costo total de almacenar todas las copias de los fragmentos.

Ésta es una formulación simplista que no es adecuada para el diseño de BDDs. Para casos reales, el problema es grande; es decir, incluye un gran número de fragmentos y sitios, por lo que se pueden obtener soluciones óptimas las cuales aún no son viables computacionalmente, por lo que hasta el momento sólo se han realizado investigaciones dedicadas a encontrar heurísticas que proveen soluciones subóptimas.

A continuación se enumeraran una serie de razones por las cuales las formulaciones simples, como la que se ha planteado anteriormente, no son adecuadas para el diseño de BDDs, sino más bien son esenciales para la distribución de archivos para redes de computadoras.

1. Uno no puede tratar los fragmentos como archivos individuales que pueden ser ubicados a la vez. La asignación de un fragmento usualmente tiene impacto sobre la decisión de asignar otros fragmentos, sobre todo cuando son accedidos juntos, es decir, impacta el costo de acceso para la permanencia de los fragmentos que puede cambiar debido a la reunión distribuida; por lo tanto, la relación entre fragmentos deberá ser tomada en cuenta.
2. El acceso a datos para aplicaciones es un modelo muy simple. La búsqueda realizada por un usuario es problema de un sitio. En los SBDDs el acceso a datos es más complicado que un simple "acceso remoto de archivos". Además, la relación entre la distribución y procesamiento de consultas debe ser modelado apropiadamente.

3. Este modelo no considera costos para mantener la integridad, aunque se localicen dos fragmentos en el mismo lugar; más aun, en dos sitios diferentes esto puede ser costoso.

4. De igual forma, el costo de mantener el mecanismo de control de concurrencia debería ser considerado como se menciona en [Rothnie, 77].

5. No hay un modelo general heurístico que tome como entrada un conjunto de fragmentos y produzca una distribución cercanamente óptima, sujeto al tipo de restricciones discutidas aquí. Los modelos desarrollados para datos, hacen un número de simplificaciones y son aplicables a determinadas formulaciones específicas.

En suma, se deben considerar las interrelaciones del problema de BDDs como se muestran en la figura 1.1.

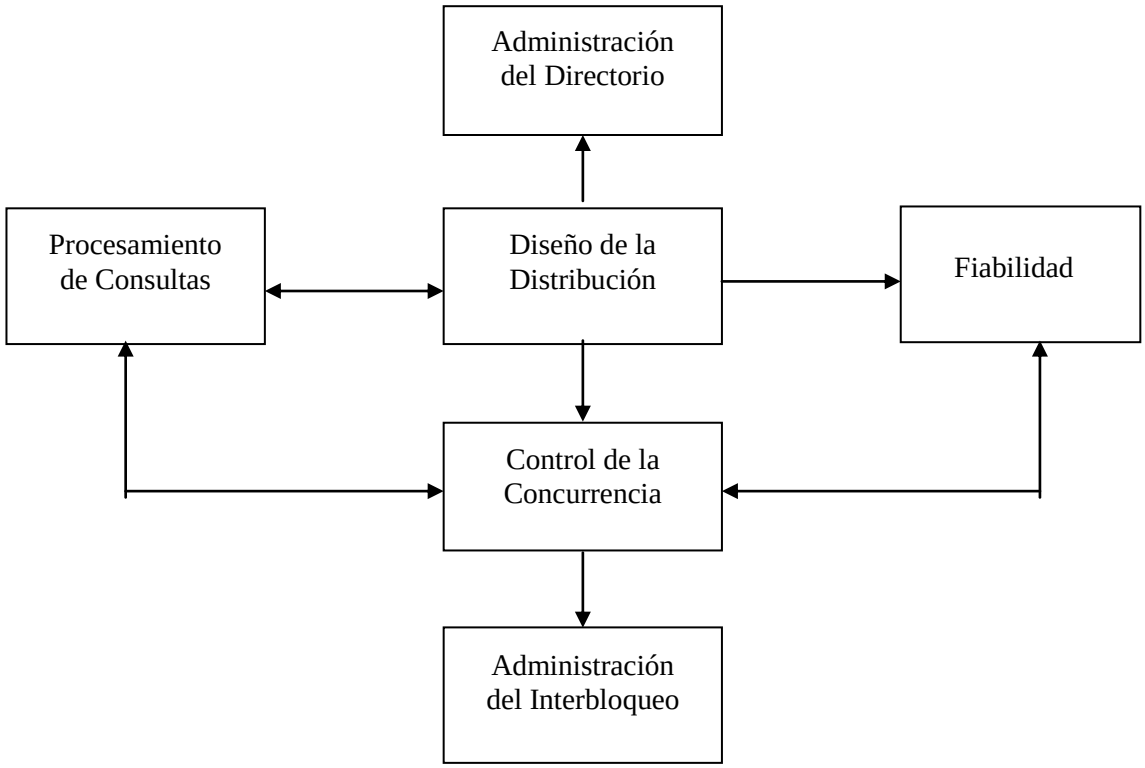

figura1.1 Relación entre problemas

1.4. Objetivos de la tesis

Objetivo general:

Formular un modelo matemático que involucre el tiempo de respuesta que permita optimizar la distribución y ubicación de fragmentos verticales en bases de datos distribuidas, y con base en este modelo diseñar y codificar un algoritmo que lo resuelva.

Objetivos particulares:

1. Justificar el modelo del tiempo de respuesta mediante la simulación de los procesos aleatorios involucrados en la generación, transmisión y procesamiento de consultas, así como en el involucrado en la transmisión de las respuestas.

2. Ajustar la fórmula propuesta para modelar el tiempo de respuesta (mediante mínimos cuadrados) a los valores obtenidos de la simulación, con el fin de comprobar qué tan aproximado es el comportamiento estimado por la fórmula propuesta versus el comportamiento simulado.

3. Desarrollar un nuevo modelo de programación matemática para el problema del diseño de la fragmentación vertical en BDDs, en particular una nueva función objetivo que involucre la fórmula propuesta para modelar el tiempo de respuesta.

4. Utilizar dos algoritmos metaheurísticos (aceptación por umbral y búsqueda tabú) para encontrar la solución al modelo, y determinar cuál de los algoritmos es mejor para resolver el problema.

5. Realizar pruebas experimentales con el modelo de programación entera para la fragmentación, ubicación y reubicación dinámica de datos (FURD) y el propuesto en esta tesis(FVU-TR), con el fin de determinar si existen diferencias entre las soluciones obtenidas para cada modelo y qué tan grandes son las diferencias.

1.5. Hipótesis

Es posible la optimización de la distribución y ubicación de fragmentos verticales en bases de datos distribuidas, considerando el tiempo de respuesta como criterio de optimización.

El trabajo de investigación en esta hipótesis es relevante por dos razones: durante muchos años se ha reconocido la importancia de considerar el tiempo de respuesta en el modelado de BDDs [Chakravarthy, 94], y una revisión de la literatura especializada ha revelado que el tiempo de respuesta no ha sido considerado por otros autores hasta antes de esta investigación (como se explica en la Sección 2.3).

1.6. Aportaciones

- Se encontró en este trabajo una fórmula que describe con bastante precisión el comportamiento del tiempo de respuesta observado en la transmisión y procesamiento de consultas, así como en la transmisión de las respuestas.
- La expresión propuesta fue validada mediante la simulación de los procesos aleatorios involucrados en la generación, transmisión, procesamiento de consultas y transmisión de las respuestas.
- Mediante el uso de la fórmula mencionada anteriormente, se desarrolló el modelo matemático denominado FVU-TR para el problema de la distribución y ubicación de fragmentos verticales en BDDs, el cual involucra el tiempo de respuesta.
- Se probó el desempeño de dos métodos metaheurísticos para la solución de este tipo de problemas: aceptación por umbral y búsqueda tabú.
- Se demostró que para este tipo de problema, con base en pruebas experimentales realizadas con los dos algoritmos (aceptación por umbral y búsqueda tabú), aceptación por umbral tiene mejor desempeño.
- Se concluyó que vale la pena el uso del modelo FVU-TR, ya que el tiempo de respuesta de las soluciones óptimas del modelo FURD son en promedio 30% más

grandes que el tiempo medio de respuesta para las soluciones óptimas del modelo FVU-TR.

- Se determinó, al realizar pruebas experimentales que compran los resultados óptimos obtenidos por los modelos FURD vs. FVU-TR, que con el modelo FVU-TR las soluciones óptimas tienen en promedio costos 47% más grandes que los del modelo FURD.
- Se llegó a la conclusión, como era de esperarse, que los objetivos de minimizar el tiempo de respuesta y el costo son antagónicos: es decir, las soluciones que tienen menor tiempo de respuesta tienen costos más grandes y viceversa; así que si un diseñador está preocupado por ambos, debería considerar un modelo donde se integren los dos objetivos.

1.7. Organización de la tesis

La organización de la tesis es de la siguiente manera:

Capítulo 2. Se presenta un panorama general de los fundamentos de las bases de datos distribuidas. Se describe el diseño de la distribución y se ubica esta tesis en el contexto de otras investigaciones relacionadas.

Capítulo 3. En este capítulo se desarrolla una fórmula matemática que considera el tiempo medio de respuesta de las consultas en un sistema de base de datos distribuída (SBDD) con fragmentación vertical en términos de los parámetros de éste. Además, se describen los procesos de simulación del tiempo de respuesta y el ajuste de la fórmula propuesta para modelar el tiempo de respuesta a los valores obtenidos de la simulación.

Capítulo 4. Se presentan dos modelos: el Modelo FURD que se usa como punto de referencia para este trabajo y el modelo propuesto en el presente trabajo de investigación (FVU-TR).

Capítulo 5. Se explican métodos exactos y metaheurísticos utilizados para resolver los dos modelos FURD y FVU-TR.
Capítulo 6. Se presentan los resultados de las pruebas experimentales realizados en esta tesis.

Capítulo 7. Se presentan las conclusiones a las que se llegaron durante esta investigación y se sugieren trabajos que pueden dar continuidad a esta investigación.

Capítulo 2

Marco teórico y estado del arte

Capítulo 2. Marco teórico y estado del arte

2.1. Fundamentos de bases de datos distribuidas

2.1.1. Antecedentes

La tecnología de bases de datos ha evolucionado desde un paradigma de procesamiento de datos en el que cada aplicación definía y mantenía sus propios datos, hasta otro en el que los datos se definían y administraban de manera centralizada. En los últimos años, se han visto una serie de rápidos desarrollos en la tecnología de redes y de comunicación de datos, tales como: Internet, comunicación móvil e inalámbrica, dispositivos inteligentes y la computación reticular. Ahora, con la combinación de estas dos tecnologías (bases de datos y redes), la tecnología de bases de datos distribuidas (BDDs) permite cambiar el modo de trabajo, pasando de un modo centralizado a otro descentralizado. Este tipo de tecnología combinada es uno de los avances principales en el área de los sistemas de bases de datos; sin embargo, no se ha logrado implantar plenamente esta tecnología en toda su potencialidad. De tal suerte que, actualmente muchas empresas cuentan con seudobases de datos distribuidas, constituidas por conjuntos de datos dispersos en una red pero sin una plena integración desde el punto de vista de las aplicaciones generadas para realizar consultas (Sagols, 92).

Una BDD, como se ha dicho, es aquélla en la que sus datos se encuentran repartidos en varias computadoras que están interconectadas por una red de comunicaciones. Desde luego sus datos están lógicamente relacionados y se presentan al usuario como si se tratara de una base de datos centralizada.

Diseñar una BDD es más complicado que diseñar una base de datos centralizada, ya que las características inherentes a la distribución se contraponen a las del centralismo (Sagols Troncoso, 1992). Es decir, los conceptos de control centralizado: independencia de datos, reducción de redundancias, estructuras físicas complejas para acceder a los datos, integridad, recuperación, control de concurrencia, y seguridad, no tienen el mismo significado en un ambiente de base de datos distribuida que en el centralizado.

En las BDDs, no sólo debe proporcionarse la independencia de datos, sino que también debe ofrecerse la transparencia de distribución. Es decir; los programas de aplicación deben poder escribirse como si la base de datos no estuviera distribuida.

Un *sistema administrador de bases de datos distribuidas* (SABDD) es aquél que administra una BDD de tal forma que hace transparentes los aspectos de distribución desde el punto de vista de los usuarios. En otras palabras, los SABDDs hacen que el usuario no se percate de que se encuentra trabajando en un ámbito distribuido y que la información con la que él trabaja pueda no estar físicamente en su computadora. Es decir, un SABDD hace que desde el punto de vista de los usuarios, la base de datos se vea como si funcionara de manera centralizada.

En los SBDDs, el problema de diseño se complica más, ya que se requiere modelar su comportamiento. Es decir, el diseño de un SBDD implica la toma de decisiones sobre la ubicación de los programas que accederán a la base de datos y sobre los propios datos que constituyen esta última, en los diferentes sitios que formen parte de una red de computadoras. La ubicación de los programas no debería suponer un excesivo problema dado que se puede tener una copia de ellos en cada máquina de la red. Sin embargo, ¿cuál es la mejor opción para colocar los datos? o ¿se puede pensar en repartir las relaciones (tablas) por toda la red?

2.1.2. Importancia de las bases de datos distribuidas

Una de las principales motivaciones para el desarrollo de sistemas de bases de datos es el deseo de integrar datos operacionales de una organización y proporcionar un acceso controlado a esos datos. Aunque la integración y el acceso controlado pueden implicar la necesidad de utilizar mecanismos de centralización, el objetivo en realidad no es primordialmente ése. De hecho, el desarrollo de redes informáticas promueve el modo descentralizado de trabajo.

Esta técnica descentralizada imita la estructura organizativa de muchas empresas, que están distribuidas de modo lógico en divisiones, departamentos, proyectos, etc. Y están físicamente distribuidas en oficinas, fabricas e instalaciones, manteniendo cada unidad sus propios datos operacionales [Date, 00]. La posibilidad de compartir los datos y la eficiencia del acceso a los mismos puede mejorarse mediante el desarrollo de SBDDs que reflejen esta estructura organizativa, que hagan que los datos de todas las unidades sean accesibles y que almacenen esos datos en algún lugar próximo al sitio o lugar donde más frecuentemente se utilicen.

Las BDDs se han desarrollado para dar respuesta a las necesidades de almacenamiento, recuperación y procesamiento de información que debe ser compartida entre diversas entidades distribuidas geográficamente bajo esquemas que garanticen un buen manejo de este recurso. Estos nuevos requerimientos funcionales han aparecido en nuestra sociedad al mismo tiempo que las grandes empresas del planeta han evolucionado, desde un esquema de funcionamiento eminentemente centralizado a esquemas corporativos.

En tiempos pasados era impensable que las empresas pudieran colaborar, debido a que eran adversarias comerciales que competían en el mercado. Ahora han formado consorcios que cooperan, bajo un mismo nombre y con un objetivo común.

En yuxtaposición con la estrategia de fusión se encuentran aquellas empresas de gran tamaño que por mucho tiempo se han desarrollado con una administración centralizada, y que ahora evolucionan en su esquema de organización, para descentralizarse y formar un esquema corporativo donde, en lugar de tener una sola empresa encargada de realizar una gran cantidad de labores, se tienen conjuntos de empresas, especializadas en un ramo de trabajo, que colaboran para formar un todo. La ventaja que esto ha producido se refleja directamente en el incremento de la productividad derivado de la especialización.

Este principio de colaboración interempresarial va más allá de los alcances de una corporación, porque una empresa especializada en la prestación de un servicio, puede resultar de utilidad para más de un corporativo.

Cabe destacar que el compartir información entre las entidades de una empresa, es un problema que tiene como base el transporte de la información. Afortunadamente, ahora se cuenta con los adelantos tecnológicos en los sistemas de telecomunicaciones que permiten transportar eficientemente la información entre las entidades que conforman un corporativo. Sin éstos, sería necesario que cada empresa contara con una matriz, en la cual se administrara la información generada en sus sucursales. Esto provocaría que la toma de decisiones en cada sucursal y en general de la empresa entera, tuviera que esperar el tiempo suficiente para procesar la información en la matriz. La administración de la información de esta forma va en detrimento de la productividad de la empresa, que necesitaría demasiado tiempo para poder tomar decisiones sobre la base del procesamiento de la información que se mantenga en la matriz.

El compartir adecuadamente la información entre las sucursales de una empresa, es una labor que va más allá de la simple transportación de la información entre las entidades, e involucra un conjunto de reglas que deben evitar los perjuicios evidentes de la fuga, pérdida o inconsistencia de la información. Precisamente la tecnología de las BDDs se ha desarrollado para dar respuesta a las necesidades de almacenamiento, recuperación y procesamiento de la información en corporativos.

El modelo de un SBDD debe tomar en cuenta los niveles de transparencia, que permitan a un usuario no percatarse de estar trabajando en un ambiente distribuido y percibir que la base de datos a consultar se encuentra almacenada en su propio nodo. La forma en la cual se organiza la base de datos distribuida contribuye a lograr que esto suceda.

La idea es que el usuario vea a la BDD como lo hace en ambientes centralizados, es decir constituida por simples relaciones. A estas relaciones percibidas por el usuario se les denomina relaciones globales, y al conjunto de las relaciones globales en la BDD se le denomina esquema global. De esta manera, el esquema global representa el nivel de transparencia en la capa superior de la arquitectura de la BDD.

Cada una de las relaciones globales se puede dividir en porciones lógicas disjuntas denominados fragmentos. Por esquema de fragmentación se hace referencia a la definición de los fragmentos de las relaciones globales. A este nivel de transparencia se le llama transparencia de fragmentación y permite que los usuarios hagan referencia a las relaciones globales en lugar de referirse a los fragmentos cuando se formulan consultas.

Estos fragmentos se encuentran distribuidos en los diferentes nodos de la red. Por esquema de distribución se hace referencia a la asignación de cada fragmento al nodo o nodos correspondientes. A este nivel de transparencia se le llama transparencia de localización, y permite que los usuarios no se preocupen en conocer en qué nodo se encuentra cada fragmento.

El modelo relacional es el modelo más conveniente para la definición del esquema global. Este modelo permite que de manera informal se visualicen gráficamente las relaciones del esquema global como tablas de datos, donde los renglones representan las tuplas de la relación y las columnas, sus atributos. El tipo de división de las relaciones globales en fragmentos se clasifica en fragmentación vertical, fragmentación horizontal y fragmentación mixta.

En la fragmentación horizontal, los fragmentos se obtienen aplicando solamente la operación de selección sobre las relaciones globales. De manera informal se puede visualizar esta fragmentación como si se dividieran las tablas horizontalmente. La reconstrucción de las relaciones globales que se fragmentaron horizontalmente se obtiene simplemente de la unión de sus fragmentos.

En la fragmentación vertical, los fragmentos se obtienen al aplicar la operación del álgebra relacional proyección (denotada por el símbolo □) sobre la relación R. El resultado de aplicar la operación de proyección a una relación específica es una relación cuyas tuplas contienen solamente los atributos especificados con un predicado dado. La fragmentación vertical se efectúa añadiendo a cada fragmento la llave primaria de la relación original o mediante un identificador de tupla. El objetivo de la fragmentación vertical es agrupar los atributos que se utilizan juntos frecuentemente.

En la fragmentación mixta, los fragmentos se obtienen aplicando la operación de proyección seguida por la operación de selección o viceversa. Su visualización sería en este caso como si se dividieran las tablas verticalmente y después horizontalmente. La reconstrucción de las relaciones globales se obtiene mediante operaciones de reuniones y uniones de sus fragmentos aplicadas en un orden y un número de veces adecuado.

Implícitamente se supone que en una BDD los datos están almacenados en diferentes sitios; es decir, cada uno de los sitios consiste lógicamente de un procesador. Los procesadores en diferentes sitios se encuentran interconectados vía una red de comunicaciones (no multiprocesadores), es decir se trata de sistemas de bases de datos en paralelo.

Un SBDD es el resultado de integrar una BDD y la tecnología de redes; asimismo, un SABDD no es otra cosa que el software que administra la BDD y proporciona un mecanismo de acceso que hace transparente esta distribución a los usuarios.

Por tanto, los usuarios al realizar consultas propician que el sistema participe en la ejecución de transacciones que acceden a datos de una o varias localidades. Cada una de las localidades mantiene un sistema de base de datos local. Además, cada localidad puede procesar transacciones locales, es decir, aquéllas que sólo acceden a datos que residen en esa localidad. Una localidad puede participar en la ejecución de transacciones globales, es decir, aquéllas que acceden a datos de varias localidades. La ejecución de transacciones globales requiere comunicación entre las localidades.

El problema de diseñar una BDD dentro de un marco general, consiste en tomar decisiones acerca de la ubicación de los datos y los programas en los sitios de una red de computadoras, así como también posiblemente el diseño de la propia red. En los SABDDs, la ubicación de las aplicaciones supone la ubicación del software de los SABDDs y la ubicación de las aplicaciones que corren sobre la base de datos.

El objetivo al diseñar una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar información sin redundancia innecesaria, pero que a la vez permitan recuperar información fácilmente.

Cada vez se tiende más a desarrollar bases de datos que manipulen la información de una manera natural, logrando que dicha información sea fácil de almacenar, localizar y usar. Por ende, el diseño de las BDDs es un factor crítico en el desempeño de este tipo de sistemas, en particular la fragmentación, ubicación y recuperación de los datos. En consecuencia, al diseñar una BDD, ésta, además de seguir los principios del diseño de bases de datos centralizadas, debe tomar en cuenta otros factores para almacenar cada relación en la BDD. Tres de ellos son los siguientes:

- **Repetición.** El sistema mantiene varias copias idénticas de una relación. Cada copia se almacena en una localidad diferente, lo que resulta en una repetición de la información. La alternativa a la repetición es almacenar una sola copia de la relación *R*.

- **Fragmentación**. La relación se divide en varias porciones lógicas disjuntas denominadas fragmentos. Existen tres tipos de fragmentación: vertical, horizontal y mixta. Cada fragmento se almacena en una localidad diferente.

- **Repetición y fragmentación**. Es una combinación de los dos conceptos antes mencionados. La relación se divide en varios fragmentos y el sistema mantiene varias copias idénticas de cada uno de los fragmentos.

En todos los tipos de fragmentación, un fragmento se puede definir por una expresión en lenguaje relacional, tomando relaciones globales como operandos y produce el fragmento como resultado.

Existen tres reglas que se han de cumplir durante el proceso de fragmentación, las cuales asegurarán la ausencia de alteraciones semánticas en la base de datos durante el proceso.

1. **Condiciones de completitud**. Si una relación *R* se descompone en una serie de fragmentos R_1, R_2, ..., R_n, cada elemento de datos que pueda encontrarse en *R* deberá poder encontrarse en uno o varios fragmentos R_i. Esta propiedad extremadamente importante asegura que los datos de la relación global se proyectan sobre los fragmentos sin pérdida alguna. Hay que considerar en el caso de la fragmentación horizontal a la tupla como elemento de datos, y para el caso de la fragmentación

vertical, al atributo. Es decir, todos los datos en la relación global deben ser mapeados en los fragmentos; en otras palabras, no debe ocurrir que una pieza de información que pertenece a una relación global no pertenezca a algún fragmento.

2. **Condiciones de reconstrucción**. Si una relación R se descompone en una serie de fragmentos R_1, R_2, ..., R_n, debe existir un operador relacional ∇ tal que

$$R = \nabla R_i, \ \ \forall R_i \in F_R$$

El operador ∇ será diferente dependiendo de las diferentes formas de fragmentación. La reconstrucción de la relación a partir de sus fragmentos asegura la preservación de las restricciones definidas sobre los datos en forma de dependencias. Es decir, siempre debe ser posible reconstruir cada relación global a partir de sus fragmentos.

3. **Condiciones de disjuntividad**. Si una relación R se descompone horizontalmente en una serie de **fragmentos** R_1, R_2, ..., R_n, y un elemento de datos d_i se encuentra en algún fragmento R_j, entonces no debe encontrarse en otro fragmento R_k ($k \neq j$). Esta regla asegura que los fragmentos horizontales sean disjuntos. Si una relación R se descompone verticalmente, los atributos de su llave primaria normalmente se repiten en todos sus fragmentos. Es conveniente que los fragmentos sean disjuntos, de manera que las réplicas de datos puedan controlarse explícitamente al nivel de distribución. Sin embargo, esta condición es útil principalmente en la fragmentación horizontal, mientras que para la fragmentación vertical se permite que algunas veces la condición sea violada.

2.1.3. Características de las bases de datos distribuidas

La intención de la tecnología de las BDDs es crear SABDDs que permitan usar de manera transparente la información almacenada en los distintos nodos de una red. El nivel de transparencia al que se hace referencia, debe hacer creer a los usuarios que la base de datos con la que ellos trabajan se encuentra almacenada en su totalidad en sus propias máquinas.

Los SABDDs deberían ayudar a resolver el problema de las islas de información. Las bases de datos se consideran en ocasiones, islas electrónicas que constituyen lugares diferenciados y generalmente inaccesibles, igual que las islas remotas. Esto puede ser resultado de la separación geográfica, de la incompatibilidad de las arquitecturas informáticas, de la incompatibilidad de los protocolos de comunicaciones, etc. Si se consigue integrar las bases de datos en un todo lógico coherente, se puede resolver el problema.

Para lograr la transparencia, un SABDD debe contar con un conjunto de características que la hagan posible. A continuación se describen estas características:

1. **Autonomía**. Este punto se relaciona con el control de la BDD que es realizado por dos entidades: el administrador de la base de datos global quien es responsable del manejo de la base de datos en su conjunto, y los administradores de las bases de datos locales en cada nodo de la red, que son los responsables de sus propias bases de datos locales. Esta característica tiene por nombre autonomía de nodo y da como resultado que cada nodo pueda controlar y procesar su información local, independientemente de cualquier otra computadora. Esto se traduce en un mejor desempeño, si se considera que los datos están localizados donde son más requeridos por las aplicaciones de los usuarios y su procesamiento se hace mediante recursos de este nodo.

   El grado de autonomía de un nodo se refleja en la medida en que éste requiera del SABDD, para realizar la coordinación entre los sistemas administradores de las bases de datos locales que cuentan con los mismos datos.

   Por supuesto, el usuario no debe percatarse de la coordinación entre los sistemas administradores de las bases de datos locales, producida por ellos mismos o por el SABDD, para actualizar su información.

2. **Redundancia controlada**. De modo contrario a lo que sucede en las bases de datos centralizadas donde se busca disminuir la redundancia, aquí los datos pueden estar repetidos en distintos nodos de la red. Esto es preferible cuando se necesita dar mayor disponibilidad a los datos, acercándolos más a las aplicaciones de los usuarios que los requieren.

   La redundancia de datos repercute en una mayor disponibilidad del sistema completo. Por ejemplo, suponga que se tiene un dato repetido en varios nodos, y que este dato es requerido por una aplicación que es activada por un usuario desde algún nodo de la red. Esta aplicación tendrá éxito si alguno de los nodos que contienen el dato se encuentra en falla, o bien si algún enlace de comunicación (no estratégico) se encuentra desactivado.

   Aunque la redundancia de datos trae beneficios, también tiene desventajas derivadas del problema que implica mantener la consistencia e integridad de la BDD. En efecto, si algún dato se encuentra repetido en varios nodos, es posible accederlo desde cualquiera de ellos; sin embargo, su actualización se debe realizar consistentemente en todos ellos.

3. **Recuperación de la base de datos**. La falla de un nodo o la desactivación de algún enlace de comunicación no impide el funcionamiento de las aplicaciones de los usuarios en los nodos restantes. Por supuesto, cuando un nodo se encuentra en falla, el SMBDD debe asegurar que el efecto de las transacciones que no se realizaron durante su caída, se refleje luego de su restablecimiento. A esto se le llama recuperación de la BDD.

4. **Manejo de transacciones**. Otros problemas relacionados con la administración de las BDDs, tienen que ver con el control de la concurrencia, la integridad y la recuperación. Estos problemas son interdependientes y su solución se enfrenta mediante el uso de transacciones. "Una transacción es una unidad atómica de ejecución" [Ceri, 84]. Una transacción es una secuencia de operaciones. Cuando se ejecuta una transacción, todas las operaciones que la integran se deben realizar por completo, pero en caso de que exista alguna complicación, ninguna de las operaciones debe reflejar su efecto sobre los datos. Las transacciones permiten trasladar a la BDD de un estado inicial consistente a un estado final consistente. Es decir, provee integridad a la BDD.

5. **Control de concurrencia**. Los factores de complicación en la preservación de la atomicidad de las transacciones, tienen que ver con las fallas de los nodos y con la ejecución concurrente de las transacciones. La recuperación de la BDD, como se vio antes, permite actualizar a un nodo que se restablece de una falla, a un estado consistente; mientras que el control de la concurrencia es una característica que asegura la atomicidad de las transacciones cuando éstas se ejecutan concurrentemente. Específicamente, el control de concurrencia asegura que la ejecución de transacciones concurrentes en el sistema se llevará a cabo en alguna forma serial (serializabilidad).

2.1.4. Características del entorno distribuido

En esta sección se hacen las consideraciones pertinentes sobre el entorno distribuido, mismas que se relacionan con el grado de heterogeneidad del ambiente distribuido sobre el modelo de datos, los lenguajes de manipulación de datos, la arquitectura de los nodos y la arquitectura de la red en que son soportados. A continuación se describe cada una de ellas.

1. **Modelo de datos.** Como punto de partida, se considera una BDD almacenada en los nodos de una red. Por supuesto el modelo de datos bajo el cual se representa a la BDD es relacional. La selección de este modelo resulta conveniente en ambientes distribuidos donde el manejo de datos es más apropiado orientarlo a conjuntos de datos que a registros.

