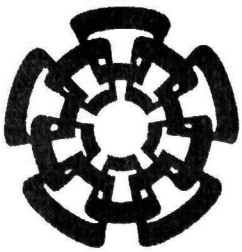
XX C11G551 1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Estudio de la paralelización de algoritmos útiles en minería de datos

Tesis que presenta:
Jorge Lauro Hernández Rojas

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jal., Abril del 2004.

Estudio de la paralelización de algoritmos útiles en minería de datos

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Jorge Lauro Hernández Rojas
Ingeniero en Computación
Universidad de Guadalajara 1994-1998

Becario del CONACyT, expediente no. **129141**

Director de Tesis
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Abril del 2004.

INDICE GENERAL

1 Introducción

La minería de datos es una disciplina de relativamente reciente creación. Tiene sus bases en un grupo de áreas de conocimiento algunas de las cuales tienen un origen no muy alejado en el tiempo como son las redes neuronales y la inteligencia artificial. Sin embargo, la aplicación de todas estas disciplinas está ligada a la realidad que plantea la tecnología de información actual. La minería de datos se encuentra en una etapa de rápido crecimiento. La naturaleza de los algoritmos de minería de datos es compleja y muy demandante de recursos.

Por otro lado, la ejecución de algoritmos en computadoras de múltiples procesadores ha sido una solución – con distintos grados de éxito – a los retos de eficiencia que plantean los algoritmos complejos. Tradicionalmente las computadoras paralelas tienen un alto costo, sin embargo, gracias al incremento en capacidad de las computadoras personales, a su costo asequible y al desarrollo de algunas herramientas que permite coordinar las acciones entre varias computadoras personales; se hace económicamente viable la creación de un sistema paralelo en la modalidad de clusters.

El propósito de este trabajo de tesis es estudiar el comportamiento de algunos algoritmos de minería de datos cuando estos se paralelizan utilizando un cluster de computadoras.

1.1 Motivación y objetivo

Dada la relevancia y potencial de la minería de datos en diferentes campos como son: procesos de investigación, negocios, búsqueda de información en Internet, etc. es importante encontrar formas efectivas y económicamente viables para utilizarla. Sin embargo dado que la minería de datos es multidisciplinaría se requiere trabajo en muchas de sus áreas. Este trabajo está orientado hacia la mejora basada en la incorporación de nuevas técnicas, para implementaciones prácticas. Como se evidenciará más adelante, el alto costo de la minería de datos se debe principalmente:

- Porque generalmente se aplica en bases de datos sumamente grandes, por ejemplo, datos almacenados a través de los años.
- Porque muchos de los algoritmos son intrínsecamente complejos

Este trabajo se enfoca al segundo aspecto y presenta un estudio sobre el comportamiento de algunos de los algoritmos utilizados en la minería de datos cuando son paralelizados para evaluar su viabilidad en aplicaciones reales.

2 Estado del Arte

2.1 Introducción

En este capítulo trataremos el estado del arte de dos ramas de las disciplinas computacionales, la minería de datos y los sistemas computacionales paralelos, ya que sirven como base para el desarrollo de esta tesis.

2.2 Minería de datos

En una sociedad cada vez más tecnificada e intercomunicada, la cantidad de datos generados y almacenados crece a tazas más y más altas. Los medios de almacenamiento masivo de datos han alcanzado un costo por unidad insignificante. Frecuentemente se prefiere aumentar la capacidad del medio de almacenamiento en vez de organizar y desechar datos. A pesar de que los esquemas de bases de datos permiten realizar un almacenamiento de una manera estructurada, estos datos requieren de una interpretación para que se conviertan en información.

El beneficio que ofrecen los sistemas de bases de datos sobre los esquemas anteriores de almacenamiento en papel es la oportunidad de la recuperación de los datos. Sin embargo, cuando se diseña una base de datos, la estructura es definida antes de que se recaben los mismos. En ese sentido, la interpretación que se les dará es definida únicamente conociendo el tipo de los datos en cuestión, pero no los datos en sí. Dos bases de datos, con la misma estructura, pero usadas en empresas diferentes guardan información diferente. Esta información describe de alguna manera las particularidades de la operación de cada empresa, algunas de las cuales pueden ser muy sutiles. Dentro de este escenario, es posible hacer un análisis manual de los datos para descubrir información útil, sin embargo, considerando el crecimiento desmesurado de las bases de datos, esto se vuelve una tarea compleja y basada en la intuición. Con las tasas actuales de crecimiento en la captura de datos, los humanos perdemos rápidamente la capacidad de reconocer patrones y regularidades en ellos.

La *minería de datos* se encarga precisamente de encontrar patrones en conjuntos de datos. Posteriormente es necesario hacer alguna interpretación de estos con el fin de que sean de utilidad. Así, la minería de datos forma parte del proceso de extracción de información.

A pesar de ser una disciplina relativamente nueva, está recibiendo mucha atención debido a las ventajas que puede ofrecer. Una de ellas y la que ha sido uno de sus principales motores son las económicas en el mundo de los negocios. No obstante, su aplicación, no se limita a este rubro ya que sus aplicaciones actualmente abarcan campos como la medicina, la recuperación de información en Internet, la astronomía, la historia, etc.

Ya que la minería de datos es multidisciplinaria, se nutre de tópicos de otras disciplinas, pero más que generar conceptos unificadores, actualmente los aglutina. Podríamos decir que su naturaleza es básicamente aplicativa. Esto no quiere decir que la disciplina esté exenta de generar conocimiento propio, esto se está dando pero todavía es fuerte la corriente de aportaciones que integran nuevos métodos provenientes de otras áreas. Por lo mismo, las técnicas son muy variadas y algunas son más adecuadas que otras dependiendo de la aplicación en particular.

El surgimiento de la minería de datos va a la par del estado actual de la tecnología informática y no se habría dado sin esta. Por un lado la tecnología ofrece alternativas para realizar el trabajo, pero por otro, crea nuevas necesidades.

Los avances en tecnología, a su vez, se ven determinados por los avances científicos, el estado tecnológico previo, y finalmente, la dinámica de los mercados. El avance científico y tecnológico requiere de fuertes sumas de dinero y los beneficios no siempre se traducen en una ganancia monetaria a corto plazo. Este financiamiento proviene de instituciones gubernamentales o privadas. Las exigencias de un mercado pueden determinar el éxito o fracaso de una tecnología.

La penetración de los sistemas informáticos en las empresas es amplia e indispensable. Es un área crítica para el funcionamiento y efectividad de una organización. Anualmente en todo el mundo se hacen fuertes inversiones en este rubro. El fenómeno de la globalización de los mercados no sería

posible sin la informática y las telecomunicaciones. Un mundo como el de los negocios, económicamente poderoso y políticamente influyente puede impulsar de manera importante un área tecnológica o científica, y en este caso, las empresas están en una búsqueda constante de elementos que le permitan obtener ventajas ante sus competidores. La información subyacente en sus datos de operación puede ofrecer las respuestas que están buscando si se diseñan soluciones o estrategias a partir de ellas.

2.2.1 Definiciones y nociones preliminares

Para comenzar a entender la minería de datos, es necesario establecer algunas definiciones y nociones preliminares. Existen varias definiciones de minería de datos. Aunque ninguna de ellas es oficial, todas delinean un contorno sobre los mismos conceptos:

- El proceso iterativo e interactivo de descubrimiento de patrones o modelos válidos, nuevos, útiles y entendibles en bases de datos masivas.
- Búsqueda de información valiosa en grandes volúmenes de datos.
- Exploración y análisis por medios automáticos o semiautomáticos de grandes cantidades de datos con la finalidad de descubrir patrones y reglas útiles [1].

En las dos primeras, se centra la atención sobre el proceso de búsqueda con la finalidad de encontrar patrones. Sin embargo, se deja de lado una parte importante. La utilización de procesos automáticos o semiautomáticos. Esta es una diferencia importante porque el análisis de datos como tal, no es nuevo y se ha realizado desde hace tiempo por otros medios; de manera formal o intuitiva. La diferencia radica en que en la minería de datos se automatiza en cierta medida el proceso, debido a las características del dominio de los problemas a resolver. Adicionalmente se está desarrollando como una actividad independiente y multidisciplinaria.

El término cluster engloba la noción de grupo. En computación puede tener diferentes significados dependiendo del contexto. Nos interesan dos acepciones:

Cluster de computadoras. Es un conjunto de computadoras interconectadas y con los recursos de software necesarios para realizar procesamiento en conjunto. Cada computadora recibe el nombre de nodo. El objetivo es tener una capacidad de cómputo combinada mayor que la de los nodos individuales. Existen varios tipos de clusters, existen servidores Web y de base de datos, estos permiten repartir la carga entre los diferentes nodos. El que nos interesa a nosotros es un cluster que permite hacer cómputo paralelo.

Clustering es un problema tipo de minería de datos. El objetivo es identificar grupos de datos. Los datos pertenecientes al mismo cluster deben ser semejantes entre si a la vez que distintos a los datos que pertenecen a otros clusters. Se trata de identificar "acumulaciones" de datos, por ejemplo, si los datos fueran las posiciones de las estrellas dadas por sus coordenadas cartesianas, entonces las galaxias serían clusters de estrellas.

Data warehouse. Es un sistema que almacena datos de manera estructurada, cuyo propósito es servir como apoyo a un proceso de toma de decisiones [3]. Tradicionalmente los datos se encuentran fragmentados en bases de datos diseñadas e implementadas en distintos formatos y medios, inclusive en formatos no digitales como el papel. Un sistema de data warehouse recopila datos de muy diversas fuentes en un formato común con definiciones consistentes para llaves y campos. Esto, aparte de darnos una visión consistente y relativamente completa de las funciones de una empresa, nos permite realizar operaciones intensivas de análisis. Es posible también, almacenar información resumida o histórica en un data warehouse. En general no es posible, y ciertamente no es recomendable, realizar operaciones de entrada y salida intensivas directamente en un sistema en producción del que depende un negocio. Los sistemas tradicionales de bases de datos almacenan datos en un formato diseñado para optimizar el desempeño de las tareas operacionales. Este formato generalmente no es adecuado para actividades de soporte de toma de decisiones. Por ejemplo, en un sistema data warehouse, las operaciones de escritura son relativamente poco frecuentes. La información es agregada periódicamente y por lotes. Sin embargo, las operaciones de lectura son intensivas. El sistema de base de datos puede estar optimizado para esto.

2.2.2 Ejemplos de Aplicación de la Minería de Datos

La minería de datos puede aplicarse a problemas de muy diversas áreas. No siempre es fácil crearse una imagen mental del tipo de cosas que la minería de datos puede realizar. La expresión de "identificación de patrones útiles o significativos" es un tanto ambigua por que el término "útil" no tiene un alcance bien definido y depende del ámbito del que se esté hablando. Debido a esto mostramos algunos ejemplos de aplicación. Este ejercicio nos sirve para delimitar el alcance de la minería de datos y por lo tanto entender lo que es y lo que no.

Segmentación o clustering de clientes

La segmentación de clientes, puede resultar de utilidad en las actividades de marketing de las empresas. La segmentación consiste en tipificar los clientes potenciales de los que se tiene información. Esto se hace mediante la detección de conjuntos de clientes con características comunes en una base de datos.

En una campaña publicitaria que utiliza un medio como el correo directo[1], donde cada mensaje enviado al público implica un costo; en la mayoría de los casos resulta incosteable enviar correspondencia a todos los posibles destinatarios. El objetivo ideal es minimizar la cantidad de mensajes que se envían y maximizar el número de respuestas positivas a la campaña. Para acercarnos al objetivo, se realiza una segmentación del universo de clientes apoyándonos en medios automáticos. Los datos a considerar normalmente son demográficos y socio-económicos, pero también puede considerarse la respuesta que hayan tenido los clientes en campañas anteriores. Todo depende de la información disponible. Finalmente se lanza la publicidad sobre el segmento al que el producto está dirigido.

La segmentación también facilita el diseño de una campaña publicitaria en medios masivos de información. El conocer las características de los clientes por un lado, permite desarrollar los spots publicitarios pensando en un público en particular. También ayuda a elegir el mejor horario o a que programa de televisión o radio asociarlo. Esta información también puede ser útil en el diseño de la distribución de mercancías en una tienda de autoservicio.

[1] Con correo directo nos referimos al envío personalizado de propaganda por el servicio de correo convencional.

La industria de la publicidad y el marketing, antes de la llegada de la minería de datos, han utilizado mecanismos semejantes a los descritos. Pero la segmentación ya está predefinida, solo es cuestión de obtener de la base de datos los elementos que cumplen con cada categoría. En cambio la minería de datos ayuda a los especialistas a definir los segmentos basándose en los datos mismos, brindando una perspectiva fresca con respecto a las segmentaciones tradicionalmente preconcebidas. Así pues, el especialista puede experimentar de manera más fina, con parámetros que normalmente no se considerarían en le momento de realizar la segmentación.

En un hospital, existen historiales clínicos que pueden ayudar a detectar patrones que originan enfermedades, por ejemplo en el caso de la mujeres en gestación existen estudios que prueban las condiciones que originan una diabetes definitiva por el proceso de_gestación y falta de una alimentación adecuada.

Aplicaciones en CRM

CRM es un acrónimo de "Costumer Relationship Management". La minería de datos tiene mucho que aportar en este rubro. La intuición detrás de CRM es la siguiente: En la relación comercial que una empresa muy pequeña tiene con sus clientes, se tienen una interesante ventaja sobre las empresas grandes. La empresa pequeña puede llevar un registro implícito de toda la relación comercial con sus clientes. Las preferencias y los hábitos de consumo del cliente son considerados. En caso de algún problema con el producto o servicio, la atención es personalizada y se le da un seguimiento detallado [2]. En cambio, en una empresa grande, lo más común es que cada interacción con un cliente, no está relacionada con las anteriores. La razón es clara, conforme el volumen de operaciones crece, nadie puede recordar los detalles de todas las transacciones y los clientes. CRM pretende acercar a las empresas grandes al funcionamiento de las empresas pequeñas en cuanto al cuidado de la relación con los clientes.

En el registro de las compras que hace una persona hay mucha información. Si cada compra está asociada a su correspondiente comprador, entonces lo que tenemos es un historial de transacciones de cada cliente que nos revelan sus hábitos de consumo. Esto es útil para realizar publicidad personalizada, considerando que actualmente existen medios adecuados para esto como lo es el

correo electrónico, teléfono, etc. Por otro lado, también es factible detectar tendencias si analizamos los hábitos de consumo de grupos de clientes. Por ejemplo se pueden identificar los productos que tengan una tendencia significativa a venderse juntos. Esta asociación puede ser en la misma compra y se puede incentivar la compra del producto complementario ofreciéndolo directamente o colocándolo a la vista de los clientes. Esta información también sirve para evitar que los productos asociados tengan rebajas de precio simultáneamente. La asociación de productos puede ocurrir en transacciones separadas y entonces se realiza un esfuerzo de publicidad personalizada por segmentos de clientes que tengan las mismas características. Por ejemplo, es relativamente sencillo determinar cuando un comprador está adquiriendo consistentemente productos para bebe y entonces se pueden ofrecer productos de la misma línea con una buena probabilidad de respuesta positiva. Más aún, eventualmente se puede predecir en cuanto tiempo aproximadamente se necesitarán artículos en función de los patrones de compra de los clientes o el tiempo de duración de los artículos.

Esta información también ayuda a diseñar mejor una campaña publicitaria, identificando las características más efectivas según el público al que se quiere llegar.

Evaluación de clientes

En los mercados donde los productos se adquieren de manera periódica, en general se desea que si un cliente compró una vez, lo haga de nuevo. Para esto tradicionalmente se recurre a promociones tales como descuentos y obsequios. Sin embargo, esto no siempre produce el efecto deseado. Esto puede deberse a que existen clientes con diferentes niveles de beneficio a mediano o largo plazo para la empresa, por ejemplo, habrá quienes solo serán fieles al proveedor mientras existan las promociones y resultará ser un cliente caro y poco conveniente. Tampoco resulta del todo conveniente hacer obsequios a clientes que de cualquier forma seguirán adquiriendo productos.

Existen mercados en los que un proveedor de bienes o servicios no se diferencia significativamente de otro, lo que ocasiona que el cliente no tenga una razón en particular para quedarse con un proveedor. El tránsito de clientes entre este tipo de compañías es intenso. Esto sucede por ejemplo entre empresas de telefonía celular y bancos.

La minería de datos puede aplicarse en este caso tratando de identificar quienes son los clientes más convenientes o quienes podrían estar a punto de cambiar a la competencia para concentrar en ellos esfuerzos y recursos.

En los bancos, una cancelación de cuenta en ocasiones es antecedida por ciertos movimientos típicos que hacen prever la pérdida de ese cliente. La detección de un comportamiento semejante puede ayudar a diseñar estrategias y ofrecer soluciones al cliente a fin de retenerlo. La minería de datos ayuda a identificar el comportamiento de los clientes perdidos. Una vez hecho esto, puede encontrar a los clientes con comportamientos semejantes.

Análisis de hábitos de compra

En una tienda departamental, los consumidores tienen hábitos de compra tales como la adquisición de productos en asociación con otros. Si esta es una tendencia presente en una parte significativa de la gente de una región, es posible acentuarla al redistribuir los anaqueles donde se muestran los productos para que los artículos que se compran juntos, también se encuentren físicamente juntos.

En una cadena de establecimientos de venta al menudeo, la minería de datos puede ayudar a elaborar un plan de distribución de mercancías procurando, en lo posible, el entregar a una determinada sucursal solo la mercancía que se desplaza, y así evitar el transportar de nuevo mercancía hacia plazas en donde sí puedan ser vendidas.

Detección de Fraudes

En los registros de actividades financieras y comerciales diarias, es posible detectar comportamientos anómalos que sugieren la posible comisión de delitos. En particular veamos algunos ejemplos.

Detección de evasiones fiscales

Debido a los volúmenes de datos que se generan de las declaraciones fiscales de un país, resulta muy difícil revisar con detenimiento las actividades de empresas o particulares. También es complicado y caro analizar la correspondencia entre las actividades fiscales de varias empresas

interrelacionadas comercialmente. La minería de datos puede ayudar a detectar comportamientos irregulares. Habrá que notar que la minería de datos no puede ser utilizada como elemento probatorio. No todos los resultados que nos arroja son correctos y por lo tanto requieren la validación de un experto. Sin embargo, son útiles en tanto que reducen el esfuerzo para detectar posibles casos fraudulentos. Los expertos concentrarán su atención en casos con una alta probabilidad de ser irregulares. Naturalmente se corre el riesgo de que nunca se detecten ciertos casos de interés, pero eso también sucede con el procedimiento manual. La ventaja de la minería de datos radica en la eficacia al trabajar con volúmenes de datos muy grandes. La cantidad de casos detectados es mayor y el costo es menor aunque los algoritmos sean menos flexibles y los criterios de búsqueda sean más rígidos que los que utiliza un especialista. Este es un caso en donde, aunque idealmente nos gustaría encontrar todas las instancias posibles, con obtener un buen número de ellas es suficiente. En la práctica lo más probable es que no se tengan los recursos para atender todos los casos. Es un fenómeno que no se puede resolver completamente, lo que interesa es simplemente controlarlo.

Detección de fraudes en registros bancarios

Es posible tipificar las secuencias de transacciones bancarias que caracterizan un fraude al alimentar un proceso de minería de datos con las transacciones comprobadamente fraudulentas. Posteriormente, se ejecutan procesos de manera rutinaria con el fin de detectar transacciones nuevas que correspondan con los patrones encontrados y por lo tanto puedan ser indicio de una actividad ilegal. La aplicación de estas estrategias depende de la legislación de cada país.

Detección de irregularidades en el consumo medido de servicios

En el caso de la industria eléctrica, existen varias formas de evadir el pago correcto por el consumo realizado. Sin embargo pueden detectarse cuentas con comportamientos sospechosos en el historial de facturación. De igual manera que en los casos anteriores, el espacio de búsqueda es demasiado grande para hacer un análisis manual.