2. **Lenguajes de manipulación de datos.** De hecho otro factor que ha influido decididamente en la popularidad del modelo de datos relacional, ha sido la aparición de los lenguajes de manipulación de datos no procedimentales que le facilitan la vida al usuario. Esta facilidad se debe a que estos lenguajes permiten indicar qué datos se requieren y no precisan que se especifique la forma de accederlos.

   Los lenguajes de manipulación de datos que son considerados, cobran relevancia luego que la BDD presenta características de fragmentación sobre las relaciones globales que la conforman y de redundancia de estos fragmentos. Gracias a estos lenguajes de manipulación de datos, el usuario ya no tiene que especificar la forma de acceder a la información y se puede olvidar del tipo de fragmentación de las

relaciones globales que posee la BDD. Las relaciones globales pueden poseer una fragmentación del tipo horizontal, vertical o mixto.

A pesar de que el modelo de datos de la BDD es relacional, los lenguajes de manipulación de datos en cada nodo de la red que se consideran pueden ser distintos, pero por simplicidad basados en el cálculo de tuplas o en el álgebra relacional. Por citar un ejemplo, algunos nodos de la red pueden contar con SQL que está basado en una mezcla de álgebra relacional y cálculo de tuplas, y otros nodos, con álgebra relacional.

3. **Arquitectura de los nodos y de la red**. La heterogeneidad de los lenguajes de manipulación de datos de la red invita a considerar la heterogeneidad en el aspecto de la red de computadoras. De hecho, se considera que la red de computadoras donde se encuentra almacenada la BDD poseerá cierta topología y no nos preocuparán los protocolos de comunicación que ésta use, más bien se pensará que la comunicación entre los nodos se efectuará a alto nivel. Esta transparencia de comunicación en la red permite considerar que la arquitectura y los sistemas operativos de los nodos pueden ser distintos.

   Cuando la arquitectura de los nodos de la red es distinta, la implementación de tal transparencia puede lograrse mediante el uso de pasadizos (gateways), al igual que con otros métodos de uso estándar como son los puentes, ruteadores y repetidores, por citar los más representativos. Estos métodos se usan frecuentemente cuando los nodos se encuentran geográficamente separados.

2.2. Diseño de la distribución

El diseño de una BDD es muy complicado, ya que muchos de los aspectos técnicos de las bases de datos centralizadas, se complican al introducir esquemas distribuidos. El problema técnico abarca desde las consideraciones de interconexión de los nodos hasta la distribución de los datos a fin de cumplir con los requerimientos originales brindando una estrategia óptima. Desde el punto de vista organizacional, las BDDs sustituyen a grandes sistemas centralizados, lo cual es deseable para reflejar la estructura descentralizada que se tiene en las grandes empresas.

El diseño de BDDs depende del tipo de sistema al que está dirigido. Cuando se integran sistemas de bases de datos ya funcionando y quizá con diferente arquitectura, se dice que se aplica una metodología *bottom up* (ascendente). En cambio, cuando se implementa un SBDD desde cero, se aplica la metodología denominada *top-down* (descendente).

La metodología descendente para el diseño de una BDD es similar a la de una centralizada y, en términos generales consiste de los pasos siguientes (como se puede ver en la figura 2.1).

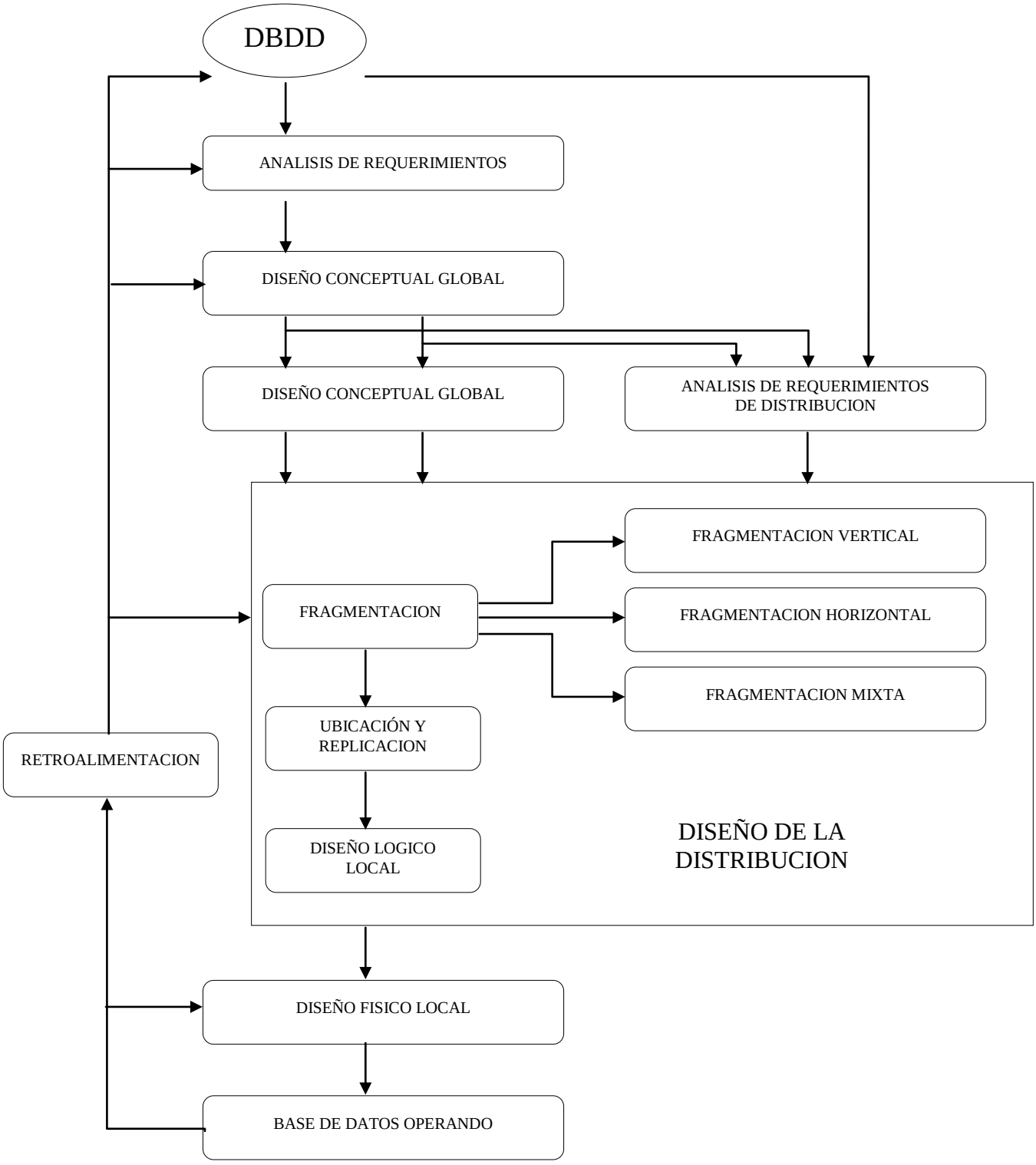

figura 2.1 Metodología de diseño de bases de datos distribuidas

Requerimientos de usuario: Es la reunión de todos los hechos relevantes para la descripción de la base de datos y que son comunicados por los usuarios en forma no estructurada. El producto de esta fase es un diccionario de datos donde se registran los hechos recolectados en forma organizada y sin ambigüedad.

- **Diseño conceptual:** También se le llama diseño de vistas e integración de vistas. Es el diseño de los esquemas para la base de datos en forma independiente de un sistema específico.

- **Diseño lógico**: Es el diseño de esquemas y métodos de acceso a partir del diseño conceptual para un sistema particular, ya sea relacional, orientado a objetos, jerárquico o de red. El modelo se escoge por razones prácticas y por requerimientos del modelo conceptual.

- **Diseño físico**: Diseño de la base de datos para un sistema concreto. Se utilizan sus capacidades específicas para la realización de un sistema de información.

Además de los pasos anteriores, el diseño de una BDD requiere colectar los requerimientos de la distribución y del diseño de ésta. El diseño de la distribución toma el esquema lógico y crea fragmentaciones y asignaciones a los nodos del sistema distribuido con el fin de que los datos estén más cerca del lugar donde se acceden y de utilizar la menor cantidad de fragmentos posible.

Estos pasos pueden insertarse en el desarrollo de la metodología general del diseño de bases de datos descrita anteriormente y como se muestra en la figura anterior, lo cual se realiza con base en considerar los siguientes hechos:

- Si la distribución de los datos ya está dada, se puede concentrar en el diseño físico. Esto sería el caso cuando el sistema es heterogéneo.

- O bien, cuando es homogéneo, se pueden probar diferentes distribuciones y probar con alguna que sea más eficiente, para lo cual se requieren algunas herramientas de decisión como la simulación de la operación del sistema.

Esto quiere decir que el diseño de la distribución debería empezar a realizarse en el diseño lógico.

Tradicionalmente se ha considerado que el diseño de la distribución consta de dos fases seriadas: la fragmentación y la ubicación de fragmentos en uno o varios sitios de la red. En este enfoque, el diseño de la distribución se compone de dos subproblemas que están relacionados entre sí:

- La fragmentación de la base de datos en subconjuntos de relaciones uniformes de información desde el punto de vista de las transacciones.

- Asignación de los fragmentos a los nodos en el sistema distribuido para su uso.

En el diseño de una BDD debe considerarse la repetición de fragmentos; es decir, la asignación a más de un nodo de un fragmento. Esto se realiza con el fin de que se tenga más de una localidad donde puedan accederse cuando otros nodos fallen.

La decisión de repetir fragmentos debe considerar también el costo de mantener la congruencia. Como se ha dicho, la actualización de fragmentos repetidos implica la distribución de las actualizaciones. Por otra parte, debe tenerse en cuenta también la capacidad adicional en dispositivos de almacenamiento estable.

En resumen, el diseño de la distribución deberá realizarse a partir de las siguientes condiciones:

- Fragmentación horizontal.
- Fragmentación vertical.
- Asignación no redundante o asignación redundante.

La fase de requerimientos de distribución analiza la frecuencia de actividad de las aplicaciones en cada nodo y trata de encontrar predicados para cada sitio que puedan indicar fragmentación horizontal. Para ello, es necesario conocer la estructura de las aplicaciones por lo que esta fase se realiza en combinación con el diseño conceptual.

En el diseño de la distribución se toman el esquema global de la base de datos y las tablas de acceso producidas en el paso del diseño lógico global, y se crean fragmentaciones para asignarlas a cada nodo. Hay que decir que el desarrollo de este proceso es no secuencial y puede requerir interacción entre una y otra fase.

Durante el análisis de la distribución se determinan los parámetros de la distribución de datos, los cuales se organizan en los siguientes productos: las tablas de frecuencias, de particionamiento y de polarización.

La tabla de frecuencias es una relación del número de veces que una aplicación se activa en un sitio dado. Cuando una aplicación no se usa en un sitio su valor en la tabla es simplemente cero.

La tabla de particionamiento determina los subconjuntos de tuplas en cada tabla que se usa en cada sitio. Puede determinarse un predicado sobre las columnas que la caracterizan, lo cual se utiliza para realizar fragmentación horizontal.

Al realizar la fragmentación horizontal, pueden distinguirse dos tipos:

- Fragmentación primaria, cuando la aplicación de un predicado en las tuplas de una entidad determina subconjuntos ajenos.

- Fragmentación secundaria, cuando una entidad está relacionada con otra que tiene fragmentación horizontal primaria, los subconjuntos resultantes deben ser

también disjuntos y en consecuencia, sólo se utilizan relaciones de uno a uno o de uno a muchos.

Las tablas de polarizaciones indican la probabilidad de que en un sitio dado una aplicación determinada utilice una fragmentación específica.

Con base en las tablas anteriores, se realiza el diseño de una distribución para determinar lo siguiente:

- Diseño de la fragmentación.
- Asignación no redundante.
- Asignación redundante.
- Reconstrucción de esquemas globales.

El diseño de la fragmentación se basa en la tabla de particionamiento y de polarización, de modo que los subconjuntos resultantes se accedan de manera uniforme. La decisión al realizar esta actividad es más bien lógica y se determina al obtener predicados de las tablas de particiones u observación de los atributos más usados en cada entidad por las aplicaciones.

La asignación no redundante se realiza al determinar el nodo donde existen aplicaciones que utilizan con mayor frecuencia cierto fragmento. En las tablas anteriores, esto se determina como el producto de la tabla de frecuencia y de polarizaciones para una partición dada.

La asignación redundante se realiza al aplicar heurísticas a las asignaciones no redundantes ya realizadas para repetir un fragmento dado si su beneficio al asignarlo en otros sitios es mayor. Este beneficio puede ser difícil de medir debido a la complejidad que hay para mantener la congruencia de las repeticiones; sin embargo, se puede obtener una apreciación si el número de accesos es mayor que el costo de actualizar un fragmento repetido.

La reconstrucción de esquemas locales construye un esquema de la asignación del fragmento final. Esta fase se encarga de la asignación de relaciones del esquema E-R global.

Suponiendo que muchos enlaces M a N se implementan como asociaciones entre los identificadores de las entidades correspondientes, se sugiere colocar las entidades auxiliares en el sitio de las entidades o fragmentos con la cardinalidad más grande, de modo que tengan que transmitirse pocos identificadores.

2.3. Trabajos relacionados sobre fragmentación vertical y ubicación de fragmentos

Como ya se mencionó, el problema de diseño en la distribución en BDDs es un problema de optimización muy complejo, por el número de variables que intervienen y las interrelaciones que existen entre éstas. Existen varias publicaciones que demuestran la complejidad del problema.

Se tiene como antecedente que los primeros trabajos son los relacionados con la ubicación de archivos (FAP, por sus siglas en inglés) en los sitios de una red de computadoras, teniendo como finalidad que los programas de aplicación accedan a esos archivos optimizando tiempos de respuesta y/o costos. En particular uno de los pioneros de estas investigaciones es W. Chu en 1969. Posteriormente L. W. Dowdy en 1982 da una excelente descripción de los trabajos en FAP [Dowdy, 82]. En estas investigaciones se divide el problema en modelos que minimizan costos de comunicaciones y/o que optimizan tiempos de respuesta.

Posteriormente S. Ceri, G. Navathe y G. Wiederhold proponen un modelo de optimización para la ubicación no replicada de datos y se introduce la replicación de datos a partir de la solución óptima no replicada [Ceri, 83]. En 1984 ellos mismos proponen diferentes algoritmos para realizar la fragmentación vertical y la ubicación, los cuales están basados en funciones objetivo empíricas [Ceri, 84].

Sin embargo más adelante conviene destacar que en diferentes años, Ceri, Apers, Ozsu, Date y March [Ceri, 84; Apers, 88; Ozsu, 91; Date, 95; March, 95] coinciden en que **tradicionalmente se ha considerado el diseño de la distribución en dos fases seriadas: la fragmentación y la ubicación de fragmentos**. Una vez hecha esta consideración, D. Xiaolin y F. J. Maryansky proponen estrategias para la ubicación y replicación de datos en una topología que cambia dinámicamente; es decir, con nodos que se desconectan de la red y que pueden volver a conectarse en otro punto distante [Xiaolin, 98].

Cabe la pena mencionar que también D. W. Cornell en 1989 trata de manera simultánea la ubicación de relaciones y la ubicación del procesamiento de las operaciones de “reunión”, y se utiliza programación lineal como método de solución [Cornell, 89].

Ya en la década de los noventa Kulkarni presenta una metodología para la fragmentación horizontal y su ubicación [Kulkarni, 91]. Para ello se utiliza conocimiento semántico para formar diferentes alternativas de diseño, las cuales son evaluadas en cuanto a su tiempo de respuesta y costo total.

Por otro lado O. Wolffon y S. Jajodia proponen un algoritmo de ubicación dinámica de datos DDA, el cual cambia dinámicamente el esquema de replicación de un objeto para ajustarlo a los cambios en los patrones de lectura-escritura [Wolffon, 92]. El algoritmo toma como base una copia primaria.

R. Muthuraj, S. Chakravarthy, R. Vadarajan y S. B. Navathe abordan formalmente el problema de la fragmentación vertical, utilizando una función objetivo que permite cuantificar esquemas de fragmentación [Muthuraj, 93].

En ese mismo año D. Ferguson, C. Nikolaou y Y. Yemini presentan un modelo donde los fragmentos de la BDD pueden ser movidos y replicados entre los nodos de una red de computadoras. El acceso a los fragmentos es efectuado por cada nodo asignando relaciones de precio-calidad a cada fragmento, la distancia de transferencia entre sitios y la demanda de los mismos determina esta relación precio-calidad [Ferguson, 93].

S. T. March y S. Rho extienden el trabajo de Cornell [Cornell, 89] para incluir la replicación, control de concurrencia y ubicación de datos y operaciones [March, 95].

También en ese año L. Xuemin y M. Orlowska tratan el problema de ubicación de fragmentos por medio de programación lineal, teniendo como objetivo encontrar las rutas de propagación de las actualizaciones de manera que se minimicen los costos de comunicación globales [Xuemin, 95].

Alguien que le da un giro novedoso al diseño de una BDD es Joaquín Pérez Ortega el cuál propone un modelo matemático (FURD) que considera a los atributos de las relaciones como unidades de datos con acceso independiente [Pérez, 97]. Se investiga la fragmentación vertical y la ubicación de datos de forma simultánea en un ambiente distribuido en donde pueden cambiar las transacciones y la cardinalidad de las tablas de manera dinámica. Esta consideración da la libertad de ubicarlos de manera óptima en los diferentes nodos de la red para satisfacer un conjunto de transacciones. Los atributos que son ubicados en un mismo nodo, forman de manera natural un fragmento vertical de la relación. Se muestra mediante ejemplos cómo el modelo FURD mejora la ubicación de datos propuesta por Navathe [Navathe, 84] y es muy similar con el diseño de la fragmentación propuesta por Muthuraj [Muthuraj, 93].

Dada la cercanía con el Dr. Pérez es que se toma como punto de partida su trabajo para extenderlo considerando ahora el tiempo de respuesta, por lo que finalmente, en la tabla 2.1 se muestra un resumen de los trabajos realizados después del año 94 a la fecha indicando el tipo de problema que tratan así como los valores que se minimizan.

tabla 2.1: Trabajos recientes por tipo de problema y valores a minimizar

Trabajos	**Tipo de Problema**			**Valores a minimizar**		**Métodos de Solución**	
	Fragmentación	**Ubicación**	**Fragmentación y Ubicación Integrada**	**Costos de Transmisión o Procesa miento**	**Tiempo de Respuesta**	**Algoritmos Heurísticos**	**Formulación de Programación Matemática**

Chakravarthy, 94	✓			✓		✓	
Tamhankar, 98			✓	✓		✓	
Golfarelli, 99	*			✓			✓
Huang, 01		✓		✓		✓	
Pérez, 03			✓	✓			✓
Ulus, 03		✓		✓		✓	
Menon, 05		✓		✓			✓
Basseda, 06		✓		✓		✓	
Ma, 06			✓	✓		✓	
Hababeh, 07	✓	✓		✓		✓	
Gorla, 08	*			✓		✓	
Tambulea, 08		✓		✓		✓	
Karimi,09		✓		✓			
Khan,10				✓			
Sevinc,10		✓		✓		✓	
Kamali,11		✓		✓		✓	
Goli,12	✓			✓		✓	
Song,13				✓			
Este trabajo,14			✓		✓		✓

* para bases de datos centralizadas

La segunda tercera y cuarta columnas de la tabla 2.1 muestran que algunos trabajos han tratado solamente el problema de fragmentación, otros trabajos han tratado el problema de la ubicación de fragmentos, y algunos otros han tratado integralmente ambos problemas. La quinta columna muestra que todos los trabajos previos han considerado costos de transmisión, acceso o procesamiento, sólo el trabajo reportado en [Golfarelli, 99] para almacenes de datos ha intentado minimizar el tiempo de respuesta, pero sólo indirectamente mediante el número de páginas de disco que deben ser accedidas para procesar una consulta. Es este caso es posible minimizar el tiempo de respuesta mediante la minimización del número de páginas accedidas; desafortunadamente, este truco tiene éxito sólo cuando se trata el problema de fragmentación sin incluir el de ubicación, como en [Golfarelli, 99]; en este caso se justifica ignorar la ubicación ya que todos los fragmentos se almacenan en la misma computadora. Finalmente, la séptima columna muestra que la mayoría de los trabajos han propuesto algoritmos heurísticos para tratar el problema de fragmentación y ubicación de BDDs, y sólo unos pocos han propuesto formulaciones de programación matemática. El modelado de un problema mediante programación matemática tiene la gran ventaja de que el problema puede ser resuelto con una gran cantidad de algoritmos metaheurísticos que existen en la actualidad.

Capítulo 3

Modelado del tiempo de respuesta

Capítulo 3. Modelado del tiempo de respuesta

El objetivo de la simulación es reproducir lo que se busca lograr o aprender del modelo. El alcance de un modelo incluye todos los objetivos e interacciones que son relevantes y necesarios para lograr los objetivos.

La parte más importante de cualquier estudio de simulación es determinar cuáles son los objetivos. Sin una clara definición y comprensión de lo que se está tratando de lograr, no se podrá definir el alcance o nivel de detalle del modelo.

El nivel de detalle de un modelo es también determinado por los objetivos del estudio. El modelo debe ser suficientemente detallado para replicar el comportamiento del sistema según sea necesario.

Algunos objetivos típicos de la simulación son los siguientes:

1. Visualización. Observar qué está sucediendo en el sistema.

2. Cálculos (analizar/optimizar). Cuantificar qué está sucediendo en el sistema. (Nota: el objetivo de la simulación realizada en este proyecto es de este tipo.)

3. Comunicación. Mostrar qué está sucediendo en el sistema.

Muchas variables en un sistema se dicen que son aleatorias (pueden tomar cualquier valor al azar). Los objetos que tienen su desempeño ligado a otros son interdependientes (cada uno afecta a los otros). La aleatoriedad e interdependencia se encuentra en el entorno. La simulación es una herramienta que ayuda a visualizar, analizar y optimizar sistemas complejos.

Tomando juntas: aleatoriedad + interdependencia = complejidad.

La simulación entra en la caja de herramientas para resolver problemas, junto con otras herramientas. Hojas de cálculo, modelos de programación lineal, simulaciones Monte Carlo, etc., son todas herramientas válidas para resolver problemas.

Este capítulo tiene como fin desarrollar una fórmula matemática para el tiempo medio de respuesta de las consultas en un sistema de base de datos distribuída (SBDD) con fragmentación vertical en términos de los parámetros de éste. Para tal efecto en la Sección 3.1 se describen de manera general los procesos de transmisión y procesamiento de consultas, así como el de transmisión de las respuestas en un SBDD. Después en la Sección 3.2 se explica la dificultad para desarrollar una fórmula matemática exacta para el tiempo medio de respuesta, debido a lo cual se introduce una suposición simplificadora que permite desarrollar una fórmula cerrada para el tiempo medio de respuesta. La Sección 3.3 justifica el modelado del tiempo de respuesta.

Para apreciar qué tan bien describe la fórmula simplificada el comportamiento del tiempo medio de respuesta, se realizaron varias pruebas experimentales en donde se comparan los valores estimados del tiempo medio de respuesta contra los valores exactos obtenidos mediante la simulación de los procesos de transmisión y procesamiento de consultas en un SBDD. Para tal efecto, en la Sección 3.4 se describe un ejemplo detallado de los procesos de transmisión y procesamiento de consultas, así como el de transmisión de las respuestas. Después en la Sección 3.5 se presenta el pseudocódigo del programa con el que se realizaron las simulaciones; mientras que en la Sección 3.6 se presenta el pseudocódigo del programa que permite ajustar la fórmula simplificada a un conjunto de valores dados (en este caso, valores obtenidos de la simulación). Finalmente, en la Sección 3.7 se presentan varias gráficas que permiten apreciar qué tan bien describe la fórmula simplificada el comportamiento del tiempo medio de respuesta.

3.1. Descripción general del proceso de transmisión y procesamiento de consultas

Para representar el proceso de transmisión y procesamiento de consultas se utilizó un diagrama, el cual representa una red de cómputo interconectada y está organizado como se ve en la figura 3.1. En dicho diagrama, cada nodo v_i representa un sitio en el que, en general, se pueden generar consultas y existe un servidor de bases de datos, el cual permite acceder a los datos de algún fragmento de una relación (tabla) almacenado en v_i. Además, cada arco e_j del diagrama representa una línea de comunicación mediante la cual se pueden trasmitir mensajes en ambos sentidos, los cuales pueden contener consultas y respuestas a las consultas que generan los servidores.

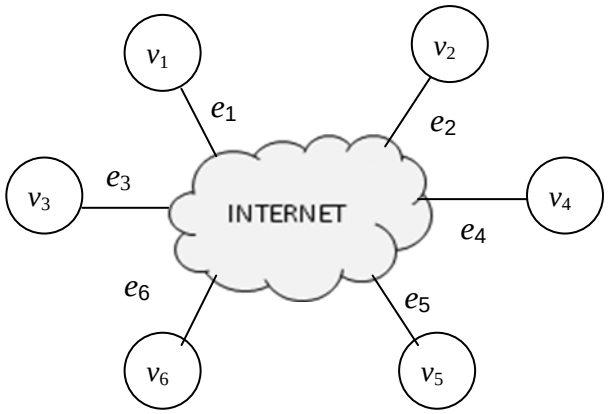

figura3.1 Diagrama del proceso de transmisión y procesamiento de consultas

Como ya se había mencionado antes, la parte más importante de un proyecto de simulación es definir el objetivo. En este caso el objetivo de nuestro ejemplo es simular y/o

representar el modelo para determinar cuantitativamente el comportamiento de las consultas en un SBDD dentro de una red con las características mostradas en el diagrama.

Utilizando un fragmento de la tabla Empleado cuyo contenido se muestra a continuación:

Empleado(Emp)	
Nombre	Nombre del empleado
Especialidad	Especialidad que maneja el empleado

Se procede a dar la descripción del proceso de simulación mismo que se presenta a continuación:

Se inicia al realizar una consulta como la que se muestra enseguida:

Consulta 1: ¿Qué especialidad tiene el Sr. Pérez?

Esta consulta expresada en SQL queda de la siguiente manera:

```
C1:   Select Especialidad
      From Emp
      Where Emp.Nombre = "Sr. Pérez";
```

Si se supone que el nodo v_4 contiene un fragmento de la tabla Emp que contiene las columnas Especialidad y Nombre, entonces ocurren los eventos descritos más adelante en el caso en que la consulta se genere en el nodo v_1.

Una vez hecha esta consideración se describen los eventos como van ocurriendo, en primer lugar para determinar el tiempo en el que se genera una consulta:

Evento 1. Se genera la consulta en el nodo v_1, y se genera el Mensaje1 que contiene la Consulta1 con destino al nodo v_4, el cual contiene un fragmento de la tabla Emp que posee la información necesaria para contestar la consulta.

Evento 2. En caso de que la línea e_1 esté ocupada por la transmisión de algún otro mensaje, el Mensaje1 espera en v_1 a ser enviado a través de Internet hacia el nodo v_4.

Evento 3. Cuando la línea e_1 se encuentra desocupada, se transmite el Mensaje1 del nodo v_1 hacia el nodo v_4 por Internet (en general, atravesando varias líneas, en las cuales el Mensaje1 posiblemente tendrá que esperar en caso de que estén ocupadas).

Evento 4. En caso de que el servidor en v_4 esté ocupado procesando alguna otra consulta, la Consulta1 espera en v_4.

Evento 5. Cuando el servidor v_4 se encuentra desocupado, se procesa la Consulta1 y se genera el Mensaje2, el cual contiene la respuesta de la consulta; es decir, el servidor de bases de datos busca en el fragmento almacenado en v_4 los datos solicitados por la Consulta1.

Evento 6. En caso de que la línea e_4 esté ocupada por la transmisión de algún otro mensaje, el Mensaje2 espera en v_4 a ser enviado a través de Internet hacia el nodo v_1.

Evento 7. Cuando la línea e_4 se encuentra desocupada, se transmite el Mensaje2 del nodo v_4 hacia el nodo v_1 por Internet (en general, atravesando varias líneas, en las cuales el Mensaje2 posiblemente tendrá que esperar en caso de que estén ocupadas).

Evento 8. Finalmente, cuando el Mensaje2 es recibido en el nodo v_1, los resultados contenidos en éste son entregados al usuario/aplicación que generó la Consulta1.

3.2. Expresión aproximada para el tiempo de respuesta

En este punto es conveniente aclarar que las consultas se generan en los nodos de la red en tiempos aleatorios, los cuales usualmente se supone que son acordes a un proceso de Poisson. Además, se supondrá que las longitudes de los mensajes correspondientes a las consultas también son variables aleatorias generadas por un proceso de Poisson. Esta suposición trae como consecuencia que las duraciones de la transmisión de los mensajes sean variables aleatorias con distribución exponencial negativa (la cual está relacionada con los procesos de Poisson). Finalmente, se supondrá que el tiempo de procesamiento de una consulta por un servidor de bases de datos así como la longitud del mensaje correspondiente a la respuesta de la consulta son variables aleatorias con una distribución exponencial negativa. Esta última suposición trae como consecuencia que las duraciones de la transmisión de los mensajes de respuesta también sean variables aleatorias con distribución exponencial negativa.

El hecho de que los tiempos de generación de las consultas, las longitudes de éstas, sus tiempos de procesamiento y las longitudes de las respuestas sean variables aleatorias trae como consecuencia que los tiempos que las consultas tienen que esperar en los nodos para ser transmitidas en las líneas también sean aleatorios, y lo mismo ocurre con los tiempos que las consultas tienen que esperar en los servidores para ser procesadas, así como con el tiempo que las respuestas tienen que esperar en las líneas para ser transmitidas de regreso a los nodos donde se generaron sus respectivas consultas. Finalmente, todo esto trae como consecuencia que el tiempo de respuesta para cada consulta (es decir, el tiempo

transcurrido desde la generación de la consulta en un nodo hasta que la respuesta de la consulta es recibida en el nodo) sea una variable aleatoria, lo cual implica que en general el tiempo de respuesta puede variar de una consulta a otra.

Aunque en general el tiempo de respuesta es variable y su valor puede cambiar de una consulta a otra, de todos modos debe existir un valor promedio para el tiempo de respuesta. Para los propósitos de esta investigación es de interés este valor promedio. Desafortunadamente, el desarrollo de una fórmula para calcular el tiempo medio de respuesta en términos de los parámetros del sistema es una tarea extremadamente compleja, para lo cual se necesitan amplios conocimientos de teoría de colas.

Para darse una idea sobre la complejidad de este problema, conviene tomar como referencia un problema similar del ámbito de las redes de comunicaciones para computadoras, el cual se describe en [Kleinrock, 07] (Capítulo 3). En dicha referencia se desarrolla una expresión para el retardo medio de los mensajes (tiempo de espera más transmisión) en un nodo, la cual involucra la transformada de Laplace. Desafortunadamente, la ecuación es extremadamente compleja y no ha podido ser resuelta.

En vista de la dificultad mencionada en [Kleinrock, 07] se ha propuesto una simplificación, la cual permite obtener una fórmula simple para el retardo medio de los mensajes. Dicha simplificación involucra una *suposición de independencia* entre los tiempos de interarribo de mensajes a los nodos y las longitudes de mensajes consecutivos conforme éstos avanzan por el grafo. Estrictamente, el proceso de llegada de mensajes a un nodo, por ejemplo el nodo Servidor de la figura 3.2.c, depende de los procesos de llegada de mensajes a los nodos que envían mensajes al nodo en cuestión, en nuestro ejemplo los nodos Fuente A, Fuente B, Fuente C, Fuente D y Fuente E. En [Kleinrock, 07] se muestra que esta dependencia entre procesos es el origen de la complejidad de la ecuación para el retardo medio de los mensajes (ver explicación en la Sección 3.3).

Si se adoptara la suposición de que el proceso de llegada de mensajes a cualquier nodo fuera independiente de los procesos de los otros nodos, esto implicaría que el proceso de llegada de mensajes en cada nodo se comportaría como un proceso de Poisson. En tales circunstancias, el comportamiento de los mensajes que son transmitidos a través de cada línea que sale de un nodo se puede modelar como una cola M/M/1 (es decir una cola con un servidor, donde el proceso aleatorio de llegada es markoviano, lo cual es satisfecho por los procesos de Poisson, y el proceso aleatorio de servicio también es markoviano). En estas condiciones el retardo medio de los mensajes en la línea (el cual incluye el tiempo de espera más el tiempo de transmisión) está dado por la siguiente fórmula [Kleinrock, 07]:

$$T = \frac{1}{\mu C - \lambda} \qquad 3.1)$$

donde:

$1/\mu$ es la longitud media de las consultas (expresada en bits por mensaje),

C es la velocidad de transmisión de la línea (expresada en bits por segundo) y,
λ es la intensidad de llegada de mensajes a ser transmitidos por la línea (expresada en mensajes por segundo).

Con relación al procesamiento de las consultas en los servidores, ocurre una situación similar a la de la transmisión de mensajes en las líneas, lo cual da lugar a una fórmula similar a (3.1) para el retardo medio de procesamiento de las consultas. Como se verá en el Capítulo 4, conociendo el retardo medio de los mensajes en cada línea y el retardo medio de procesamiento de las consultas en los servidores, es posible calcular el tiempo medio de respuesta, el cual es una suma ponderada de aquéllos.

La adopción de una simplificación en un modelo ocasiona que el comportamiento de éste se aleje de la realidad en menor o mayor medida. Para determinar qué tanto afecta al modelo la adopción de la suposición de independencia, se realizó en esta investigación un estudio similar al descrito en [Kleinrock, 07], el cual consiste en realizar pruebas experimentales para comparar los retardos estimados por la fórmula (3.1) versus los retardos "reales" obtenidos mediante la simulación de los procesos aleatorios que ocurren con diferentes tipos de redes y diversos valores para los parámetros del sistema (velocidad de transmisión de las líneas, tiempo medio de procesamiento en los servidores, intensidad de llegada de mensajes).

A semejanza de un experimento de simulación para redes de comunicaciones para computadoras [Kleinrock, 07], para esta investigación se consideraron tres configuraciones de nodos fuente y servidores de bases de datos como se muestra en la figura 3.2.

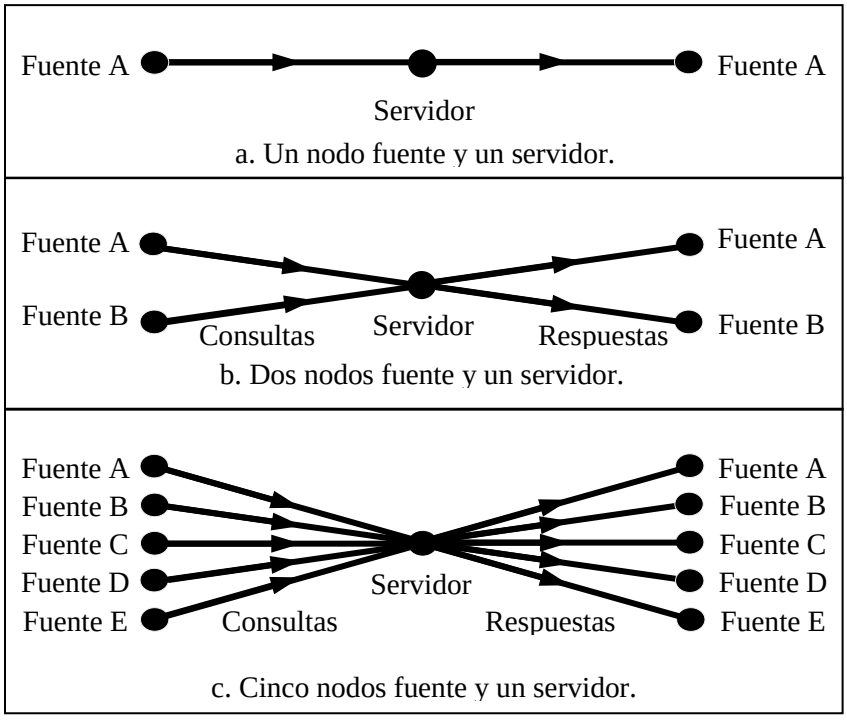

figura 3.2 Diagramas de los experimentos realizados.

Como se muestra en la figura 3.2 anterior se realizaron tres grupos de experimentos para la simulación. En el primero se consideró sólo un nodo y un servidor; en el segundo, dos nodos y un servidor; y un tercer caso en el que se utilizan cinco nodos y un servidor. En la figura 3.2 los nodos fuente son aquéllos donde se generan las consultas. Cada consulta es transmitida a través de una línea de comunicación al nodo servidor, donde es procesada para generar una respuesta que contiene los datos solicitados por la consulta. Cada respuesta es enviada del servidor (mediante una línea) al nodo fuente donde se generó la consulta correspondiente a la respuesta. Conviene aclarar que en la figura 3.2 cada nodo fuente se muestra desdoblado en dos: los nodos fuente que aparecen a la izquierda del servidor se usan para representar el proceso de envío de las consultas de los nodos fuente al servidor; mientras que los nodos fuente que aparecen a la derecha sirven para representar el proceso de envío de las respuestas del servidor a los nodos fuente.

3.3. Justificación del modelado del tiempo de respuesta

El diseño de una red de comunicación, requiere de una descripción matemática del comportamiento del tiempo de mensajes, el cual es considerado el lapso que tiene que esperar una consulta para saber si el servidor está disponible o no. Una forma de hacerlo es a través de un conjunto adecuado de variables de estado. Ese conjunto de variables debe cumplir dos condiciones: 1) el comportamiento del retardo de mensaje, y 2) sus propiedades markovianas (el conocimiento de las variables de estado en el tiempo t y el conocimiento de todas las llegadas de mensajes de fuentes externas a la red en el intervalo cerrado (t, t')). [Kleinrock, 07].

Se considera una red de comunicación con los nodos N y M con canales de un solo sentido. En este caso el estado de la red en cualquier instante de tiempo debe incluir información detallada sobre el número de mensajes en cada cola, la longitud de cada mensaje y el tiempo necesario para completar la transmisión en curso en cada canal. Además, se supone que cada mensaje se etiqueta con un origen, un destino y una prioridad.

La solución a este problema permanece aún sin resolverse en virtud de que éste conduce a ecuaciones cuya solución aún no ha sido encontrada. Lo que es más, el comportamiento estocástico del problema se incrementa al introducir una dependencia entre algunas de las variables aleatorias, lo cual lleva a la conclusión de que es necesario incluir funciones de densidad de probabilidad exponencial, con lo que se puede inferir que la fuente de la dificultad radica entre los intervalos de tiempo de emisión de los mensajes y las longitudes de los mensajes internos, al menos en lo que a redes se refiere; no obstante el punto general a resaltar es que, suponer una independencia entre ellos, elimina esta dificultad.

La complejidad de la descripción de las variables de estado se debe en parte a la restricción de que cada mensaje, al entrar en la red, tiene una longitud constante. El mensaje mantiene esta misma longitud a través de la red. La identificación de una longitud constante con cada mensaje no sólo aumenta la dimensionalidad de la descripción del estado, también complica el comportamiento estocástico de la red mediante la introducción

de una dependencia entre algunas de las variables aleatorias que describen el comportamiento de la red.

Si se tienen en cuenta dos mensajes sucesivos que llegan a un nodo *i* de la red, entonces el tiempo de llegada entre estos mensajes no es independiente de la longitud del segundo mensaje, ya que hay que considerar el tiempo de espera para completar la transmisión del mensaje, esto incrementa la dificultad del problema.

3.4. Ejemplo de procesos de transmisión y procesamiento de consultas

Para ilustrar los procesos que ocurren en la transmisión de consultas, el procesamiento de éstas en los servidores, y la transmisión de las respuestas, se considerarán el grafo de la figura 3.2b y varias consultas generadas en la Fuente A y la Fuente B. Las tablas 3.1 y 3.2 muestran algunos datos obtenidos de una simulación que sirven para ilustrar los procesos involucrados en la transmisión y procesamiento de consultas, incluyendo el proceso de transmisión de respuestas.

Específicamente, la tabla 3.1 muestra los valores de los tiempos de interarribo de las consultas (es decir, el tiempo transcurrido entre la llegada de una consulta y la llegada de la siguiente), las duraciones de transmisión de las consultas, las duraciones del procesamiento de las consultas en el servidor y las duraciones de transmisión de las respuestas. Todos estos tiempos fueron generados aleatoriamente con diferentes distribuciones exponenciales negativas. (Nota: como se verá en la siguiente subsección, en realidad no se genera la duración de transmisión de las consultas sino la longitud de éstas, de la cual se puede calcular su duración).

tabla 3.1: Datos aleatorios de un ejemplo del proceso de transmisión y procesamiento de consultas

Evento	Fuente	Tiempo Interarribo	Duración Transmisión Consulta	Duración Procesamiento Consulta	Duración Transmisión Respuesta
1	A	11.394	0.057	7.824	1.307
2	A	1.165	0.246	0.446	0.969
3	A	8.916	0.040	5.627	10.962
4	B	1.054	1.482	0.742	1.548
5	B	1.744	2.198	2.158	7.954
6	B	6.733	2.198	0.685	0.269
7	B	1.508	0.399	0.446	1.674
8	B	4.308	0.146	0.742	15.219

tabla 3.2: Valores calculados de un ejemplo del proceso de transmisión y procesamiento de consultas

Even-to	Fuen-te	Arribo Consulta	Inicio Trans. Consulta	Fin Trans. Consulta	Inicio Proc. Consulta	Fin Proc. Consulta	Inicio Trans. Respuesta	Fin Trans. Respuesta
1	A	11.394	11.394	11.451	11.451	19.275	19.275	20.582
2	A	12.560	12.560	12.806	20.406	20.853	20.853	21.821
3	A	21.476	21.476	21.516	21.595	27.222	27.222	38.184
4	B	1.054	1.054	2.536	2.536	3.278	3.278	4.826
5	B	2.797	2.797	4.995	4.995	7.153	7.153	15.107
6	B	9.531	9.531	11.729	19.275	19.960	19.960	20.229
7	B	11.039	11.729	12.128	19.960	20.406	20.406	22.080
8	B	15.347	15.347	15.492	20.853	21.595	22.080	37.300

Con base en los datos de la tabla 3.1, el programa de simulación calculó para cada consulta el tiempo de arribo, el tiempo de inicio de transmisión, el tiempo de fin de transmisión, el tiempo de inicio de procesamiento, el tiempo de fin de procesamiento, el tiempo de inicio de transmisión de la respuesta correspondiente a la consulta y el tiempo de fin de transmisión de la respuesta. Estos valores calculados se muestran en la tabla 3.2.

Para entender cómo se calcularon los valores de la tabla 3.2, considérese primero el evento 1; en este caso el valor del tiempo de arribo de la consulta se hace igual al tiempo de interarribo (11.394); el tiempo de inicio de transmisión es igual al tiempo de arribo (11.394), ya que se encuentra desocupada la línea de transmisión de la fuente A hacia el servidor; el tiempo de fin de transmisión de la consulta es igual al tiempo de inicio más la duración de transmisión (11.394+0.057 = 11.451); el tiempo de inicio de procesamiento es igual a 11.451, ya que es el tiempo en que termina de transmitirse la consulta hacia el procesador, y como éste se encuentra desocupado, se inicia de inmediato el procesamiento de esta consulta; el fin del procesamiento de la consulta es igual al tiempo de inicio del procesamiento más la duración de éste (11.451+7.824 = 19.275); el tiempo de inicio de transmisión de la respuesta es 19.275, ya que la línea del servidor a la fuente A se encuentra desocupada; finalmente, el tiempo de fin de transmisión de la respuesta es igual al tiempo de inicio más la duración de la transmisión (19.275 + 1.307 = 20.582). En este punto conviene destacar que el tiempo de respuesta para el evento 1 es igual al tiempo de fin de transmisión de la respuesta menos el tiempo de arribo de la consulta, es decir, 20.582– 11.394 = 9.188. Para el evento 4 el cálculo es similar al del evento 1. En general, el tiempo de respuesta para cada evento es la diferencia entre los valores de la novena columna de la tabla 3.2 y los valores de la tercera columna.

En los eventos anteriores, éstos encuentran las líneas de transmisión y el servidor libres y no tienen que esperar para ser atendidos; sin embargo, para otros eventos no ocurre lo mismo; así que en estos casos los cálculos se realizan de manera diferente. Por ejemplo,

considérese el evento 2; en este caso el tiempo de arribo de la consulta es igual al tiempo de arribo del evento anterior en la misma fuente (evento 1) más el tiempo de interarribo del evento 2 (11.394+1.165 = 12.559, +0.001 por efecto de redondeo); el tiempo de inicio de transmisión es igual al tiempo de arribo (12.560), ya que se encuentra desocupada la línea de transmisión de la fuente A hacia el servidor (nótese que la consulta 1 terminó de transmitirse en 11.451); el tiempo de fin de transmisión de la consulta es igual al tiempo de inicio más la duración de transmisión (12.560+0.246 = 12.806); el tiempo de inicio de procesamiento no puede ser igual al tiempo en que esta consulta llega al servidor (12.806), porque en éste ya se encuentran la consulta 6 (con tiempo de llegada al servidor en 11.729) y la consulta 7 (con tiempo de llegada al servidor en 12.128), así que tiene que esperar al final del procesamiento de esta última consulta (20.406), lo cual implica que el tiempo de inicio de procesamiento de la consulta 2 sea 20.406; el fin de procesamiento de la consulta es igual al tiempo de inicio del procesamiento más la duración de éste (20.406+0.446 = 20.852, +0.001 por efecto de redondeo); el tiempo de inicio de transmisión de la respuesta es 20.853, ya que la línea del servidor a la fuente A se encuentra desocupada (nótese que la respuesta 1 terminó de transmitirse en 20.582; finalmente, el tiempo de fin de transmisión de la respuesta es igual al tiempo de inicio más la duración de la transmisión (20.853+0.969 = 21.822, –0.001 por efecto de redondeo).

Otro ejemplo parecido ocurre con el evento 8; en este caso el tiempo de arribo de la consulta es igual al tiempo de arribo del evento anterior en la misma fuente (evento 7) más el tiempo de interarribo del evento 7 (11.039+4.308 = 15.347); el tiempo de inicio de la transmisión es igual al tiempo de arribo (15.347), ya que se encuentra desocupada la línea de transmisión de la fuente B hacia el servidor (nótese que la consulta 7 terminó de transmitirse en 11.729); el tiempo de fin de transmisión de la consulta es igual al tiempo de inicio más la duración de transmisión (15.347+0.146 = 15.493. –0.001 por efecto de redondeo); el tiempo de inicio de procesamiento no puede ser igual al tiempo en que esta consulta llega al servidor (15.492), porque en éste ya se encuentran la consulta 1 (con tiempo de llegada al servidor en 11.451), la consulta 6 (con tiempo de llegada al servidor en 11.729), la consulta 7 (con tiempo de llegada al servidor en 12.128), y la consulta 2 (con tiempo de llegada al servidor en 12.806), así que tiene que esperar al fin de procesamiento de esta última consulta (20.853), lo cual implica que el tiempo de inicio de procesamiento de la consulta 8 sea 20.853; el fin de procesamiento de la consulta es igual al tiempo de inicio del procesamiento más la duración de éste (20.853+0.742 = 21.595); el tiempo de inicio de transmisión de la respuesta no puede ser igual al tiempo en que esta consulta termina de ser procesada en el servidor (21.595), porque en la línea de transmisión del servidor a la fuente B se encuentra en transmisión la respuesta 7 (con tiempo de llegada a la línea en 20.406), así que tiene que esperar al fin de transmisión de ésta (22.080), lo cual implica que el tiempo de inicio de transmisión de la respuesta 8 sea 22.080; finalmente, el tiempo de fin de trasmisión de la respuesta es igual al tiempo de inicio más la duración de la transmisión (22.080+15.219 = 37.299, +0.001 por efecto de redondeo).

3.5. Simulación del tiempo de respuesta

A continuación se describe mediante un seudocódigo la simulación de los procesos aleatorios involucrados en la generación de cada consulta, su transmisión en la línea de comunicación del nodo fuente hacia el servidor (incluyendo su posible espera en el nodo fuente), su procesamiento en el servidor (incluyendo su posible espera en el servidor) y la consecuente generación de la respuesta, y finalmente la transmisión de la respuesta del servidor al nodo fuente (incluyendo su posible espera en el nodo servidor). Para esta simulación se consideran los tres grafos que se muestran en la figura 3.2 (con un servidor y una fuente, con un servidor y dos fuentes, y con un servidor y cinco fuentes) y se generan 10,000 eventos para cada fuente.

Seudocódigo para la Simulación del tiempo de Respuesta

```
1)  inicio
2)  // NúmeroFuentes: número de fuentes en el grafo
3)  para Fuente←1 hasta NúmeroFuentes hacer
4)     // Cálculo del No. de evento inicial y final para cada fuente
5)     EventoInicial ← 10,000*(Fuente-1) + 1
6)     EventoFinal ← 10,000*Fuente
7)     // Generación de valores aleatorios para las consultas y respuestas
8)     para Evento←EventoInicial hasta EventoFinal hacer
9)        // Generación de tiempos de arribo de las consultas
10)       TiempoInterarriboConsulta(Evento) ← número aleatorio
11)       si Evento=EventoInicial entonces
12)          TiempoArriboConsulta(Evento) ← TiempoInterarriboConsulta(Evento)
13)       si no
14)          TiempoArriboConsulta(Evento) ← TiempoArriboConsulta(Evento-1) +
             TiempoInterarriboConsulta(Evento)
15)       fin si
16)       // Generación de tamaños de las consultas
17)       TamañoConsulta(Evento) ← número aleatorio
18)       // Generación de duraciones de procesamiento de las consultas
19)       DuraciónProcesamientoConsulta(Evento) ← número aleatorio
20)       // Generación de tamaños de las consultas
21)       TamañoRespuesta(Evento) ← número aleatorio
22)       Evento ← Evento + 1
23)    fin hacer
24)    Fuente ← Fuente + 1
25) fin hacer
26)
27) // Cálculo de los tiempos de inicio y fin de transmisión de las consultas
28) para Fuente←1 hasta NúmeroFuentes hacer
29)    EventoInicial ← 10,000*(Fuente-1) + 1
30)    EventoFinal ← 10,000*Fuente
```

```
31)      para Evento←EventoInicial hasta EventoFinal hacer
32)         DuraciónTransmisiónConsulta ←
            TamañoConsulta(Evento)/VelocidadTransmisión
33)         si Evento=EventoInicial entonces
34)            InicioTransmisiónConsulta(Evento) ← TiempoArriboConsulta(Evento)
35)         // El tiempo de inicio de transmisión de la consulta es el mayor del tiempo
36)         // de arribo de la consulta y el fin de transmisión de la consulta anterior
37)         si no
38)            InicioTransmisiónConsulta(Evento) ←
               max(TiempoArriboConsulta(Evento), FinTransmisiónConsulta(Evento-1))
39)         fin si
40)         FinTransmisiónConsulta(Evento) ← InicioTransmisiónConsulta(Evento) +
41)         DuraciónTransmisiónConsulta
42)         Evento ← Evento + 1
43)      fin hacer
44)      Fuente ← Fuente + 1
45)   fin hacer
46)
47)   // Ordenar todos los eventos para aplicar política FIFO en el servidor
48)   Ordenar los eventos con respecto al valor de FinTransmisiónConsulta
49)
50)   // Cálculo de los tiempos de inicio y fin de procesamiento de las consultas
51)   // Nota: El valor de FinTransmisiónConsulta es también el tiempo de arribo de
52)   // las consultas al servidor
53)   para Fuente←1 hasta NúmeroFuentes hacer
54)      EventoInicial ← 10,000*(Fuente-1) + 1
55)      EventoFinal ← 10,000*Fuente
56)      para Evento←EventoInicial hasta EventoFinal hacer
57)         si Evento=EventoInicial entonces
58)            InicioProcesamientoConsulta(Evento) ← FinTransmisiónConsulta(Evento)
59)         // El tiempo de inicio del procesamiento de la consulta es el mayor del tiempo
60)         // de arribo de la consulta y el fin del procesamiento de la consulta anterior
61)         si no
62)            InicioProcesamientoConsulta(Evento) ←
               max(InicioProcesamientoConsulta(Evento),
               FinProcesamientoConsulta(Evento-1))
63)         fin si
64)         FinProcesamientoConsulta(Evento) ← InicioProcesamientoConsulta(Evento)
            + DuraciónProcesamientoConsulta(Evento)
65)         Evento ← Evento + 1
66)      fin hacer
67)      Fuente ← Fuente + 1
68)   fin hacer
69)
70)   // Ordenar todos los eventos por fuente para separar los eventos de cada fuente
```

```
71)   Ordenar los eventos con respecto a la fuente
72)   Para todos los eventos de cada fuente, ordenarlos con respecto al valor de
      FinProcesamientoConsulta
73)
74)   // Cálculo de los tiempos de inicio y fin de transmisión de las respuestas
75)   // Nota: El valor de FinProcesamientoConsulta es también el tiempo de "arribo" de
76)   // las respuestas al servidor
77)   para Fuente←1 hasta NúmeroFuentes hacer
78)      EventoInicial ← 10,000*(Fuente-1) + 1
79)      EventoFinal ← 10,000*Fuente
80)      para Evento←EventoInicial hasta EventoFinal hacer
81)         DuraciónTransmisiónRespuesta ←
            TamañoRespuesta(Evento)/VelocidadTransmisión
82)         si Evento=EventoInicial entonces
83)            InicioTransmisiónRespuesta(Evento) ← FinProcesamientoConsulta(Evento)
84)         // El tiempo de inicio de transmisión de la respuesta es el mayor del fin del
85)         // procesamiento de la consulta y el fin de transmisión de la respuesta anterior
86)         si no
87)            InicioTransmisiónRespuesta(Evento) ←
               max(FinProcesamientoConsulta(Evento),
               FinTransmisiónRespuesta(Evento-1))
88)         fin si
89)         FinTransmisiónRespuesta(Evento) ← InicioTransmisiónRespuesta(Evento) +
90)         DuraciónTransmisiónRespuesta
91)         Evento ← Evento + 1
92)      fin hacer
93)      Fuente ← Fuente + 1
94)   fin hacer
```

En la tabla 3.3 se presentan los diferentes valores utilizados para la simulación, la cual se realizó para 10,000 eventos; mientras que las tablas 3.4, 3.5 y 3.6 muestran los valores obtenidos de la simulación correspondiente al retardo de transmisión de las consultas, el retardo de procesamiento de las consultas, y el retardo de transmisión de las respuestas. En estas últimas tablas, se muestra un valor de T_i para cada valor de λ_i, para cada i = 1, 2, ..., 6.

tabla 3.3: Valores utilizados para las simulaciones

Prueba	No. de Fuentes	Velocidad de Transmisión	Tiempo de Procesamiento	Intensidades de Arribo*					
				λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
1	1	10	10	0.100	0.300	0.500	0.700	0.900	1.100
2	2	20	5	0.100	0.500	0.900	1.300	1.700	2.100
3	5	50	2	0.100	1.100	2.100	3.100	4.100	5.100
4	1	10	10	0.100	0.600	1.100	1.600	2.100	2.600
5	1	10	10	0.050	0.060	0.070	0.080	0.090	1.000
6	2	20	5	0.050	0.060	0.070	0.080	0.090	1.000

7	5	50	2	0.050	0.060	0.070	0.080	0.090	1.000
8	1	10	2	0.050	0.063	0.075	0.088	0.100	0.113
9	2	20	5	0.050	0.060	0.070	0.080	0.090	0.100
10	5	10	10	0.050	0.063	0.075	0.088	0.100	0.113
* Intensidad de arribo = (Tiempo de interarribo)$^{-1}$									

tabla 3.4: Retardos medios observados de la transmisión de las consultas

Prueba	Tiempos de Retardo Observados(seg)					
	T_1	T_2	T_3	T_4	T_5	T_6
1	0.813	0.972	1.203	1.576	2.304	4.753
2	0.391	0.463	0.564	0.724	1.020	1.819
3	0.155	0.182	0.221	0.286	0.406	0.717
4	0.813	1.365	4.753	731.7	1448.2	1893.8

tabla 3.5. Retardos medios observados del procesamiento de las consultas

Prueba	Tiempos de Retardo Observados(seg)					
	T_1	T_2	T_3	T_4	T_5	T_6
5	18.907	23.345	30.575	44.479	84.041	395.022
6	9.061	11.126	14.535	21.253	39.426	197.042
7	3.735	4.632	6.129	9.089	17.865	154.044

tabla 3.6. Retardos medios observados de la transmisión de las respuestas

Prueba	Tiempos de Retardo Observados(seg)					
	T_1	T_2	T_3	T_4	T_5	T_6
8	5.184	5.445	5.765	6.146	6.607	6.965
9	2.309	2.360	2.412	2.469	2.526	2.593
10	0.876	0.886	0.894	0.904	1.651	104.335

3.6. Ajuste de curva a tiempos de retardo

Una vez que se concluyó la simulación usando los datos de la tabla 3.3, se procedió a realizar el ajuste de curva a los tiempos de retardo (tablas 3.4, 3.5 y 3.6). Para tal efecto se utilizó el método de mínimos cuadrados, el cual se explica a continuación.

Para ilustrar el proceso de ajuste, considérense los tiempos de retardo de transmisión de consultas de la prueba 1 (tabla 3.4), los cuales se muestran en la figura 3.3. En esta figura en el eje de las ordenadas se muestran los valores T_1, T_2, T_3, T_4, T_5 y T_6, correspondientes respectivamente a λ_1, λ_2, λ_3, λ_4, λ_5 y λ_6 (tabla 3.3), cuyos valores se encuentran en el eje de las abscisas.