Cruzamiento de bases de datos

En un proyecto de minería de datos, ocasionalmente se realiza una combinación de bases de datos. Esto ofrece una nueva perspectiva y es posible descubrir información que no existe en las bases de datos por separado. En este caso del tipo "el todo es mayor que la suma de las partes"

Estos son algunos de los problemas que pueden ser resueltos con minería de datos. Como podemos notar, se aplican en muy diversas áreas y sus beneficios son notables. Sin embargo, la aplicación de los métodos de minería de datos, no es un proceso automático. Requiere de un experto con conocimiento tanto en el área de minería de datos como en el campo de aplicación específica. Esto permite que los algoritmos se apliquen de manera adecuada y que la interpretación que se le da a los resultados del proceso de minería de datos sea más provechosa. Además, el experto debe guiar a los algoritmos para que el proceso se conduzca por un camino que lleve a un buen resultado. Ante todo el hacer minería de datos es realizar un proceso de búsqueda dirigido por un analista de modo que las herramientas computacionales hagan más manejable la complejidad, pero el factor humano sigue y seguirá teniendo un peso decisivo en la disciplina.

2.2.3 Proceso de minería de datos

En minería de datos existen varias maneras de resolver un problema. De entre ellas, no existe una que sea la mejor en general. La mejor solución dependerá del problema en particular. En muchas ocasiones es necesario realizar un tratamiento a los datos antes de ser procesados. Por ejemplo las bases de datos suelen tener datos de tipo cualitativo y cuantitativo mientras que los algoritmos normalmente funcionan con parámetros cuantitativos. En estos casos es necesario crear una representación cuantitativa apropiada de los datos cualitativos. Otro caso típico es cuando las bases de datos contienen inconsistencias o les faltan datos. Se deben ponderar estas contingencias en la determinación de los resultados.

La minería de datos es un proceso de búsqueda por aproximaciones sucesivas, y en ocasiones por prueba y error. Los resultados de una técnica pueden servir de entrada para otra o para la misma pero en un nivel distinto. También puede ser necesario ejecutar varias veces el mismo algoritmo. Los algoritmos de minería de datos suelen exigir muchos recursos computacionales. Esto aunado al

hecho de que los datos requieren frecuentemente un preprocesamiento, ocasiona que no se recomiende realizar minería directamente en las bases de datos de producción.

De un modo muy general, podríamos resumir los pasos generales para realizar minería de datos de la siguiente manera

- *Plan de búsqueda.* Se debe crear un plan inicial con el fin de definir que tipo de patrones se desean encontrar, esto establece el alcance del proceso. Al definir que tipo de resultados esperamos se puede determinar también que datos son necesarios de entrada.
- *Recopilación de información.* En últimas fechas, se han difundido los sistemas de tipo data warehouse, en el que se crean sistemas de manejo y recuperación de datos totales, que abarcan uniforme y coherentemente todos los datos de la empresa. Si no se tienen los datos de esta forma, es necesario realizar un proceso de homogeneización y captura de datos.
- *Selección del modelo.* Una vez definidos los tipos de patrones que se necesitan o las preguntas a realizar, se selecciona el o los modelos, algoritmos y técnicas que prometan responder al las necesidades planteadas.
- *Selección y limpieza de los datos.* Cuando se buscan patrones o se busca la respuesta a una pregunta en minería de datos, se deben seleccionar los datos relevantes. Además, en ocasiones es necesario completar datos y cruzar bases de datos externas a fin de tener un espacio de búsqueda con atributos significativos. Una vez que se ha hecho una selección, quizá no se tomen en cuenta la totalidad de los datos disponibles a la vez. No se desea alentar el proceso utilizando datos no relevantes para el tipo de preguntas que se realizan. También se debe eliminar información no válida, o que pueda conducir a errores.
- *Preprocesamiento y transformación.* También sucede que algunos algoritmos trabajan con atributos en un formato que no es manejable directamente, por lo que se requiere un preprocesamiento o transformación, por ejemplo, algunos algoritmos funcionan con argumentos cuantitativos, mientras que los datos pueden ser cualitativos. Se deben representar de una forma que sea congruente.
- *Reducción.* Suele ser necesario una etapa en la que los datos se acondicionan para ser manipulados por los sistemas de bases de datos de una manera eficiente, por ejemplo, se puede obtener una tabla a partir de otras para un manejo más conveniente. Estas reducciones son con

fines de mejorar el desempeño y obedecen a limitaciones y criterios técnicos de la tecnología particular con la que se lleva a cabo la minería.

- *Búsqueda de patrones.* Aquí se ejecutan propiamente los algoritmos de minería de datos. Es probable que se aplique más de uno.
- *Evaluación e interpretación.* Después de obtener los resultados arrojados por el proceso de minería, es necesario evaluar e interpretar los resultados, con el fin de determinar que parte de estos es útil, si alguna lo es. Con esto se establece si se debe continuar con el proceso de minería, si se debe reformular el planteamiento o bien, si se da por concluido el proceso.
- *Consolidación y uso del conocimiento extraído.* Finalmente, se deben sacar conclusiones y aplicar las respuestas obtenidas a una estrategia o una decisión apoyada en la minería de datos. La minería de datos no necesariamente da soluciones directas a un problema, pero si da información que puede ser útil para elaborar una solución con base en la nueva información disponible.

Al confrontar las soluciones generadas contra la realidad, es probable que obtengamos más datos que a su vez puedan contribuir a una nueva instancia de un proceso de minería de datos. Esto inclusive se puede extender en varios ciclos, cada uno de los cuales es más refinado que el anterior. No es conveniente utilizar a la minería de datos como una herramienta para validar una hipótesis. La minería de datos debe ser vista como un proveedor de sugerencias, que indica posibles caminos a seguir, pero será en definitiva el analista quien decida sobre el medio para fundamentar la hipótesis o dar una explicación a los resultados.

Es importante destacar que al aplicar una técnica de minería de datos en la solución de un problema, no siempre se obtiene una solución óptima. La naturaleza de los problemas que interesa abordar es típicamente muy compleja. Por lo que lo que se busca es una solución lo suficientemente buena para fines prácticos.

Como hemos mencionado, el crecimiento de la minería de datos ha sido en buena medida por el interés que ha despertado en la comunidad de negocios. Sin embargo, en el calor del entusiasmo, también se han generado falsas expectativas de los beneficios que esta puede aportar. Naturalmente, la minería de datos, no ofrece beneficios instantáneos ni soluciones mágicas. No hay

recetas ni métodos que se apliquen en todas las empresas por igual. Es un proceso que se desarrolla y ajusta para cada empresa o aplicación. Por lo mismo, es un proceso viable, pero no barato.

Es importante mencionar que la aplicación de minería de datos en ciertos ámbitos tiene implicaciones éticas y legales. En este sentido, no todo lo que es técnicamente viable debe ser implantado. Toda ciencia o disciplina tecnológica tiene un compromiso social en alguna medida, pero la minería de datos es especialmente sensible en este rubro porque se trabaja con información que puede ser confidencial. En algunos casos la información es obtenida en el entendido de que se le dará un uso acotado y no debe ser utilizada en aplicaciones que puedan infringir leyes o violentar los derechos de las personas. Se debe respetar, por ejemplo, guardas legales como es el secreto bancario y se debe evitar realizar algún tipo de discriminación moralmente cuestionable con base en la información aportada por la minería de datos. No olvidemos que la minería de datos por sobre todas las cosas clasifica y a nosotros nos interesa crear clases para calificar y discriminar en distintos niveles y ámbitos. El profesional en esta área debe ser receptivo a estas consideraciones y tiene la obligación de respetar y hacer respetar las leyes.

2.2.4 Técnicas usadas en minería de datos

Desde antes de que se comenzara a registrar la historia, el ser humano ha analizado datos y buscado patrones en ellos. Es la forma más natural en que empezamos a entender un fenómeno. Primero se observa y se registran atributos del mismo buscando inconscientemente regularidades que nos permitan describirlo mejor y predecir su comportamiento aún sin tener una explicación completa de su funcionamiento. Este es un mecanismo de razonamiento por experiencia. Es anterior al método científico y por supuesto es más propenso a fallas que este. En la historia se han dado muchas explicaciones erróneas de fenómenos, sin embargo todos los pueblos del mundo, por más primitivos que sean han sabido sacar provecho de este mecanismo. Vemos que en todo el mundo se han encontrado métodos de caza y agricultura efectivos; han creado medicinas tradicionales sorprendentemente útiles y por supuesto han creado una explicación de su mundo mediante religiones, mezclando conocimiento útil con explicaciones místicas. Con el advenimiento de la

ciencia, lo que ha cambiado es la manera en que observamos el mundo, registramos los datos y los interpretamos.

Cualquier tipo de ciencia comprende un cúmulo de técnicas para encontrar patrones y explicaciones. En ese sentido la minería de datos puede ser de ayuda en muchas de ellas. Más recientemente la informática ha cambiado drásticamente la manera y eficiencia en que registramos y recuperamos los datos.

Los métodos de minería pueden ser clasificados en dos tipos. Los descriptivos y los predictivos. El objetivo de los primeros es facilitar el entendimiento de los datos, explicarlos o hacer una tarea de descubrimiento de conocimiento. Esto implica encontrar patrones que describen los datos. Por otro lado, los métodos predictivos, son capaces de clasificar un dato nuevo, sin existencia previa en las bases de datos.
También pueden ser clasificados como dirigidos y no dirigidos. Los primeros se caracterizan por la presencia de un único campo objetivo cuyo valor será predicho en términos de otros campos en la base de datos. En los algoritmos no dirigidos, no hay campo objetivo. El algoritmo simplemente se ejecuta sobre los datos en espera de que eventualmente descubrirá una estructura significativa.
Las técnicas utilizadas por la minería de datos provienen de áreas como la inteligencia artificial, las redes neuronales, la teoría de probabilidad y la estadística entre otras.

Detección de clusters

La detección de clusters es la construcción de modelos que encuentran registros de datos que son similares a otros. Estos cúmulos de datos similares son llamados clusters. Es un problema de minería de datos no dirigida ya que, el objetivo es encontrar similitudes previamente desconocidas en los datos. Existen varias técnicas para encontrar clusters.

En aplicaciones de marketing, estos datos podrían ser atributos demográficos como edad, rango de poder adquisitivo, propiedades inmuebles, compras anteriores etc.

Existe una noción gráfica de clustering que ayuda a entender el concepto de semejanza en los datos. Esta es solo efectiva cuando los datos cuentan con dos o tres atributos ya que se hace una

representación gráfica en dos o tres dimensiones. Si los atributos son representados como valores en un sistema de coordenadas cartesianas y la semejanza está dada por la distancia euclidiana entre la posición de los elementos, entonces los clusters visualmente se perciben como aglomeraciones de puntos. La intuición de este esquema es que los elementos cercanos entre si, en general se comportarán de manera semejante en un cierto escenario.

La detección de clusters en dos y tres dimensiones puede ser trivial para el ojo humano y seguramente que un procedimiento manual ofrecerá mejores resultados que un procedimiento automático. La ventaja de la minería de datos es que nos permite extender esta noción más allá de las tres dimensiones y con una gran cantidad de datos.
La detección de Clusters es una técnica apropiada para iniciar casi cualquier análisis de datos. La configuración de clusters puede mostrar indicios de la forma que tienen los datos y sugerir formas de utilización.

Árboles de decisión e inducción de reglas

Los árboles de decisión son un tipo de modelos producidos por una variedad de técnicas. Los árboles de decisión son utilizados para realizar minería de datos dirigida, en particular, para clasificación. Se dividen los registros del conjunto de entrenamiento en subconjuntos disjuntos cada uno de los cuales es descrito por una regla.

Una de las ventajas de los árboles de decisión es que, el modelo es susceptible de ser explicado ya que toma la forma de reglas explícitas. Esto permite que los resultados puedan ser evaluados, identificando atributos importantes en el proceso. Esto también es útil cuando los datos de entrada son de dudosa calidad. Resultados espurios son obvios con reglas explícitas. Las reglas por si mismas pueden ser fácilmente representadas como enunciados lógicos o en lenguajes como SQL de manera que puedan ser aplicadas directamente sobre nuevos registros.

Redes neuronales artificiales

Las redes neuronales artificiales son un modelo matemático *inspirado* en el funcionamiento y estructura del sistema nervioso de los seres vivos. La unidad básica en este esquema es la neurona

artificial. Las redes neuronales artificiales no son ni pretenden ser una réplica exacta de las redes neuronales naturales. Aún no se conoce a cabalidad el funcionamiento de las neuronas orgánicas. Adicionalmente, las neuronas artificiales tienen características que se han agregado por conveniencia y no están presentes en las neuronas biológicas.

Una de las principales ventajas de las redes neuronales es su amplia aplicabilidad, sin embargo, tienen dos fuertes desventajas. La primera es la dificultad que se tiene para explicar los modelos que genera. La segunda es su sensibilidad hacia el formato de los datos de entrada. Diferentes representaciones de datos pueden producir diferentes resultados. Debido a esto, el acondicionamiento de los datos es una parte muy significativa del trabajo que se debe realizar al utilizar esta técnica.

Dentro de las características generales de las redes neuronales tenemos:

- *Pueden reconocer patrones, clasificar datos y generalizar.* Después de una fase de entrenamiento donde la red neuronal se ajusta, esta es capaz abstraer regularidades de los datos y clasificar datos nuevos, es decir, no pertenecientes al conjunto de datos de entrenamiento original.
- *Pueden manejar datos imprecisos y con ruido.* Frecuentemente los datos que es preciso manejar contienen variables no deseadas y aún así, normalmente puede identificar tendencias.
- *Son generales.* En principio, una red neuronal puede calcular cualquier función computable.
- *Una vez entrenadas pueden operar rápidamente.* El proceso de entrenamiento es iterativo y suele ser lento. Sin embargo, una vez entrenadas, procesan la información rápidamente.
- *Son naturalmente paralelas*. Pueden ser implementadas en plataformas paralelas, mejorando su desempeño. Las redes neuronales son ejemplo del procesamiento distribuido. El conocimiento esta también distribuido. Una red neuronal diseñada en hardware seria un dispositivo en el que cada neurona podría trabajar en paralelo con las demás. En la práctica, lo común es hacer implementaciones secuenciales por la disponibilidad del equipo existente. Distribuir una red neuronal en una computadora multiprocesador es entonces natural.
- *Son tolerantes a fallas.* El conocimiento abstraído en una red neuronal tiene un grado de redundancia, lo cual tiene como efecto secundario, que estos sistemas sean tolerantes a fallas.

- *Mejora en tiempo de desarrollo*. El tiempo de desarrollo con esta técnica puede ser muy corto y en algunos casos no es necesario programar. Por un lado, existen modelos en los que esta demostrado el tipo de información que pueden procesar, y por otro, el desarrollador no tiene que programar el comportamiento del sistema ya que este lo adquiere mediante un algoritmo de aprendizaje ya establecido. Además existen herramientas que facilitan la implementación de redes neuronales de varias formas:

  Es posible separar las etapas de entrenamiento y ejecución de una red neuronal. Si así lo permite la aplicación, se puede diseñar y entrenar la red en una herramienta. Así el programador solo tiene que implementar la ejecución de la red y no los algoritmos de entrenamiento, los cuales son considerablemente más complicados.

  - Otras herramientas generan código que implementa la red por completo.
  - En ocasiones no es necesario desarrollar una aplicación autónoma y todo el proceso se desarrolla en una herramienta existente.

Una red neuronal, ver figura 2.1, está compuesta de elementos simples operando en paralelo llamados *neuronas*. Las neuronas están interconectadas entre si haciendo las veces de conexiones sinápticas del cerebro. Las neuronas tienen *entradas*, que son los puntos por donde la neurona capta los datos que procesará. Estos datos pueden provenir del exterior o de otras neuronas. El efecto que tienen las entradas en la respuesta de la neurona se ve afectado por un grupo de elementos llamados *pesos*. Es en estos elementos donde se almacena el conocimiento. El valor que tendrán los pesos será determinado por un algoritmo de aprendizaje que mediante aproximaciones sucesivas los ajusta hasta producir una *salida* satisfactoria. En cada neurona existe una función que determina el *umbral de disparo* de la neurona, es decir, indica que valor de entrada producirá una salida. En el caso de las redes neuronales supervisadas, la salida se *compara* con un valor *objetivo* que es el valor de salida al que se quiere llegar.

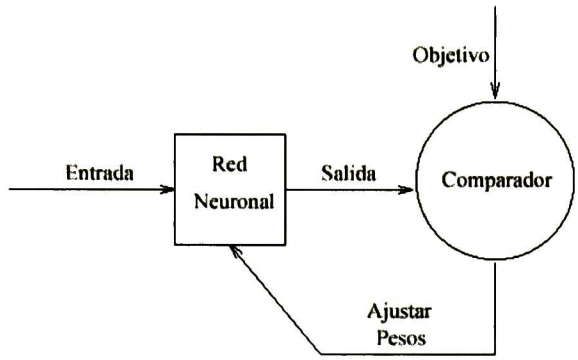


Figura 2.1. Proceso de entrenamiento de una red neuronal. La *salida* se compara con un valor de referencia u *objetivo* y se ajustan los *pesos* hasta producir la salida

En la figura 2.2 vemos una neurona. Consiste de un conjunto de valores de entrada $E = \{E_1, E_2,...,E_n\}$, un conjunto de pesos $P = \{P_1,P_2,...,P_n\}$, una función f de umbral de disparo y una salida S. Una vez entrenada, la neurona procesa los datos como la suma de los productos de cada entrada con su respectivo peso. Al resultado se le aplica la función y con esto se obtiene la salida.

$$S = f\left(\sum_{i=1}^{n}(E_i P_i)\right)$$

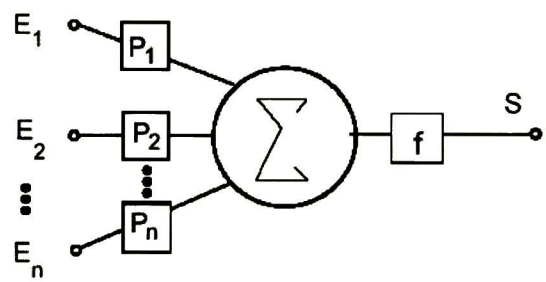


Figura 2.2. Neurona Artificial. Es el elemento básico de procesamiento en una red neuronal.

Análisis de secuencias

El análisis de secuencias consiste en seguir las relaciones entre registros para desarrollar modelos que las describan. Esta es una aplicación de la teoría de grafos. Un área de aplicación del análisis de secuencias son las telecomunicaciones y los motores de búsqueda del WWW.

Algoritmos genéticos

En los algoritmos genéticos, se aplican los mecanismos de genética y selección natural para realizar un proceso de búsqueda de valores óptimos. Esta es una técnica predictiva. En este caso, la forma de la solución debe ser conocida con antelación. Las soluciones se codifican en cadenas llamadas *genomas*. Los algoritmos genéticos utilizan las operaciones de selección, cruza y mutación para evolucionar generaciones sucesivas de posibles soluciones. Existe además una función de evaluación de los genomas para determinar que tan buena solución representa un genoma dado. Esta función nos permite comprar la calidad de las posibles soluciones permitiendo así a la operación de selección determinar que solo las soluciones de mejor calidad trasciendan de generación en generación. La operación de cruza permite que las soluciones combinen sus características generando a su vez soluciones válidas. Con estos mecanismos se pretende obtener generaciones con soluciones cada vez mejores. Lo que hace a los algoritmos genéticos diferenciarse de técnicas del descenso del gradiente es la operación de mutación. Su función es evitar que el algoritmo converja demasiado rápido atascándose en un mínimo local en vez de alcanzar el mínimo global, agrega variedad a las posibles soluciones y también ocasiona que la ejecución reiterada de este algoritmo con los mismos parámetros nos pueda dar soluciones distintas.