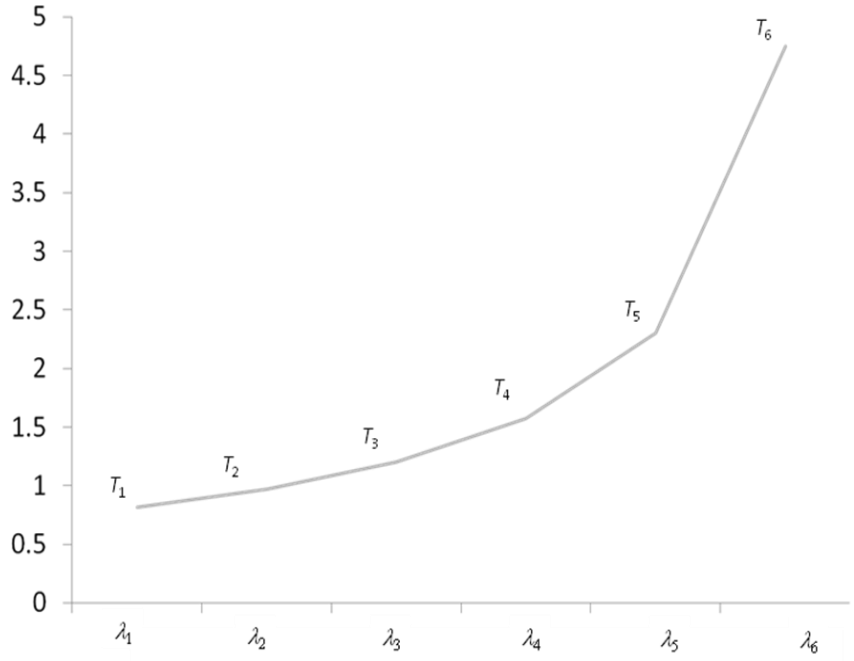

figura 3.3 Comportamiento del retardo de transmisión de consultas para la prueba 1

En la Sección 3.2 se mencionó que el tiempo de retardo T se puede calcular aproximadamente mediante la siguiente expresión (ver 3.1):

$$T = \frac{1}{\mu C - \lambda}$$

Para el proceso de ajuste se utilizará una expresión similar:

$$T' = \frac{P_2}{P_3 - \lambda} \tag{3.2}$$

en donde P_2 y P_3 son dos constantes (con valores adecuadamente seleccionados).

En tales circunstancias, para una intensidad de arribo dada λ_i la expresión (3.2) dará un valor T'_i para el retardo de transmisión, el cual deberá ser aproximadamente igual al valor observado T_i (para i = 1, 2, ..., 6). El proceso de ajuste consiste en encontrar los valores de P_2 y P_3 tales que los valores calculados T'_i sean lo más aproximados posible a los valores observados T_i (para i = 1, 2, ..., 6).

Para tal efecto, se usará la conocida técnica de minimización del error cuadrático. En este caso el error cuadrático está dado por la siguiente fórmula:

$$EC = \sum_{i=1}^{6} (T_i - T_i')^2 \tag{3.3}$$

Por otra parte, de la expresión (3.2) se tiene que el valor T'_i está dado por

$$T_i' = \frac{P_2}{P_3 - \lambda_i}, \quad i = 1,2,\ldots,6 \tag{3.4}$$

así que sustituyendo (3.4) en (3.3) se obtiene,

$$EC = \sum_{i=1}^{6} \left(T_i - \frac{P_2}{P_3 - \lambda_i} \right)^2 \tag{3.5}$$

Entonces, formalmente el proceso de ajuste consiste en encontrar los valores de P_2 y P_3 que minimicen el error cuadrático (3.5).

En lo que resta de esta sección se presenta una descripción general del proceso de ajuste de curva a los tiempos de retardo observados (es decir, los valores de las T_i obtenidos en la simulación).

Si en lugar de seis muestras del proceso de simulación (λ_1, T_1), (λ_2, T_2), ..., (λ_6, T_6), se tuvieran sólo dos (λ_a, T_a) y (λ_b, T_b), entonces los valores que minimizan el error cuadrático, se podrían obtener fácilmente haciendo $T'_a = T_a$ y $T'_b = T_b$, lo cual según (3.2) conduce a las siguientes expresiones:

$$T_a = T'_a = \frac{P_2}{P_3 - \lambda_a} \tag{3.6}$$

$$T_b = T'_b = \frac{P_2}{P_3 - \lambda_b} \tag{3.7}$$

De las dos expresiones anteriores mediante manipulaciones algebraicas se obtienen las siguientes dos expresiones:

$$T_a P_3 - T_a \lambda_a = P_2 \tag{3.8}$$

$$T_b P_3 - T_b \lambda_b = P_2 \tag{3.9}$$

Al restarle a (3.8) la expresión (3.9) se obtiene

$$T_aP_3 - T_a\lambda_a - T_bP_3 + T_b\lambda_b = 0$$

de la cual se puede despejar P_3, cuya expresión es

$$P_3 = \frac{T_b\lambda_b - T_a\lambda_a}{T_b - T_a} \tag{3.10}$$

finalmente, de (3.8) se obtiene la siguiente expresión para P_2:

$$P_2 = T_a(P_3 - \lambda_a). \tag{3.11}$$

De lo anterior resulta que el proceso de minimización del error cuadrático (3.5) se podría iniciar usando las dos expresiones anteriores (en donde λ_a, T_a, λ_b y T_b se sustituirían por los valores de λ_1, T_1, λ_6 y T_6 respectivamente), con lo cual se obtendrían unos valores preliminares para P_2 y P_3. Por ejemplo, para la prueba 1 (ver tablas 3.3 y 3.4) los valores que se obtendrían con este proceso son P_2 = 0.980759 y P_3 = 1.306345, con los cuales se obtendría un error cuadrático de 0.013919. Con estos valores se obtendría un ajuste de curva como el que se muestra en la figura 3.4. Desafortunadamente los valores obtenidos de esta manera en general no minimizan el error cuadrático, ya que solamente se han tomado en cuenta dos puntos (λ_1=0.1, T_1=0.813) y (λ_6=1.1, T_6=4.753) en lugar de todos los seis puntos. De hecho, los valores de P_2 y P_3 que minimizan el error cuadrático son P_2 = 0.940405 y P_3 = 1.298283, con lo cual se obtiene un error cuadrático de 0.005706. Sin embargo, los valores de P_2 y P_3 obtenidos con las expresiones (3.10) y (3.11) se pueden usar como punto de partida para encontrar el mínimo del error cuadrático.

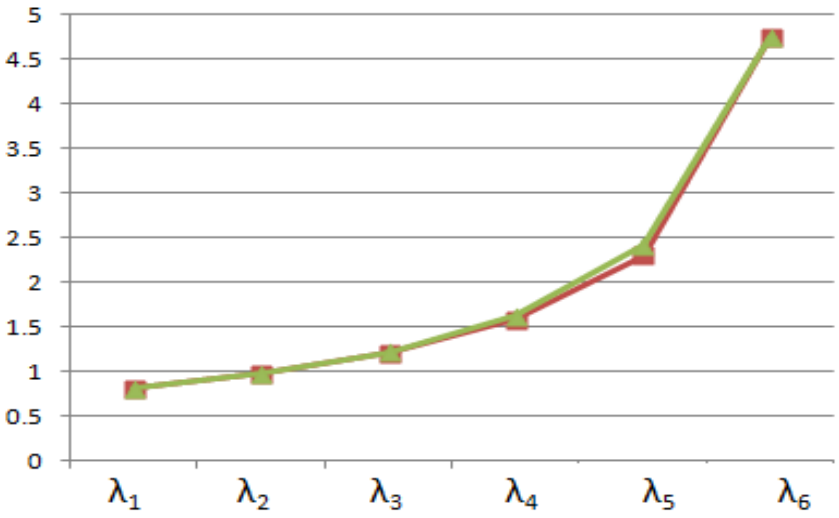

figura 3.4 Ajuste de curva para los resultados de la prueba 1 con valores preliminares de P_2 y P_3

Para encontrar los valores de P_2 y P_3 que minimizan el error cuadrático se puede usar un proceso de refinamientos sucesivos. Dicho proceso se efectúa de la siguiente manera: primero se fija el valor del parámetro P_2 y enseguida se aumenta o reduce el valor de P_3 de tal manera que se reduzca el error cuadrático *EC*, después se mantiene fijo el nuevo valor de P_3 y entonces se aumenta o reduce el valor de P_2 de tal forma que se reduzca *EC*; finalmente, este proceso se repite hasta que ya no sea posible reducir el valor de *EC*.

En cada paso del proceso descrito anteriormente es necesario determinar para cada parámetro la magnitud Δ del cambio y el sentido de éste, es decir, aumento o reducción. Para tal efecto, se pueden utilizar la primera y segunda derivadas de *EC* con respecto a P_2 y P_3, cuyas fórmulas se presentan a continuación:

$$\frac{\partial EC}{\partial P_2} = -2\sum_i \frac{1}{P_3 - x_i}\left(T_i - \frac{P_2}{P_3 - x_i}\right) \tag{3.12}$$

$$\frac{\partial^2 EC}{\partial P_2^2} = 2\sum_i \left(\frac{1}{P_3 - x_i}\right)^2 \tag{3.13}$$

$$\frac{\partial EC}{\partial P_3} = 2\sum_i \frac{P_2}{(P_3 - x_i)^2}\left(T_i - \frac{P_2}{P_3 - x_i}\right) \tag{3.14}$$

$$\frac{\partial^2 EC}{\partial P_3^2} = 2\sum_i \frac{3P_2^2 - 2P_2T_i(P_3 - \lambda_i)}{(P_3 - \lambda_i)^4} \tag{3.15}$$

Para ilustrar el uso de estas fórmulas, supóngase que para ciertos valores de P_2 y P_3 el valor de $\partial EC/\partial P_2$ fuera positivo, esto implicaría que el valor de *EC* aumentaría al incrementar P_2 y al contrario, *EC* disminuiría al reducirse P_2. Por lo tanto, si se desea disminuir el valor de *EC* es necesario reducir P_2 cuando $\partial EC/\partial P_2 > 0$, y al revés, es necesario aumentar P_2 cuando $\partial EC/\partial P_2 < 0$. De manera similar para disminuir *EC* se requiere reducir P_3 cuando $\partial EC/\partial P_3 > 0$, y al revés, se necesita aumentar P_3 cuando $\partial EC/\partial P_3 < 0$.

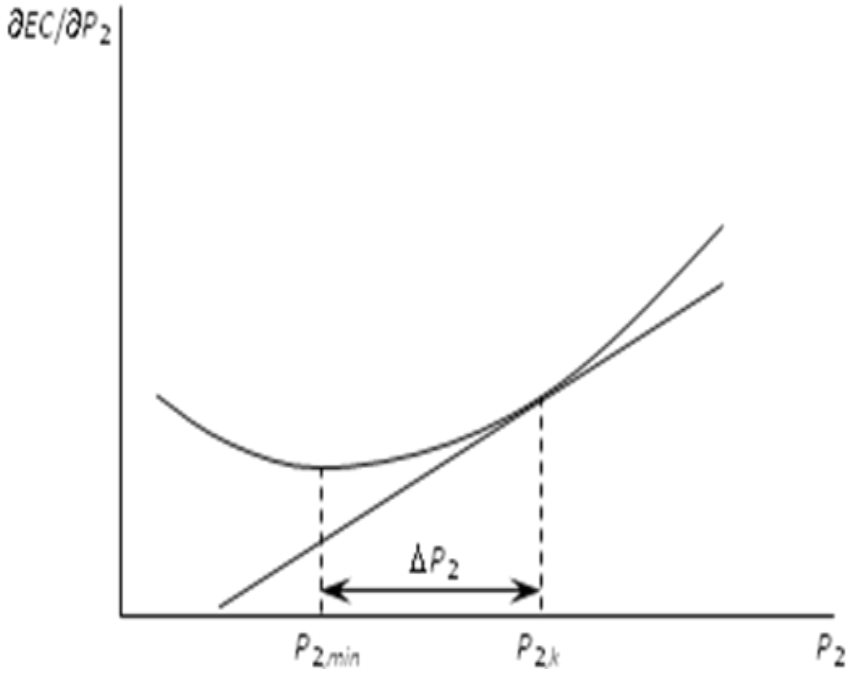

figura 3.5. Gráfica de $\partial EC/\partial P_2$ como función de P_2

Para determinar una magnitud ΔP_2 adecuada para aumentar o disminuir el parámetro P_2, considérese la figura 3.5, la cual muestra la gráfica de $\partial EC/\partial P_2$ como una función de P_2. Ahora, suponiendo que el valor actual de P_2 fuera $P_{2,k}$, habría que reducir P_2 hasta el punto $P_{2,min}$ que es donde $\partial EC/\partial P_2 = 0$ (es decir, donde se encuentra el valor mínimo de *EC*). Como muestra la figura 3.5, la siguiente expresión da un valor aproximado del cambio que se necesita para llegar al valor de P_2 que minimiza *EC*:

$$\Delta P_2 = \frac{\partial EC / \partial P_2}{\partial^2 EC / \partial P_2^2} \tag{3.16}$$

Naturalmente, como ΔP_2 no es exactamente el cambio que se necesita para llegar al valor de P_2 que minimiza *EC*, puede ser necesario repetir varias veces el proceso de cálculo de ΔP_2.

De manera similar, para determinar la magnitud del cambio de P_3 se puede usar la siguiente expresión:

$$\Delta P_3 = \frac{\partial EC / \partial P_3}{\partial^2 EC / \partial P_3^2} \tag{3.16}$$

Finalmente, el proceso de refinamientos sucesivos para encontrar los valores óptimos de P_2 y P_3 queda detallado en el siguiente pseudocódigo:

Seudocódigo para encontrar P_2 y P_3

```
1)  inicio
2)  Asignar un valor inicial a P2
3)  Asignar un valor inicial a P3
4)  // DirecciónCambio: =1 si el parámetro (P2 o P3) debe aumentarse para reducir EC
5)  // DirecciónCambio: =–1 si el parámetro (P2 o P3) debe reducirse para reducir EC
6)  para k←1 hasta 100 hacer
7)     // Determinación de un nuevo valor de P3 que reduzca el error cuadrático EC
8)     Calcular ∂EC/∂P3
9)     si ∂EC/∂P3 > 0 entonces
10)       DirecciónCambio ← –1
11)    si no
12)       DirecciónCambio ← 1
13)    fin si
14)    Calcular ΔP3
15)    P3 ← P3 + DirecciónCambio * ΔP3
16)    // Determinación de un nuevo valor de P2 que reduzca el error cuadrático EC
17)    Calcular ∂EC/∂P2
18)    si ∂EC/∂P2 > 0 entonces
19)       DirecciónCambio ← –1
20)    si no
21)       DirecciónCambio ← 1
22)    fin si
23)    Calcular ΔP2
24)    P2 ← P2 + DirecciónCambio * ΔP2
25)  fin hacer
```

Desafortunadamente el error cuadrático es una función de P_2 y P_3 que en general tiene varios óptimos locales, así que el algoritmo de optimización anterior permite encontrar sólo uno de éstos. Por lo tanto, para encontrar el menor de los óptimos locales, es necesario ejecutar este algoritmo varias veces, cada una con diferentes valores iniciales para P_2 y P_3. La tabla 3.7 muestra los valores de λ y T usados para calcular los valores iniciales de P_2 y P_3 en cada una de las búsquedas de óptimos locales.

tabla 3.7. Valores de λ y T usados para calcular los valores iniciales de P_2 y P_3

	λ_a	T_a	λ_b	T_b
1ra búsqueda de óptimo local	λ_1	T_1	λ_5	T_5
2da búsqueda de óptimo local	λ_1	T_1	λ_6	T_6
3ra búsqueda de óptimo local	λ_2	T_2	λ_5	T_5
4ta búsqueda de óptimo local	λ_2	T_2	λ_6	T_6
5ta búsqueda de óptimo local	λ_3	T_3	λ_5	T_5
6ta búsqueda de óptimo local	λ_3	T_3	λ_6	T_6

3.7. Gráficas del comportamiento del tiempo medio de respuesta

Para cada una de las pruebas 1, 2 y 3 (tabla 3.4) se ejecutó el algoritmo de ajuste de curva descrito en la Sección 3.6. Los tiempos de retardo estimados con la función ajustada se muestran en la tabla 3.8. De manera similar, para cada una de las pruebas de las tablas 3.5 y 3.6 se obtuvieron los resultados que se encuentran en las tablas 3.9 y 3.10.

tabla 3.8. Retardos medios estimados de la transmisión de las consultas

Prueba	Parámetros		Tiempos de Retardo Estimados(seg)					
	P_2	P_3	T'_1	T'_2	T'_3	T'_4	T'_5	T'_6
1	0.9406	1.2983	0.785	0.942	1.178	1.572	2.361	4.742
2	0.9674	2.6331	0.382	0.454	0.558	0.726	1.037	1.815
3	0.9573	6.4368	0.151	0.179	0.221	0.287	0.410	0.716

tabla 3.9. Retardos medios estimados del procesamiento de las consultas

Prueba	Parámetros		Tiempos de Retardo Estimados(seg)					
	P_2	P_3	T'_1	T'_2	T'_3	T'_4	T'_5	T'_6
5	1.0321	0.1026	19.618	24.222	31.649	45.646	81.838	395.154
6	0.4835	0.1025	9.218	11.389	14.898	21.534	38.826	197.105
7	0.1952	0.1013	3.808	4.731	6.244	9.180	17.327	154.087

tabla 3.10. Retardos medios estimados de la transmisión de las respuestas

Prueba	Parámetros		Tiempos de Retardo Estimados(seg)					
	P_2	P_3	T'_1	T'_2	T'_3	T'_4	T'_5	T'_6
8	1.0186	0.2444	5.240	5.600	6.014	6.493	7.055	7.724
9	1.0815	0.2564	5.240	5.506	5.802	6.131	6.499	6.915
10	0.0330	0.1128	0.525	0.656	0.873	1.303	2.575	104.421

La figura 3.6 muestra las gráficas de los valores del retardo medio de transmisión de las consultas correspondientes a la prueba 1. Los valores observados (T_i) se indican con puntos cuadrados (tabla 3.4), mientras que los valores estimados (T'_i) se indican con puntos triangulares (tabla 3.8). Como muestra la figura, resulta difícil distinguir las dos gráficas ya que están casi empalmadas. De manera similar las figuras 3.7 y 3.8 muestran las gráficas correspondientes a las pruebas 2 y 3.

Las figuras 3.9, 3.10 y 3.11 muestran las gráficas de los valores del retardo medio de procesamiento de las consultas correspondientes respectivamente a las pruebas 5, 6 y 7 (tablas 3.5 y 3.9). Por último, las figuras 3.12, 3.13 y 3.14 muestran las gráficas de los valores del retardo medio de transmisión de las respuestas correspondientes respectivamente a las pruebas 8, 9 y 10 (tablas 3.6 y 3.10).

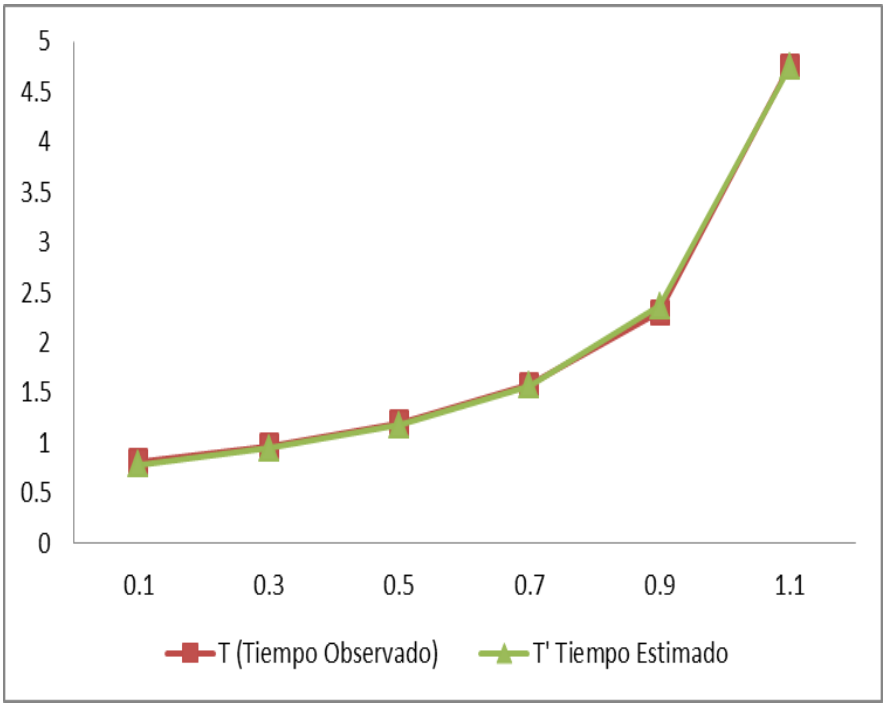

figura 3.6. Gráficas del retardo de transmisión de consultas (observado y estimado) para la prueba 1

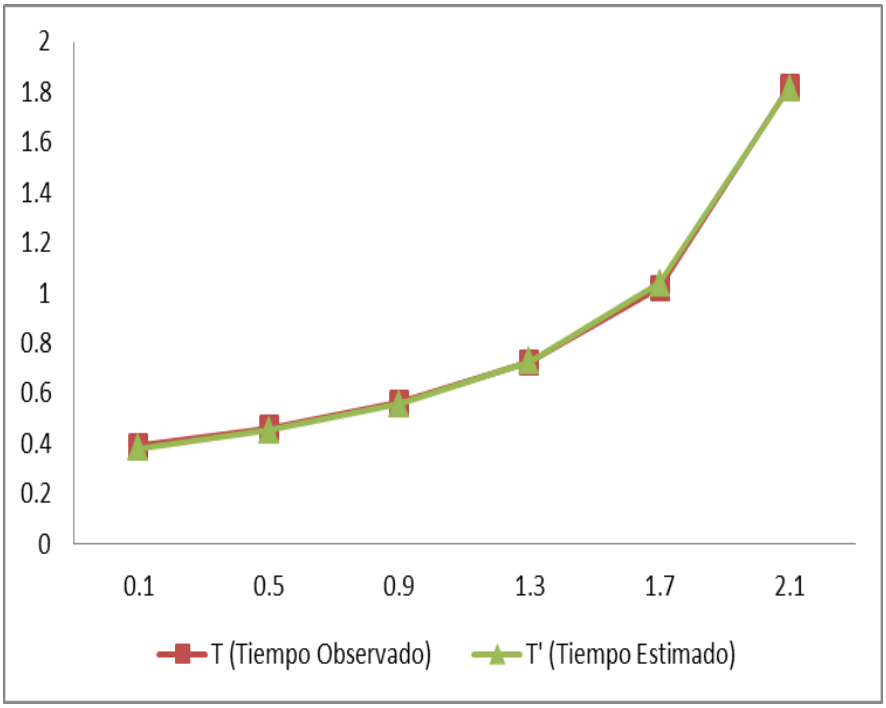

figura 3.7. Gráficas del retardo de transmisión de consultas (observado y estimado) para la prueba 2

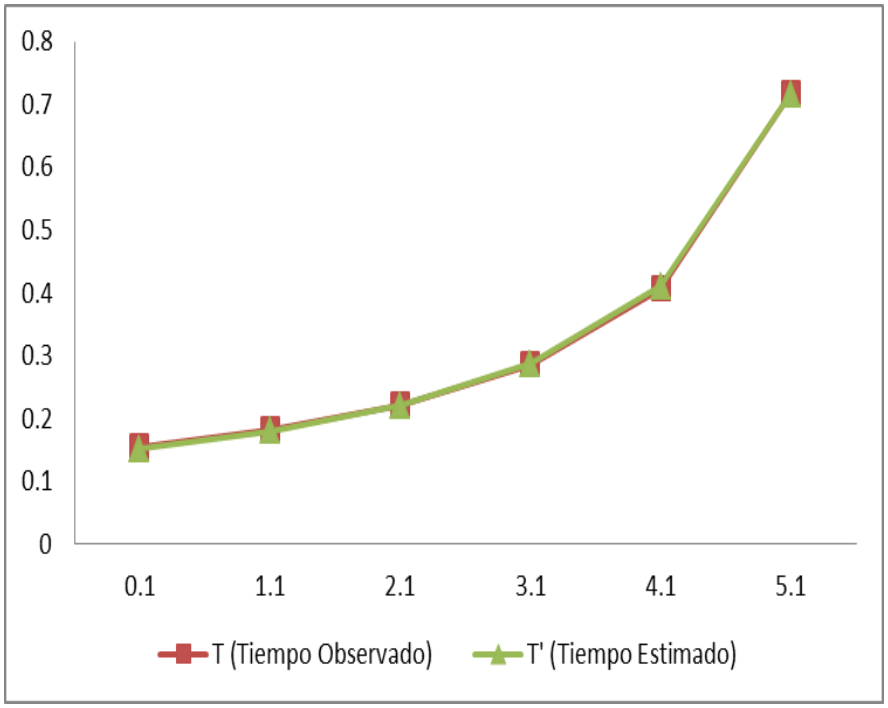

figura 3.8. Gráficas del retardo de transmisión de consultas (observado y estimado) para la prueba 3

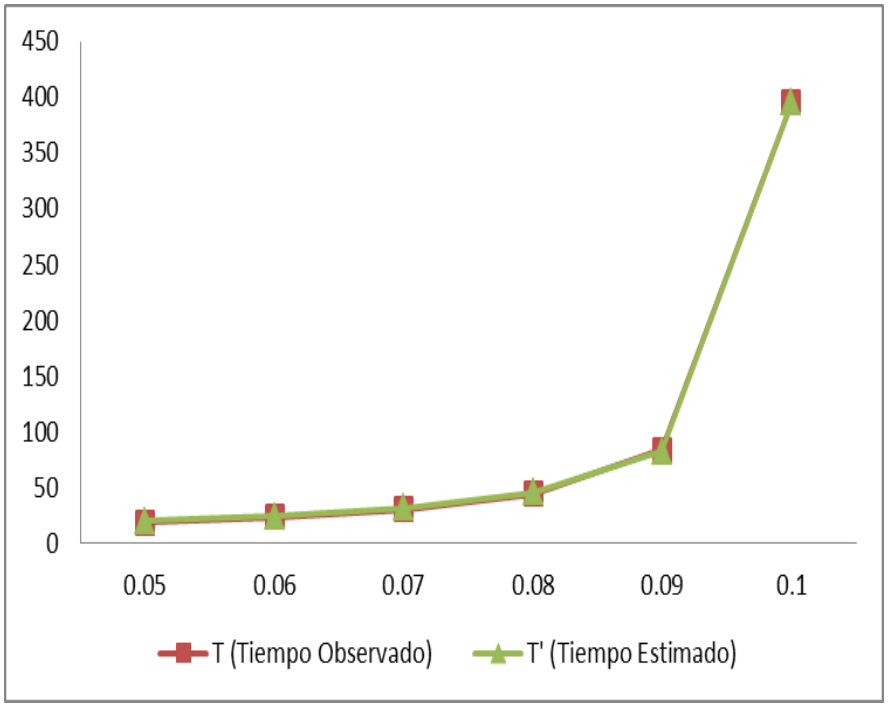

figura 3.9. Gráficas del retardo de procesamiento de consultas (observado y estimado) para la prueba 5

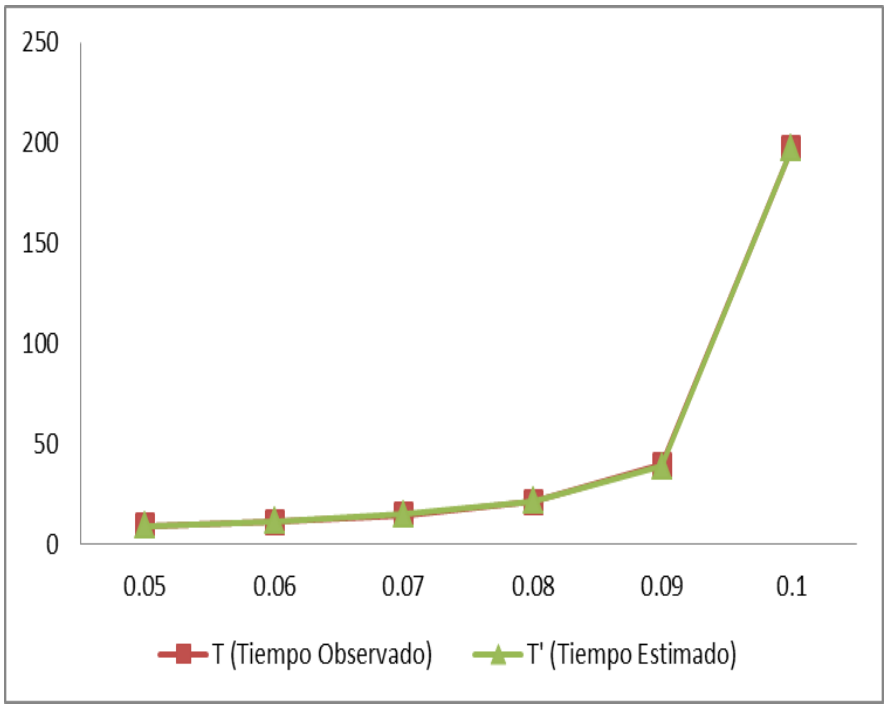

Figura 3.10. Gráficas del retardo de procesamiento de consultas (observado y estimado) para la prueba 6

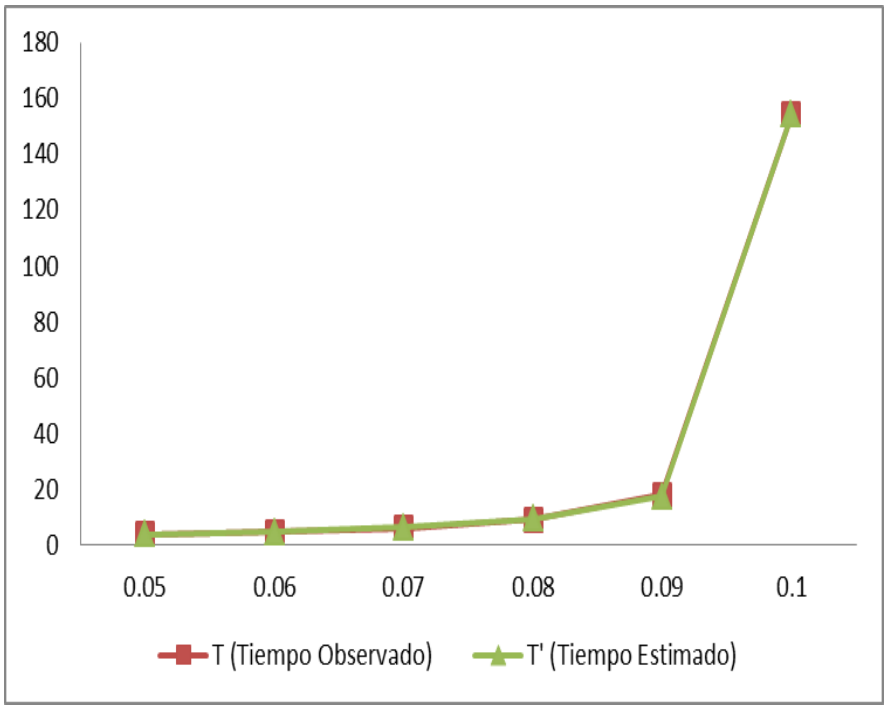

figura 3.11. Gráficas del retardo de procesamiento de consultas (observado y estimado) para la prueba 7

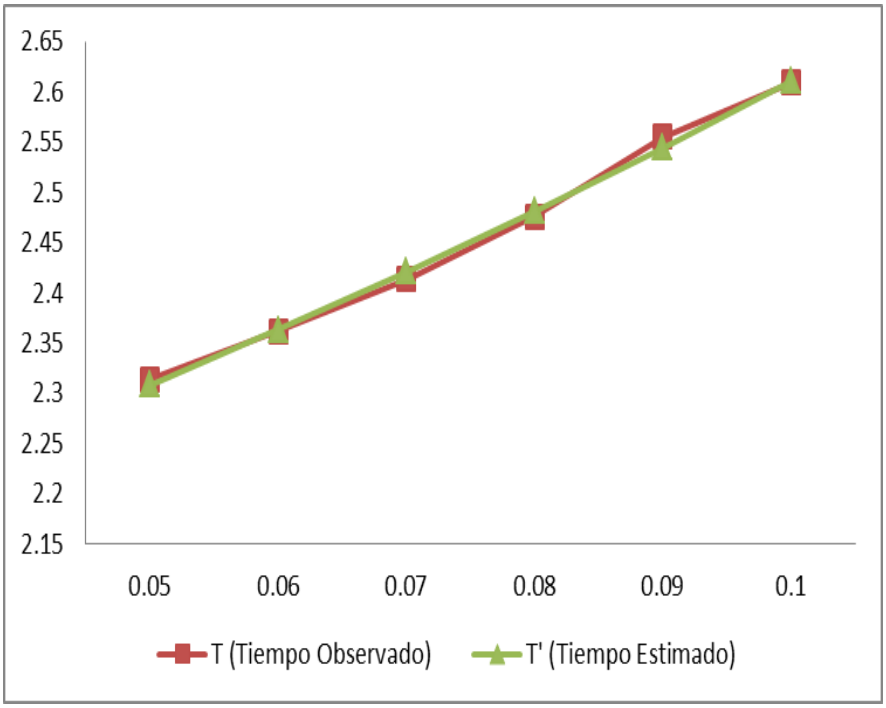

figura 3.12. Gráficas del retardo de transmisión de respuestas (observado y estimado) para la prueba 8

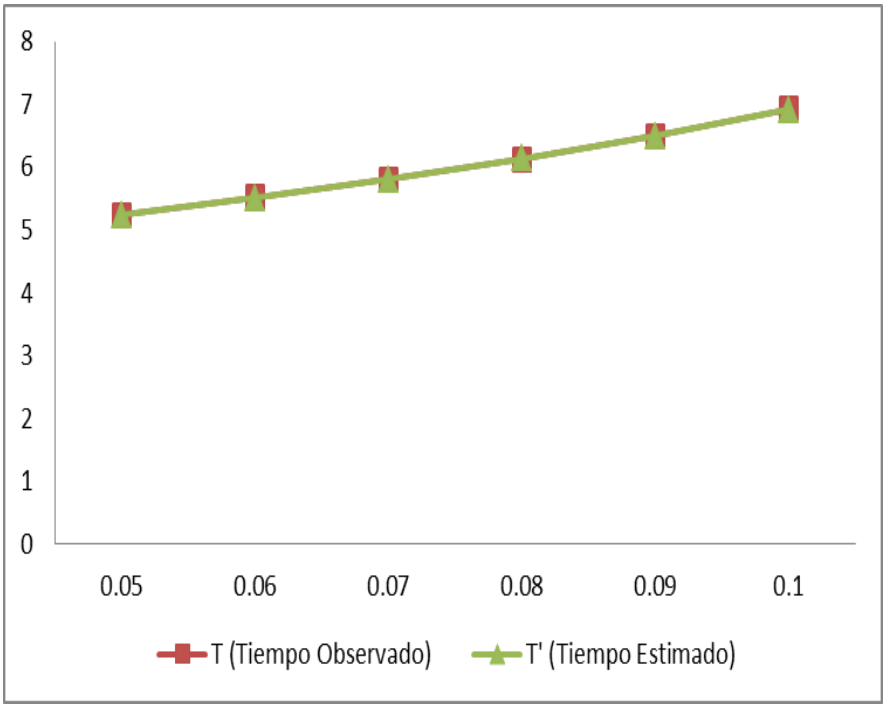

figura 3.13. Gráficas del retardo de transmisión de respuestas (observado y estimado) para la prueba 9

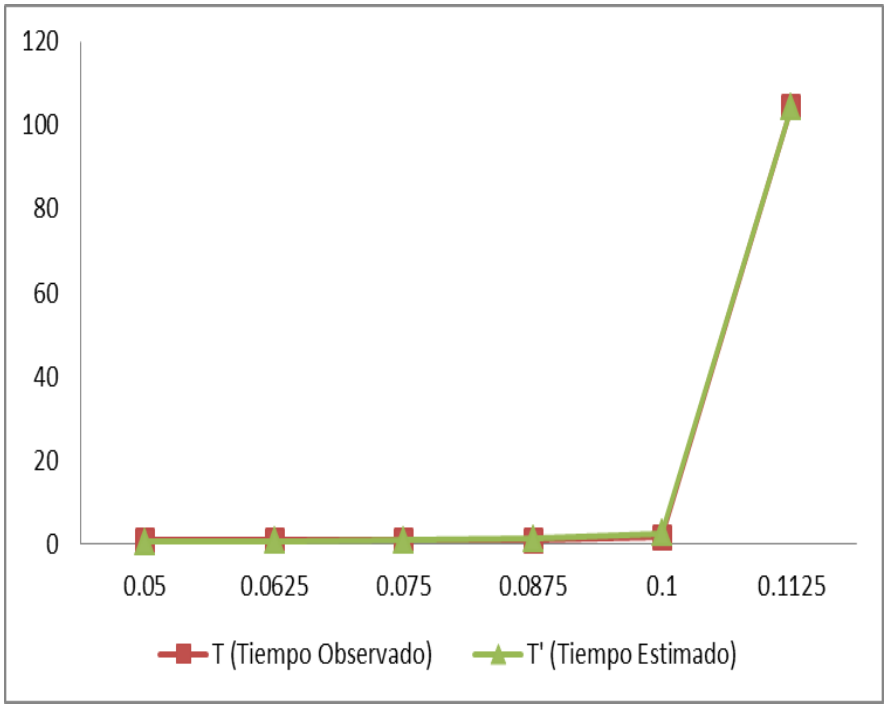

figura 3.14. Gráficas del retardo de transmisión de respuestas (observado y estimado) para la prueba 10

Las figuras anteriores (de la 3.6 a la 3.14) muestran que los tiempos estimados con la expresión (3.2) son bastante aproximados a los que se obtienen mediante simulación; por lo tanto, se justifica el uso de dicha expresión para modelar tiempos de retardo: retardo de transmisión de consultas, retardo de procesamiento de consultas, y retardo de transmisión de respuestas.

Antes de concluir este capítulo es conveniente aclarar que en las simulaciones como la descrita en la Sección 3.5 existe el problema de que los resultados obtenidos algunas veces discrepan de los resultados esperados de acuerdo con la teoría. Esto no significa que el programa de simulación tenga algún error o que la teoría esté equivocada.

Por ejemplo en el caso que nos ocupa, se busca simular un proceso que involucra tres subprocesos aleatorios: la transmisión de consultas de sus fuentes al servidor de bases de datos, el procesamiento de las consultas en el servidor, y la transmisión de las respuestas del servidor a las fuentes. En particular el proceso de transmisión de consultas de las fuentes al servidor (figura 3.2) es un fenómeno bien estudiado y se puede modelar perfectamente como una cola M/M/1 (como se mencionó en la Sección 3.2). En tales circunstancias, el retardo medio para este tipo de proceso está dado por la expresión (3.1), cuya gráfica se muestra en la figura 3.15.

Es importante destacar que dicha figura muestra que el retardo medio tiende a infinito cuando la intensidad de arribo λ se acerca a μC. Sin embargo, los resultados de una

simulación no exhiben este mismo comportamiento como lo ejemplifica la figura 3.16, la cual muestra los resultados correspondientes a la prueba 4 (tabla 3.4).

La explicación de tal discrepancia es la siguiente: la figura 3.15 muestra el comportamiento de una cola en estado estable, es decir, el estado que alcanza el sistema después de un tiempo arbitrariamente grande, lo cual implica la llegada de un número arbitrariamente grande de mensajes (idealmente un número infinito); desafortunadamente, en una computadora no es posible simular un proceso que involucre un número infinito de eventos; así que esta limitación de las simulaciones es la causa de la discrepancia entre los valores obtenidos por una simulación y los que predice la expresión (3.1). Esta discrepancia es insignificante cuando la intensidad de arribo es pequeña y aumenta sustancialmente cuando la intensidad de arribo crece, de tal manera que es imposible mediante una simulación obtener valores medios del retardo que tiendan a infinito.

Una vez explicado el comportamiento anómalo de este tipo de simulaciones, conviene aclarar que los valores de las pruebas presentadas en este capítulo fueron escogidos de tal manera que se evitara la región donde la discrepancia se vuelve significativa; es decir, valores de intensidad de arribo en donde el valor medio del retardo empiece a aumentar para tender a infinito.

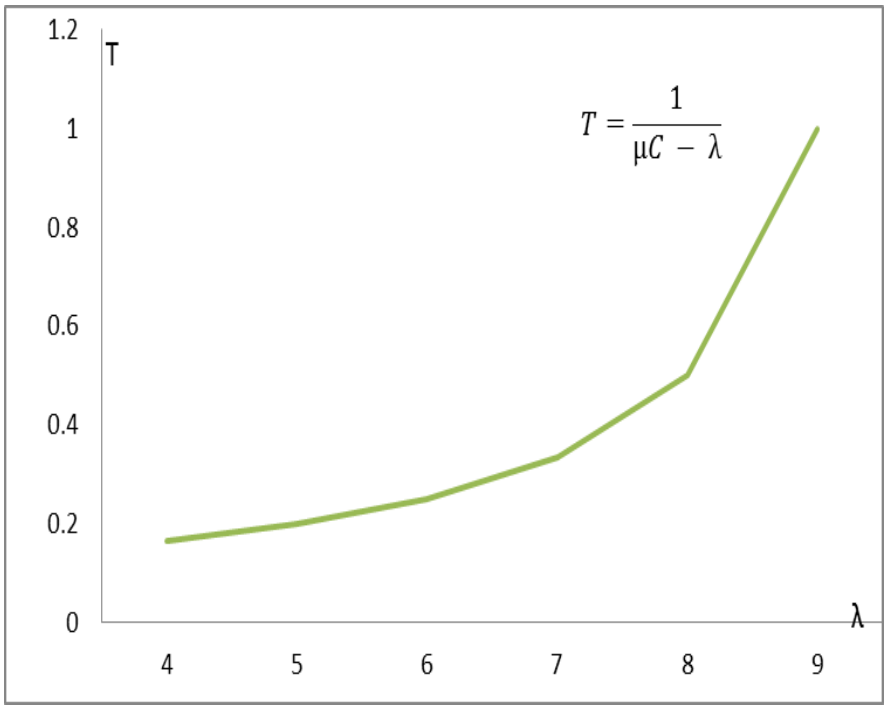

Figura 3.15. Gráfica del retardo de una cola M/M/1

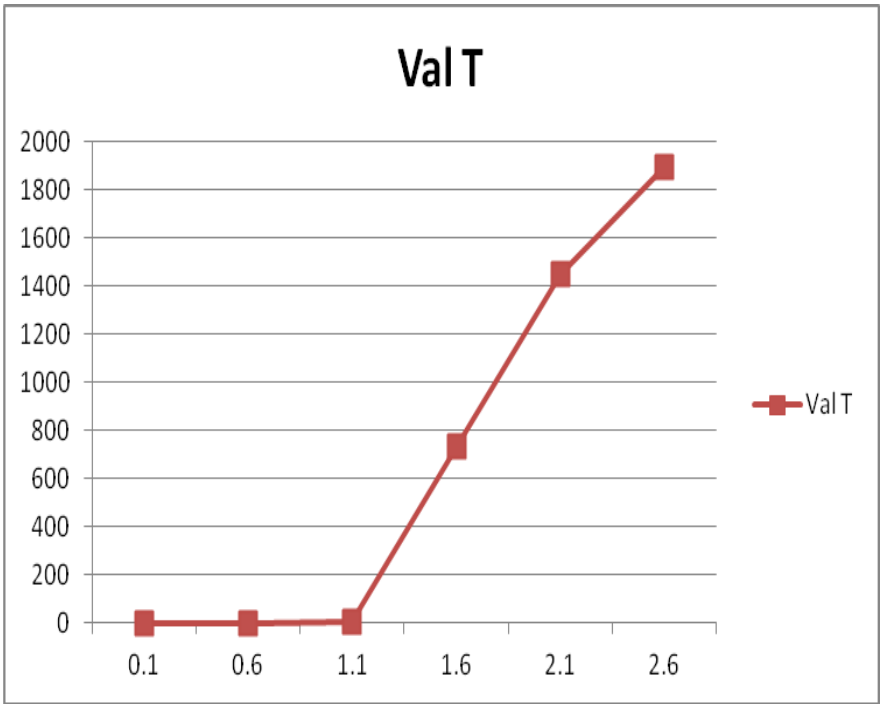

Figura 3.16. Gráfica del retardo de transmisión de consultas (observado) para la prueba 4

Capítulo 4

Modelos de fragmentación y ubicación

Capítulo 4. Modelos de fragmentación y ubicación

En la primera sección de este capítulo se toma como referencia uno de los modelos de fragmentación y ubicación (modelo FURD), el cual aborda desde una perspectiva nueva la fragmentación y ubicación de los atributos en una sola fase, ya que el resto de los trabajos anteriores lo hacen en dos fases. Es oportuno mencionar que el modelo FURD fue desarrollado en un proyecto doctoral documentado en [Pérez, 99].

Este modelo optimiza su desempeño al ubicar los datos en los sitios más adecuados, esto puede ser implementado como un módulo de una herramienta computarizada de ayuda al diseño de bases de datos distribuidas.

También modela la migración de atributos a otros sitios, la cual tiene como propósito el adecuarse a los cambios en los patrones de transacciones del período anterior. Este cambio, da lugar a una disminución en los costos de comunicación.

En la segunda sección de este capítulo se presenta el modelo propuesto en esta tesis, la cual trata de minimizar el tiempo de respuesta (FVU-TR). Como se verá en dicha sección, la diferencia entre los modelos FURD y FVU-TR se encuentra principalmente en la función objetivo; en tanto que las tres restricciones del modelo FVU-TR son iguales a las del modelo FURD.

Deliberadamente se buscó que las restricciones del nuevo modelo FVU-TR fueran iguales a las del modelo FURD, con el propósito de comparar los resultados experimentales obtenidos con ambos modelos (presentadas en el Capítulo 6), con el fin de determinar qué tanta diferencia existe en los resultados de ambos modelos.

4.1. Modelo FURD

El modelo FURD es útil al cuantificar el beneficio de reubicar los datos acorde con los cambios en su explotación y cardinalidad. Este modelo hace las siguientes contribuciones al problema del diseño de la distribución:

1. Aborda el diseño de la distribución desde una perspectiva completamente nueva; esto es, se considera que los atributos de una relación al ubicarse en un nodo dado, formarán el conjunto de elementos del fragmento vertical de la relación para dicho nodo. Esto difiere del enfoque tradicional que considera dos fases: fragmentación y ubicación.

2. Se formula una función objetivo sencilla que permite ubicar o reubicar los atributos de las relaciones en los nodos de la red de manera optimizada, minimizando los costos de comunicación. El modelo permite su solución con paquetes comerciales de software de programación entera.

3. Se muestra que el proceso de diseño de la distribución puede ser automatizado, con los siguientes beneficios: a) la función puede ser implementada como un módulo de los manejadores de bases de datos distribuidas, permitiéndoles optimizar su desempeño al ubicar los datos en los sitios más adecuados, y b) el modelo puede ser implementado como un módulo de una herramienta computarizada de ayuda al diseño de bases de datos distribuidas.

En el modelo FURD se supone que existen un conjunto de fragmentos $F = \{f_1, f_2, ..., f_n\}$ de una BDD y una red formada por sitios $S = \{s_1, s_2, ..., s_m\}$ y en éstos un conjunto de aplicaciones ejecutándose $Q = \{q_1, q_2, ..., q_q\}$, las cuales generan consultas a ser procesadas en los servidores de BDs que residen en los sitios. En estas circunstancias, el problema consiste en encontrar la distribución "óptima" de *F* en *S*, tal que se minimicen los costos de transmisión y procesamiento de las consultas.

4.1.1. Función objetivo

La función objetivo está definida por la expresión (4.1). Esta función modela los costos de transmisión, almacenamiento y acceso, usando cuatro términos: el primer término modela los costos de transmisión de los datos necesarios para satisfacer cada una de las consultas que se generan en los sitios; el segundo término modela los costos incurridos durante el procesamiento de una consulta que necesita acceder a varios fragmentos que están almacenados en diferentes sitios; el tercer término modela los costos incurridos por el almacenamiento de fragmentos en los sitios de la red; y el cuarto término modela los costos de transmisión incurridos por la reubicación de atributos de un sitio a otro. Esta reubicación tiene como propósito el adecuarse a los cambios en los patrones de transacciones del período anterior.

$$\min z = \sum_k \sum_i f_{ki} \sum_\ell \sum_j u_{k\ell} l_{k\ell} c_{ij} x_{\ell j} + c_1 \sum_i \sum_k \sum_j f_{ki} y_{kj} + c_2 \sum_j w_j + \sum_\ell \sum_i \sum_j a_{\ell i} c_{ij} d_\ell x_{\ell j} \qquad (4.1)$$

Donde los índices *k*, *i*, *j* y ℓ denotan consultas, sitios de consultas, sitios de atributos y atributos, respectivamente. Las letras *I*, *J*, *K*, *L* denotan el número total de consultas, total de sitios de consultas, total de sitios de atributos y total de atributos, respectivamente.

$i = 1, 2, ..., I$
$j = 1, 2, ..., J$
$k = 1, 2, ..., K$
$\ell = 1, 2, ..., L$

$x_{\ell j}$= variable de decisión, $x_{\ell j} = 1$ si el atributo ℓ se almacena en el sitio *j*, y $x_{\ell i} = 0$ en caso contrario;

f_{ki}= frecuencia de emisión de la consulta *k* desde el sitio *i*, para un periodo de tiempo dado;

$u_{k\ell}$= parámetro de uso, $u_{k\ell}$ = 1 si la consulta k usa el atributo ℓ, en caso contrario $u_{k\ell}$ = 0;

c_{ij}= costo de transmitir un paquete de información entre el sitio i y el sitio j;

c_1= costo por acceder a cada fragmento para satisfacer una consulta; p.e., el costo de procesar una consulta que accede a dos o más fragmentos;

c_2= costo por ubicar un fragmento en un sitio; p.e., el costo derivado de tener una copia de la llave primaria en cada fragmento;

w_j= variable de decisión, w_j = 1 si existe uno o más atributos en un sitio j, y w_j = 0 en caso contrario;

y_{kj}= variable de decisión,
y_{kj} = 1 si la consulta k accede a uno o más atributos localizados en el sitio j, y
y_{ki} = 0 en caso contrario;

$a_{\ell i}$= variable que indica la ubicación previa de un atributo,
$a_{\ell i}$ = 1 si el atributo ℓ está actualmente localizado en el sitio i,
en caso contrario $a_{\ell i}$ = 0;

d = número de paquetes de comunicación requeridos para mover el atributo ℓ a otro sitio si es necesario

$$d_\ell = (p_\ell * CA)/PA$$

donde
p_ℓ = tamaño en bytes del atributo ℓ.
CA = cardinalidad de la relación; también expresada como $|A|$.
PA = tamaño del paquete de comunicación en bytes;

$l_{k\ell}$= número de paquetes de comunicación necesarios para transportar el atributo ℓ requerido por la consulta k.

$$l_{k\ell} = (p_\ell\, s_k)/PA$$

donde
s_k = selectividad de la consulta k (número de tuplas retornadas cuando se ejecuta la consulta k).

4.1.2. Restricciones intrínsecas del problema

En virtud de que no se considera la replicación de atributos, se tiene un grupo de restricciones que específica que cada atributo se ubicará solamente en un sitio (4.2). Adicionalmente, cada atributo se debe ubicar en un sitio que al menos ejecute una consulta que acceda al atributo (4.3) y sin exceder la capacidad del nodo (4.6). Estas restricciones se expresan de la siguiente manera:

$\sum_j x_{\ell j} = 1$ $\forall \ell$ — Cada atributo debe almacenarse solamente en un sitio. (4.2)

$x_{\ell i} \le \sum_k u_{k\ell}\varphi_{ki}$ $\forall \ell, i$ — Donde (4.3)

$$\varphi_{ki} = \begin{cases} 1, & \text{si } f_{ki} > 0 \\ 0, & \text{si } f_{ki} = 0 \end{cases}$$

cada atributo debe ser ubicado en un sitio *i* que ejecute al menos una consulta que involucre al atributo.

$t\ w_j - \sum_\ell x_{\ell j} \ge 0$ $\forall j$ — esta restricción fuerza al valor de w_j a 1 cuando cualquier x_j es igual a 1, e induce a w_j a 0 en cualquier otro caso, donde (4.4)

t = número de atributos.

$t\ y_{kj} - \sum_\ell u_{k\ell}\ x_{\ell j} \ge 0$ $\forall k, j$ — Esta restricción fuerza y_{kj} igual a 1 cuando cualquier $u_{k\ell}x_j$ es igual a 1, e induce y_{kj} a 0 en cualquier otro caso. (4.5)

$CA\sum_\ell x_{\ell j}\, p_\ell \le CS_j$ $\forall j$ — La suma de los tamaños de los fragmentos asignados al sitio *j* no debe exceder la capacidad de éste, donde (4.6)

CS_j = capacidad del sitio *j*.

4.2. Modelo de fragmentación y ubicación considerando tiempo de respuesta

La hipótesis principal de este trabajo, es que el enfoque tradicional de diseño de BDDs no conduce a una sola solución óptima del problema, ya que el método que se ha seguido sólo consiste en adecuaciones y extensiones a las metodologías de las bases de datos centralizadas. Pensar de esta manera hace difícil el manejo de la complejidad del problema y por lo tanto dificulta la obtención de soluciones que optimizan su solución.

Este trabajo plantea un nuevo enfoque de solución al diseño de BDDs, partiendo de la idea de incorporar a la función de optimización algunas variables que involucren el tiempo de respuesta para obtener un modelo de la operación de BDDs basado en la teoría de colas.

Además, hay una similitud entre ese modelo y el utilizado en las redes de computadoras, por lo que se pueden aprovechar los diferentes métodos de redes para

simplificar el modelo, y seguir el mismo proceso para el diseño de la BDD, con la finalidad de obtener un modelo simplificado.

En consecuencia, el diseño de la BDD debe orientarse a la minimización del tiempo de respuesta en el sistema de información para el que sirve.

Debe considerarse no sólo el diseño de la base de datos, sino también el medio sobre el que funcionará, que en gran medida determina su eficiencia y que se debe considerar invariable.

Se formulará el problema de la fragmentación vertical y su ubicación utilizando el modelo simplificado. Además se adaptarán métodos de optimización para resolver el problema de ubicación y fragmentación.

El producto de este análisis es una fragmentación de las tablas y su asignación a las diferentes computadoras, lo cual representa la solución de problemas de complejidad NP-duros.

El algoritmo que modelará la ejecución de transacciones debe tomar en cuenta todos estos factores y escoge, de los fragmentos cuyas entidades son referidas y que estén localizados local o remotamente, a aquéllos que al ser utilizados minimicen el tiempo de respuesta.

Se han ideado una gran cantidad de modelos para resolver este problema, pero no todos tienen por objetivo minimizar el tiempo de respuesta.

Actualmente, una línea activa de investigación es la extensión de heurísticas a los procesos de búsqueda local con el fin de acelerar su convergencia y/o mejorar su factor de aproximación, entre este número de propuestas se encuentran las búsquedas locales no obvias, la búsqueda tabú, el recocido simulado y los algoritmos genéticos. Estos métodos presentan la particularidad de no ser alternativas robustas, en el sentido de que con ligeros cambios en sus parámetros, tales heurísticas encuentran soluciones muy diferentes.

En las siguientes páginas se describe el modelo para optimizar el diseño de bases de datos distribuidas considerando tiempo de respuesta, al cual se denominará FVU-TR. En este punto conviene aclarar que la derivación de la función objetivo para este modelo sigue un procedimiento similar al que se presenta en [Kleinrock, 76] para obtener el retardo promedio de los mensajes en una red de comunicaciones para computadoras.

Específicamente en el modelo FVU-TR (a semejanza del modelo FURD) se supone que existen un conjunto de fragmentos $F = \{f_1, f_2, ..., f_n\}$ de una BDD y una red formada por sitios $S = \{s_1, s_2, ..., s_m\}$ y en éstos un conjunto de aplicaciones ejecutándose $Q = \{q_1, q_2, ..., q_q\}$, las cuales generan consultas a ser procesadas en los servidores de BDs que residen en los sitios. Sin embargo en el modelo FVU-TR, el problema consiste en encontrar la distribución "óptima" de F en S, tal que se minimicen los tiempos de transmisión y procesamiento de las consultas.

4.2.1. Derivación de la función objetivo

La función objetivo está dada por la siguiente expresión:

$$z = T_{TC} + T_{PC} + T_{TR} \tag{4.7}$$

donde

T_{TC} = tiempo medio de transmisión de consultas,
T_{PC} = tiempo medio de procesamiento de consultas,
T_{TR} = tiempo medio de transmisión de respuestas.

A continuación, se obtendrá una expresión para z en términos de las variables de decisión, para lo cual, se introducen las siguientes definiciones:

C_j = capacidad de procesamiento del servidor j (expresada en consultas/seg.),

C_{ij} = velocidad de transmisión de la línea de comunicación del nodo i al nodo j (expresada en bits/seg.),

$1/\mu_Q$ = longitud media de las consultas (expresada en bits/consulta),

$1/\mu_R$ = longitud media de las respuestas (expresada en bits/respuesta),

f_{ki} = intensidad de llegada de la consulta k al nodo fuente i (expresada en las consultas/seg.),

f = suma de las intensidades de llegada de todas las consultas = $\sum_{ki} f_{ki}$,

γ_{ki} = intensidad de emisión de la consulta k desde el nodo fuente i (= f_{ki}, $2f_{ki}$, $3f_{ki}$, etc.),

γ = suma de las intensidades de emisión de todas las consultas = $\sum_{ki} \gamma_{ki}$.

El tiempo medio de procesamiento de consultas en los servidores se define de la siguiente manera:

$$T_{PC} = \frac{1}{f} \sum_k \sum_i f_{ki} T_{ki}$$

donde T_{ki} representa el tiempo medio de procesamiento de las consultas k emitidas desde el nodo i; pero, dada la dificultad para calcular T_{PC} de manera exacta, se usará la siguiente aproximación, dado que f_{ki} y γ_{ki} son aproximadamente iguales:

$$T^*_{PC} = \frac{1}{\gamma} \sum_k \sum_i \gamma_{ki} T_{ki}$$

Una manera fácil de calcular T^*_{PC} es usando el resultado de Little [Kleinrock, 07], entonces

$$\gamma T^*_{PC} = n = \sum_j n_j$$

donde n representa el número medio de consultas que se encuentran en los servidores de bases de datos (ya sea en espera o en procesamiento), y n_j representa el número medio de consultas que se encuentran en el servidor j. Aplicando nuevamente el resultado de Little a n_j, se obtiene la siguiente expresión:

$$\gamma T_{PC}^{*} = \sum_{j} \lambda_{*j} T_{PCj} \tag{4.8}$$

donde λ_{*j} representa la intensidad de llegada de consultas al servidor j y T_{PCj} representa el retardo medio de procesamiento de las consultas en el servidor j.