Razonamiento basado en memoria

El razonamiento basado en memoria es una técnica que utiliza instancias conocidas como un modelo para hacer predicciones acerca de instancias desconocidas. Esta técnica busca a los elementos más parecidos a la nueva instancia y determina la solución más común a ellos. Esta misma solución es asociada al elemento nuevo, obteniendo no solo una solución sino que el nuevo elemento puede pasar a formar parte del conjunto de elementos conocidos.

Se tiene un conjunto de eventos u objetos con atributos definidos[1]. Un evento puede pensarse como un suceso que desencadenará una reacción o nos llevará a tomar una decisión, a esta

consecuencia la llamaremos solución. El resolver un problema nuevo es encontrar la reacción más adecuada[2] mediante la combinación[3] de las soluciones de los eventos más parecidos, la semejanza se determina mediante una función de distancia. Comúnmente se utiliza la distancia euclidiana. El conjunto de los eventos parecidos a un evento problema se llama vecindario. Una vez encontrada la solución se procede con una de las dos opciones siguientes:

- Si el sistema es supervisado, entonces se requiere una validación de la correctud de la solución antes de agregarla al conjunto de eventos que constituyen la experiencia.
- Si el sistema es no supervisado, entonces se integrará a la experiencia inmediatamente.

Un evento conocido es aquel que forma parte de la experiencia y por lo tanto se sabe qué solución tiene asociada. En este método, se debe conocer de antemano la forma que tendrá la solución.

Una de las principales ventajas del razonamiento basado en memoria es su adaptabilidad a casi cualquier tipo de dato. Los dos elementos principales en esta técnica son la función de distancia utilizada para encontrar a los vecinos más cercanos y la función de combinación que combina valores al vecino más cercano para hacer una predicción.

Esta técnica tiene su origen en el área de inteligencia artificial y trata de imitar el proceso de toma de decisiones basándose en experiencias anteriores. Consiste en identificar casos similares de experiencias previas a fin de aplicar la información de esos casos para resolver un problema nuevo.

Probabilidad y estadística. Desde el principio estas nos han acercado a un mejor entendimiento de los conjuntos grandes de datos y al manejo de la incertidumbre. Varias de las técnicas aquí referidas tienen su fundamento conceptual en estas disciplinas. Sin embargo, se han dado casos en los que el origen histórico de algunas técnicas proviene inicialmente de otras áreas [19].

---

[1] Los atributos frecuentemente son una lista de características con un valor asociado a cada uno de ellos
[2] La reacción adecuada dependerá de la aplicación en particular
[3] El método de combinación depende de la aplicación en particular

Herramientas Existentes

Existen herramientas de varios tipos y enfocadas a distintas especialidades, por ejemplo:

- Clementine. Es un toolkit dos veces ganador del premio "SMART award for innovation" otorgado por el departamento de comercio e industria del Reino Unido. Las aplicaciones de Clementine incluyen segmentación y tipificación de clientes, detección de fraudes y predicción de utilidades entre otras.
- Trajecta. Utiliza redes neuronales para ofrecer servicios de predicción del comportamiento de clientes y tendencias de mercados.
- Nuggets. Utiliza algoritmos de búsqueda propietarios llamados SiftAgents(TM) para desarrollar reglas del tipo si- entonces. Estos algoritmos utilizan métodos genéticos y técnicas de aprendizaje para buscar hipótesis válidas con las cuales formular reglas. Los criterios con los que se define la validez de las reglas son establecidos por los usuarios. Nuggets también provee un conjunto de herramientas para utilizar las reglas en predicción con datos nuevos, clasificación y segmentación de datos.

Las posibles áreas de aplicación así como las técnicas utilizadas en minería de datos son muy variadas. Mientras algunas herramientas utilizan redes neuronales para crear modelos predictivos otros utilizan técnicas probabilísticas. Otros más utilizan árboles de decisión para crear reglas. En los algoritmos en los que trabajamos se utilizan técnicas de optimización global estocástica. La utilización de un enfoque u otro depende del caso en particular de aplicación.

A diferencia de los proyectos mencionados, este trabajo utiliza dos algoritmos de clustering originales de Trejos Murillo y Piza de la universidad de Costa Rica [15]. Exploramos la paralelización de estos algoritmos específicamente y no pretendemos, de momento, que el resultado sea una herramienta de uso general.

2.3 Paralelismo

El propósito principal del procesamiento paralelo es realizar cómputos más rápidos de los que pueden ser hechos con un solo procesador mediante la utilización de un cierto número de

procesadores trabajando concurrentemente. La necesidad de soluciones a problemas complejos que involucran grandes cantidades de información es evidente en diferentes áreas. Estas incluyen dinámica de fluidos, predicción del clima, modelado y simulación de sistemas grandes. Cabe también mencionar que el paralelismo está también presente en sistemas que utilizamos todos los días y han sido absorbidos por las computadoras personales comunes. Este es el caso del *pipeline* y las interrupciones [9].

Las computadoras paralelas están constituidas por una colección de procesadores, típicamente del mismo tipo, interconectadas de algún modo para permitir la coordinación de actividades y el intercambio de datos. Existen muy diversos tipos de computadoras paralelas y tienen diferencias fundamentales más allá de la velocidad y el tamaño. Se diferencian por su modelo de programación, la topología de interconexión de los procesadores y su organización en general. Existen diferentes tipos de paralelismo y las computadoras están diseñadas para explotar algunos en especial.

2.3.1 Tipos de paralelismo

El aprovechamiento del paralelismo se puede dar en distintos niveles y diversas formas. Encontramos las siguientes:

- *Procesadores especializados.* Es posible diseñar y utilizar procesadores de propósito específico que se encargarán de una tarea particular, liberando de carga al procesador central. En este segmento tenemos los coprocesadores gráficos, matemáticos o de compresión de vídeo.
- *Pipeline.* Es la superposición de las fases de ejecución de series de instrucciones a nivel del ciclo de fetch. Hablando del nivel más bajo de ejecución de un programa, tenemos el juego de instrucciones de un procesador. La ejecución de una instrucción se descompone en varias fases: *Fetch*, *Addressing*, *execute* and *write back.* Cada una de estas etapas pueden ser realizadas por unidades de ejecución distintas y comenzar a ejecutar una instrucción antes de que se termine la anterior. De este modo se da uso prácticamente todo el tiempo a cada sección de la unidad de ejecución.

- *A nivel de instrucción de alto nivel.* Es posible paralelizar instrucciones que son independientes entre si. Dos instrucciones son independientes si la ejecución de una no requiere de la ejecución previa de la otra. Se acostumbra indicar explícitamente bloques de código que pueden ser ejecutados de este modo. Típicamente se utilizan instrucciones del tipo PARBEGIN y PAREND, la primera para definir el inicio del bloque de código paralelo y la segunda como punto final, donde se sincronizan todos los procesadores. Si alguno termina antes que los demás, esperará a que todos terminen.
- *Aritmética.* Es sencillo determinar las relaciones de dependencia que existen en la evaluación de una expresión aritmética, por lo que con frecuencia pueden ser paralelizadas. Por ejemplo el calculo de una sumatoria puede ser realizado si cada procesador toma dos a dos los números de la serie obteniendo una siguiente serie que puede ser procesada de la misma manera hasta tener el resultado total.
- *A nivel de procedimiento.* En este caso se distribuyen los procedimientos independientes entre los distintos procesadores para su ejecución en paralelo. En este esquema se asume que existe un espacio de memoria compartido entre los microprocesadores para realizar la comunicación.
- *A nivel de programa.* La granularidad de este nivel de paralelización es el más grueso. Aquí se distribuyen programas completos entre los distintos procesadores. La comunicación entre los procesadores puede ser mediante el paso de mensajes.

La programación paralela es de más bajo nivel que la programación secuencial, ya que se requiere un nivel de descripción mayor en un programa para lograr un fin deseado. Se debe tener conocimiento de la arquitectura de hardware subyacente para el diseño y la implementación de los programas. La implementación de un algoritmo depende del tipo de paralelismo que se explote.

2.3.2 Comunicación y sincronía entre elementos de procesamiento paralelo.

Debido a que, en un programa paralelo, los procesadores cooperan para llevar a cabo una tarea en común, es necesario que tengan mecanismos de comunicación para sincronizar sus actividades, así como para compartir datos. Estos mecanismos difieren entre los modelos y tipos de computadoras paralelas que existen:

- *Señales*. Es una forma de recibir notificaciones de eventos asíncronos de software o de hardware. En este rubro caen las interrupciones. Es la forma en que se comunican los diferentes elementos procesadores que existen en una computadora convencional.
- *Memoria Compartida*. Es un espacio de direcciones de memoria que es común a dos o más procesadores. En este espacio, los procesadores pueden acceder y modificar los datos de forma transparente. Se deben tomar providencias especiales para sincronizar el acceso y evitar inconsistencia de datos.
- *Semáforos*. Es un elemento de sincronización por lo que también es un mecanismo de comunicación. Son elementos implementados en software o hardware que permiten llevar el control de uso de recursos compartidos por varios procesadores o procesos.
- *Barreras*. Instrucciones de sincronización que sirven para asegurar que todos los procesos de un grupo estén listos antes de que cualquiera de ellos continúe con su proceso.
- *Mensajes*. Los mensajes entre los procesadores pueden transportar señales de sincronización o datos.
- *Archivos*. Puede haber comunicación entre procesos mediante archivos. Para evitar conflictos de acceso se establecen bloqueos.

2.3.3 Clasificación de computadoras paralelas

Las computadoras paralelas pueden clasificarse de acuerdo a varias características arquitecturales y modos de operación.

- *El tipo y número de procesadores*. La mayoría de las computadoras paralelas cuentan con un cierto número de procesadores idénticos. La principal ventaja de este esquema es que los tiempos de respuesta de cada uno son similares disminuyendo los tiempos muertos que podrían generarse al momento de sincronizar actividades.

  Otro tipo de computadoras, como la que se utiliza en este trabajo de tesis, es con procesadores heterogéneos. Es posible aminorar el problema mencionado anteriormente mediante técnicas de programación que asignen dinámicamente la carga de trabajo. Sin embargo esto aumenta la complejidad del código.

- *Por su modelo de programación.* Hay una variedad de esquemas de programación además del imperativo. Existe el modelo de programación vectorial y el declarativo.
- *La interconexión entre procesadores.* Debido a que la programación paralela frecuentemente considera la arquitectura subyacente, la forma de interconexión entre los procesadores es un factor de clasificación importante. Hay esquemas de programación totalmente orientado a la disposición número e interconexión de los procesadores[6]

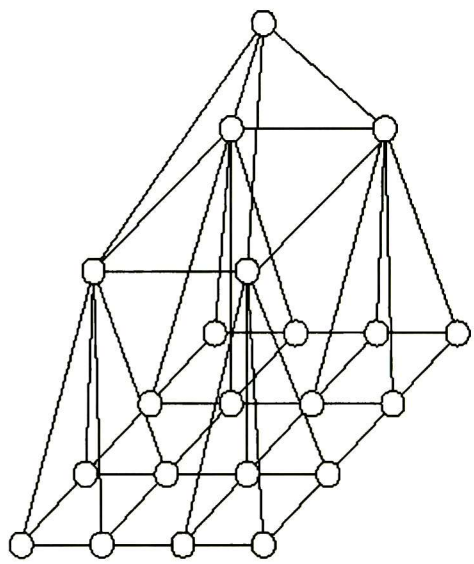


Figura 2.5. Pirámide

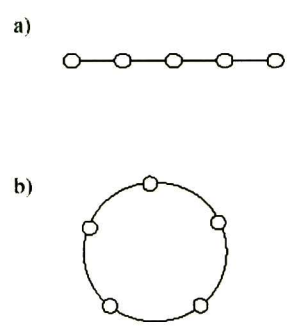


Figura 2.4. Arreglo lineal y anillo

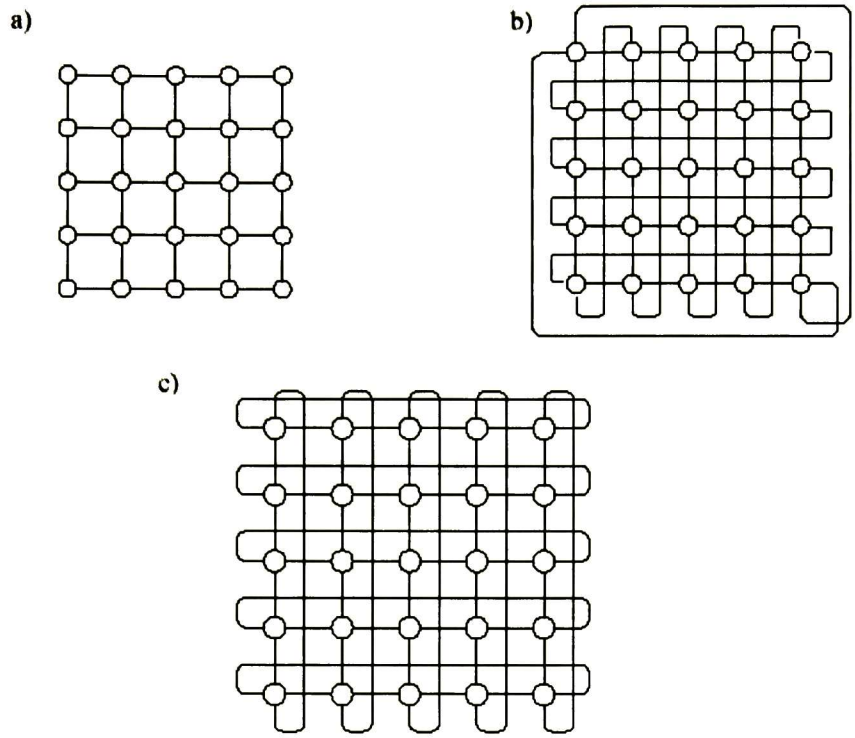


Figura 2.3 Retículas

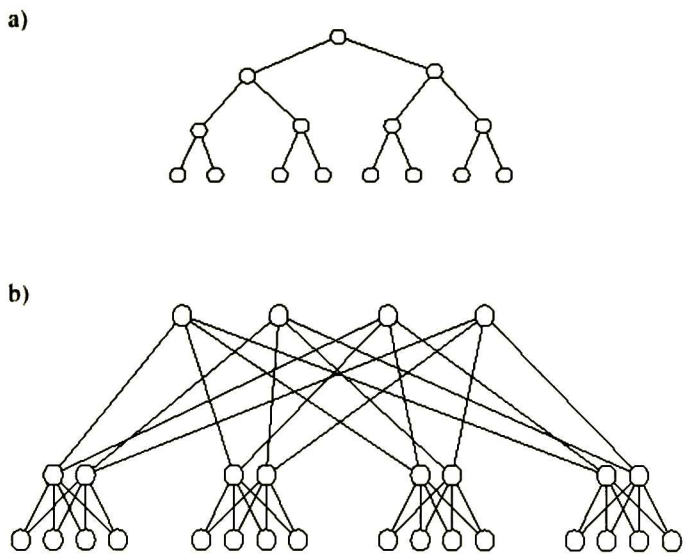


Figura 2.6. Árboles

Hay varios aspectos en la organización de los procesadores que determinan las bondades y desventajas de una topología particular. Si la representamos mediante un grafo donde los nodos son los procesadores y los arcos son los enlaces de comunicación entre los procesadores tenemos:

- Diámetro. Es la distancia máxima entre dos procesadores, en términos de la cantidad de arcos que los separan. Es deseable minimizar tanto como sea posible el diámetro para asegurar que la información fluya entre los procesadores tan rápido como sea posible.
- Bisección. Es la cantidad de enlaces entre procesadores, necesarios para dividir al grafo en dos. Resulta una desventaja tener una bisección pequeña, debido a que estas sen posibles cuellos de botella en la comunicación.
- Número máximo de conexiones por procesador. Debe ser en lo posible la misma en todos para evitar que el programador tenga que preocuparse en diferenciar las operaciones de comuncación dependiendo de la posición del procesador en particular dentro de la topología. Además una diferencia en el número de conexiones puede conducir a una diferencia de desempeño entre los mismos lo que genera tiempos muertos.

  Longitud máxima de un arco. Determina la latencia de comunicación entre dos procesadores adyacentes. Entre menor sea en valor se obtiene un desempeño mejor.

Existen métodos bien definidos de ruteo de mensajes en las distintas topologías, lo que hace más sencilla la implementación de operaciones de comunicación frecuentemente utilizadas como boradcasts, y reducciones. Inclusive es posible simular ciertas topologías en términos de otras. Esto mejora la portabilidad de algunos algoritmos a otras plataformas. Los esquemas comunicación y control general. Los esquemas de comunicación definen el paradigma de programación.

Actualmente, la densidad de integración de los circuitos integrados ha permitido tener sistemas muy complejos y potentes en un espacio muy reducido. El mercado de las microcomputadoras ha hecho esta tecnología estándar y ha puesto estos dispositivos al alcance de todos. El avance en el desempeño de los procesadores es tan alto que sobrepasan por mucho a los procesadores individuales de las computadoras paralelas que hay en el mercado. La tecnología de redes no es

ajena a este fenómeno y tiene un comportamiento semejante. El resultado es que hay una tendencia a hacer cómputo paralelo con estas herramientas estándares [7]. Las computadoras ensambladas bajo este esquema tienen ventajas y desventajas:

Ventajas:

- Bajo costo de compra de hardware
- Posibilidad de reutilizar equipo en desuso
- Es posible integrar más elementos procesadores en cualquier momento
- Muy bajo costo de herramientas de software
- Posibilidad de integrar equipos nativamente paralelos
- Posibilidad de aprovechar ventajas específicas de una arquitectura
- Permiten un modelo de programación más general

Desventajas:

- Menos eficientes que las computadoras nativamente paralelas
- La comunicación entre los procesadores es un cuello de botella
- Los esquemas de programación que aprovechan topologías particulares no siempre son aplicables
- Debido a la relativa lentitud de la comunicación de los procesadores el modelo de paralelización necesariamente es de grano grueso.

Con estos sistemas, la cantidad de operaciones por segundo es muy alta. Existen equipos desarrollados que sobrepasan por mucho la capacidad en Gigaflops de las computadoras nativamente paralelas, sin embargo la comunicación no es tan eficiente como en las primeras. Sin embargo suficiente para muchas aplicaciones útiles.

Entre las aplicaciones que se le han dado a este tipo de computadoras estan:

- Simulación de impactos de automóviles (DaimlerChrisler)

- Creación de animaciones para cine (DreamWorks, Pixar)
- Simulación de mecánica de fluidos (English Airforce)
- Simulación de la dinámica de las galaxias (Observatorio de Génova)
- Modelado de los efectos de sismos en el suelo (Universidad Clarkson)
- Simulación de dinámica molecular (Universidad de Southampton EU)

Este es el tipo de sistema que desde el principio se utiliza en este trabajo de tesis por lo que nos concentraremos en su modelo de programación y no abundaremos en los detalles de los demás tipos de computadoras paralelas.

2.3.4 Modelos de programación paralela aplicables a clusters de computadoras

En la programación paralela nos interesa que los distintos procesadores efectúen conjuntamente un trabajo. Esto nos permite, en general, una disminución del tiempo de ejecución al repartir la carga de trabajo. Un programa que consume muchos recursos o es muy complejo algorítmicamente o trabaja con una gran cantidad de datos. La carga de trabajo de un programa es entonces, de control o de datos y esta carga de trabajo es justamente lo que podemos repartir entre los procesadores [9].

Cuando se paraleliza el control, la algorítmica del programa se divide en tareas independientes que cooperan para realizar una tarea. Cuando lo que se paraleliza son los datos, típicamente se ejecutan varias instancias del mismo programa cada uno de los cuales procesa una parte de los datos.