Por otro lado, considerando que el procesamiento de consultas en los servidores se puede modelar como una cola M/M/1, entonces el retardo medio de procesamiento de las consultas en el servidor j está dado por (ver expresión (3.1)

$$T_{PCj} = \frac{1}{C_j - \lambda_{*j}} \tag{4.9}$$

Después, sustituyendo la expresión (4.9) para T_{PCj} en (4.8), y despejando T^*_{PC} se obtiene

$$\begin{aligned} T_{PC}^{*} &= \frac{1}{\gamma} \sum_{j} \frac{\lambda_{*j}}{C_j - \lambda_{*j}} \\ &= \frac{1}{\gamma} \sum_{j} \frac{1}{C_j / \lambda_{*j} - 1} \end{aligned} \tag{4.10}$$

Por otra parte, la expresión para λ_{*j} en términos de las intensidades de llegada de las consultas a los nodos fuente y de la ubicación de los atributos en los servidores puede calcularse de la siguiente manera:

$$\lambda_{*j} = \sum_{k} \sum_{i} f_{ki} y_{jk} \tag{4.11}$$

donde y_{jk} es una variable de decisión dependiente (y_{jk} = 1 si en el sitio j se almacenan uno o más atributos usados por la consulta k, y y_{jk} = 0 en caso contrario).

Además, nótese que la suma de las intensidades de emisión de todas las consultas emitidas desde los nodos fuente (γ) es igual a la suma de las intensidades de llegada de todas las consultas que llegan a los servidores (λ_{*j}), y considerando que λ_{*j} está dada por la expresión (4.11); por lo tanto

$$\gamma = \sum_{j} \lambda_{*j} = \sum_{j} \sum_{k} \sum_{i} f_{ki} y_{jk} \tag{4.12}$$

Por último, sustituyendo (4.11) y (4.12) en (4.10) se obtiene la siguiente expresión para T^*_{PC} en función de la ubicación (y_{jk}) de los atributos en los servidores:

$$T^*_{PC} = \frac{1}{\sum_j \sum_k \sum_i f_{ki} y_{jk}} \sum_j \frac{1}{\frac{C_j}{\sum_k \sum_i f_{ki} y_{jk}} - 1} \tag{4.13}$$

Con respecto al retardo medio de transmisión de consultas, éste se define de la siguiente manera:

$$T_{TC} = \frac{1}{f} \sum_k \sum_i f_{ki} T_{ki}$$

donde T_{ki} representa el retardo medio de transmisión de las consultas k que llegan al nodo fuente i.

A semejanza del proceso para T_{PC}, se usará la siguiente aproximación de T_{TC}:

$$T^*_{TC} = \frac{1}{\gamma} \sum_k \sum_i \gamma_{ki} T_{ki}$$

Usando el resultado de Little, se obtiene la siguiente expresión

$$\gamma T^*_{TC} = n = \sum_{ij} n_{ij}$$

donde n representa el número medio de consultas que se encuentran en las líneas de transmisión (ya sea en espera o en transmisión), y n_{ij} representa el número medio de consultas que se encuentran en la línea del nodo fuente i al servidor j. Aplicando nuevamente el resultado de Little a n_{ij}, se obtiene la siguiente expresión:

$$\gamma T^*_{TC} = \sum_{ij} \lambda_{ij} T_{TCij} \tag{4.14}$$

donde λ_{ij} representa la intensidad de llegada de consultas a la línea de i a j, y T_{TCij} representa el retardo medio de transmisión de las consultas en la línea de i a j.

Por otro lado, considerando que la transmisión de consultas en las líneas se puede modelar como una cola M/M/1, entonces el retardo medio de transmisión de las consultas en la línea de i a j está dado por

$$T_{TCij} = \frac{1}{\mu_C C_{ij} - \lambda_{ij}} \tag{4.15}$$

Después, sustituyendo la expresión (4.15) para T_{TCij} en (4.14), y despejando T^*_{TC} se obtiene

$$T_{TC}^{*} = \frac{1}{\gamma}\sum_{ij}\frac{\lambda_{ij}}{\mu_{C}C_{ij} - \lambda_{ij}}$$

$$= \frac{1}{\gamma}\sum_{ij}\frac{1}{\mu_{C}C_{ij} / \lambda_{ij} - 1} \tag{4.16}$$

Por otra parte, la expresión para λ_{ij} en términos de las intensidades de llegada de las consultas a los nodos fuente y de la ubicación de los atributos en los servidores puede calcularse de la siguiente manera:

$$\lambda_{ij} = \sum_{k} f_{ki} y_{jk} \tag{4.17}$$

donde y_{jk} es una variable de decisión dependiente (y_{jk} = 1 si en el sitio *j* se almacenan uno o más atributos usados por la consulta *k*, y y_{jk} = 0 en caso contrario).

Por último, sustituyendo (4.12) y (4.17) en (4.16) se obtiene la siguiente expresión para T^{*}_{TC} en función de la ubicación (y_{jk}) de los atributos en los servidores:

$$T_{TC}^{*} = \frac{1}{\sum_{j}\sum_{k}\sum_{i} f_{ki} y_{jk}} \sum_{ij} \frac{1}{\frac{\mu_{C}C_{ij}}{\sum_{k} f_{ki} y_{jk}} - 1} \tag{4.18}$$

Siguiendo un proceso similar, se puede obtener la siguiente expresión para T^{*}_{TR}:

$$T_{TR}^{*} = \frac{1}{\sum_{j}\sum_{k}\sum_{i} f_{ki} y_{jk}} \sum_{ij} \frac{1}{\frac{\mu_{R}C_{ij}}{\sum_{k} f_{ki} y_{jk}} - 1} \tag{4.19}$$

Al final se puede obtener una expresión para la función objetivo en términos de la ubicación de los atributos, sustituyendo las expresiones (4.13), (4.18) y (4.19) en (4.7), lo cual da como resultado

$$\min z = \frac{1}{\sum_{j}\sum_{k}\sum_{i} f_{ki} y_{jk}} \left(\sum_{ij} \frac{1}{\frac{\mu_{C}C_{ij}}{\sum_{k} f_{ki} y_{jk}} - 1} + \sum_{j} \frac{1}{\frac{C_{j}}{\sum_{k}\sum_{i} f_{ki} y_{jk}} - 1} + \sum_{ij} \frac{1}{\frac{\mu_{R}C_{ij}}{\sum_{k} f_{ki} y_{jk}} - 1} \right) \tag{4.20}$$

Por último, es oportuno aclarar que, a diferencia del modelo FURD, esta función objetivo no considera la reubicación de fragmentos.

4.2.2. Restricciones intrínsecas del problema

Dado que no se considera la replicación de atributos, se tiene un grupo de restricciones que específica que cada atributo se ubicará solamente en un sitio (4.21). Adicionalmente, cada atributo se debe ubicar en un sitio que al menos ejecute una consulta que acceda al atributo (4.22). Estas restricciones se expresan de la siguiente manera:

$\sum_j x_{\ell j} = 1$ $\forall \ell$	Cada atributo debe almacenarse solamente en un sitio.	(4.21)
$x_{\ell i} \le \sum_k u_{k\ell}\varphi_{ki}$ $\forall \ell, i$	Donde $\varphi_{ki} = \begin{cases} 1, & \text{si } f_{ki} > 0 \\ 0, & \text{si } f_{ki} = 0 \end{cases}$ cada atributo ℓ debe ser ubicado en un sitio i que ejecute al menos una consulta que involucre al atributo.	(4.22)
$t\ y_{jk} - \sum_\ell u_{k\ell}\ x_{\ell j} \ge 0$ $\forall j, k$	Esta restricción fuerza al valor de y_{jk} a 1 cuando algún producto $u_{kl}\ x_{\ell j}$ es igual a 1 (es decir, algún atributo ℓ usado por la consulta k está ubicado en el servidor j), e induce y_{jk} a 0 en cualquier otro caso, donde t = número de atributos.	(4.23)

Como se había mencionado, conviene mencionar que las restricciones (4.21), (4.22) y (4.23) son iguales que las restricciones (4.2), (4.3) y (4.5) del modelo FURD.

.

Capítulo 5

Métodos de solución

Capítulo 5. Métodos de solución

En general existen métodos de solución iterativos, los cuales se muestran a continuación:

Método	En qué consiste	Tipo de solución
Exacto	Procedimiento que está sujeto a una serie de presupuestos, pero converge a una solución óptima.	Exacta
Heurístico	Procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un costo computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación. Se usa el calificativo heurístico en contraposición a exacto	Aproximada
Metaheurístico	El sufijo "meta" significa "más allá", a un nivel superior. Las metaheurísticas son estrategias para diseñar o mejorar los procedimientos heurísticos con miras a obtener un alto rendimiento.	Aproximada

La optimización combinatoria es una rama de la optimización en matemáticas aplicadas y en ciencias de la computación, relacionada a la investigación de operaciones, teoría de algoritmos y teoría de la complejidad computacional. También está relacionada con otros campos, como la inteligencia artificial e ingeniería de software. Los algoritmos de optimización combinatoria resuelven ejemplares de problemas que se creen ser difíciles en general, explorando el espacio de soluciones (usualmente grande) para estos ejemplares. Los algoritmos de optimización combinatoria logran esto reduciendo el tamaño efectivo del espacio, y explorando el espacio de búsqueda eficientemente.

Mediante el estudio de la teoría de la complejidad computacional es posible comprender la importancia de la optimización combinatoria. Los algoritmos de optimización combinatoria se relacionan comúnmente con problemas NP-duros. Dichos problemas en general no son resueltos eficientemente; sin embargo, varias aproximaciones de la teoría de la complejidad sugieren que ciertos ejemplares (ejemplares pequeños) de estos problemas pueden ser resueltos eficientemente. Dichos ejemplares a menudo tienen ramificaciones prácticas muy importantes.

Los métodos de solución utilizados en esta tesis son un método exacto, y dos métodos meta heurísticos. La razón es que los primeros han sido ampliamente estudiados y se consideran prácticos para ejemplares pequeños, mientras que los segundos son considerados prometedores para ejemplares de grandes magnitudes [Pérez, 02]. A

continuación se describe el funcionamiento y utilización de los siguientes tipos de métodos de solución que se utilizan para resolver problemas NP-duros: métodos exactos, métodos heurísticos y métodos metaheurísticos.

5.1. Tipos de métodos de solución

5.1.1. Métodos exactos

Los métodos exactos resuelven problemas que requieren usualmente de una búsqueda exhaustiva dentro del cómputo de todas las soluciones potenciales. Éstos usan una computadora para contestar preguntas como: ¿cuántos caminos hay para..?, ¿listar todas las posibles soluciones para..?, ¿hay un camino para..?

En los últimos años se vienen estudiando y aplicando métodos de optimización para tratar de dar solución a problemas de optimización combinatoria complejos en campos tan diversos como son la economía, el comercio, la ingeniería, la computación, la medicina o la industria. Para este tipo de problemas, no existen algoritmos exactos que proporcionen una solución en un tiempo polinomial.

Dentro de los métodos para afrontar este tipo de problemas, existen los métodos exactos y los aproximados. Los algoritmos exactos intentan encontrar una solución óptima y luego demostrar que es un óptimo global. Dentro de este tipo de algoritmos se encuentran procesos de retroceso (*backtracking*), procesos de ramificación y acotamiento (*branch-and-bound*), programación dinámica, etc. Dado el bajo rendimiento que proporcionan los algoritmos exactos, se desarrollaron algoritmos aproximados, que se pueden clasificar en dos tipos: constructivos (que generan soluciones desde cero añadiendo nuevas construcciones a la solución, paso a paso) y algoritmos de búsqueda local (que pretenden de forma iterativa encontrar soluciones mejores migrando a soluciones vecinas (esperando que sean mejores).

Es recomendable el uso de métodos exactos cuando el tamaño del problema es pequeño, ya que es posible, en general, encontrar la solución exacta del problema en un tiempo de computo razonable. La solución de problemas de tamaño grande, con métodos exactos, es impráctica por la gran cantidad de tiempo de cómputo y/o de memoria que requiere [Pérez, 99].

Existen dos técnicas para organizar búsquedas: la primera "retroceso" (*backtracking*), la cual trabaja tratando continuamente de extender una solución parcial. La segunda "tamiz o criba" (*sieves*), es el complemento lógico de "retroceso" en que se tratan de eliminar las no-soluciones en lugar de encontrar la solución.

Se debe tener en mente que estas dos técnicas son solamente técnicas generales, su aplicación resultará en algoritmos cuyos requerimientos en tiempo son prohibitivos, y en general la velocidad de las computadoras no es práctica para una búsqueda exhaustiva de más de 10^8 elementos.

Para que estas técnicas sean útiles, deben considerar solamente una estructura dentro de la cual se aproxima el problema. La estructura debe ser hecha a la medida, a menudo con gran ingeniosidad para cuadrar con el problema particular así que el algoritmo resultante sea de uso práctico.

Los algoritmos exactos se utilizan en aplicaciones del mundo real, y generalmente éstas son problemas de gran tamaño, lo cual implica que pueden requerir una enorme cantidad de tiempo. Luego entonces, una alternativa la constituyen los métodos heurísticos, los cuales en general obtienen buenas soluciones, pero no garantizan una solución óptima.

Actualmente los problemas exigen soluciones que deben ser encontradas rápida e precisamente, teniendo un bajo costo y encontrando la solución óptima. Para tanto, algoritmos que utilizan técnicas para reducir el espacio de búsqueda del problema, haciendo podas en las zonas aún no exploradas hasta que la solución óptima sea encontrada, son los más eficientes. En este sentido los algoritmos branch-and-bound tienen una gran aplicabilidad e eficiencia.

El método branch-and-bound (en español ramificación y acotamiento) se interpreta como un árbol de soluciones, donde cada rama nos lleva a una posible solución posterior a la actual. La característica de esta técnica con respecto a otras (y a la que debe su nombre) es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para acotar esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima, aborda la resolución de modelos de programación entera a través de la resolución de una secuencia de modelos de programación lineal que consistirán de nodos o subproblemas del problema entero. Si bien el procedimiento es extendible a un número mayor de variables, para efectos prácticos de su aplicación este algoritmo separa el problema en partes manejables:

- *divide (branch)*: se separa el conjunto factible con restricciones adicionales;
- *acota o poda (bound)*: muestra que un subproblema sólo puede entregar soluciones peores.

El método exacto utilizado en esta tesis es un método de búsqueda exhaustiva que se explica en la Sección 5.2.

5.1.2. Métodos heurísticos

Existen muchos problemas para los cuales no se conocen algoritmos que puedan encontrar la solución de forma eficiente: problemas NP-duros. La solución exacta puede requerir un orden factorial o exponencial: el problema de la explosión combinatoria. Se hace necesario utilizar algoritmos heurísticos: un algoritmo heurístico (o simplemente heurística) puede producir una buena solución (quizá la óptima) pero también puede que no produzca ninguna solución o dar una solución no muy buena. Normalmente, se basa en un conocimiento intuitivo del programador sobre un determinado problema. El estudio y

diseño de los algoritmos heurísticos (heurísticas) es uno de los paradigmas de la Inteligencia Artificial (IA).

El término heurístico está relacionado con la tarea de resolver problemas inteligentemente utilizando la información disponible. El término proviene de la palabra griega heuriskein que significa encontrar o descubrir, de la cual se deriva eureka, la famosa exclamación de Arquímedes al descubrir su principio. La palabra heurística aparece en más de una categoría gramatical. Cuando se usa como sustantivo, identifica el arte o la ciencia del descubrimiento, una disciplina susceptible de ser investigada formalmente. Cuando aparece como adjetivo, se refiere a cosas más concretas, como estrategias heurísticas, reglas heurísticas o silogismos y conclusiones heurísticas. Claro está que estos dos usos están íntimamente relacionados, ya que la heurística usualmente propone estrategias heurísticas que guían el descubrimiento.

La popularización del concepto se debe al matemático George Pólya, con su libro Cómo resolverlo (How to solve it). Habiendo estudiado tantas pruebas matemáticas desde su juventud, quería saber cómo los matemáticos llegan a ellas. El libro contiene la clase de recetas heurísticas que trataba de enseñar a sus alumnos de matemáticas. Cuatro ejemplos extraídos de él ilustran el concepto mejor que ninguna definición:

- Si no consigues entender un problema, dibuja un esquema.
- Si no encuentras la solución, haz como si ya la tuvieras y mira qué puedes deducir de ella (razonando a la inversa).
- Si el problema es abstracto, prueba a examinar un ejemplo concreto.
- Intenta abordar primero un problema más general (es la "paradoja del inventor": el propósito más ambicioso es el que tiene más posibilidades de éxito).

Los métodos heurísticos son estrategias generales de resolución y reglas de decisión utilizadas por los solucionadores de problemas, basadas en la experiencia previa con problemas similares. Estas estrategias indican las vías o posibles enfoques a seguir para alcanzar una solución.

Los procedimientos heurísticos son acciones que comportan un cierto grado de variabilidad y su ejecución no garantiza la consecución de un resultado óptimo como, por ejemplo, reducir el espacio de un problema complejo a la identificación de sus principales elementos.

Heurístico es "un procedimiento que ofrece la posibilidad de seleccionar estrategias que nos acercan a una solución".

Los métodos heurísticos pueden variar en el grado de generalidad. Algunos son muy generales y se pueden aplicar a una gran variedad de dominios, otros pueden ser más específicos y se limitan a un área particular del conocimiento. La mayoría de los programas de entrenamiento en solución de problemas enfatizan procesos heurísticos generales como los planteados por Polya (1965) o Hayes (1981).

Los métodos heurísticos específicos están relacionados con el conocimiento de un área en particular. Este incluye estructuras cognoscitivas más amplias para reconocer los problemas, algoritmos más complejos y una gran variedad de procesos heurísticos específicos.

A continuación se listan unas categorías amplias de métodos heurísticos, no excluyentes, donde se ubican los métodos heurísticos más conocidos:

- **Métodos de descomposición**.- El problema original se descompone en subproblemas más sencillos de resolver, teniendo en cuenta, aunque sea de manera general, que éstos pertenecen al mismo problema.

- **Métodos inductivos**.- La idea de estos métodos es generalizar de versiones pequeñas o más sencillas al caso completo.

- **Métodos de reducción**.- Consisten en identificar propiedades que se cumplen mayoritariamente en las buenas soluciones e introducirlas como restricciones del problema. El objeto es restringir el espacio de soluciones simplificando el problema. El riesgo obvio es dejar fuera las soluciones óptimas del problema original.

- **Métodos constructivos**.- Consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración. Estos métodos han sido muy utilizados en problemas clásicos como el del agente viajero.

- **Métodos de búsqueda local**.- A diferencia de los métodos anteriores, los procedimientos de búsqueda o mejora local, comienzan con una solución del problema y la mejoran progresivamente. El método finaliza cuando no existe ninguna solución accesible que mejore la anterior.

El método heurístico más intuitivo es el de la búsqueda exhaustiva: dado un problema, analizar todas las posibles soluciones y escoger la mejor, como puede intuirse también, este proceso será factible sólo para aquellos problemas simples que tengan un número pequeño de soluciones alternativas. Este se describe a detalle en la Sección 5.2.

5.1.3. Métodos metaheurísticos

En los últimos años han aparecido o han sido derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos una serie de métodos bajo el nombre de metaheurísticas con el propósito de obtener mejores resultados que los alcanzados por los métodos heurísticos tradicionales.

El sufijo “meta” significa “más allá”, “a un nivel superior”. Las metaheurísticas son estrategias para diseñar o mejorar los procedimientos heurísticos con miras a obtener un alto rendimiento. El término metaheurística fue introducido por Fred Glover en 1986 y a

partir de entonces han aparecido muchas propuestas de pautas o guías para diseñar mejores procedimientos de solución de problemas combinatorios.

Los profesores Osman y Kelly (1995) introducen la siguiente definición: Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos.

Las metaheurísticas son estrategias para diseñar y/o mejorar los procedimientos heurísticos, por lo tanto, el tipo de metaheurística está en función de qué tipo de heurística comprende y se puede hacer una clasificación como la que se lista a continuación:

- **Metaheurísticas constructivas**. Son las que van incorporando elementos a una estructura inicialmente vacía que representa la solución. La metaheurística constructiva da pautas para seleccionar los componentes que debe tener una buena solución. Un ejemplo de metaheurística constructiva es la GRASP (*Greedy Randomized Adaptative Search Procedures*), procedimientos de búsqueda basados en funciones voraces aleatorizadas que se adaptan. Esta metaheurística es una de las más populares debido a su sencillez y facilidad de implementación.

- **Metaheurísticas evolutivas**. Son métodos que van construyendo un conjunto de soluciones a diferencia de los otros métodos que sólo pasan de una solución a otra en cada iteración. El procedimiento consiste en generar, seleccionar, combinar y reemplazar un conjunto de soluciones. Ejemplos de metaheurísticas evolutivas: algoritmos genéticos o búsqueda dispersa (*scatter search*).

- **Metaheurísticas de búsqueda**. Son métodos que presuponen que existe una solución y realizan procedimientos de búsqueda, la diferencia con los métodos analíticos es que no necesariamente se encontrará la solución óptima. Uno de los riesgos al usar un algoritmo de búsqueda es el de alcanzar un óptimo local del que ya no sea posible salir. Las principales metaheuristicas de búsqueda global surgen de las tres formas principales de escapar de los óptimos locales: a) volver a comenzar la búsqueda desde otra solución inicial (*multistart*), b) modificar la estructura de entornos (metaheurística de entornos variables), y c) permitir movimientos de empeoramiento de la solución actual. Ejemplos de metaheurísticas de este tercer grupo: búsqueda tabú (*tabu search*) o recocido simulado (*simulated annealing*). Una variante de recocido simulado se presenta en la Sección 5.3, mientras que la metaheuristica búsqueda tabú se explica en la Sección 5.4.

5.2. Algoritmo de búsqueda exhaustiva

La búsqueda exhaustiva, búsqueda combinatoria, búsqueda por fuerza bruta o simplemente fuerza bruta, es una técnica trivial pero a menudo usada, que consiste en enumerar

sistemáticamente todos los posibles candidatos para la solución de un problema, con el fin de determinar si dicho candidato satisface la solución al mismo.

La búsqueda exhaustiva es sencilla de implementar y, siempre que exista, encuentra una solución. Sin embargo, su costo de ejecución es proporcional al número de soluciones candidatas, el cual puede ser exponencialmente proporcional al tamaño del problema. Por el contrario, la búsqueda exhaustiva se usa habitualmente cuando el número de soluciones candidatas no es elevado, o bien cuando éste puede reducirse previamente usando algún otro método heurístico.

Es un método utilizado también cuando es más importante una implementación sencilla que una mayor rapidez. Éste puede ser el caso en aplicaciones críticas donde cualquier error en el algoritmo puede acarrear serias consecuencias. La búsqueda exhaustiva no se debe confundir con *backtracking*, método que descarta un gran número de conjuntos de soluciones, sin enumerar explícitamente cada una de las mismas.

Algoritmo de búsqueda exhaustiva

1. Hágase $x_{opt} = \infty$ (en caso de un problema de minimización como FURD y FVU-TR).

2. Hágase $k = k+1$; generar una solución x_k, tal que sea factible. Nota: la solución x_k se genera de manera sistemática y determinista, de tal suerte que $x_k \neq x_{k-1}$, $x_k \neq x_{k-2}$, ..., $x_k \neq x_1$.

3. Si $f(x_k) < f(x_{opt})$, hacer $x_{opt} = x_k$.

4. Si existen más soluciones que no han sido generadas, ir al paso 2.

5.3. Algoritmo de aceptación por umbral

La aceptación de una solución que difiere de la actual en una cantidad inferior a un umbral predefinido, es un algoritmo propuesto por Dueck y Scheuer (1990) que, a diferencia del descenso de máximo gradiente, tolera opciones de peor calidad para eludir los óptimos locales (ver figura 5.1). Esta degradación estratégica de la solución en curso es compartida por la cristalización simulada, aunque en este caso el criterio de aceptación es probabilística, dependiendo de la diferencia de los valores de la función objetivo en la solución actual y la candidata, y de un parámetro de control denominado "temperatura", ver [Kirkpatrick, 83].

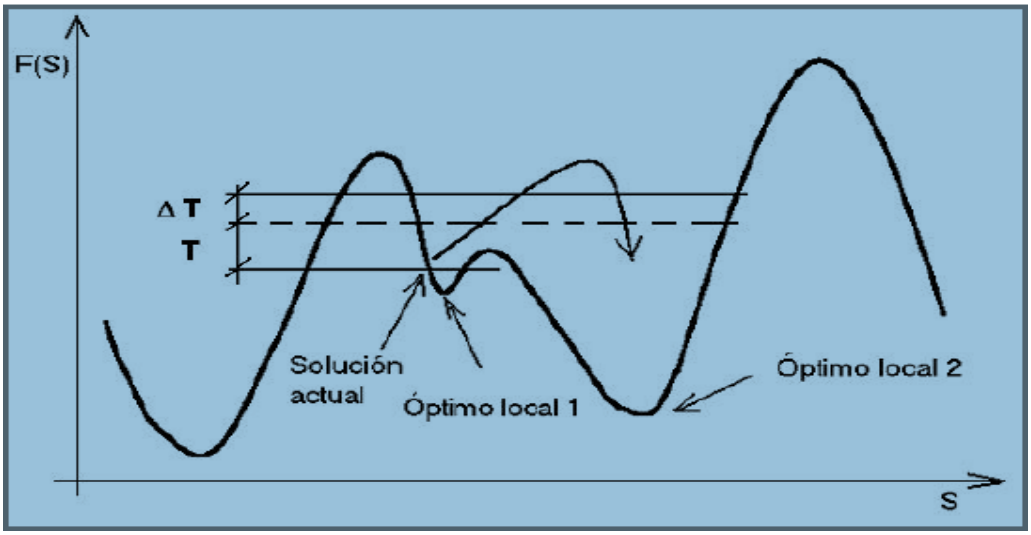

figura 5.1. Búsqueda local mejorada por el criterio de aceptación por umbral

El algoritmo de aceptación por umbral (AU, *threshold accepting* en inglés), está considerado como una técnica heurística de propósito general que permite solucionar problemas de optimización combinatoria, el cual obtiene soluciones de mejor calidad que recocido simulado y consume menor tiempo computacional. Es una variante de la metaheurística recocido simulado (RS, *simulated annealing* en inglés), consume menos tiempo computacional que el método original de RS, el cual se inspira en la simulación del proceso de recocido de metales.

La implementación de AU es sencilla, se requiere (al igual que en RS) entre otras cosas, la elección de un esquema de enfriamiento, es decir, la especificación de una temperatura inicial, la estrategia para el equilibrio térmico, el factor de disminución de la temperatura y las condiciones de paro del algoritmo. Sin embargo, proponer un programa de enfriamiento efectivo, que logre un balance entre calidad y tiempo de procesamiento es una tarea ardua. Requiere de un conocimiento amplio de la naturaleza del problema a resolver, y puede no ser obvio.

Tradicionalmente el esquema de enfriamiento se ha desarrollado mediante prueba y error, por lo que es necesario profundizar en el estudio del algoritmo para obtener sus parámetros de una manera analítica.

El proceso de recocido de materiales se aplica en la física del estado sólido, se aplica para que los metales formen una estructura o red cristalina, en la que los átomos tengan una disposición geométrica repetida periódicamente.

El proceso físico de recocido consiste en calentar un sólido a una temperatura elevada para reblandecerlo y eliminar las tensiones internas del metal o rigidez de la red, y posteriormente enfriarlo de manera lenta para que las partículas se coloquen por sí mismas

en su estado fundamental (bajo nivel de energía) y produzcan un material con gran ductilidad.

Al ser enfriado con lentitud, el material puede tener la suficiente energía para causar el crecimiento de grano. Los granos grandes tienen menor energía libre que los granos pequeños. En condiciones ideales, el estado menor de energía para un metal es aquél que esté formado por un solo cristal.

Dentro del proceso de recocido, cuando se cambia la temperatura hay un crecimiento máximo del grano, esto sucede porque la fuerza que impulsa el crecimiento de grano se encuentra en equilibrio con la rigidez de la red. La idea de trabajar un proceso de recocido es obtener estados de baja energía en un sólido; es decir, disponer de estructuras cristalinas uniformes libres de defectos o en su estado fundamental.

El algoritmo general de aceptación por umbral tiene un procedimiento, el cual se describe a través del algoritmo presentado en la figura 5.2, donde X representa al conjunto de todas las posibles soluciones, x es una solución factible actual tal que $x \in X$, y $N(x) \subset X$ es una vecindad asociada a x.

De esta manera, dada una solución actual x, se genera una solución factible vecina $y \in N(x)$ (línea 6); si el valor de la solución se encuentra dentro de un umbral delimitado por T, donde T es un parámetro de temperatura seleccionado apropiadamente (línea 7), entonces se acepta satisfactoriamente como la nueva solución actual (línea 8), de lo contrario la solución actual quedaría sin cambio (línea 10). El valor de T (línea 14) decrece cada vez que se alcance el equilibrio térmico y se reduce, multiplicando repetidamente por un factor de enfriamiento $\mu < 1$ mientras el sistema no se congele.

Algoritmo general de aceptación por umbral

```
1     Inicialización:  real T0, μ;  entero i, S
2     T0 = temperatura inicial (de tamaño adecuadamente grande);
3     μ = factor de enfriamiento, tal que 0 < μ < 1;
4     S = tamaño de lote;
5     T = T0
6     x = solución inicial factible
7     repetir
8        repetir
9           para i = 1 hasta S
10             y = solución vecina de x generada aleatoriamente
11             si z(y) - z(x) < T entonces
12                x = y
13             en caso contrario
14                la solución y es rechazada y se conserva x
15             fin si
16          fin para
```

17 **hasta** alcanzar el equilibrio térmico
18 $T = \mu T$
19 **hasta** alcanzar el congelamiento

Este algoritmo se implementó en un programa en C, tanto para el modelo FURD como para el modelo propuesto en esta tesis.