Como habíamos mencionado anteriormente hay varias formas de comunicación entre los procesadores. En un ambiente de clusters de computadoras la comunicación es mediante el paso de mensajes. En este esquema cada procesador tiene su propio espacio de memoria que no puede ser visto directamente por los demás procesadores. La comunicación se establece mediante mensajes. Las primitivas de comunicación tienen la forma *send(procesadores_destino, mensaje)* y *receive()*.

La primera envía un mensaje a los procesadores especificados. La segunda espera recibir un mensaje.

Como mencionamos anteriormente, la granularidad de la programación de los clusters de computadoras solo puede ser a nivel de proceso. La frecuencia de comunicación crece conforme la programación disminuye de granularidad. Por ejemplo, si paralelizamos el cálculo de una ecuación, se necesitaría un mensaje de red por cada operador lo cual es inaceptable en este esquema ya que hablamos de procesadores potentes y comunicaciones lentas comparativamente, sin embargo, en una aplicación como la renderización de una secuencia de vídeo generado por computadora, la comunicación es relativamente poca, pero el trabajo que tiene que hacer cada procesador es alto.

2.3.5 Conclusiones del capítulo

El trabajo en minería de datos es muy amplio y sigue alimentándose de otras disciplinas que tienen vida independiente y que tienen propósitos afines, como lo es la inteligencia artificial y las redes neuronales.

3 Herramientas Teórico-Prácticas

En esta sección se presentan las diversas herramientas que se utilizaron. Estas son tanto herramientas de implementación como herramientas teóricas.

3.1 Plataforma de Paralelización

Uno de los intereses en esta tesis fue la utilización de una plataforma de paralelización económica. Esto debido a que las computadoras nativamente paralelas tienen costos relativamente altos.

La plataforma de paralelización que se utiliza en este trabajo de tesis es llamada PVM. Se trata de una herramienta que permite agrupar un conjunto de computadoras (cluster) posiblemente de distintas características de modo que funcionen como una sola computadora paralela. Es el resultado del Heterogeneous Network Computing research project auspiciado de manera compartida por el Oak Ridge National Laboratory, la Universidad de Tennessee, la Universidad Emory y la Universidad Carnegie Mellon. PVM se distribuye gratuitamente y es utilizada en una variedad de aplicaciones computacionales por todo el mundo. PVM es capaz de correr en una amplia variedad de plataformas de cómputo, entre las que se encuentran las siguientes:

Las computadoras paralelas más poderosas del mundo son las de tipo MPP (Massively Parallel Processors) en donde se concentran un gran número de procesadores en un solo espacio interconectadas con esquemas muy sofisticados, desarrollados expresamente para disminuir el costo en comunicación. PVM no puede aspirar a competir con estos sistemas, sin embargo, es posible obtener resultados satisfactorios, sobre todo en el aspecto económico, con equipo ya existente y comunicado mediante una red. Más aún, es capaz de conectar varios equipos del tipo MPP, permitiendo utilizar su poder de manera combinada.

Alliant FX/8	Data general Aviion
DEC Alpha	Encore 88000
Sequent Balance	HP-9000 model 300
BBN Butterfly TC2000	HP-9000 PA-RISC
IBM 370 (AIX)	Intel iPSC /860
Thinking Machines CM2 y CM5	Intel iPSC /2 386 host
Convex C-Series IEEE f.p.	Kendall Square KSR-1
C-90 YMP T3D	x86 corriendo Linux
Cray-2, Cray-S-MP	Maspar
SPARCstation (SunOS 4.2)	Sequent Symetry
SPARC multiprocessor (shrd mem)	SG IRIS (IRIX4.x y, IRIX5.x)
MIPS4680	NeXT
DECstation 3100,5100	IBM/RS6000
DEC Micro VAX	SGI multiprocessor
Stardent Titan	x86 UNIX (BSDI, 386BSD, NetBSD)
IBM RT	Sun3 (SunOs 4.2), Sun4
Intel Paragon	

En una MPP, cada procesador es exactamente como cualquier otro en capacidad, recursos, software y velocidad de comunicación. No es el caso de una red. Las computadoras disponibles en una red pueden ser de diferentes plataformas o tener diferentes compiladores y diferentes formatos de representación de los datos. En general, en una red encontramos varios niveles de heterogeneidad:

- En arquitectura. El conjunto de computadoras disponibles puede incluir un amplio rango de tipos de arquitecturas, como PC convencionales, estaciones de trabajo, multiprocesadores de memoria compartida, supercomputadoras vectoriales, y MPPs. Cada arquitectura tiene su propio método óptimo de programación. Pueden existir formatos binarios incompatibles por lo que se requiere compilar un programa para cada máquina diferente.
- En el formato de datos. Los formatos de datos en computadoras distintas son frecuentemente incompatibles. Esta incompatibilidad es un punto importante en computación distribuida porque los datos enviados desde una computadora pueden ser ilegibles en la computadora

receptora. Un esquema de paso de mensajes desarrollado para un ambiente de cómputo heterogéneo debe asegurarse que todas las computadoras entienden los datos intercambiados.

- En la velocidad computacional. Aún en el caso de que todas las computadoras que componen a la máquina virtual sean del mismo formato de datos, es posible que tengan distintas velocidades de procesamiento.

Sin embargo, una máquina virtual tiene varias ventajas:

- El utilizar hardware existente permite que el costo de un cómputo sea bajo.
- El desempeño puede ser optimizado asignando a cada tarea individual la arquitectura más apropiada.
- Es posible explotar la naturaleza heterogénea del cómputo, por ejemplo, se puede tener acceso a bases de datos distintas para aquellas partes de la aplicación que pueden correr únicamente en cierta plataforma.
- Los recursos de la computadora virtual pueden crecer en etapas y tomar ventaja de las últimas tecnologías de cómputo y de redes.
- El desarrollo de programas puede ser facilitado utilizando un ambiente familiar. Los programadores pueden usar aplicaciones familiares disponibles en las máquinas individuales, tales como compiladores, editores, depuradores.

Dentro de las características particulares de PVM tenemos:

- Grupos de anfitriones configurados por el usuario. Una aplicación paralela se divide en tareas. Estas se ejecutan en un conjunto de máquinas o anfitriones que son seleccionados por el usuario para una corrida en particular de un programa en PVM. El grupo de anfitriones puede ser alterado agregando y suprimiendo máquinas durante la corrida. Esta es una característica para la tolerancia a fallas.
- Acceso transparente al hardware. Los programas de aplicación pueden ver el ambiente de hardware como una colección de elementos virtuales de procesamiento sin atributos particulares, dependientes de una arquitectura. Por otro lado, si se desea, se pueden explotar las capacidades de máquinas específicas en el grupo asignando ciertas tareas en las computadoras más apropiadas.

- Cómputo basado en procesos. La unidad de paralelismo en PVM es una tarea, un hilo de control secuencial que alterna entre comunicación y control. Varias tareas pueden ejecutarse en un solo procesador.
- Modelo de paso de mensajes explícitos. Las tareas que componen una aplicación se coordinan y comparten datos enviando y recibiendo explícitamente mensajes entre ellos. El tamaño de los mensajes está solamente limitado por la cantidad de la memoria disponible.
- Soporte de heterogeneidad. PVM soporta heterogeneidad en términos de máquinas y redes. Con respecto al paso de mensajes, PVM permite mensajes que contienen más de un tipo de datos que pueden ser intercambiados por máquinas con diferentes representaciones de datos.

- Soporte para computadoras multiprocesador. PVM utiliza las facilidades para paso de mensajes nativas en multiprocesadores para aprovechar el hardware disponible. Algunos fabricantes proveen sus propias versiones optimizadas de PVM, que, no obstante, pueden comunicarse con la versión pública.

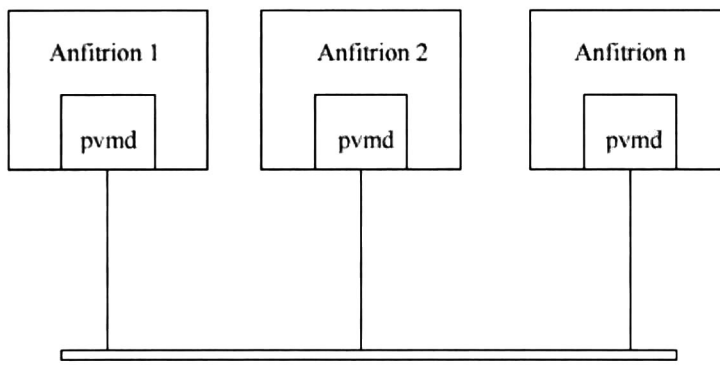


Figura 3.1 Cada anfitrión de PVM contiene un daemon, este se encarga de la administración de las tareas asignadas a este.

PVM esta compuesto de dos partes. La primera es un *daemon*, llamado pvmd, que reside en todas las computadoras que conforman una máquina virtual. La segunda parte del sistema es una biblioteca de rutinas de interfaz de PVM. Contiene un repertorio funcionalmente completo de

primitivas que se necesitan para cooperación entre tareas de una aplicación. Esta biblioteca contiene rutinas para paso de mensajes, creación de procesos, coordinación de tareas y modificación la máquina virtual.

Modelo computacional. El modelo computacional de PVM se basa en la noción de que una aplicación consiste de varias tareas. Cada tarea es responsable de por una parte de la carga de trabajo computacional de la aplicación. Esta carga de trabajo puede ser control o datos. De ahí que en PVM se puedan crear programas con el paradigma de *paralelismo de control* o funcional y *paralelismo de datos.*

Interfaz de programación. PVM actualmente soporta los lenguajes de programación C, C++ y Fortran. Todas las tareas de PVM están identificadas por un número entero llamado *task identifier* (TID). Los mensajes son enviados y recibidos utilizando estos TID's. Debido a que los TID's deben ser únicos en toda la máquina virtual, estos son asignados por el pvmd y no son escogidos por el usuario. PVM contiene varias rutinas que regresan TID's de modo que las aplicaciones puedan identificar otras tareas en el sistema.

Facilidades para manejo de grupos. PVM soporta también un conjunto de operaciones de grupos definidos por el usuario. Cuando una tarea se une agrega a un grupo, le es asignado un número de instancia único en el ámbito del grupo. Los números de instancia se numeran de 0 en adelante.

Una tarea puede entrar o salir de un grupo en cualquier momento y sin tener que informar a ninguna otra tarea. Los grupos se pueden superponer de modo que una tarea puede pertenecer a más de un grupo. También las tareas pueden enviar mensajes en *broadcast* a grupos a los que no pertenecen.

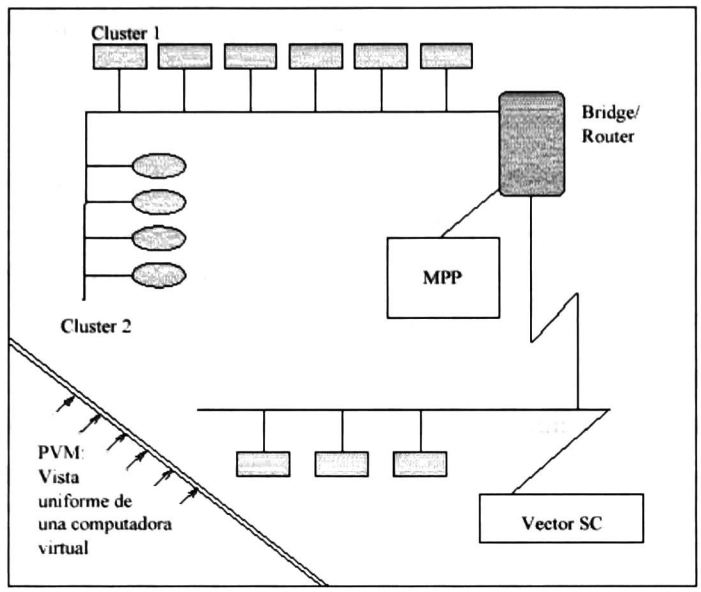


Figura 3.2 Arquitectura de PVM

Un programa en PVM tiene una forma general que es como sigue: Un usuario escribe uno o más programas secuenciales en C, C++, o Fortran 77 que contiene llamadas a la biblioteca de PVM. Cada programa corresponde a una tarea de la aplicación. Estos programas son compilados para cada arquitectura en la máquina virtual, y los archivos objeto resultantes son puestos en un lugar accesible para otros hosts en la máquina virtual. Para ejecutar una aplicación, un usuario típicamente inicia una copia de una tarea. Usualmente la tarea *maestra.* Este proceso inicia otras tareas de PVM, resultando eventualmente en una colección de tareas activas que se ejecutan e intercambian mensajes con otras para resolver el problema. Debe notarse que aunque, lo mencionado arriba es un escenario típico, pueden ser iniciadas manualmente tantas tareas como sea necesario.

3.2 Metodología de Paralelización

Existen varias técnicas y metodologías abocadas a la paralelización de algoritmos. Pero no es posible utilizarlas en todos los casos. Hay algunas mas apropiadas que otras según la aplicación.

Existen métodos de paralelización automática, estos se basan en un análisis de dependencias, desafortunadamente no se eligieron por dos razones:

- La literatura no los recomienda en general debido a su poca eficiencia. Los programas resultantes suelen ser más lentos que sus contrapartes hechas manualmente. Pueden ser utilizados de forma práctica por ejemplo en compiladores paralelos especializados como *High Performance Fortran* y *C**. Estos son diseñados para correr en arquitecturas nativamente paralelas lo cual no va de acuerdo con los alcances de esta tesis.
- Estas paralelizaciones se utilizan generalmente en los ciclos por lo que no son aplicables en la plataforma de paralelización seleccionada. Recordemos que en PVM las unidades de paralelización son tareas o procesos.

Sin embargo, la creación de un programa paralelo a nivel de procesos no tiene por que ser una actividad totalmente libre y realizada sin ninguna guía. Se pueden seguir lineamientos generales que ayudan a obtener, en muchos casos, un buen grado de paralelización. Decidimos trabajar con la metodología mostrada a continuación [10]. En esta, el proceso de paralelización se divide en cuatro etapas:

- Descomposición. Se divide el control o los datos al máximo de tareas que sean mutuamente independientes, es decir, se encuentra la máxima paralelización. Desafortunadamente no siempre la máxima paralelización logra los mejores resultados, porque normalmente el costo en comunicaciones se vuelve muy alto. Sin embargo, este ejercicio ayuda a visualizar el potencial de paralelización del programa.
- Agrupamiento. Se reagrupa datos o control, de modo que exista un compromiso entre paralelización y costo de comunicación. Se debe determinar la mayor velocidad de ejecución. En este punto, el programa normalmente es independiente de la plataforma todavía.
- Orquestación. Se hace un análisis tomando en cuenta la arquitectura, el modelo de programación y el lenguaje de programación. Se toman decisiones acerca de la organización de las estructuras de datos y la calendarización de tareas.

- Mapeo. Cada proceso se asigna a un procesador de manera que se trata de maximizar la utilización de los procesadores y los costos de comunicación y sincronización

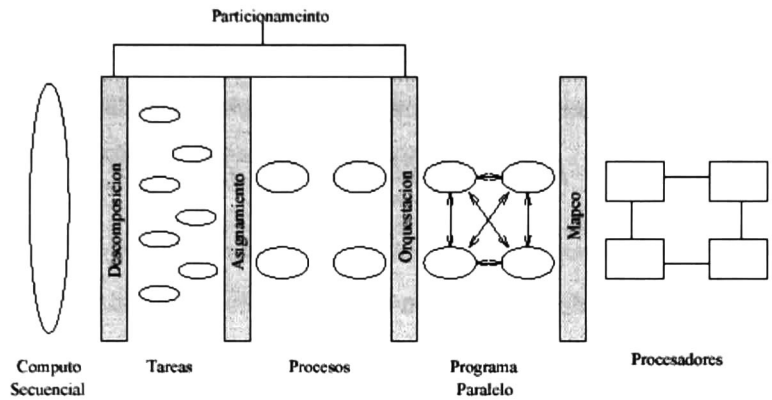


Figura 3.3 Pasos en la creación de un programa paralelo

3.3 Evaluación de desempeño de un algoritmo paralelo

Sea P un problema computacional dado y sea n la cantidad de datos de entrada. Denotamos la complejidad secuencial de P como $T^*(n)$. Esto es, existe un algoritmo secuencial que resuelve P dentro de ese tiempo, además, es posible probar que no existe un algoritmo secuencial que resuelva a P más rápido. Sea A un algoritmo paralelo que resuelve P en tiempo $T_p(n)$ en una computadora paralela con p procesadores. Entonces, la *aceleración* alcanzada por A se define [6]:

$$S_p(n) = \frac{T^*(n)}{T_{p(n)}} \qquad (3.1)$$

$S_p(n)$ mide el factor de aceleración obtenido por el algoritmo A cuando se dispone de p procesadores. Como $S_p(n) \leq p$ es deseable diseñar algoritmos en donde $S_p(n) \approx$ p tanto como sea posible. En realidad, existen varios factores que introducen ineficiencias. Estas incluyen, retardos introducidos por la comunicación y tiempo invertido en la sincronización de las actividades de los procesadores.

Notese que $T_1(n)$, el tiempo de ejecución del algoritmo paralelo A cuando el número de los p procesadores es igual a 1, no es necesariamente el mismo que $T^*(n)$. Comúnmente, cuando la complejidad del problema no se conoce, se reemplaza $T^*(n)$ por el tiempo del mejor algoritmo *conocido*.

Otra medida de desempeño de un algoritmo paralelo A es la *eficiencia*, definida por

$$E_p(n) = \frac{T_1(n)}{pT_p(n)} \qquad (3.2)$$

Esta medida provee un indicio de la utilización efectiva de los p procesadores relativa al algoritmo dado. Un valor de $E_p(n)$ aproximadamente igual a 1, para algún p, indica que el algoritmo A corre aproximadamente p veces más rápido utilizando p procesadores de lo que lo hace con un procesador. Luego cada procesador está realizando un "trabajo útil" la mayor parte del tiempo dedicado a resolver el algoritmo A.

Existe un límite de tiempo, denotado por $T_\infty(n)$, más allá del cual el algoritmo no puede correr más rápido, no importa cuantos procesadores se utilicen. Ya que $T_p(n) \geq T_\infty(n)$, para cualquier valor de p, y la eficiencia $E_p(n)$ satisface $E_p(n) \leq T_1(n) / pT_\infty(n)$. Es por esto que la eficiencia de un algoritmo decae rápidamente conforme crece p más allá de $T_1(n) / pT_\infty(n)$.

Este es el esquema que utilizaremos para medir el desempeño de los algoritmos paralelizados.

4 Algoritmos

Como se vio anteriormente, existen muchas y muy diversas técnicas aplicadas a la minería de datos, algunas de ellas son áreas de estudio completas, por lo que, la elección de los algoritmos podría implicar una revisión profunda de todas las disciplinas lo cual queda fuera de los alcances de esta tesis por razones de tiempo y concordancia con los objetivos.

En este punto nos apoyamos en el trabajo de Javier Trejos, Eduardo Piza y Alex Murillo de la Universidad de Costa Rica [15]. Ellos trabajan sobre algunos métodos de optimización global estocástica aplicados al problema de clustering. Se han tomado dos de sus métodos:

Simulated annealing aplicado al clustering, y *Búsqueda tabú aplicada al clustering.* Además, como punto de referencia, agregamos un algoritmo clásico de clustering, el *k-means[1].* Es necesario presentar algunas bases para poder mostrar sus métodos.

4.1 El Problema de clustering

En nuestro caso trabajamos con el problema de *clustering.* De manera informal podemos decir que dado un conjunto de datos, el objetivo es encontrar los subconjuntos que agrupen a los elementos que más "se parecen entre si" A este tipo especial de conjuntos les llamaremos *clusters.* El parecido lo determina un criterio de similitud. Frecuentemente el criterio de similitud es ser la distancia euclidiana entre los elementos representados en un sistema cartesiano. Así pues, entre más pequeña sea la distancia entre dos elementos, se considera que su grado de similitud es mayor.
Resaltemos algunos de los conceptos que son importantes en las siguientes discusiones:

> **Partición.** Un conjunto S de elementos se divide en k subconjuntos $A_1, \ldots, A_k$ tal que $A_i \cap A_j = \emptyset$ y $A_1 \cup \ldots \cup A_k = S$. A este arreglo de elementos se le llama partición o configuración. Existen k particiones distintas para un conjunto dado, donde k es la cantidad de clusters y n es la cantidad de elementos. En la figura 4.1 se observan todas las particiones posibles para $n = 4$ y $k = 2$. Puede observarse que algunas agrupaciones de

elementos se repiten, solo cambia el nombre del cluster al que pertenecen. Sin embargo, las consideramos como particiones distintas. Así, en la misma figura tenemos que las particiones de la *a* a la *h* son semejantes al rango de la *i* a la *j* respectivamente.

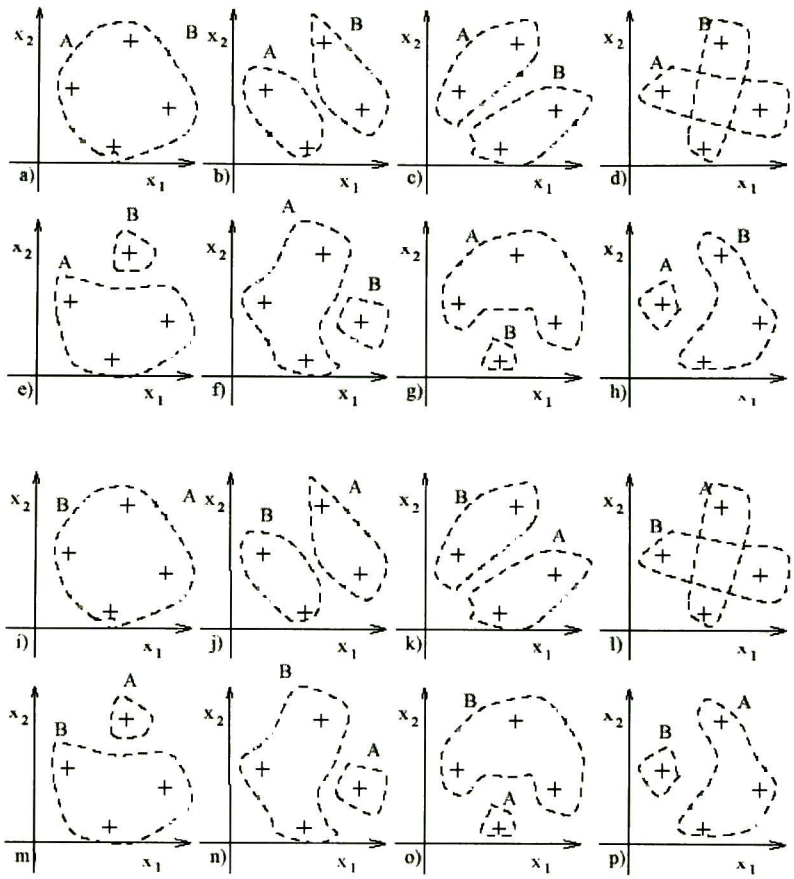


Figura 4.1 Todas las posibles particiones para un conjunto de $n = 4$ y $k = 2$. Se incluye el caso en la que un cluster tiene todos los elementos y el otro ninguno.

Cluster. Es cada uno de los subconjuntos que permiten identificar elementos que tienen alguna característica similar y permite conocer mejor el conjunto de datos. Para poder determinar el grado de similitud, los atributos que definen a los elementos deben ser cuantitativos, o en su defecto, deben ser una codificación cuantitativa de una característica cualitativa. Un elemento solo puede pertenecer a un cluster a la vez y necesariamente cualquier elemento pertenece a un cluster. Esto último implica que la técnica de clustering,

tal como se define aquí no es útil para descartar elementos atípicos. Los elementos atípicos serán incluidos en algún cluster aunque no sean bien representados por este.

Distancia. Con ella se determina que tan distintos son los elementos entre sí. En nuestro caso, la distancia siempre será euclidiana. Para las p - tuplas ordenadas $a = (a_1,a_2,\ldots,a_p)$, $b = (b_1,b_2,\ldots,b_p)$, la distancia $d : \Re^p \times \Re^p \mapsto \Re$ se define como

$$d(a,b) = \sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\ldots+(a_p-b_p)^2} \qquad (4.1)$$

Inercia. Es una medida que nos permite establecer que tan bien elegidos están los clusters. Entre menor sea la inercia, la partición asociada estará mejor dividida.

Centroide. Una vez determinado un cluster, puede encontrarse su centroide ubicando el punto en el espacio $\Re^p$ "más céntrico" dentro del mismo cluster obteniendo una medida de resumen representativa. Dado un cluster $C = \{x_1,x_2,\ldots,x_n\}$ con n elementos. Donde cada elemento x_i es una p - tupla ordenada $x_i = \{x_{i,1},x_{i,2},\ldots,x_{i,p}\}$. El centroide g es el número en $\Re^p$ siguiente:

$$g(C) = \left(\frac{\sum_{i=1}^{n} x_{i,0}}{n}, \frac{\sum_{i=1}^{n} x_{i,1}}{n}, \ldots, \frac{\sum_{i=1}^{n} x_{i,p}}{n} \right) \qquad (4.2)$$

Más formalemente, el problema del particionamiento lo podemos expresar así [15]: Dado un conjunto $\Omega = \{x_1,x_2,\ldots,x_n\}$ en $\Re^p$, con pesos ϖ_i, se busca una partición $P = (C_1, \ldots, C_k)$ de Ω en k clases, tal que la inercia

$$\varpi(P) = \sum_{l=1}^{k} \sum_{x_i \in C_l} w_i d^2(x_i, g_l) \qquad (4.3)$$

sea mínima, donde g_l es el centroide de C_l y d es la distancia euclidiana en $\Re^p$

Es importante hacer notar que desde la definición no se garantiza el óptimo particionamiento debido a que el número de particiones k está predefinido. Así, por ejemplo, una posible partición con $k = 3$ sería como en la figura 4.2. Con $k = 2$ tendríamos un resultado semejante al de la figura 4.3. Es evidente que la primera solución es mejor, sin embargo, en estos algoritmos, el número de

clusters no es una variable a determinar. Los tres algoritmos que se presentan son aproximados, aunque no siempre se obtiene la solución óptima, se logra por lo general una partición lo suficientemente buena para un fin en particular.

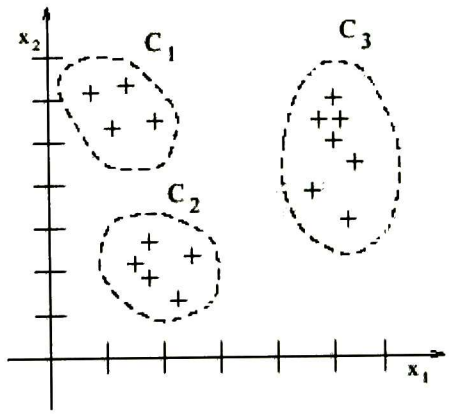


Figura 4.2 Partición constituida por los clusters Formula C_1,C_2,C_3. Los elementos más cercanos entre si pertenecen a la mismo cluster. En este caso, $k = 3$, la inercia es mínima.

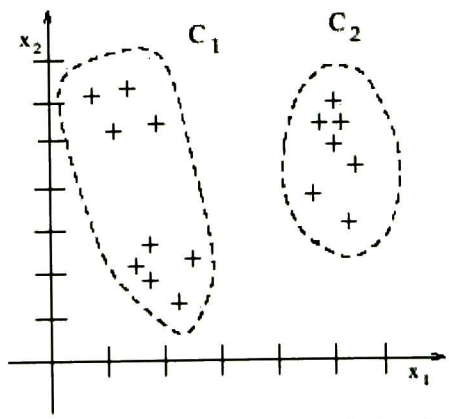


Figura 4.3 Posible partición con $k = 2$. La inercia no es óptima.

En un conjunto pequeño de datos, un ser humano es capáz de captar las semejanzas entre ellos. Sin embargo, cuando crece el número de los mismos, o cuando los datos constan de muchos atributos, es difícil visualizar las aglomeraciones. Una noción muy natural como lo es visualmente diferenciar grupos de puntos en un espacio de dos o tres dimensiones se vuelve complicada cuando estamos hablando de espacios de grado mayor. La utilidad de técnicas automáticas o semi-automáticas que resuelvan o aproximen la solución del problema de clustering se vuelve evidente en las circunstancias citadas. El interpretar la información generada por un mecanismo de clustering sigue siendo tarea para un ser humano, sin embargo, los resultados pueden ser los datos de entrada de alguna otra técnica de minería de datos. Esto sin descartar a los mismos métodos de clustering que se pueden aplicar en varios niveles o en múltiples ocasiones con distintos parámetros con la intención de refinar el resultado.

4.2 El Algoritmo K-Means

El algoritmo k-means es el algoritmo clásico de clustering. Es un intento de solución muy natural e intuitivo.

Algoritmo 4.1. Determinar la partición cuya inercia sea mínima dados un conjunto de elementos $\Omega = \{x_1, x_2, \ldots, x_n\}$ *en* R^p*; el número entero k de clusters que se desea obtener; una condición de paro normalmente en función de la inercia y una función de distancia*

1. *Se eligen al azar k elementos* $x_1, x_2, \ldots, x_k$ *y se designan como los centroides iniciales* $g_1, g_2, \ldots, g_k$
2. *A cada centroide se le asocian los elementos más cercanos, según la función de distancia, formando los clusters* $C_1, C_2, \ldots, C_k$.
3. *Se calculan nuevamente los centroides de cada cluster*
4. *Se calcula la inercia*
5. *Se repite desde el paso 2 hasta que se cumpla la condición de paro*
6. *Se muestra la partición mediante los clusters* $C_1, C_2, \ldots, C_k$ *y su inercia correspondiente.*

La condición de paro se establece según las necesidades. Puede ser que el algoritmo itere un cierto número de veces. Que la inercia de la partición en curso sea lo suficientemente baja para los propósitos de una aplicación en particular o que la inercia se estabilice, es decir, que llegue un momento en que la velocidad con que cambia sea despreciable.

El resultado de este algoritmo está determinado por los centroides iniciales elegidos al principio de la ejecución. Es el único punto donde el azar interviene. Si se hace una mala elección es muy probable que el algoritmo converja demasiado pronto al encontrar un mínimo local.

En las figuras 4.4 y 4.5 se ilustra el comportamiento del algoritmo con un ejemplo donde el número de elementos n es *36*. La cantidad de clústeres deseados k es *3* y los elementos están en R^2

A. En la figura 4.4 se eligen los 3 centroides iniciales y cada elemento se compara con estos para asociarlos al más cercano y de esta forma crear los clusters iniciales.
B. Se recalculan los centroides y reconfiguran los clusters.

C. En la figura 4.5 se observa como paulatinamente los clusters preliminares se aproximan al particionamiento óptimo sugerido visualmente por las gráficas.

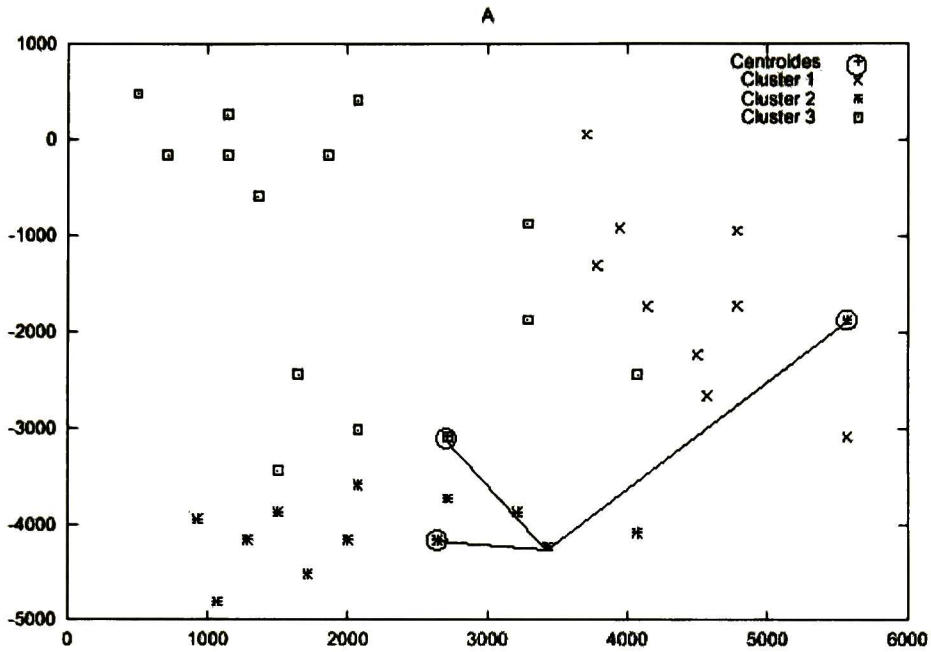


Figura 4.4. Se eligen *k* elementos al azar y se designan como centroides iniciales. Después se asocian a ellos los elementos que estén cerca.

D.

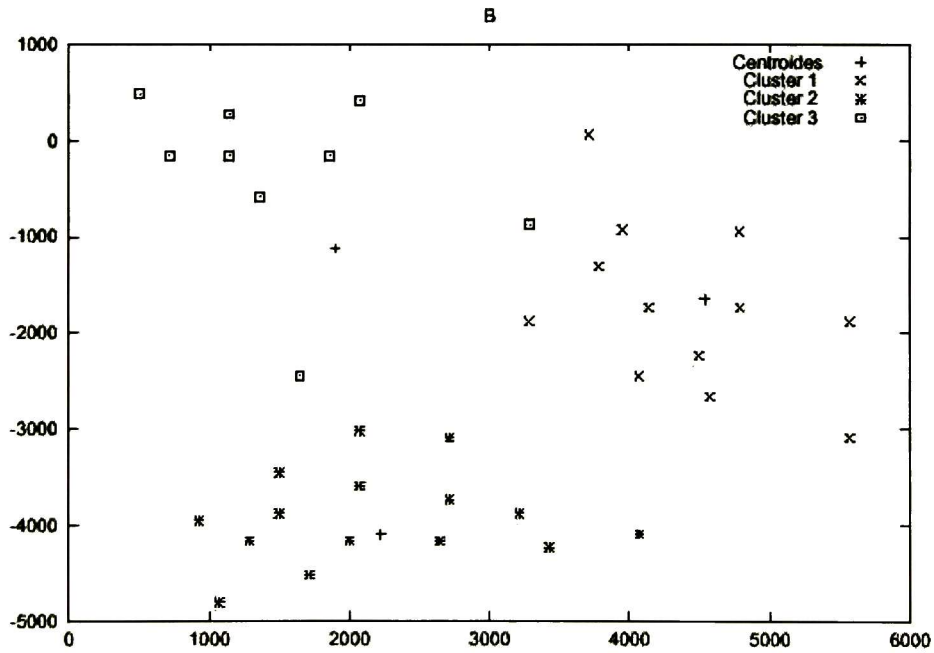


Figura 4.5. Se reposicionan los centroides y se reconfiguran los clusters.

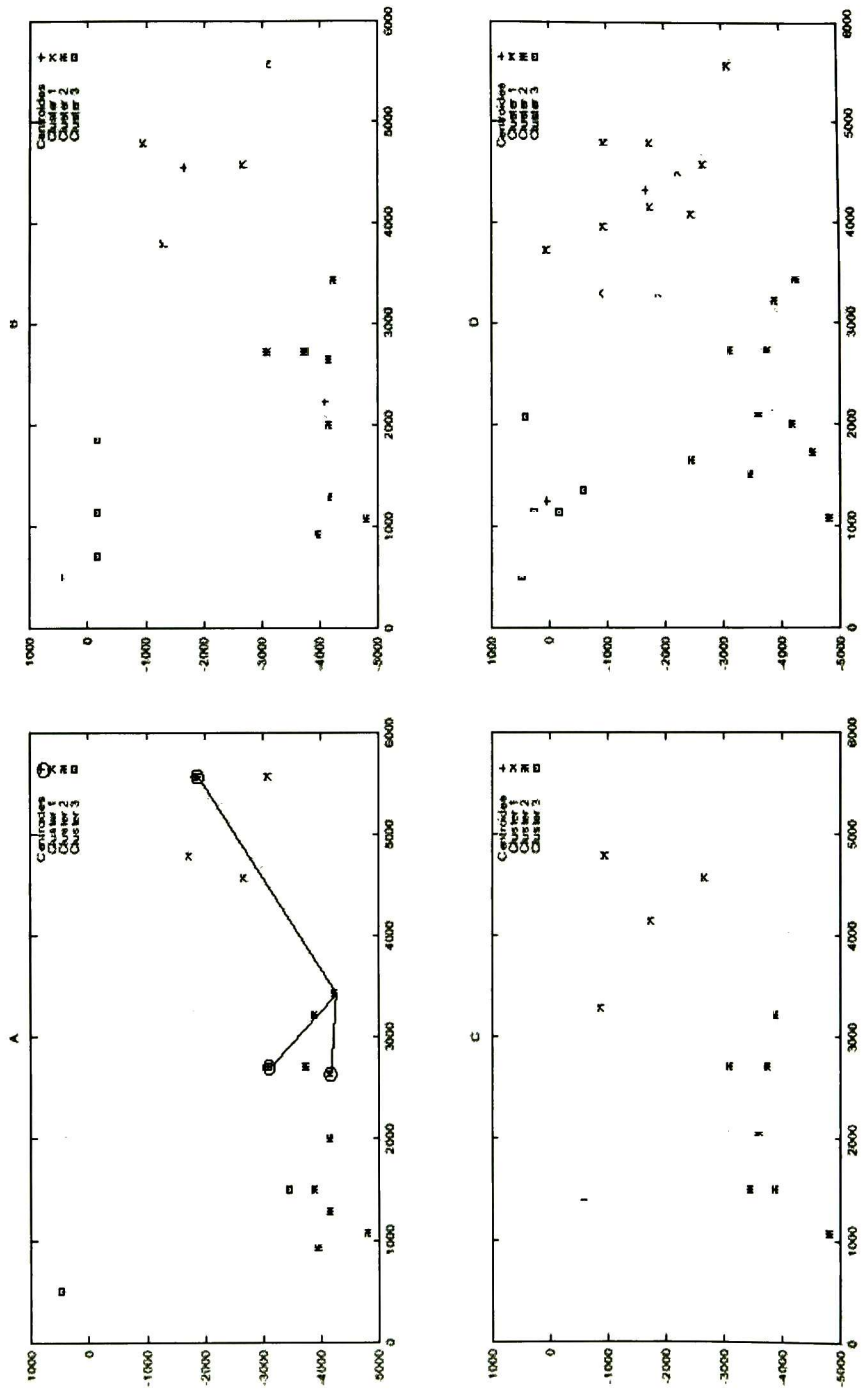


Figura 4.6. La evolución completa

D. En la figura 4.7 mostramos como evoluciona el valor de la inercia con el paso de las iteraciones.

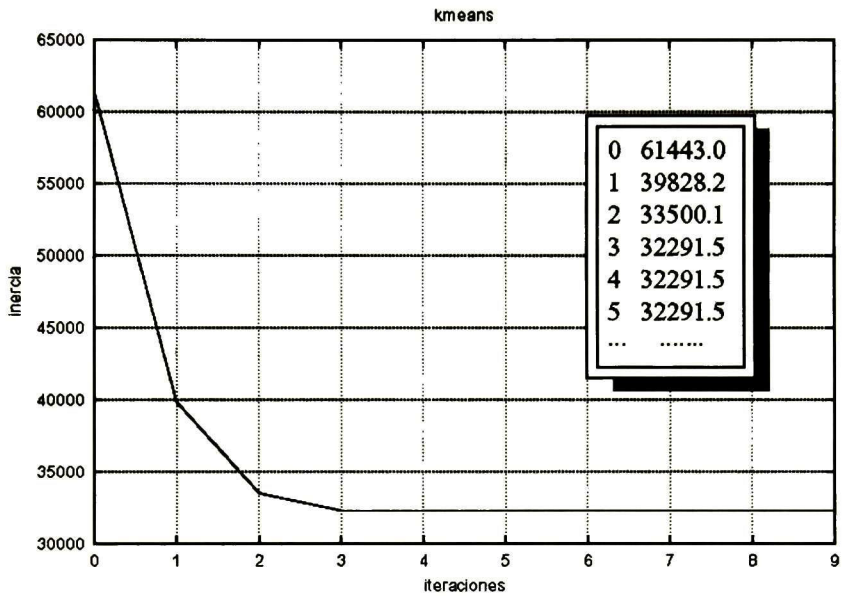


Figura 4.7 Evolución de las inercias para el algoritmo kmeans.