5.4. Algoritmo de búsqueda tabú

La búsqueda tabú (BT) es una metaheurística utilizada frecuentemente en problemas de optimización combinatoria. Se basa en un procedimiento iterativo de exploración guiada en el espacio de soluciones del problema a tratar. La forma actual en que se utiliza el método fue presentada en 1986 por Glover.

Por el momento no se conoce una demostración formal sobre su buen comportamiento. Sin embargo, se utiliza exitosamente en numerosas aplicaciones teóricas como coloreo de grafos, agente viajero, clique máximo y otras.

El algoritmo basa su funcionamiento en los conceptos de exploración inteligente y memoria adaptativa. Ambas ideas sugieren la noción de aprendizaje, en el sentido de que se utilizan los datos recolectados al realizar la tarea (memoria) para influir en la toma de decisiones (exploración inteligente) que se espera lleven a buenos resultados.

En la BT también puede hacerse una distinción entre la utilización de una memoria a corto o largo plazo. A corto plazo la memoria sirve para almacenar los movimientos inversos a los realizados recientemente a fin de evitar volver a visitar una solución ya analizada. La memoria a largo plazo suele estar relacionada con la frecuencia de ocurrencia de determinados atributos en las soluciones visitadas, y se le puede utilizar para privilegiar o prohibir movimientos que lleven a soluciones con atributos que fueron vistos frecuentemente en iteraciones pasadas.

El algoritmo se mueve hacia la mejor solución que pueda hallar en la vecindad de la solución actual que esté considerando. A veces puede realizar movimientos en los que la solución obtenida es peor en busca del mínimo global.

Esta sucesión de movimientos podría producir la aparición de ciclos en las soluciones a analizar (visitar alternadamente las mismas soluciones). Para evitar esta situación el algoritmo mantiene un conjunto de movimientos prohibidos (lista tabú) que no deben realizarse durante una determinada cantidad de iteraciones (tenencia tabú).

Dada una función $f(x)$ a ser optimizada, en un conjunto X, el proceso empieza de la misma manera que cualquier búsqueda local, procediendo iterativamente de una solución a otra hasta satisfacer un criterio dado de terminación.

Cada $x \in X$ tiene una vecindad asociada $N(x) \subseteq X$, y cada solución vecina $x' \in X$ se puede alcanzar desde x mediante una operación llamada movimiento.

BT rebasa la búsqueda local, empleando una estrategia de modificación de $N(x)$ a medida que la búsqueda progresa, remplazándola por otra vecindad $N^*(x)$. Un aspecto clave de BT es el empleo de estructuras especiales de memoria que sirven para determinar $N^*(x)$, y así organizar la manera en la cual se explora el espacio.

Las soluciones que son admitidas en $N^*(x)$ por estas estructuras de memoria se determinan de varias maneras. Una de ellas identifica soluciones encontradas sobre un horizonte especificado y les prohíbe pertenecer a $N^*(x)$, clasificándolas como tabú.

El proceso mediante el cual las soluciones adquieren un estatus tabú tiene varias facetas, diseñadas para promover un examen agresivo y guiado de nuevas soluciones. Una manera útil de visualizar e implementar este proceso es el remplazo de la evaluación original de soluciones mediante evaluaciones tabú, las cuales introducen penalizaciones para desalentar en forma significativa la elección de soluciones tabú (es decir, aquéllas que serán preferentemente excluidas de la vecindad $N^*(x)$, de acuerdo a su dependencia de los elementos que conforman el estatus tabú).

Además, las evaluaciones tabú incluyen también periódicamente incentivos para estimular la elección de otros tipos de soluciones con resultado de niveles de aspiración e influencias de largo plazo.

Algoritmo de búsqueda tabú

1. Dada una solución inicial x_i; $x_{opt} = x_i$; $k = 0$.
2. Hágase $k = k+1$; generar el conjunto de soluciones vecinas aleatorias $N_k^*(x_i)$, tal que cumpla las condiciones impuestas por la lista tabú o cumpla el criterio de aspiración.
3. Elegir el mejor x_j de N_k^* de acuerdo a $\varphi(x_j)$ y hacer $x_i = x_j$. (Nota φ puede ser igual a f o diferente.)
4. Si $f(x_j) < f(x_{opt})$, hacer $x_{opt} = x_j$.
5. Actualizar la lista tabú.
6. Si no se cumple el criterio de paro ir al paso 2.

Para este tipo de algoritmo, el criterio de paro suele estar dado por la realización de una determinada cantidad de iteraciones en las que no se encuentre una mejora de la solución. Otros criterios pueden ser la utilización de una cota de tiempo o un valor de umbral en el que se considere que la solución obtenida es suficientemente buena para la aplicación en cuestión.

Este algoritmo se inspira en la administración de la memoria para la búsqueda de soluciones, esto con el fin de no repetir cosas malas que se hicieron en el pasado (tabú), evitando retrocesos y llegar a un mismo punto.

5.5. Calidad de la implementación de los algoritmos

Las pruebas realizadas se diseñaron considerando las recomendaciones especificadas para este tipo de experimentos [Barr, 01, McGeoch, 02, Johnson, 02, Moret, 03]. Algunos de los principios aplicados son los siguientes: los experimentos deben estar vinculados con la literatura, se deben poder reproducir y comparar con los realizados en otras investigaciones, deben utilizar implementaciones razonablemente eficientes y casos de prueba estándar.

Capítulo 6

Pruebas experimentales

Capítulo 6. Pruebas experimentales

6.1. Descripción del escenario de prueba

Para los experimentos que se presentan en este capítulo se generaron aleatoriamente 390 (=13×30) casos de prueba para el modelo FURD (descrito en la Sección 4.1) y otros tantos para el modelo FVU-TR.

La tabla 6.1 muestra los datos que se usaron para generar los casos de prueba para el modelo FVU-TR; es decir para generar los coeficientes de las variables para las expresiones (4.20)-(4.23). Los 390 casos de prueba se dividen en 13 grupos con 30 casos en cada grupo, de tal suerte que los casos en cada grupo son similares entre sí.

Para mayor claridad se usará el término ejemplar para referirse a un caso de prueba, y se usará el término problema para referirse a un ejemplar genérico representativo de un grupo de ejemplares con características similares. Así, el primer renglón de la tabla contiene los datos usados para generar los 30 ejemplares del problema P1.

Los valores para P1 contenidos en las columnas de la segunda a la cuarta indican que el número de atributos, el número de sitios y el número de consultas son 2, 3 y 2 respectivamente para cada uno de los 30 ejemplares correspondientes a P1. Además, los valores en la quinta columna indican que cada uno de los valores de la matriz de uso [u_{km}] se eligió aleatoriamente entre 0 y 1; mientras que los valores de la sexta columna indican que cada uno de los valores de la matriz de frecuencias [f_{ki}] se eligió aleatoriamente entre 6 y 21; de esta manera cada ejemplar generado tiene matrices [u_{km}] y [f_{ki}] con valores diferentes de los del resto de los ejemplares. Finalmente, el resto de las columnas indican que C_j, C_{ij}, $1/\mu_Q$ y $1/\mu_R$ adoptaron el mismo valor en todos los ejemplares de P1. Una situación similar ocurre con la generación de los ejemplares para el resto de los problemas. Por último, es conveniente mencionar que los valores que se muestran en la tabla son valores típicos que se pueden encontrar en casos reales.

tabla 6.1: Datos de entrada para generar los casos de prueba

para el modelo FVU-TR

Problema	No. atributos L (=t)	No. sitios I	No. consultas K	Uso de atributos u_{km}	Frecuencia de emisión f_{ki}	Capacidad de procesamiento C_j	Velocidad de transmisión C_{ij}	Longitud media de consultas $1/\mu_Q$	Longitud media de respuestas $1/\mu_R$
P1	2	3	2	0-1	6-21	50	100,000	1,000	5,600
P2	10	2	3	0-1	0-19	50	100,000	1,000	5,600
P3	3	4	5	0-1	1-25	500	1,000,000	1,000	5,600
P4	10	3	4	0-1	2-21	500	1,000,000	1,000	5,600
P5	23	6	5	0-1	0-24	500	1,000,000	1,000	5,600

P6	14	5	13	0-1	0-25	500	1,000,000	1,000	5,600
P7	12	9	10	0-1	1-25	5,000	1,000,000	1,000	5,600
P8	17	10	5	0-1	0-25	500	1,000,000	1,000	5,600
P9	23	5	10	0-1	0-25	5,000	1,000,000	1,000	5,600
P10	18	11	9	0-1	0-25	5,000	1,000,000	1,000	5,600
P11	26	11	7	0-1	0-25	5,000	1,000,000	1,000	5,600
P12	17	13	10	0-1	0-25	5,000	1,000,000	1,000	5,600
P13	28	9	12	0-1	0-25	5,000	1,000,000	1,000	5,600

La tabla 6.2 muestra los datos que se usaron para generar los 390 casos de prueba para el modelo FURD. A semejanza de lo que ocurre para el modelo FVU-TR, en este caso para cada problema (de P1 a P13) se generaron aleatoriamente 30 ejemplares. Así, el primer renglón de la tabla contiene los datos usados para generar los 30 ejemplares del problema P1. El significado de los datos que contiene cada celda de la tabla 6.2 es similar al descrito anteriormente para la tabla 6.1, donde la definición de cada parámetro se encuentra en la Sección 4.1. Nota: es conveniente mencionar que los valores de $a_{\ell i}$ son todos cero, lo cual equivale a ignorar el cuarto término de la función objetivo del modelo FURD (4.1), el cual modela los costos de reubicación de los atributos.

tabla 6.2: Datos de entrada para generar los casos de prueba

para el modelo FURD

Problema	No. atributos L (=t)	No. sitios I	No. consultas K	c_{ij}	c_1	c_2	Uso de atributos $u_{k\ell}$	Frecuencia de emisión f_{ki}	$l_{k\ell}$	d_ℓ	$a_{\ell i}$	CA	CS_j
P1	2	3	2	1-5	1	1	0-1	6-21	3-8	3-8	0	1	135-171
P2	10	2	3	1-5	1	1	0-1	0-19	7-20	7-20	0	1	206-214
P3	3	4	5	1-5	1	1	0-1	1-25	5-18	5-18	0	1	112-220
P4	10	3	4	1-5	1	1	0-1	2-21	2-18	2-18	0	1	108-206
P5	23	6	5	1-5	1	1	0-1	0-24	2-18	2-18	0	1	130-234
P6	14	5	13	1-5	1	1	0-1	0-25	3-20	3-20	0	1	112-238
P7	12	9	10	1-5	1	1	0-1	1-25	12-20	12-20	0	1	106.231
P8	17	10	5	1-5	1	1	0-1	0-25	3-19	3-19	0	1	101-225
P9	23	5	10	1-5	1	1	0-1	0-25	3-19	3-19	0	1	118-216
P10	18	11	9	1-5	1	1	0-1	0-25	3-19	3-19	0	1	108.213
P11	26	11	7	1-5	1	1	0-1	0-25	3-20	3-20	0	1	109-213
P12	17	13	10	1-5	1	1	0-1	0-25	2-19	2-19	0	1	100-237
P13	28	9	12	1-5	1	1	0-1	0-25	2-20	2-20	0	1	100-231

Al observar los datos correspondientes al problema P1 de la tabla 6.2 destaca el hecho de que los valores de las columnas 2, 3, 4, 9 y 10 son respectivamente iguales a los valores de las columnas 2, 3, 4, 5 y 6 de la tabla 6.1 para el problema P1; y lo mismo ocurre para el resto de los problemas (de P2 a P13). La razón de esta coincidencia es que se

pretendía tener ejemplares similares para los dos modelos FVU-TR y FURD con el fin de poder hacer comparaciones entre las soluciones óptimas para ambos modelos.

Finalmente, conviene aclarar que no todos los ejemplares generados se usaron en los experimentos que se describen en las siguientes secciones. En la descripción de cada experimento se indicará cuáles ejemplares se utilizaron.

6.2. Experimento comparativo de AU y BT para el modelo FVU-TR

Como se ha establecido en la propuesta, el objetivo principal de este trabajo es "Formular un modelo matemático que involucre el tiempo de respuesta que permita optimizar la distribución y ubicación de fragmentos verticales en bases de datos distribuidas, y con base en este modelo diseñar y codificar un algoritmo que lo resuelva".

Desafortunadamente, el problema del modelo FVU-TR es NP-duro; y por lo tanto, solamente se pueden obtener soluciones mínimas para ejemplares pequeños; es decir, ejemplares en los que el número de atributos, el número de sitios y el número de consultas son pequeños. En este caso, para resolver ejemplares grandes es necesario usar algoritmos heurísticos, los cuales no garantizan la obtención de soluciones mínimas. En tales circunstancias, la calidad de un algoritmo se mide por la cercanía a la solución óptima de las soluciones generadas por el algoritmo. Desafortunadamente, no existe un algoritmo heurístico que sea mejor que cualquier otro para cualquier problema que se pretenda resolver. Así que para cada problema es necesario determinar un algoritmo de buena calidad para resolverlo.

En nuestro caso, se seleccionaron dos algoritmos (aceptación por umbral y búsqueda tabú) como posibles candidatos para resolver el problema del modelo FVU-TR. El propósito de este experimento es comparar la calidad de los dos algoritmos para determinar cuál es el mejor para dicho problema.

Se realizaron experimentos para 13 ejemplares: tres ejemplares para cada uno de los problemas de la tabla 6.1. Es importante destacar que aceptación por umbral (AU) y búsqueda tabú (BT) son algoritmos aleatorios; es decir, se mueven aleatoriamente de la solución actual a la siguiente (línea 6 del algoritmo AU en la Sección 5.3, y paso 2 del algoritmo BT en la Sección 5.4); por lo tanto, usualmente en diferentes ejecuciones generan soluciones diferentes para el mismo problema. Así que con el fin de obtener resultados estadísticamente significativos, se efectuaron 30 ejecuciones diferentes para cada ejemplar.

La experimentación consistió en resolver cada ejemplar usando AU y BT. La tabla 6.3 muestra los resultados obtenidos con este experimento, el cual se realizó en una computadora portátil con las siguientes características: CPU Intel Core Duo T5450 a 1.66 GHz, con 2038 MB de RAM, y sistema operativo Vista Home Premium de 32 bits. La primera columna de la tabla indica el problema. La columnas de la segunda a la cuarta contienen los valores mínimo, promedio y máximo de la función objetivo para las soluciones obtenidas por AU, y la quinta columna contiene el tiempo promedio de ejecución en segundos. Las columnas de la sexta a la novena contienen los valores

correspondientes para BT. Finalmente, es oportuno mencionar que los ejemplares de los problemas P1, P2, P3 y P4 se resolvieron con el algoritmo de búsqueda exhaustiva descrito en la Sección 5.2, con el fin de obtener las soluciones mínimas para estos ejemplares y poder determinar la calidad de las soluciones obtenidas por AU y BT.

tabla 6.3: Resultados comparativos de los algoritmos

aceptación por umbral y búsqueda tabú

Prob.	Aceptación por Umbral				Búsqueda Tabú			
	Función Objetivo (seg)			Tiempo	Función Objetivo (seg)			Tiempo
	Mínimo	Promedio	Máximo		Mínimo	Promedio	Max.	
P1.1	0.2282*	**0.2282***	0.2282*	734	0.2282*	0.2405	0.2896	937
P2.1	0.1182*	**0.1182***	0.1182*	946	0.1182*	0.2603	0.4461	1,183
P3.1	0.0073*	**0.0073***	0.0073*	664	0.0073*	**0.0073***	0.0073*	4,026
P4.1	0.0062*	**0.0063**	0.0067	1,119	0.0067	0.0069	0.0074	14,917
P5.11	0.0083	**0.0089**	0.0101	5,230	0.0099	0.0108	0.0115	175,498
P6.11	0.0094	**0.0119**	0.0132	4,848	0.0182	0.0281	0.0985	2,210
P7.1	0.0065	**0.0067**	0.0070	8,392	0.0076	0.0079	0.0081	130,749
P8.1	0.0093	**0.0097**	0.0111	15252	0.0115	0.0142	0.0194	61241
P9.1	0.0066	**0.0078**	0.0104	5,918	0.0088	0.0119	0.0144	125,851
P10.1	0.0067	**0.0070**	0.0073	16,099	0.0079	0.0084	0.0088	400,454
P11.1	0.0067	**0.0071**	0.0080	18,981	0.0082	0.0085	0.0087	951,114
P12.1	0.0069	**0.0073**	0.0083	21,271	0.0081	0.0087	0.0092	503,224
P13.1	0.0074	**0.0103**	0.0123	15,690	0.0119	0.0149	0.0209	101,629

Ya que del problema P1 al P4 se conocen los valores óptimos, un asterisco indica cuando un algoritmo pudo encontrar la solución óptima. Los resultados muestran que el algoritmo de aceptación por umbral siempre encuentra los óptimos para los problemas P1-P3 y algunas veces los encuentra para el problema 4; sin embargo, BT siempre encuentra los óptimos solamente para el problema P3, a veces encuentra los valores para los problemas P1 y P2, y nunca para el problema P4. Al ver los valores promedio de la función objetivo, es claro que AU obtiene mejores resultados que BT usualmente con un menor tiempo de ejecución.

Los resultados experimentales muestran que aceptación por umbral obtiene mejores resultados que búsqueda tabú usualmente con un valor menor de tiempo de ejecución. Sin embargo, es importante señalar que la implementación de búsqueda tabú es una versión básica que no incluye estrategias sofisticadas de intensificación y diversificación; por lo tanto, tal vez una implementación más sofisticada podría producir mejores resultados.

6.3. Experimentos comparativos de los modelos FVU-TR y FURD

El propósito del primer experimento es determinar si vale la pena usar un modelo tan complejo como FVU-TR en lugar de uno más simple como FURD para obtener el diseño óptimo de un SBDD. La respuesta a esta interrogante depende de qué tan diferente es la solución óptima para el SBDD cuando éste se modela usando FVU-TR en comparación con la solución óptima cuando se usa el modelo FURD.

Para tal efecto, se realizaron experimentos para obtener los diseños óptimos para 80 ejemplares de los 120 generados, ya que sólo 80 tuvieron soluciones factibles para ambos modelos, y para cada uno de los 80 ejemplares se obtuvieron las soluciones mínimas para ambos modelos ($x^*_{FVU\text{-}TR}$ para el modelo FVU-TR y x^*_{FURD} para el modelo FURD) con el fin de determinar qué tan diferentes son. En este momento es oportuno aclarar que no se realizaron pruebas para los problemas P5 a P13 debido a que son demasiado grandes para ser resueltos con el algoritmo de búsqueda exhaustiva implementado.

La tablas 6.4a, 6.4b, 6.4c muestran los resultados de este experimento. La primera y segunda columnas indican el identificador de cada ejemplar. Para el primer renglón de la tabla 6.4a, la tercera columna contiene el tiempo de respuesta mínimo $z_{FVU\text{-}TR}(x^*_{FVU\text{-}TR})$, donde $z_{FVU\text{-}TR}$ representa el valor de la función objetivo del modelo FVU-TR (expresión 4.20) y $x^*_{FVU\text{-}TR}$ representa la solución mínima del ejemplar 1 de P1 modelado con FVU-TR (expresiones (4.20)-(4.23)) obtenida usando el algoritmo de búsqueda exhaustiva; mientras que la cuarta columna contiene el tiempo de respuesta $z_{FVU\text{-}TR}(x^*_{FURD})$, donde x^*_{FURD} representa la solución mínima del mismo ejemplar 1 de P1 modelado con FURD (expresiones (4.1)-(4.6)) obtenida usando el algoritmo de búsqueda exhaustiva. Para el resto de los renglones de las tablas (6.4a, ..., 6.4d), el significado de los valores en cada celda es similar. La quita columna contiene la diferencia de los valores de la tercera y cuarta columnas expresada como porcentaje.

tabla 6.4a: Resultados comparativos de tiempos de respuesta de los modelos FVU-TR y FURD (problema P1)

Proble-ma	Ejem-plar	Tiempo de Respuesta(seg)		
		Modelo FVU-TR	Modelo FURD	% Diferencia
P1	1	0.2282	0.2336	54
P1	5	0.3716	0.3746	30
P1	8	0.1274	0.1294	20
P1	9	0.1270	0.1370	100
P1	11	0.1464	0.1668	204
P1	13	0.1264	0.1294	30
P1	14	0.2859	0.2859	0
P1	19	0.3314	0.3465	151
P1	21	0.3240	0.3251	11
P1	22	0.1242	0.1680	438
Promedio				**104**

tabla 6.4b: Resultados comparativos de tiempos de respuesta de los modelos FVU-TR y FURD (problema P2)

Proble-Ma	Ejem-plar	Tiempo de Respuesta(seg)		
		Modelo FVU-TR	Modelo FURD	% Diferencia
P2	1	0.1182	0.1182	0

P2	4	0.1137	0.1137	0
P2	5	0.1844	0.1844	0
P2	8	0.0784	0.0784	0
P2	11	0.0787	0.0787	0
P2	13	0.0868	0.0868	0
P2	17	0.1435	0.1435	0
P2	22	0.1927	0.5277	173
P2	23	0.1752	0.1752	0
P2	28	0.1802	0.1802	0
Promedio				**17**

tabla 6.4c: Resultados comparativos de tiempos de respuesta de los modelos FVU-TR y FURD (problema P3)

Proble-ma	Ejem-plar	Tiempo de Respuesta(seg)		
		Modelo FVU-TR	Modelo FURD	% Diferencia
P3	1	0.0073	0.0082	9
P3	2	0.0074	0.0074	0
P3	3	0.0073	0.0087	14
P3	4	0.0071	0.0082	11
P3	5	0.0073	0.0081	8
P3	6	0.0062	0.0074	12
P3	7	0.0076	0.0082	6
P3	8	0.0071	0.0084	13
P3	9	0.0069	0.0082	13
P3	10	0.0068	0.0082	14
P3	11	0.0071	0.0086	15
P3	12	0.0077	0.0079	2
P3	13	0.0073	0.0073	0
P3	14	0.0076	0.0081	5
P3	15	0.0072	0.0080	8
P3	16	0.0078	0.0078	0
P3	17	0.0078	0.0080	2
P3	18	0.0075	0.0086	11
P3	19	0.0073	0.0087	14
P3	20	0.0071	0.0080	9
P3	21	0.0072	0.0084	12
P3	22	0.0066	0.0078	12
P3	23	0.0076	0.0079	3
P3	24	0.0073	0.0096	23
P3	25	0.0072	0.0094	22
P3	26	0.0068	0.0086	18
P3	27	0.0074	0.0074	0

P3	28	0.0074	0.0078	4
P3	29	0.0076	0.0083	7
P3	30	0.0075	0.0086	11
Promedio				**9**

tabla 6.4d: Resultados comparativos de tiempos de respuesta de los modelos FVU-TR y FURD (problema P4)

Proble-ma	Ejem-plar	Tiempo de Respuesta(seg)		
		Modelo FVU-TR	Modelo FURD	% Diferencia
P4	1	0.0062	0.0071	9
P4	2	0.0068	0.0081	13
P4	3	0.0068	0.0069	1
P4	4	0.0057	0.0094	37
P4	5	0.0059	0.0078	19
P4	6	0.0062	0.0063	1
P4	7	0.0065	0.0065	0
P4	8	0.0068	0.0068	0
P4	9	0.0060	0.0086	26
P4	10	0.0067	0.0083	16
P4	11	0.0063	0.0063	0
P4	12	0.0068	0.0068	0
P4	13	0.0067	0.0073	6
P4	14	0.0067	0.0087	20
P4	15	0.0069	0.0075	6
P4	16	0.0067	0.0081	14
P4	17	0.0062	0.0084	22
P4	18	0.0065	0.0078	13
P4	19	0.0061	0.0061	0
P4	20	0.0065	0.0087	22
P4	21	0.0066	0.0085	19
P4	22	0.0064	0.0078	14
P4	23	0.0066	0.0080	14
P4	24	0.0065	0.0079	14
P4	25	0.0062	0.0081	19
P4	26	0.0055	0.0066	11
P4	27	0.0069	0.0086	17
P4	28	0.0062	0.0092	30
P4	29	0.0057	0.0100	43
P4	30	0.0067	0.0081	14
Promedio				**14**

La última columna de la tabla 6.4a muestra que, para los ejemplares del problema P1, las soluciones óptimas del modelo FURD dan en promedio tiempos de respuesta 104%

más grandes que los de las soluciones óptimas del modelo FVU-TR. Además, las tablas 6.4b, 6.4c y 6.4d revelan que las soluciones óptimas del modelo FURD producen en promedio tiempos de respuesta 17%, 9% y 14% mayores que los del modelo FVU-TR para los problemas P2, P3 y P4. Considerando los 80 ejemplares de los cuatro problemas, se tiene que con el modelo FURD las soluciones óptimas tienen en promedio tiempos de respuesta 30% más grandes que los de las soluciones óptimas del modelo FVU-TR. Por lo tanto, puede concluirse que vale la pena el uso del modelo FVU-TR.

El segundo experimento es al revés del primero; es decir, se pretende determinar qué tan grande es la diferencia entre los costos (transmisión, acceso y almacenamiento) de las soluciones óptimas obtenidas con ambos modelos. Para tal efecto (a semejanza del experimento anterior), se realizaron experimentos para obtener los diseños óptimos para 100 ejemplares de los 120 generados, ya que sólo 100 tuvieron soluciones factibles para ambos modelos, y para cada uno de los 100 ejemplares se obtuvieron las soluciones mínimas para ambos modelos ($x^*_{\text{FVU-TR}}$ para el modelo FVU-TR y x^*_{FURD} para el modelo FURD).

Las tablas 6.5a, 6.5b, 6.5c y 6.5d muestran los resultados del segundo experimento. La primera y segunda columnas indican el identificador de cada ejemplar. Para el primer renglón de la tabla 6.5a, la tercera columna contiene el costo (transmisión, acceso y almacenamiento) mínimo $z_{\text{FURD}}(x^*_{\text{FURD}})$, donde z_{FURD} representa el valor de la función objetivo del modelo FURD (expresión 4.1) y x^*_{FURD} representa la solución mínima del ejemplar 1 de P1 modelado con FURD (expresiones (4.1)-(4.6)) obtenida usando el algoritmo de búsqueda exhaustiva; mientras que la cuarta columna contiene el tiempo de respuesta $z_{\text{FURD}}(x^*_{\text{FVU-TR}})$, donde $x^*_{\text{FVU-TR}}$ representa la solución mínima del mismo ejemplar 1 de P1 modelado con FVU-TR (expresiones (4.20)-(4.23)) obtenida usando el algoritmo de búsqueda exhaustiva. Para el resto de los renglones de las tablas (6.5a, ..., 6.5d), el significado de los valores en cada celda es similar. La quita columna contiene la diferencia de los valores de la tercera y cuarta columnas expresada como porcentaje.

tabla 6.5a: Resultados comparativos de costos de los modelos FVU-TR vs. FURD (problema P1)

Proble-Ma	Ejem-plar	Costo Trans., Acceso y Alm. ($)		
		Modelo FURD	Modelo FVU-TR	% Diferencia
P1	1	559	887	59
P1	4	676	773	14
P1	5	839	1352	61
P1	6	286	399	40
P1	8	939	1307	39
P1	9	321	839	161
P1	10	944	1377	46
P1	11	394	849	115
P1	13	731	1026	40
P1	14	534	534	0

P1	18	1024	1041	2
P1	19	502	1650	229
P1	21	881	895	2
P1	22	606	1122	85
P1	23	587	731	25
P1	24	1064	1185	11
P1	26	344	394	15
P1	27	1202	1731	44
P1	28	627	1106	76
P1	29	367	564	54
P1	30	989	2194	122
Promedio				**59**

tabla 6.5b: Resultados comparativos de costos de los modelos FVU-TR vs. FURD (problema P2)

Proble-Ma	Ejem-plar	Costo Trans., Acceso y Alm. ($)		
		Modelo FURD	Modelo FVU-TR	% Diferencia
P2	1	893	1214	36
P2	4	859	859	0
P2	5	961	961	0
P2	6	956	1576	65
P2	8	1095	1095	0
P2	9	2423	4122	70
P2	11	978	978	0
P2	12	2118	3402	61
P2	13	928	928	0
P2	17	1790	1790	0
P2	18	4049	4296	6
P2	19	3797	4248	12
P2	20	2617	4294	64
P2	22	859	1191	39
P2	23	5144	5144	0
P2	25	2585	2762	7
P2	26	2544	3078	21
P2	27	723	723	0
P2	28	2923	5003	71
Promedio				**24**

tabla 6.5c: Resultados comparativos de costos de los modelos FVU-TR vs. FURD (problema P3)

Proble-Ma	Ejem-plar	Costo Trans., Acceso y Alm. ($)		
		Modelo	Modelo	%

		FURD	FVU-TR	Diferencia
P3	1	6849	9491	39
P3	2	11870	11870	0
P3	3	8827	13264	50
P3	4	4504	5379	19
P3	5	3254	4457	37
P3	6	1910	2589	36
P3	7	3785	4825	27
P3	8	3632	4222	16
P3	9	5214	5863	12
P3	10	5806	7316	26
P3	11	7129	8972	26
P3	12	5628	11038	96
P3	13	3464	3464	0
P3	14	3625	5887	62
P3	15	5967	7351	23
P3	16	5058	10278	103
P3	17	5238	5927	13
P3	18	3703	5112	38
P3	19	5511	7841	42
P3	20	3888	6683	72
P3	21	12508	12647	1
P3	22	2791	5155	85
P3	23	4977	9085	83
P3	24	7124	8130	14
P3	25	8207	8824	8
P3	26	2442	4408	81
P3	27	4192	4192	0
P3	28	3410	7685	125
P3	29	3028	4034	33
P3	30	4861	6083	25
Promedio				**40**

tabla 6.5d: Resultados comparativos de costos de los modelos FVU-TR vs. FURD (problema P4)

Proble-Ma	Ejem-plar	Costo Trans., Acceso y Alm. ($)		
		Modelo FURD	Modelo FVU-TR	% Diferencia
P4	1	7798	8889	14
P4	2	9753	16637	71
P4	3	9062	12542	38
P4	4	6944	14526	109
P4	5	3157	3655	16
P4	6	6974	7009	1

P4	7	11863	11863	0
P4	8	10320	10531	2
P4	9	8302	16238	96
P4	10	10156	14746	45
P4	11	9485	9485	0
P4	12	14706	15546	6
P4	13	1803	5686	215
P4	14	12616	19850	57
P4	15	5983	12607	111
P4	16	18274	31296	71
P4	17	11447	27817	143
P4	18	9707	16860	74
P4	19	9433	9433	0
P4	20	4465	12545	181
P4	21	18820	29221	55
P4	22	6816	10830	59
P4	23	8467	8908	5
P4	24	3083	4765	55
P4	25	17353	33262	92
P4	26	4243	4261	0
P4	27	7522	16307	117
P4	28	9406	15256	62
P4	29	8303	14462	74
P4	30	4044	5635	39
Promedio				**60**

La última columna de la tabla 6.5a muestra que, para los ejemplares del problema P1, las soluciones óptimas del modelo FVU-TR dan en promedio costos 59% más grandes que los de las soluciones óptimas del modelo FURD. Además, las tablas 6.5b, 6.5c y 6.5d revelan que las soluciones óptimas del modelo FVU-TR producen en promedio costos 24%, 40% y 60% mayores que los del modelo FURD para los problemas P2, P3 y P4. Considerando los 100 ejemplares de los cuatro problemas, se tiene que con el modelo FURD las soluciones óptimas tienen en promedio costos 47% más grandes que los de las soluciones óptimas del modelo FVU-TR.