4.3 Técnicas de Optimización Global Estocástica

Las técnicas de optimización global estocástica, son técnicas de búsqueda. El objetivo es encontrar un valor *óptimo* o *cercano al óptimo* en un espacio que frecuentemente es demasiado grande para ser explorado exhaustivamente. Por esta razón se utiliza un enfoque *estocástico*. Un valor óptimo equivale a un valor mínimo o bien a uno máximo de todo el espacio de búsqueda.

Aunque el propósito de estas técnicas es el de encontrar un óptimo global, no existe forma computacionalmente eficiente de saber si el valor encontrado efectivamente es óptimo. Frecuentemente lo que se busca es un valor lo suficientemente cercano al óptimo global de modo que sea adecuado para un problema en específico. En otras ocasiones, el resultado utilizado es el que arroja el algoritmo después de un número predeterminado de iteraciones. Generalmente en este tipo de algoritmos se ha demostrado que, entre más tiempo se ejecutan, se acercan asintóticamente al óptimo global.

El hecho de encontrar el valor mínimo o máximo de un espacio de búsqueda cuando este es grande tiene muchas aplicaciones, porque es una técnica que realmente puede resolver problemas. Si tenemos un problema que tiene varias soluciones y todas ellas tienen la misma forma, entonces el objetivo es encontrar cual de ellas es la mejor solución. Sabemos que hay algunas soluciones mejores que otras porque unas nos reportan un mayor beneficio. Debe entonces existir una función de evaluación que nos indique la calidad de una solución dada. Ahora bien, si el espacio de búsqueda lo constituyen las calidades de todas las posibles soluciones de un problema entonces encontrar un valor mínimo o máximo en ese espacio equivale a encontrar una solución óptima.

Una forma tradicional de encontrar un valor mínimo en un espacio de búsqueda continuo trataría de recorrer dicho espacio ávidamente en busca de valores cada vez más pequeños. Estos métodos están basados en el *descenso del gradiente*. En el caso de la existencia de mínimos locales, la lógica que siguen estos algoritmos nos puede llevar al estancamiento del proceso. Si estamos situados en un mínimo local, todas las soluciones vecinas que pudiera seguir el método tienen un

valor más alto. Dentro de este escenario es muy probable obtener un resultado no óptimo. Más aún, este resultado puede estar demasiado alejado del mínimo global. En la figura 4.8 se ve este caso en un espacio de búsqueda donde se destaca con un trazo los diferentes valores o *estados* explorados por un algoritmo de optimización basado en el descenso del gradiente.

Un método de optimización, aunque con una tendencia a buscar valores mínimos, debe ser capaz de modificar su camino aceptando convenientemente estados de valor más alto. Así se evaden mínimos locales; lo que le da la oportunidad de alcanzar el mínimo global, figura 4.9. Sin embargo, también se corre el riesgo de que el mecanismo que ayuda a eludir mínimos locales, también evite el mínimo global dando como resultado una solución indeseable, figura 4.10

Tanto el sobrecalentamiento simulado como la búsqueda tabú tratan de alcanzar este comportamiento ideal

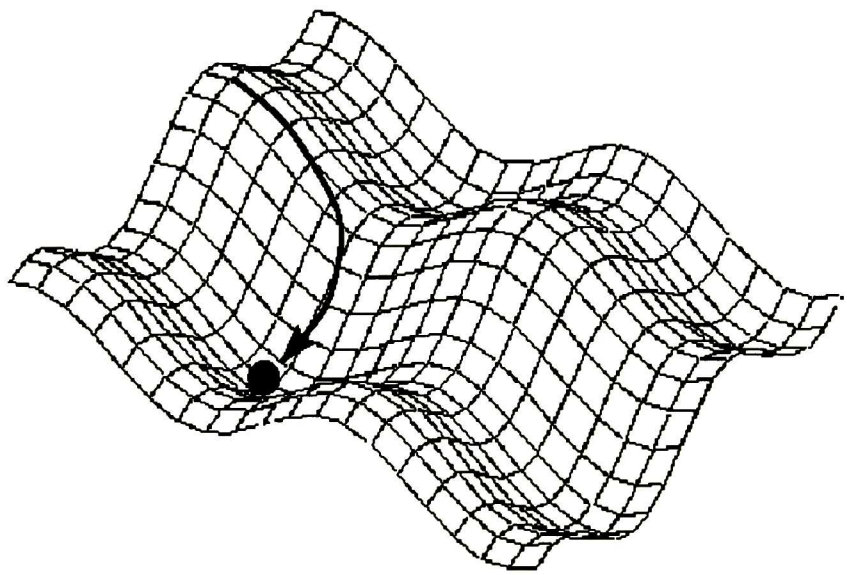


Figura 4.8 Comportamiento general de una búsqueda basada en el descenso del gradiente.

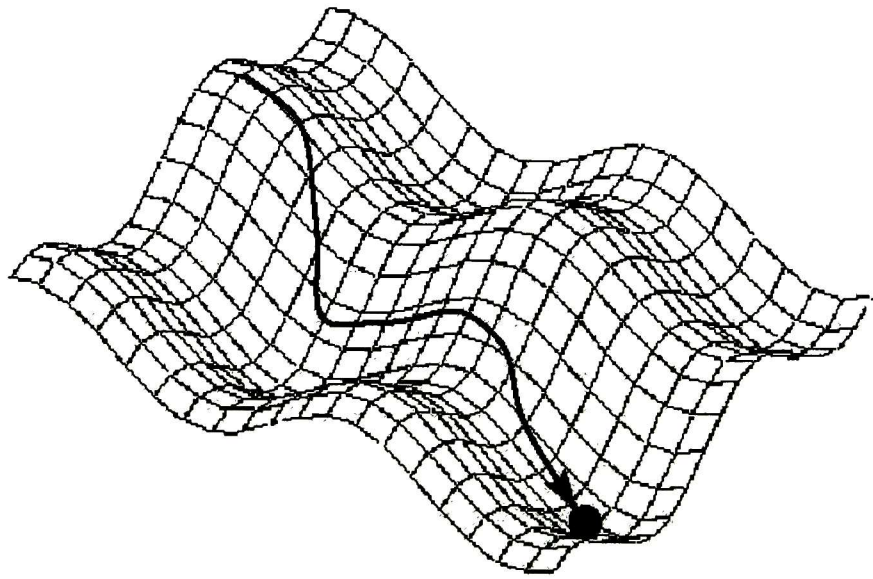


Figura 4.9 Comportamiento ideal. Un algoritmo de optimización debe escapar de los mínimos locales para poder llegar a los mínimos globales.

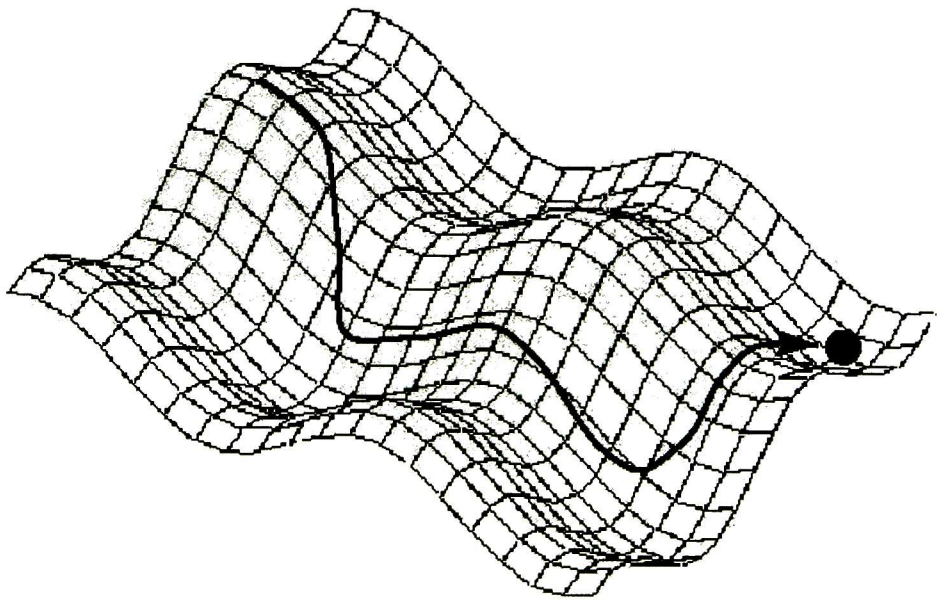


Figura 4.10 . Comportamiento no deseado. Un algoritmo de optimización idealmente debe evitar que el mecanismo que ayuda a eludir los mínimos locales, no permita llegar al mínimo global.

4.3.1 Simulated annealing

Al variar la temperatura o la presión, todos los cuerpos sufren modificaciones internas denominadas cambios de estado. Los principales estados físicos de la materia son el sólido, el líquido y el gaseoso. Pero estos cambios no son los únicos, sino sólo los más aparentes. En el estado sólido debe distinguirse el amorfo y el cristalino. Los cuerpos pueden adoptar diversas formas cristalinas según las condiciones generales de presión, temperatura y dependiendo también del proceso seguido en el cambio de estado. También puede adoptar un estado amorfo en el cual las partículas que lo componen carecen de orientación de conjunto unas respecto a otras. Este estado amorfo se parece al estado líquido o al gaseoso, en los cuales las moléculas gozan de una libertad de movimientos mucho mayor que en el estado sólido y en el que no puede subsistir un determinado orden geométrico.

La técnica de *simulated annealing* se basa en el proceso metalúrgico de solidificación controlada de algunos metales para producir un sólido con características especiales. En particular nos interesan las características de la estructura cristalina del material. Si un material se solidifica en un ambiente donde la temperatura baja bruscamente entonces se obtiene una cristalización muy fina con configuración de energía alta.
En cambio si la disminución de temperatura es paulatina y controlada, se obtiene una cristalización gruesa con una configuración de energía baja. El propósito del recalentamiento simulado es reproducir este proceso físico pero llevado al extremo de tratar de encontrar la configuración de energía mínima, en la que en principio se obtendría un solo cristal. En el mundo metalúrgico, este proceso ha sido modelado y es posible obtener metales con las características deseadas de manera consistente mediante un algoritmo llamado metrópolis.

Esto se puede traducir a un algoritmo de optimización aplicable en otros contextos al considerar los valores de energía posibles como un espacio de búsqueda, el cual se recorre para encontrar el mínimo valor de energía de la misma forma que el proceso físico lo hace. Utilizando modelos de cadenas de Markov se ha probado que el algoritmo simulated annealing encuentra, asintóticamente el óptimo global del problema [15][16].

Los problemas que pueden ser resueltos por este algoritmo deben tener varias posibles soluciones o estados, todas ellas con una forma común y bien definida. Dentro del conjunto de soluciones podemos ver que unas tienen un mayor grado de pertinencia que otras. Este grado de bondad o coste debe estar bien determinado y le llamaremos en lo sucesivo la *energía* del estado. Entre menor sea la energía, mayor será la pertinencia o calidad del estado.

La idea general del algoritmo es ir navegando entre los distintos estados en busca de los que tienen una menor energía. Como veremos, ocasionalmente se aceptan estados con mayor energía. Se requiere que dado un estado i con energía E_i, el siguiente estado j con energía E_J sea generado a partir del primero aplicando una pequeña perturbación aleatoria. Además cualesquiera dos estados deben estar conectados por una secuencia finita de estados de modo que la probabilidad de transición de un estado al estado siguiente sea mayor que cero. El problema debe tener la propiedad de *reversibilidad*, es decir, si el sistema pasa de un estado a a un estado b con una probablilidad p, entonces el sistema debe poder regresar del estado b al estado a con la misma probabilidad p. [13][16]

Cuando el algoritmo se topa con un estado con energía mayor puede aceptar este estado con una probabilidad dada por:

$$P(E) = e^{-\frac{E_j - E_i}{kt}} \qquad (4.4)$$

Donde *Ej y Ej* son los estados de energía siguiente y actual, t es la temperatura del sistema y k es la constante de Boltzman. Para el caso del recalentamiento simulado el producto kt puede ser sustituido por un solo parámetro de control c.

En el proceso físico, la temperatura debe bajar paulatinamente, dependiendo de la forma en la que baje, variará la calidad del resultado. De igual forma, en recalentamiento simulado disminuirá gradualmente el parámetro de control c. Este decremento se define según el problema de aplicación. Finalmente el momento en que termina el algoritmo se determina según los requerimientos de la aplicación.

Resumiendo, para que el algoritmo de recalentamiento simulado pueda ser utilizado se tienen los siguientes requerimientos:

- Una representación común para todas las posibles soluciones del problema
- Un generador de soluciones aleatorias
- Un valor a minimizar determinable por una función de evaluación.
- La propiedad de conectividad.
- La propiedad de reversibilidad.
- Una función de disminución del parámetro de control.
- Un criterio de paro.

Algoritmo 4.2. Dados un espacio de búsqueda $S = \{s_1, s_2, \ldots, s_n\}$ *donde* s_i *es una posible solución, una función de evaluación* $f: S \rightarrow \Re$ *que indica la calidad de una solución, una función de disminución del parámetro de control c y una condición de paro.*

1. *Se genera un estado inicial* s_i *.determinado aleatoriamente o por algún método heurístico.*
2. *Se determina un valor inicial para el parámetro de control c*
3. *Se genera un cambio aleatorio para formar el siguiente estado* s_{i+1}.
4. *Si* $f(S_{i+1}) \leq f(S)$ *el siguiente estado es aceptado, es decir, el estado siguiente ahora es el estado actual.*
5. *Si* $f(s_{i+1}) > f(s)$ *se acepta el siguiente estado con una probabilidad de* $e^{-\frac{E_{i+1}-E_i}{c}}$
6. *Se actualiza el parámetro de control.*
7. *Si no se cumple la condición de paro se prosigue con el paso 3.*
8. *Se muestra el estado final y su energía correspondiente.*

El algoritmo de recalentamiento simulado, en su búsqueda del óptimo global, tenderá a transitar por una secuencia de estados con energía cada vez menor, pero ocasionalmente elegirá valores de energía mayor permitiéndole escapar de algunos mínimos locales. La frecuencia con que el

algoritmo acepta estados con energía mayor a la energía actual disminuye conforme avanza el algoritmo, debido a que el parámetro de control *c* disminuye.

En la figura 4.11 se ve la curva de la función de probabilidad. Entre mayor sea la diferencia de energía *Ej-Ei,* entre un estado actual y el siguiente o menor el parámetro de control, la probabilidad de aceptación es menor.

Si se elige un valor inicial de *c* muy pequeño, la probabilidad de aceptación será despreciable. En términos prácticos nunca se aceptaría un estado con energía mayor, por lo que el algoritmo se reduciría en uno basado en el descenso del gradiente.

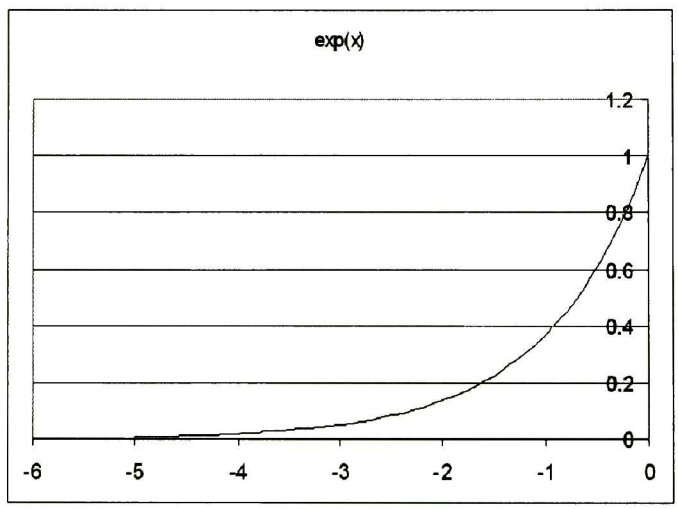


Figura 4.11. Distribución de probabilidad utilizada en simulated annealing.

En la gráfica 4.12 se ilustra la evolución del algoritmo. A lo largo de la trayectoria se observan puntos de los que se desprenden flechas de distintas longitudes. La longitud indica que tan probable es que tomen el camino señalado por la dirección de la flecha. Notese que conforme nos acercamos al punto final de la trayectoria, las flechas de probabilidad que apuntan "hacia arriba" son más pequeñas. Esto indica que conforme avanza el algoritmo disminuye el parámetro de control y por lo mismo la probabilidad de un salto hacia arriba.

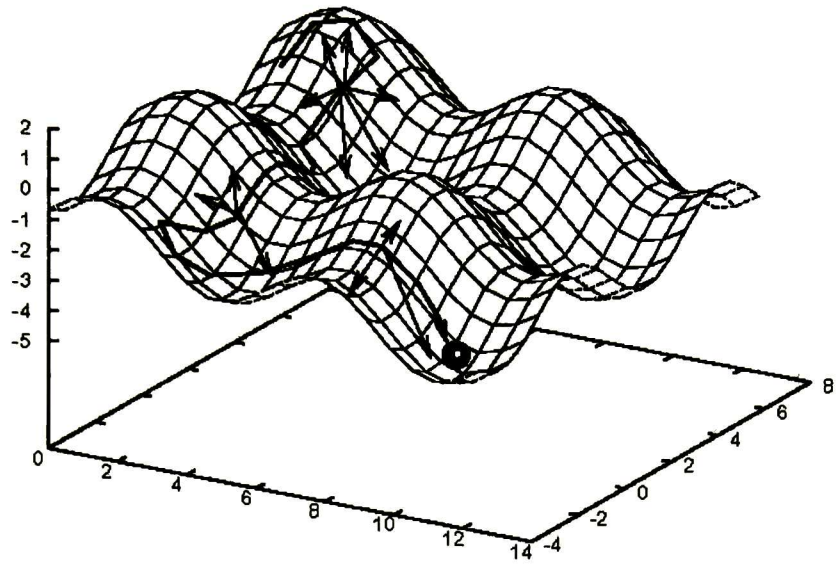


Figura 4.12. Evolución del algoritmo simulated annealing. La longitud de las líneas delgadas es proporcional a la probabilidad de tomar estados de mayor energía.

Debido a que la gráfica es ilustrativa y no está aplicada a un problema en particular hay algunos rasgos que pueden sugerir propiedades no necesariamente existentes. El espacio de búsqueda no necesariamente es continuo.

La transición de un estado a otro es aleatoria con algunas restricciones, es decir, de un estado particular se puede pasar a cualquier otro excepto a los que establezca la restricción, la vecindad entre dos estados entonces queda determi nada por la posibilidad de pasar de uno a otro en un solo paso y no necesariamente, como lo puede sugerir la figura, por la cercanía en un espacio.

Usualmente no hay una forma de representar en una superficie las soluciones de un problema combinatorio. Un problema combinatorio es aquel en el que existe un conjunto de configuraciones discretas formadas por combinaciones de elementos.

4.3.2 Búsqueda Tabú

La búsqueda tabú es una metaheurística utilizada para resolver problemas de optimización discreta combinatoria [12][14]. Ha sido usada para obtener soluciones óptimas y cercanas a óptimas en problemas que incluyen scheduling y optimizaciones de distribuciones de circuitos entre otras aplicaciones. La idea básica de la búsqueda tabú es explorar el espacio de búsqueda de todas las posibles soluciones mediante una secuencia de movimientos. Para escapar de óptimos locales pero no del óptimo global, y para disminuir la posibilidad de entrar en ciclos durante la búsqueda, algunos movimientos, en iteraciones particulares, son clasificados como prohibidos o tabú. Los movimientos tabú se basan en una memoria que, en un momento dado, puede olvidar estados tabú o puede decidirse, de manera extraordinaria elegir un estado tabú.
El autor refiere varias aplicaciones exitosas aplicando su metaherística. Los resultados prácticos impulsan el desarrollo de esta técnica, sin embargo, la efectividad de la misma depende de la representación del problema en cada estructura del método [18].
El concepto básico de la búsqueda tabú es una meta-heurística superpuesta en otra heurística. Como tal, la búsqueda tabú es muy general y puede englobar a otras técnicas. El método tabú fue motivado en parte por la observación del comportamiento humano, ya que, aparentemente opera con un elemento aleatorio que conduce a comportamientos inconsistentes dadas circunstancias similares. No siempre se resuelven los problemas de la misma forma, aun en circunstancias similares. La tendencia resultante de desviarse de un curso preestablecido, puede ser considerada como una fuente de error, pero puede ser también fuente de mejores resultados. El método tabú opera de esta manera con la excepción de que nuevos cursos no son escogidos aleatoriamente. La búsqueda tabú trabaja bajo la suposición que no tiene sentido el aceptar nuevos soluciones de mala calidad a menos que sea para evitar un camino ya investigado. Esto asegura que nuevas regiones del espacio de búsqueda sean investigadas con el objetivo de evitar mínimos locales y en última instancia, encontrar las soluciones deseadas.

La búsqueda tabú comienza con un descenso del gradiente para acercarse a los mínimos locales. Para evitar el regreso a estados ya visitados, este método registra las movidas más recientes en una o más listas tabú. El intento original de esta lista no era evitar que las movidas previas fueran repetidas, el objetivo era que el camino no se recorriera de igual manera pero en sentido contrario.

El rol de la lista tabú puede cambiar conforme el algoritmo progresa. Al principio del algoritmo, el objetivo es hacer una exploración ligera del espacio de solución, pero tan pronto como estados candidatos a óptimo son localizados, la búsqueda es más focalizada para producir mínimos locales. En ocasiones, la diferencia entre distintas implementaciones de la búsqueda tabú tiene que ver con el tamaño, la variabilidad y la adaptabilidad de la memoria tabú para un dominio particular de problema.

Los elementos principales del algoritmo de búsqueda tabú son:

- *Representación de la solución.* Cada posible solución del problema de optimización debe tener una única representación dentro del espacio de búsqueda.
- *Función de costo.* Una función de costo haciendo un mapeo entre cada solución posible y un valor representando su costo de optimización. El objetivo del algoritmo es encontrar una solución que minimiza este valor.
- *Vecindario.* Una función que relaciona cada posible solución S con un conjunto de otras soluciones. Cada vez que el algoritmo tiene que considerar una nueva solución, es escogida del vecindario de la solución actual.
- *Lista tabú.* Una lista que contiene los últimos T movimientos realizados, que por esta razón, están prohibidos. Una solución obtenida de la solución actual S con una movida contenida en la lista tabú no puede, en general, ser un miembro del vecindario de S
- *Criterio de aspiración.* Si un movimiento que pertenece a la lista tabú satisface este criterio, entonces la solución obtenida mediante la aplicación de este movimiento puede ser considerado para pertenecer al vecindario y ser entonces elegible. El criterio típico es que si el movimiento produce la mejor solución obtenida hasta el momento entonces se puede aceptar.
- *Criterio de terminación.* El algoritmo se detiene cuando el criterio de terminación es satisfecho.

Algoritmo 4.3. Sea N(s) el conjunto de estados que pueden ser alcanzados mediante un movimiento a partir del estado s T(s), el conjunto de estados tabú; A(s) el conjunto de aspiraciones. F(s), una función de evaluación de s y s' el estado siguiente elegido.

1. *Generar una solución s aleatoriamente*
2. *Identificar N(s).*
3. *Identificar A(s)*
4. *Escoger s' ∈ (N(s) - T(s)) ∪ A(s), para el cual F(s') es mínimo*
5. *Actualizar la lista de tabú*
6. *Si no se cumple el criterio de paro ejecutar el paso 2*
7. *Mostrar s*

4.3.3 Técnicas de Optimización Global Estocástica Aplicadas al clustering

Las técnicas mostradas, simulated annealing y búsqueda tabú, fueron aplicadas para el problema de clustering por Trejos, Piza y Murillo de la siguiente manera [15].

Simulated annealing aplicado a clustering.
En este caso el espacio de búsqueda está dado por todas las posibles particiones en que se puede configurar un conjunto de datos en k clusters. La función de evaluación está definida por la ecuación 4.3. que determina que tan buena es una partición dada. La transición entre dos estados se realiza seleccionando aleatoriamente un elemento y un cluster. El elemento seleccionado se asigna al cluster seleccionado. Con estos parámetros definidos, se ejecuta el algoritmo simulated annealing como está definido en el algoritmo 4.2

El cálculo de la inercia puede ser simplificado ya que una vez definida la inercia del estado actual, la inercia del estado siguiente no se calcula a partir de los datos, sino a partir de la inercia anterior a través de la siguiente ecuación [15]

$$\Delta W = W(\overline{P}) - W(P) = \frac{|C_j|}{n(|C_j|-1)} \| g(C_j) - x \|^2 - \frac{|C_l|}{n(|C_l|+1)} \| g(C_l) - x \|^2 \quad (4.5)$$

Dado un conjunto $\Omega = \{ x_1, x_2, \ldots , x_n \}$de elementos y siendo k la cantidad de clusters que se desea obtener, Cada posible partición o configuración constituye el espacio de solución.

Para obtener una nueva partición P' a partir de una partición P:

1. Se Selecciona aleatoriamente un objeto $x_i \in \Omega$
2. Seleccionar al azar un índice $l \in \{1,2,...,k\}$;
3. Colocar a x_i en la clase c_i

Búsqueda tabú aplicada al clustering

El espacio de soluciones lo forman el conjunto de particiones posibles. Un movimiento es la transferencia de un elemento de una clase a otra. La lista tabú se compone de los m últimos movimientos. La lista de aspiraciones son los elementos que tengan la mejor calificación dada por la ecuación 4.3. En este algoritmo también aplica la simplificación del cálculo de la inercia de la ecuación 4.5

5 Implementación

Existen varias estrategias para paralelizar algoritmos, típicamente se paraleliza el control o los datos. Cuando se paraleliza el control, se divide la funcionalidad del algoritmo en varias partes, dando lugar a un conjunto de programas distintos. Luego, cada unas de las partes se ejecuta simultanea y coordinadamente. Por otro lado, la paralelización de datos consiste en ejecutar varias instancias del mismo programa en diferentes segmentos de datos para, al final, realizar un proceso de consolidación de resultados. La utilización de una u otra estrategia depende del modelo de programación, el costo de comunicación y la estructura intrínseca del algoritmo.

Los algoritmos utilizados en este trabajo de tesis son algoritmos probabílisticos. Nunca se tiene la certeza de que en un momento dado los algoritmos hayan encontrado una solución óptima. La efectividad de los algoritmos está demostrada en los trabajos previos referidos. Nuestro interés está en la paralelización del procesamiento. Para medir el tiempo de ejecución de los algoritmos fijamos la cantidad de iteraciones que los algoritmos ejecutan, independientemente de si han encontrado la solución o no. Esto nos permite hacer las mediciones de manera consistente sin perder generalidad. En el caso de que el algoritmo encuentre el óptimo antes de terminar con las iteraciones establecidas, puede seguir en ejecución realizando el mismo trabajo computacional que antes de encontrar la solución.
Los algoritmos son probados en un ambiente multitarea, por lo que los tiempos de ejecución varían de ejecución en ejecución. Podemos notar adicionalmente que estas condiciones de ejecución tienen un alto impacto con respecto a la diferencia en los tiempos de ejecución con distinta cantidad de procesadores.

En todos los casos se utiliza un conjunto de cuatro mil datos de prueba generados artificialmente, los datos son de dos dimensiones por lo que pueden ser graficados. Los datos sugieren la existencia de tres clusters.

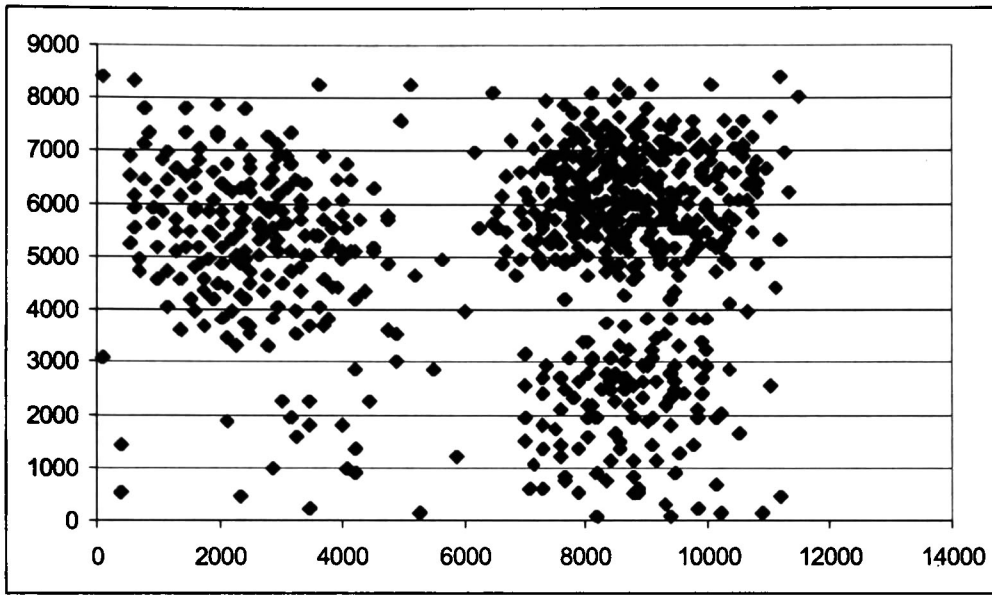


Figura 5.1 Datos de prueba. Se sugiere la formación de tres clusters

Debido a que cada elemento necesariamente está en uno de k clusters el espacio de búsqueda es de 3^{4000} estados o posibles soluciones.

El espacio de búsqueda es combinatorio, no continuo. En general no existe una forma ordenada de mostrarlo, por lo que arbitrariamente desplegamos los primeros cien elementos siguiendo un esquema de numeración base 3. En la gráfica se unen los puntos para una mejor visualización, pero no existen estados intermedios entre dos puntos contiguos, es una gráfica de puntos discretos.

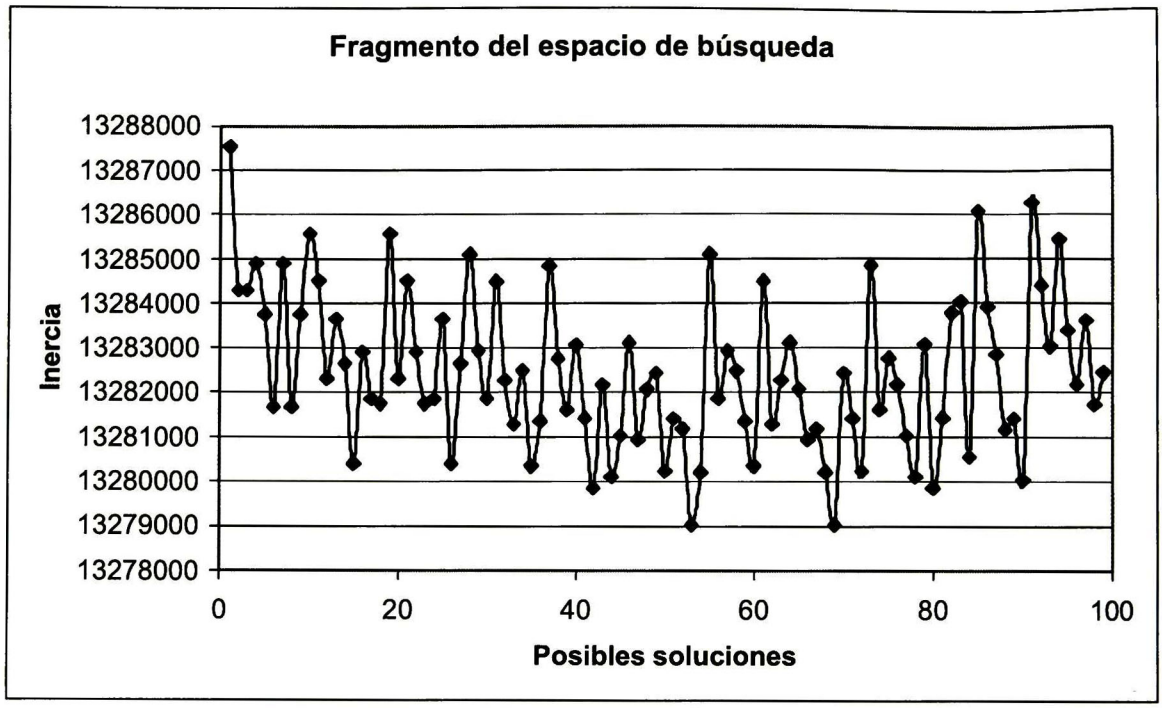


Figura . 5.2 Fragmento del espacio de búsqueda

5.1 Kmeans

Las partes computacionalmente más demandantes del algoritmo kmeans son: la asignación de cada elemento al centroide más cercano, para formar los clusters en cada iteración y calcular los nuevos centroides. Los cálculos realzados en una iteración dependen de los resultados de la iteración anterior. Cada iteración requiere del procesamiento de todos los datos. Todo lo anterior implica que debe existir un punto de sincronización en cada iteración.

En este algoritmo se optó por realizar una paralelización tanto de control como de datos sin embargo, la paralelización que realmente se explota es la de datos, siendo la de control un tanto accesoria.

Para calcular los centroides es necesario realizar una suma dimensión por dimensión de todos los elementos que pertenecen a un cluster y luego dividirlos entre la cantidad de elementos que pertenecen al cluster correspondiente. La suma inicial puede ser paralelizada, sin embargo, para hacer el cálculo final de los centroides, se requiere conocer la cantidad de elementos en cada cluster.

Elementos paralelizables:

- La asociación de los elementos de cada elemento con su centroide más cercano.
- El cálculo parcial de los centroides
- El cálculo de la inercia

Elementos intrínsecamente secuenciales:

- El cálculo final de los centroides

Se optó por paralelizar de la siguiente manera:

- El procesamiento está dividido en dos tipos de procesos, un proceso padre y varios procesos hijo.

Proceso padre:

1. Obtiene los datos de un dispositivo de almacenamiento
2. Crea p instancias de procesos hijo
3. Divide y envía los datos a cada proceso hijo
4. Calcula los centroides iniciales
5. Envía los centroides a cada proceso hijo
6. Espera los resultados de los procesos hijo
7. Recibe los centroides parciales de cada proceso hijo
8. Recalcula los centroides a partir de los datos del paso 5 del hijo
9. Si no se ha cumplido la condición de paro,
   continua con el paso 5.
10. Recaba resultados de procesos hijo
11. Muestra los resultados

Procesos hijo:

1. Recibe el segmento de datos que le ha sido asignado
2. Recibe los centroides del proceso padre
3. Asocia cada dato al centroide más cercano
4. Realiza un cálculo parcial de los nuevos centroides

5. Envía el cálculo parcial de los centroides al proceso padre para que continúe en el paso 7 del proceso padre
6. Si nos se cumple la condición de paro

   continúa con el paso 2
7. Envía los resultados parciales al proceso padre

Resultados:

Kmeans			
Ejecuciones	**Secuencial**	**3 procesadores**	**4 procesadores**
1	265.8403399	175.57814	221.79634
2	266.3266101	175.72421	236.2679901
3	263.9744098	176.2983999	251.27107
4	265.8430598	177.2725599	190.6723101
5	305.9002299	190.8350801	220.75806
6	264.9105201	176.4179199	178.51052
7	314.8771	176.0443702	178.4814
8	263.8801301	176.1470599	217.6210401
9	265.44397	184.6240401	207.4973199
10	265.4078598	184.60324	183.1036901
Promedio:	274.2404229	179.354502	208.597974
Aceleración:		1.5290412	1.314684019
Iteraciones:	50,000	50,000	50,000
Cantidad de datos:	4,000	4,000	4,000
Dimensiones:	2	2	2
Cluster buscados:	3	3	3
Cantidad de estados	3^{4000}	3^{4000}	3^{4000}

Tabla . 5.1 Resultados de la paralelización del algoritmo kmeans

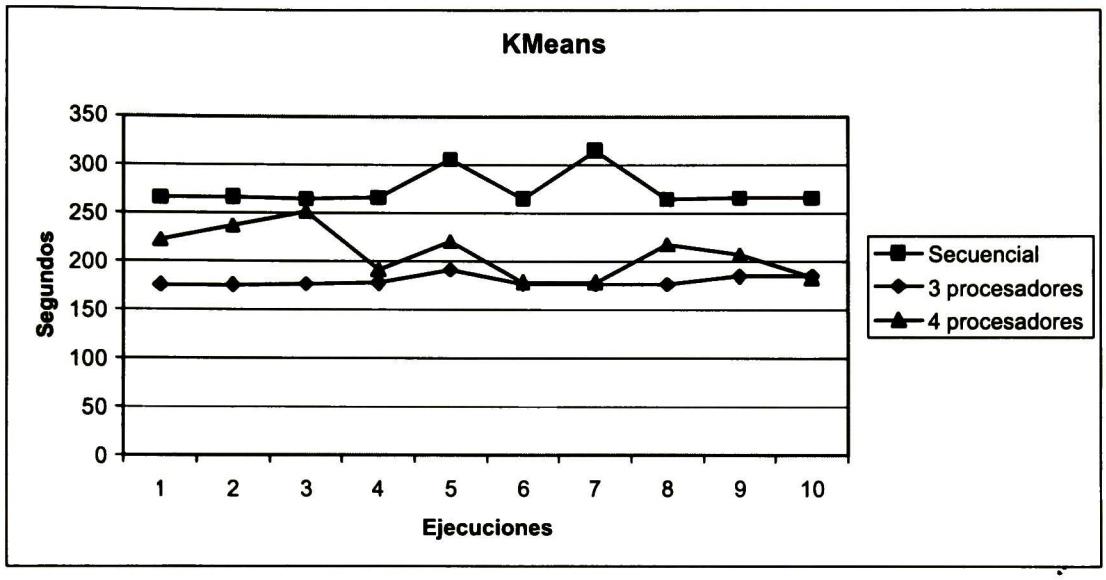


Figura . 5.3 Resultados de la paralelización del algoritmo kmeans

Conclusiones de la paralelización del algoritmo Kmeans:

- Se require un punto de sincronización para cada iteración, lo que ocasiona un fuerte costo de comunicación.
- Se logró una mejora con tres procesadores, dos ejecutando al proceso hijo y un tercero al padre, La mayor parte de la carga de trabajo se encuentra en los procesos hijo.
- Conforme crece la comunicación entre procesos, el desempeño es más pobre, en particular, a partir de los cuatro procesadores.