Finalmente puede concluirse que, como era de esperar, los objetivos de minimizar el tiempo de respuesta y el costo son antagónicos; es decir, las soluciones que tienen menor tiempo de respuesta tienen costos más grandes y viceversa; así que si un diseñador está preocupando por ambos, debería considerar un modelo donde se integren los dos objetivos.

6.4. Experimento comparativo del modelo FVU-TR y un método heurístico

Una amplia búsqueda de la literatura sobre fragmentación vertical reveló que la mayoría de los trabajos tratan solamente con fragmentación o ubicación de fragmentos; muy pocos tratan con ambos, como el nuestro. Uno de los artículos más recientes que trata con ambos es [Ma, 2006].

Sería inequitativo comparar nuestro método con uno que solamente tratara con fragmentación o ubicación, ya que estos dos aspectos son tan interdependientes que el diseño óptimo solamente se puede obtener cuando se tratan ambos aspectos. Por lo tanto, se decidió realizar un experimento para comparar los resultados de nuestro método y los del reportado en [Ma, 2006]. Los valores de los parámetros para el ejemplo presentado en [Ma, 2006] son los siguientes: número de nodos = 4, número de atributos = 10, número de consultas = 8; la matriz de uso, la matriz de frecuencias y la matriz de costos de comunicación se muestran en las tablas 6.6, 6.7 y 6.8.

tabla 6.6: Matriz de uso [u_{kl}] para el ejemplo

Consulta	**Atributos**									
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
q_1	1	0	0	0	1	0	1	0	0	0
q_2	0	1	1	0	0	0	0	1	1	0
q_3	0	0	0	1	0	1	0	0	0	1
q_4	0	1	1	0	0	0	1	1	0	0
q_5	1	1	0	0	1	0	1	1	1	0
q_6	1	0	0	0	1	0	0	0	0	0
q_7	0	0	1	0	0	0	0	0	1	0
q_8	0	0	1	1	0	1	0	0	1	1

tabla 6.7: Matriz de frecuencias [f_{ki}] para el ejemplo

Consulta	**Nodos**			
	s_1	s_2	s_3	s_4
q_1	10	15	0	0
q_2	10	20	10	10
q_3	0	0	15	10
q_4	10	0	15	10
q_5	5	10	5	5
q_6	10	5	5	5
q_7	5	10	5	5
q_8	5	5	3	2

tabla 6.8: Matriz de costos de comunicación [c_{ij}] para el ejemplo

Nodos	**Nodos**

	s_1	s_2	s_3	s_4
s_1	0	10	25	20
s_2	10	0	20	15
s_3	25	20	0	15
s_4	20	15	15	0

Para el modelo FVU-TR (Sección 4.2), los valores de los parámetros son los siguientes: número de nodos = 4, número de atributos = 10, número de consultas = 8, longitud media de consultas ($1/\mu_Q$) = 1000; longitud media de respuestas ($1/\mu_R$) = 5587; la matriz de uso, la matriz de frecuencias, la matriz de velocidades de transmisión y el vector de velocidades de procesamiento se muestran en las tablas 6.6, 6.7, 6.9 y 6.10.

tabla 6.9: Matriz de velocidades de transmisión [C_{ij}] para el ejemplo

Nodos	**Nodos**			
	s_1	s_2	s_3	s_4
s_1	∞	400,000	200,000	250,000
s_2	400,000	∞	250,000	300,000
s_3	200,000	250,000	∞	300,000
s_4	250,000	300,000	300,000	∞

tabla 6.10: Vector de velocidades de procesamiento [C_j] para el ejemplo

Nodos			
s_1	s_2	s_3	s_4
200	200	200	200

La fragmentación y ubicación óptima obtenidas por un algoritmo exacto para el modelo FVU-TR se encuentran en la tabla 6.11, la cual muestra los valores de las variables x_{ij} (donde cada i denota un atributo y cada j representa un nodo) del modelo descrito en la Sección 4.2. Cada renglón de la tabla indica en qué nodo debe ubicarse cada atributo; por ejemplo, el primer renglón indica que el atributo 1 debe ubicarse en el nodo s_4, y de igual manera los atributos 2, 3, 5, 7, 8 y 9, los cuales constituyen así el fragmento $F_1 = \{a_1, a_2, a_3, a_5, a_7, a_8, a_9\}$; similarmente, el cuarto, el sexto y el décimo renglón definen el fragmento $F_2 = \{a_4, a_6, a_{10}\}$ que debe ubicarse en el nodo s_1. El tiempo de respuesta de esta solución es 0.016029 seg. Por otra parte, la fragmentación y ubicación reportadas en [Ma, 2006] para este ejemplo es la siguiente: el fragmento $F_1 = \{a_1, a_2, a_3, a_5, a_7, a_8, a_9\}$ ubicado en el nodo s_2, y el fragmento $F_2 = \{a_4, a_6, a_{10}\}$ ubicado en el nodo s_3. El tiempo de respuesta para esta solución calculado con la expresión (4.20) es 0.020302 seg. Obsérvese que ambos métodos generan la misma fragmentación; sin embargo, la ubicación de los fragmentos es diferente, lo cual ocasiona que el tiempo de respuesta obtenido con el algoritmo heurístico sea 27% mayor que el obtenido con nuestro método. Desafortunadamente, no se reportan casos de prueba adicionales en [Ma, 2006]; y por lo tanto, no se pueden derivar conclusiones sólidas de este experimento.

tabla 6.11: Valores de las variables x_{ij} para la solución óptima del modelo FVU-TR

Valor de *i* (Atributos)	**Valor de *j* (Nodos)**			
	1	2	3	4
1	0	0	0	1
2	0	0	0	1
3	0	0	0	1
4	1	0	0	0
5	0	0	0	1
6	1	0	0	0
7	0	0	0	1
8	0	0	0	1
9	0	0	0	1
10	1	0	0	0

Capítulo 7

CONCLUSIONES

Capítulo 7. Conclusiones

Se concluye que se pudo modelar del tiempo de respuesta mediante la simulación de los procesos aleatorios involucrados en la generación, transmisión y procesamiento de consultas, así como aquéllos involucrados en la transmisión de las respuestas.

Para tal efecto, se ajusto la fórmula propuesta para modelar el tiempo de respuesta (mediante mínimos cuadrados) a los valores obtenidos de la simulación, con el fin de comprobar qué tan aproximado es el comportamiento estimado por la fórmula propuesta versus el comportamiento simulado.

Se desarrolló un nuevo modelo de programación matemática para el problema del diseño de la fragmentación vertical en BDDs, en particular una nueva función objetivo que involucra la fórmula propuesta para modelar el tiempo de respuesta.

Se utilizaron dos algoritmos metaheurísticos (aceptación por umbral y búsqueda tabú) para encontrar la solución al modelo, y determinar cuál de los algoritmos es mejor para resolver el problema.

Finalmente se realizaron pruebas experimentales con el modelo de programación entera para la fragmentación, ubicación y reubicación dinámica de datos (FURD) y el propuesto en esta tesis (FVU-TR), con el fin de determinar si existen diferencias entre las soluciones obtenidas para cada modelo y qué tan grandes son las diferencias.

7.1. Conclusiones sobre los resultados

Aunque desde hace décadas se ha considerado conveniente el desarrollo de un modelo que permita encontrar el diseño de una BDD de tal manera que se optimice el tiempo de respuesta de las consultas [Dowdy, 82], [Chakravarthy, 94]; sin embargo, no se había desarrollado dicho modelo, tal como lo revela un estudio de la literatura especializada en el tema. En este proyecto de tesis doctoral, por primera vez se desarrolló un modelo matemático (denominado FVU-TR) para encontrar el diseño óptimo de la fragmentación vertical y su ubicación de tal manera que se minimice el tiempo de respuesta de las consultas (Sección 4.2), con lo cual se alcanzó el primero de los objetivos generales de este proyecto de tesis (Sección 1.4).

Con el fin de encontrar la solución al modelo mencionado, se implementó un algoritmo de búsqueda exhaustiva (descrito en la Sección 5.2). Aunque este algoritmo permite encontrar la solución óptima, desafortunadamente sólo sirve para resolver ejemplares pequeños, ya que para ejemplares grandes tarda demasiado tiempo, lo cual se debe a que el problema modelado es NP-duro. En estas circunstancias es recomendable usar algoritmos metaheurísticos, los cuales permiten encontrar buenas soluciones con un tiempo de ejecución razonable. Para tal efecto, se implementaron dos algoritmos metaheurísticos: aceptación por umbral y búsqueda tabú (descritos en las Secciones 5.3 y 5.4). Las pruebas experimentales de los dos algoritmos demostraron que aceptación por umbral tiene mejor desempeño que búsqueda tabú: encuentra mejores soluciones con un

tiempo generalmente menor. Con esto se alcanzó el segundo de los objetivos generales (Sección 1.4).

Finalmente, quedaba una duda: ¿realmente conviene usar un modelo matemático que minimice el tiempo de respuesta en lugar de uno que minimice los costos de transmisión y procesamiento como los modelos tradicionales? Esta pregunta es importante porque los resultados de las corridas con los algoritmos metaheurísticos muestran que tardan más tiempo en resolver los ejemplares del modelo FVU-TR que los ejemplares similares del modelo FURD. Las pruebas experimentales (tablas 6.4a, 6.4b, 6.4c y 6.4d) muestran que para algunos ejemplares los dos modelos tienen el mismo tiempo de respuesta para sus soluciones óptimas; sin embargo, el tiempo de respuesta de las soluciones óptimas del modelo FURD son en promedio 30% más grandes que el tiempo medio de respuesta para las soluciones óptimas del modelo FVU-TR. Con los resultados de las pruebas comparativas, se concluye que en general los dos modelos tienen soluciones óptimas diferentes y que es significativa la diferencia en tiempos de respuesta para las soluciones de ambos modelos; por lo que vale la pena el uso del modelo FVU-TR.

7.2. Aportaciones

Las principales aportaciones alcanzadas con este proyecto de tesis son los siguientes:

- Se encontró en este trabajo una fórmula que describe con bastante precisión el comportamiento del tiempo de respuesta observado en la transmisión y procesamiento de consultas, así como en la transmisión de las respuestas.
- La expresión propuesta fue validada mediante la simulación de los procesos aleatorios involucrados en la generación, transmisión, procesamiento de consultas y transmisión de las respuestas.
- Mediante el uso de la fórmula mencionada anteriormente, se desarrolló el modelo matemático denominado FVU-TR para el problema de la distribución y ubicación de fragmentos verticales en BDDs, el cual involucra el tiempo de respuesta.
- Se probó el desempeño de dos métodos metaheurísticos para la solución de este tipo de problemas: aceptación por umbral y búsqueda tabú.
- Se demostró que para este tipo de problema, con base en pruebas experimentales realizadas con los dos algoritmos (aceptación por umbral y búsqueda tabú), aceptación por umbral tiene mejor desempeño.
- Se concluyó que vale la pena el uso del modelo FVU-TR, ya que el tiempo de respuesta de las soluciones óptimas del modelo FURD son en promedio 30% más grandes que el tiempo medio de respuesta para las soluciones óptimas del modelo FVU-TR.
- Se determinó, al realizar pruebas experimentales que compran los resultados óptimos obtenidos por los modelos FURD vs. FVU-TR, que con el modelo FVU-TR las soluciones óptimas tienen en promedio costos 47% más grandes que los del modelo FURD.
- Se llegó a la conclusión, como era de esperarse, que los objetivos de minimizar el tiempo de respuesta y el costo son antagónicos: es decir, las soluciones que tienen menor tiempo de respuesta tienen costos más grandes y viceversa; así que si un

diseñador está preocupado por ambos, debería considerar un modelo donde se integren los dos objetivos.

7.3. Trabajos futuros

Uno de los trabajos propuestos en el futuro es el que el modelo propuesto aquí se puede mejorar de varias maneras:

1. La primera es formular una nueva función objetivo que combine adecuadamente el tiempo de respuesta, y los costos de transmisión, acceso y procesamiento de consultas.

2. Incluir una nueva restricción que involucre el tiempo de respuesta, y que la función objetivo involucre sólo los costos de transmisión, acceso y procesamiento.

3. Y que finalmente se incluya la replicación de los fragmentos en las alternativas 1 y 2, incluyendo las "consultas de actualización", ya que el modelo aquí propuesto sólo considera "consultas de lectura" (ver Sección 1.3).

7.4. Trabajos publicados y presentados derivados de esta tesis

1. "Minimizing Roundtrip Response Time in Distributed Databases with Vertical Fragmentation", Journal of Computational and Applied Mathematics, vol. 259, ISSN 0377-0427, Elsevier, Mar. 2014, pp. 905-913.
2. "Vertical Fragmentation Design of Distributed Databases Considering the Nonlinear Nature of Roundtrip Response Time", Lecture Notes in Artificial Intelligence, vol. 6277, ISSN 0302-9743, Springer-Verlag, Sep. 2010, pp. 173-182.
3. "Modeling the Nonlinear Nature of Response Time in the Vertical Fragmentation Design Of Distributed Databases", Advances in Soft Computing, vol. 50, ISBN 978-3-540-85862-1, Springer, 2008, pp. 605-612.

REFERENCIAS

[Apers, 88] Apers Peter M. G., Vrije, Data Allocation in Distributed Database Systems, Universiteit, *ACM Transactions on Database Systems*, Vol. 13, No. 3, September 1988, pp 263-304.

[Barr 01] Barr R.S., Golden, B.L., Kelly, J., Steward, W.R., Resende, M.: Guidelines for Designing and Reporting on Computational Experiments with Heuristic Methods. *Proceedings of International Conference on Metaheuristics for Optimization. Kluwer Publishing, Norwell*, MA (2001) 1-17

[Basseda, 06] Basseda R., *Fragment Allocation in Distributed Database Systems*. Technical Report. Faculty of Electrical and Computer Eng., School of Engineering, Database Research Group, University of Tehran, Iran (2006).

[Bell, 00] Bell David, Jane Grimson, *Distributed Database System*, Addison – Wesley,2000.

[Ceri, 84] Ceri Stefano, Pelagatti Giuseppe, *Distributed Databases Principles and Sytems*,. McGraw-Hill, 1984.

[Ceri, 87] Ceri Stefano, Barbara Percini, and Gio Wiederhold, Distributed Database Design Methodologies, *Proceedings of the IEEE*, Vol 75, No. 5, May 1987.

[Ceri, 93] Ceri Stefano, Shamkant Navathe, and Gio Wiederhold, Distribution Design of Logical Database Schemas, *IEEE Transactions on Software Engineering*, vol. Se-9, No. 4, July 1993.

[Chakravarthy, 92] Chakravarthy S. Muthuraj, J. Varadarajan, R. Navathe S.: An Objetive Function For Vertically Partitioning Relations in Distributed Databases and its Analysis, Department of Computer and Information Sciences, *Computer Science Engineering Building*, University of Florida (1992)

[Chaturvedi, 94] Chaturvedi Alok R., Ashok Choubey and Jinssheng Roan, Scheduling the Allocation of Data Fragments in a Distributed Database Enviroment: A Machine Learning Aproach, *IEEE Transactions of Engineering Management*, Vol. 41 No.2, Mayo 1994.

[Date, 08] Date C.J., *Introducción a los Sistemas de Bases de Datos*, Volumen 1, 5ª. Edición, Addison – Wesley, 2008.

[Dowdy, 82] Dowdy Lawrence W. and Derrell V. Foster, Comparative Models of the file Assignment Problem, *ACM Computing Survey*, Vol.14 No. 2, June 1982, pp: 287-313.

[Dueck, 90] Dueck, G., Scheuer, T.: Threshold Accepting: a General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *Journal of Computational Physics*, Vol. 90, No. 1 (1990) 161–175.

[Fraire, 05] Fraire Huacuja Héctor Joaquín, *Una metodología para el diseño de la fragmentación y ubicación en grandes bases de datos distribuidas*, Tesis Doctoral, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Mor., 2005.

[García, 92] García Hong-Mei C., "*A Semantics - Based Methodology for Integrated Distributed Database Design: Toward Combined Logical and Fragmentation Design and Design Automation"*, Tesis Doctoral, Universidad de Arizona, Julio de 1992.

[Goli, 12] M. Goli, S.M.T. Rouhani, A new vertical fragmentation algorithm based on ant collective behavior in distributed database systems, *Knowledge and Information Systems* 30 (2) (2012) 435–455.

[Golfarelli, 99] Golfarelli M., Maio, D., Rizzi, S.: Vertical Fragmentation of Views in Relational Data Warehouses, Universitá di Bologna (1999)

[Gorla, 08] Gorla N., Wing-yan, B.P.: Vertical Fragmentation in Databases Using Data-Mining Technique. Int. *J. of Data Warehousing and Mining*, Vol. 4, No. 3 (2008) 35–53.

[Guk, 91] Guk Dong, Shin and Keki B. Fragmentating Relations Horizontally Ussing a Knowledge-Bases Approach, Irani, *IEEE Transactions on Software*, vol 17, No.9, September 1991.

[Hababeh, 07] Hababeh I., Ramachandran, M., Bowring, N.: Application Design for Data Fragmentation and Allocation in Distributed Database Systems. Distributed Database Systems, *Research and Practice Conference*. Leeds, U.K. (2007).

[Huang, 01] Huang Y.F., Chen, J.H.: Fragment Allocation in Distributed Database Design. *J. of Information Science and Engineering*, Vol. 17, No. 3 (2001) 491–506.

[Johnson, 02] Johnson D.S., McGeoch, L.A.: Experimental Analysis of Heuristics for the STSP. In: Gutin, G., Punnen, A. (eds.): The Traveling Salesman Problem and its Variations. *Kluwer Academic Publishers*, Dordrecht (2002) 369-443

[Kamali, 11] S. Kamali, P. Ghodsnia, K. Daudjee, Dynamic data allocation with replication in distributed systems, in: *Proc. 30th IEEE International Performance Computing and Communication Conf.*, Orlando, USA, 2011, pp. 1–8.

[Karimi, 99] R. Karimi, S.M.T. Rouhani, A new ant colony optimization based algorithm for data allocation problem in distributed databases, *Knowledge and Information Systems* 20 (3) (2009) 349–373.

[Khan,10] S.U. Khan, I. Ahmad, Replicating data objects in large distributed database systems: an axiomatic game theoretic mechanism design approach, *Distributed and Parallel Databases* 28 (2–3) (2010) 187–218.

[Kleinrock, 76] Kleinrock Leonard, *Queueing Systems, Vol. II: Computer Applications*. John Wiley & Sons, E.U.A. (1976) 314–322.

[Kleinrock, 07] Kleinrock, Leonard, *Communication Nets: Stochastic Message Flow and Delay*. Dover Publications, E.U.A. (2007).

[Kulkarni, 89] Kulkarni Uday Ravidranath, *An integrated support system for design of distributed data bases*, Disertación Doctoral, Universidad de Wisconsin-Milwaukee, 1989.

[Kulkarni, 91] Kulkarni Uday Ravidranath and Helmant K. Jain, Using Semantic Knowdeledge in Partitioning and Allocation of Data in Distributed Databases, *IEEE*, 1991.

[Lee, 90] Lee Heeseok and Olivia R. Liu sheng, Optimal Data Allocation in a Bus Computer Network, University of Arizona,IEEE Mayo 1990.

[Lee, 92] Lee Heeseok y Olivia R. Liu Sheng,Optimal Data Allocation in a Bus Computer Network, Proc. Second *Workshop on the Management of Replicated Data*. Pp 62-65, 1992.

[Lin, 93] Lin X., Orlowska, M.E., Zhang, Y.: On Data Allocation with the Minimum Overall Communication Costs in Distributed Database Design. Proc. 5th *International Conference on Computing and Information* (1993) 539–544.

[Ma, 06] Ma H., Schewe, K.-D., Kirchberg, M.: A Heuristic Approach to Vertical Fragmentation Incorporating Query Information. Proc. 7th *Int. Baltic Conf. on Databases and Information Systems* (2006) 69–76.

[McGeoch, 02] McGeoch C.C., Experimental Analysis of Algorithms. In: Pardalos, P.M., Romeijn, H.E. (eds.): *Handbook of Global Optimization*, Vol. 2(2002) 489-513

[March, 95] March Salvatore T. and Sangkyu Rho, Allocating Data and Operations to Nodes in distributed Database Design, IEEE *Transactions on Knowledge and Data Engineering,* Vol 7, No.2, April 1995.

[Menon, 05] Menon S., Allocating Fragments in Distributed Databases. IEEE *Trans. Parallel and Distributed Systems,* Vol. 16, No. 7 (2005) 577–585.

[Moret, 03] Moret B.M.E., Toward a Discipline of Experimental Algorithmics. In: Goldwasser, M.H., Johnson, D.S., McGeoch, C. (eds.): Data Structures, Near Neighbor Searches, and Methodology: *Fifth and Sixth DIMACS Implementation Challenges*, Series DIMACS, Vol. 5 (2003) 197-214

[Mondragón, 01] Mondragón Ibarra Rosa Lina, *Un modelo matemático generador de esquemas de Replicación para bases de datos distribuidas*, Tesis de Maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Mor., 2001

[Muthuraj, 93] Muthuraj J., S.Chakravarthy, R. Varadarajan, S.B. Navathe, A Formal Approach to the Vertical Partitioning Problem in Distributed Database Design, *IEEE*, Enero 1993.

[Navathe, 84] Navathe Shamkant, Stefano Ceri, Gio Wiederhold, and Jinglie Dou, Vertical Partitioning Algorithms for Database Design, ACM *Transactions on Database Systems*, Vol. 9, No. 4, december 1984.

[Ozsu, 91] Ozsu M. Tamer y Patrick Valduriez, *Principles of Distributed Database Systems*, Prentice Hall, 1991.

[Ozsu, 10] Ozsu M. Tamer y Patrick Valduriez, *Principles of Distributed Database Systems*, Third Edition, Springer, 2010.

[Osman, 96] Osman I.H. and Kelly, Meta-*Heuristics: Theory and Applications*, J.P. (eds.), Boston USA Ed. Kluwer Academic, (1996).

[Pazos, 99] Pazos Rangel Rodolfo A., Joaquín Pérez O., Guillermo Rodríguez O., Juan Frausto S. y Ana Guadalupe Vélez, Fragmentación Vertical y Reubicación Dinámica en Bases de Datos Distribuidas, *Instituto de Investigaciones Eléctricas, CENIDET- Cuernavaca Morelos, ITESM Campus Morelos*, 1999

[Pazos, 09] Pazos, R.A., Vázquez, G., Pérez, J., Martínez, J.A.: Modeling the Nonlinear Nature of Response Time in the Vertical Fragmentation

Design of Distributed Databases. *Advances in Soft Computing,* Vol. 50 (2009) 605–612.

[Pazos, 10] Pazos, R.A., Vázquez, G., Pérez, J., Martínez, J.A.: Prueba y validación del modelo que involucra el tiempo de respuesta en el diseño de la fragmentación vertical en bases de datos distribuidas. *Advances in Soft Computing,* Vol. 52 (2010).

[Pazos, 14] Pazos, R.A., Vázquez, G., Pérez, J., Martínez, J.A. Gilberto Martínez-Luna: Minimizing roundtrip response time in distributed databases with vertical fragmentation. *Journal of Computational and Applied Mathematics 259,* (2014).

[Place, 74] Placemente of Records in a File and File Allocation in a Computer Network, *Information Processing'74* Stockholm, 1974 pp. 304-307.

[Pérez, 98] Perez Joaquín, Juan Frausto, Guillermo Rodríguez, David Romero, Rodolfo Pazos y Fernando Reyes, Dynamic Allocation of vertical Fragments in distributed databases using the Threshold Accepting Algorithm, *Instituto de Investigaciones Eléctricas, CENIDET - Cuernavaca, ITESM Campus Morelos e Instituto de Matemáticas de la UNAM*.

[Pérez, 99] Pérez, J.: *Integración de la Fragmentación Vertical y Ubicación en el Diseño Adaptativo de Bases de Datos Distribuídas*, tesis de doctorado, Inst. Tecnológico y de Estudios Superiores de Monterrey, Campus Morelos, Cuernavaca, Mor., 1999.

[Pérez, 00] 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, MICAI 2000: *Advances in Artificial Intelligence* (2000).

[Pérez, 01] Pérez O J., R. A. Pazos R., D. Romero, y L. Cruz R., "Predicción del Desempeño de Algoritmos Exactos y Heurísticos: un Enfoque Estadístico", *Memoria del 8vo. Congreso Internacional de Investigación en Ciencias Computacionales*, Inst. Tecnológico de Colima, Colima, México, Nov. 2001, pp. 241-252.

[Pérez, 02] Pérez, J., Pazos, R., Velez, L., Rodríguez, G.: Automatic Generation of Control Parameters for the Threshold Accepting Algorithm. *Lecture Notes in Computer Science,* Vol. 2313 (2002) 125–144.

[Pérez, 03] Pérez, J., Pazos, R., Santaolaya, R. et al.: Data-Object Replication, Distribution, and Mobility in Network Environments. *Lecture Notes in Computer Science* Vol. 2890 (2003) 539–545.

[Reyes, 96] Reyes Ramos Fidel, *Un evaluador en tiempo de respuesta de diseños de bases de datos distribuidos*., 1996, Tesis de maestría en ciencias de la computación de la Benemérita Universidad Autónoma de Puebla, en la Facultad de Ciencias de la Computación.

[Rothnie, 77] Rothnie J. B. and Goodman, N. (1977). A Survey of Research and Development in Distributed Database Management, Information Processing 3rd. *International Conference on Very Large Databases,*Tokio, Japan,1977, pp 48-62.

[Sagols, 92] Sagols Troncoso Feliú, *Introducción a bases de datos centralizadas y distribuidas* 1992, Informe Técnico No. 39, Serie verde, Departamento de Ingeniería Eléctrica, Centro de Investigación y de Estudios Avanzados del IPN.

[Sevinc, 10] E. Sevinc, A. Cosar, Distributed database design with genetic algorithm and relation clustering heuristic, *Lecture Notes in Electrical Engineering* 62 (2010) 133–136.

[Song, 13] S. Song, Design of distributed database systems: an iterative genetic algorithm, *Journal of Intelligent Information Systems* (2013).

[Tambulea, 08] Tambulea L., Horvat-Petrescu, M.: Redistributing Fragments into a Distributed Database. *Int. J. of Computers, Communications & Control*, Vol. 3, No. 4 (2008) 384–394.

[Tamhankar, 98] Tamhankar A.M.; Ram, S.,*Systems, Man and Cybernetics,* Part A, IEEE Transactions on, Volume 28, (1998) Page(s):288 – 305

[Ulus, 03] Ulus T., Uysal, M.: Heuristic Approach to Dynanic Data Allocation in Distributed Database Systems, Pakistan *Journal of Information and Technology* (2003), Asian Netsork for Scientific Information.

[Wolffson, 92] Wolffson O. y S. Jajodia, An Algorithm for Dinámica Data Distribution, Proc. Second *Workshop on the Management of Replicated Data*, pp. 62-65, 1992.

[Xiaolin, 98] Xiaolin D., F.J. Maryanski, Data Allocation in a Dynamically Reconfigurable Environment, Proc. *Fourth International Conference on Data Engineering*, pp. 74-81, 1998.

[Yu, 93] Yu Philip S., Dias, Daniel M. y Lavenberg, Stephen S., Digital Object Identifier On the analytical modeling of database concurrency control. *JACM*, Septiembre de 1993.