5.2 Simulated Annealing

El algoritmo simulated annealing aplicado a clustering es un algoritmo en el que la carga de trabajo realizada en cada iteración es relativamente pequeña comparada con el kmeans. En este caso, la mayor carga de trabajo se encuentra en la cantidad de iteraciones que se deben realizar para llegar a un resultado satisfactorio. Para ejecutar una iteración particular, se requiere de los resultados de la iteración anterior. Por estas dos razones, el paralalelismo de control resulta inviable.

Elementos paralelizables:

- Actualización del annealing schedule con respecto a la evaluación de la función de probabilidad. Sin embargo, ambas operaciones deben estar dentro de la misma iteración.

Elementos intrínsecamente secuenciales:

- Determinación del estado siguiente a partir del estado actual
- Evaluación de la función de probabilidad con respecto a la determinación del nuevo estado

La paralelización de control en este algoritmo no resulta viable debido a la poca carga de la iteración principal. La paralelización de datos convencional tampoco resulta viable debido a que la división de los mismos no permite un cálculo más rápido del ciclo principal del algoritmo.

Tampoco es posible dividir el espacio de búsqueda sin afectar los resultados del algoritmo, la división siempre sería arbitraria. Recordemos que el espacio de búsqueda es combinatorio y no continuo. Sin embargo, debido a la forma en que se calcula el estado siguiente el espacio tiene una cierta suavidad. Existe un pequeño cambio entre los estados vecinos lo que hace que las inercias no cambien bruscamente.

Aunque no es viable hacer una paralelización de la que se obtenga una mejora en el rendimiento, se explora la posibilidad de aprovechar el paralelismo en la mejora de la calidad del resultado y no en la mejora del tiempo de ejecución.

El algoritmo simulated annealing carece de memoria que permita llevar un registro de donde haya estado el cursor. La elección del estado siguiente en el simulated annealing está determinada en parte por una función de probabilidad cambiante. Sin embargo, este cambio no está bien definido en el algoritmo original. No obstante, podemos utilizar información conjunta de los cursores para definir con más precisión la forma en que cambia dicha función de probabilidad. El escenario es el siguiente: Suponiendo que se ejecutan simultáneamente varios procesos y que estos pueden conocer el valor de la inercia actual de los otros procesos, entonces si un proceso dado está en una planicie o valle no puede decidir si ha llegado al mínimo global, pero si puede saber si está en un mínimo local siempre y cuando exista otro proceso en un estado con una inercia menor. En ese

caso puede tomar la decisión de intentar salir del mínimo local con la intención de localizar otro camino hacia el mínimo global u otro mínimo local menor. No existe garantía de que lo encontrará, pero la intención es mejorar las posibilidades de encontrar un mejor resultado.

El algoritmo es como sigue:

- Se ejecutan p procesos simultáneamente, cada uno de los cuales realiza lo siguiente:
    1. Obtiene los datos de un medio de almacenamiento físico
    2. Genera un estado aleatorio
    3. Genera aleatoriamente un estado siguiente
    4. Si la inercia del estado siguiente es menor,
        el estado siguiente se convierte en el estado actual
       En otro caso
        se acepta el nuevo estado con una probabilidad indicada por la función de probabilidad de SA
    5. Difunde a los demás procesos la inercia de su propio estado actual
    6. Recibe las inercias actuales de los demás procesos
    7. Si existe por lo menos un proceso que tenga una inercia menor
        disminuye la tasa de decremento del parámetro de control
    8. Se actualiza el parámetro de control
    9. Si no se cumple la condición de paro
        continúa en el paso 2
    10. Muestra los resultados

- Con el objetivo de disminuir los costos de comunicaciones, la sincronización de los procesos no se hace cada cierta cantidad de iteraciones.
- Un algoritmo determina si está en un valle o planicie llevando un historial de las inercias de los últimos estados que ha visitado. Si en este historial se detecta una marcada monotonía en los resultados, se asume que se está en un valle o planicie. Esta decisión está basada en

información parcial por lo que es falible. Los siguientes parámetros son ajustables, pero aún no tenemos un criterio estricto para determinarlos:

1. El tamaño de la historia
2. El grado de monotonía a partir del cual se decide que se está en un valle o planicie

Resultados:

Simulated Annealing			
Ejecuciones	**Secuencial**	**2 procesadores**	**3 procesadores**
1	60.8064301	227.0392799	378.28753
2	61.99574995	237.3456402	377.65052
3	65.75935006	287.0980699	376.97283
4	62.35429001	270.26877	361.21370
5	63.10119009	190.4534099	378.45967
6	62.86930013	232.7327499	378.53558
7	66.11024022	232.7327499	378.39013
8	65.83485985	154.85974	380.81439
9	63.96353006	218.3911099	382.35591
10	62.74208021	223.05987	380.26188
Promedio:	63.55370207	227.398139	377.294214
Aceleracion:		0.279482068	0.168446002
Iteraciones:	50,000	50,000	50,000
Cantidad de datos	4,000	4,000	4,000
Dimensiones:	2	2	2
Clusters buscados:	3	3	3
Cantidad de estados:	3^{4000}	3^{4000}	3^{4000}

Tabla . 5.2 Resultados de la paralelización del algoritmo Simulated Annealing

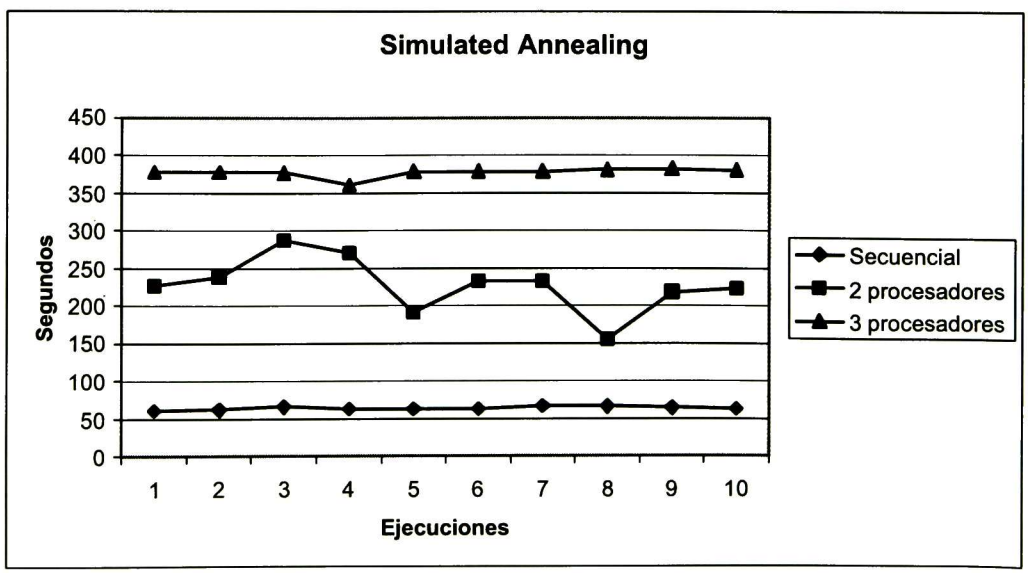


Figura . 5.4 Resultados de la paralelización del algoritmo kmeans

La variante del algoritmo simulated annealig propuesta se comporta en tiempo de la manera recién presentada, sin embargo, no hemos determinado si efectivamente se obtienen mejores resultados.

Para ilustrar el comportamiento mostramos dos ejecuciones con diez procesos cada una.

En la implementación se ha limitado la cantidad de oportunidades que un proceso puede incrementar su parámetro de control debido a que se observó que un proceso que ha incrementado una vez su parámetro de control lo seguirá haciendo debido a que de entrada su inercia actual tiende a ser todavía mayor de lo que era antes. Entre más oportunidades tenga un proceso de subir su parámetro de control más pronunciado el efecto de intento de escape de los mínimos locales. En la primer gráfica se ejecutó con una cantidad de oportunidades para incrementar el parámetro de control deliberadamente alta. Doscientas oportunidades para cada proceso.

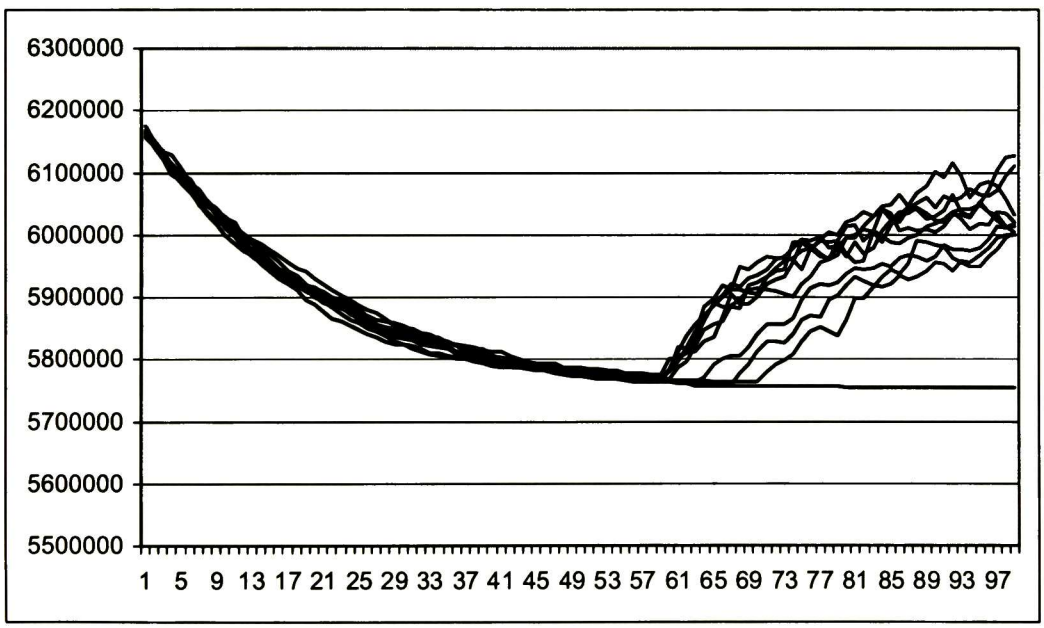


Figura . 5.5 Gráfica de ejemplo del comportamiento del algoritmo SA en paralelo. Los parámetros deliberadamente están escogidos para destacar el comportamiento de los procesos que elevan su parámetro de control.

En esta otra gráfica, tan solo se dan 20 oportunidades a cada proceso de incrementar su parámetro de control. En este ejemplo todos los procesos confluyen a resultados muy cercanos y es más

difícil apreciar el efecto mencionado. Se requiere un estudio estadístico para establecer si esta variación del algoritmo simulated annealing efectivamente tiende a encontrar mejores resultados.

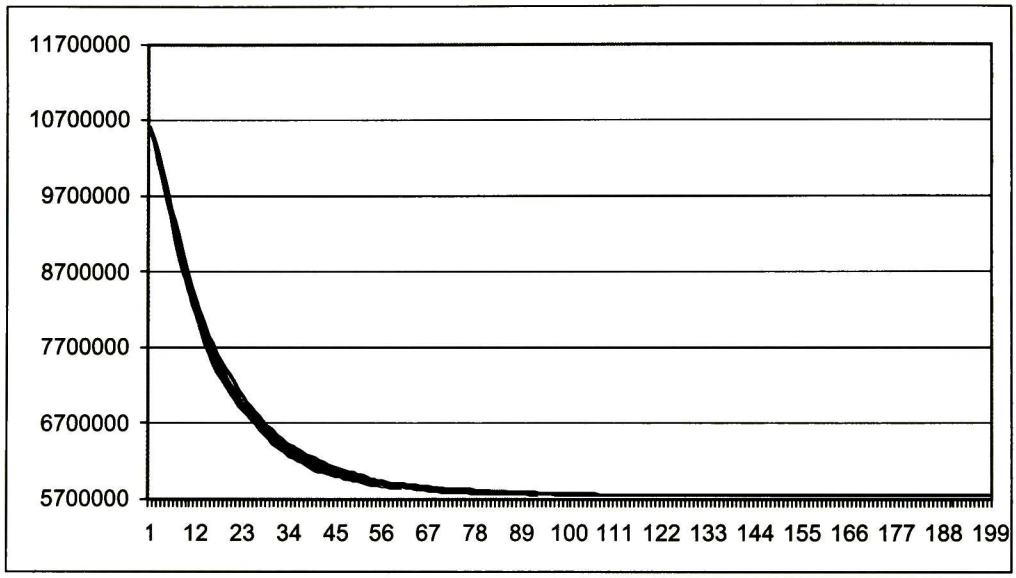


Figura . 5.6 Gráfica de ejemplo del comportamiento del algoritmo SA en paralelo. Ejecución normal del algoritmo.

Conclusiones de simulated annealing aplicado a clustering:

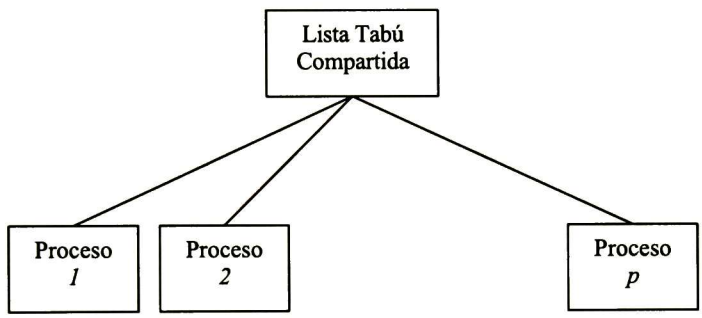
- No se obtuvo una mejora en tiempo
- Los procesos se ejecutan en un espacio de búsqueda común
- El tiempo adicional requerido para la ejecución en paralelo se debe a las comunicaciones adicionales.
- Se requiere un estudio extensivo de la propuesta de algoritmo para establecer si efectivamente tiende a encontrar mejores resultados que el algoritmo original.

Tabu search

En este algoritmo también existe un ciclo iterativo principal y al igual que el algoritmo simulated annealing, la mayor carga de trabajo está en la cantidad de iteraciones que se deben ejecutar antes de llegar a un resultado útil. Existe un solo cursor de búsqueda y cada iteración requiere de los

resultados de la iteración anterior. Esto descarta el paralelismo de control, pero hay una oportunidad de paralelizar los datos.

Es posible que los varios procesos de ejecución compartan la misma lista tabú, en este caso, los procesos evitarán no solo los estados tabú que hayan resultado de su propia exploración, sino que también se aprovecha en parte el resultado de la exploración de los otros procesos. Esto permite dividir el espacio de búsqueda. Esta división está determinada dinámicamente durante la ejecución de cada instancia.



En la figura se ve como todos los proceso comparten una misma lista tabú.

Se ejecutan p procesos simultáneamente, cada uno de los cuales realiza lo siguiente:

1. Obtiene los datos del medio de almacenamiento físico
2. Genera un estado aleatorio
3. Se identifica el vecindario y se elige el mejor estado del vecindario que no esté en la lista tabú a menos que sea el mejor estado conocido
4. Se agrega el nuevo estado a la lista tabú
5. Se determina el mejor estado conocido
6. Se difunde a los demás procesadores el mejor estado conocido y la lista tabú
7. Se reciben los mejores estados alcanzados y las listas tabú de todos los procesadores y se combinan

8. Si no se cumple el criterio de paro

   ir al paso 3

9. Mostrar el estado o partición encontrada

Resultados:

Solo se implementó la versión secuencial del algoritmo tabu search por lo que no se realizó la comparativa y solo se propone la estrategia de paralelización. Además se prevé que la estrategia descrita será mejor aplicada a un modelo de programación paralela de memoria compartida debido al alto costo en comunicación que la actualización de la lista tabú implica.

Resumen de Implementación	
Lenguaje:	C++
Sistema Operativo:	Linux
Plataforma de paralelización:	PVM
Líneas de Código	
KM Secuencial:	91
KM Paralelo:	Padre: 231 Hijo: 126
SA Secuencial:	156
SA Paralelo:	410
TS Secuencial:	137
Bibliotecas de soporte:	1200

Tabla 5.3. Resumen de Implementación

6 Conclusiones y trabajo futuro

Conclusiones:

- Se obtuvo una mejora del 66% aproximadamente en el algoritmo k-means con 3 procesadores.
- La paralelización en un modelo de programación de paso de mensajes y paralelismo a nivel de tarea no es el más adecuado para los algoritmos simulated annealing y tabu search.
- Se realizó una propuesta para mejorar la calidad de los resultados arrojados por el algoritmo simulated annealing. Se requiere un análisis más extensivo para determinar contundentemente si realmente se mejoraron los resultados.
- Sólo se implementó la versión secuencial del algoritmo tabu search y se realizó el diseño de la versión paralela.

Trabajo futuro:

- Paralelizar bajo un modelo de programación de memoria compartida y granularidad más fina.
- Extender este estudio a otros algoritmos, ya que hay una amplia variedad de ellos.
- Crear un marco genérico para la incorporación de algoritmos paralelos de minería de datos.

BIBLIOGRAFIA

[1] Berry Michael, y Linoff Gordon. "Data Mining techniques for Marketing, Sales, and Customer Support". John Wiley & Sons, Inc. 1997

[2] Berson Alex, Smith Stephen y Thearling Kurt. "Building Data Mining Applications for CRM". McGraw-Hill Osborne Media, 1999

[3] Poe Vidette, Klauer Patricia y Brobst Stephen. "Building a data warehouse for decision support" Prentice Hall 1998

[4] Adriaans Pieter y Zantingue Dolf.. "Data mining". Addison-Wesley 1996

[5] Ian H. Witten y Eibe Frank. "Data mining: practical machine learning tools and techniques with Java implementations", Morgan Kaufmann, 2000

[6] JáJá Joseph "An introduction to Parallel Algorithms" Addison-Wesley Pub Co, 1992

[7] Geist Al, Berguelin Adam, Dongarra Jack, Jiang Weicheng, Mancheck Robert y Sunderam Vaidy. "PVM Parallel Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing", The MIT Press, 1994

[9] "Paralelismo". Notas del curso. Escuela de Invierno de Sistemas Distribuidos. Guadalajara México, 1998

[10] "Paralelismo" Notas del curso. Internacional Symposium of Advanced Distributed Systems and School, 2000

[11] Vipin Kumar y Mahesh Joshi, "Tutorial on High Performance Data Mining" http://www-users.cs.umn.edu/~mjoshi/hpdmtut/ , 2000

[12]Gray Perry, Hart William, Painton Laura, Phillips Cindy, Trahan Mike, Wagner John "Global Optimization Survey" http://www.cs.sandia.gov/opt/survey/sa.html , Sandia National Laboratory, 1997

[13] Luke Brian. Ph.D. "Simulated annealing" http://members.aol.com/btluke/simann1.htm

[14] Glover Fred. "Tabu search: fundamentals and uses" University of Colorado, 1995

[15] Trejos Javier, Murillo Alex, Piza Eduardo "Global Stochastic Optimization Techniques applied to partitioning" CIMPA Universidad de Costa Rica

[16] Kirkpatrick S., Gelatt C. y Vecchi M. "Optimization by Simulated Annealing" SCIENCE, 1983

[17] Trejos Javier, Murillo Alex "Heuristics of Combinatorial Optimization and Applications to Data Analysis" CIMPA, Universidad de Costa Rica

[18] Nicholas J. Radcliffe "Computer Science Today: Recent Trends and Developments", J. van Leeuwen, Springer-Verlag LNCS 1000, 1995

[19] David J. Hand "Data Mining, statistics and more", The American Statistician, 1998

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis:

ESTUDIO DE LA PARALELIZACIÓN DE ALGORITMOS UTILES EN MINERIA DE DATOS

del (la) C.

Jorge Lauro HERNANDEZ ROJAS

el día 2 de Abril de 2004.

Dr. Jose Luis LEYVA MONTIEL
Investigador Cinvestav 3B
CINVESTAV GDL
Jalisco

Dr. Deni Librado TORRES ROMÁN
Profesor Investigador 3A
CINVESTAV GDL
Jalisco

Dr. Félix Francisco RAMOS CORCHADO
Investigador Cinvestav 2B
CINVESTAV GDL
Jalisco