UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE QUIMICA

QUIMICA COMPUTACIONAL: UN ANALISIS CRITICO.

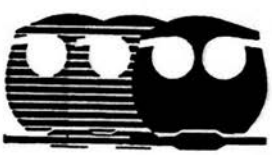
T E S I S

QUE PARA OBTENER EL TITULO DE:

Q U I M I C O

P R E S E N T A :

EDMUNDO SEGUNDO CARRERA MARTINEZ



MEXICO, D.F. **2004**

Jurado asignado:

Presidente	Dra. Sara Elvia Meza Galindo
Vocal	Dr. Andoni Garritz Cruz
Secretario	Dr. Carlos Amador Bedolla (Asesor)
1er. Suplente	Dr. Francisco Miguel Castro Martínez
2ndo. Suplente	Dr. Luís Emilio Orgaz Baque.

Sitio en donde se desarrolló el tema
Departamento de Física y Química Teórica. Facultad de Química.
UNAM.

Dr. Carlos Amador Bedolla
Asesor de tesis

Edmundo Segundo Carrera Martínez
Sustentante

TABLA DE CONTENIDO

AGRADECIMIENTOS

Considero que una de las partes más difíciles de escribir en esta tesis, al menos para mí, es sin duda los agradecimientos; ya que es muy difícil expresar en un pequeño espacio todo lo que quisiera decir y mencionar a todas las personas que me han ayudado de alguna u otra forma para obtener el grado de químico.

Quisiera agradecer todo el apoyo y ayuda que recibí de mi asesor, amigo y muy buen maestro el Dr. Carlos Amador Bedolla, por las pláticas en Unión Química y Fundamentos Espectroscópicos, de Matemáticas, de Monte Carlo Cuántico, la UNAM, etc. Espero que después de la titulación, todavía me sigas aguantando.

A mi mejor amigo y compadre Edgar Efrén Hernández Prado y familia, por todo el apoyo y cariño, por los años en los que hemos compartido anécdotas y aprendizajes, por la gran amistad que tenemos y brindo por que continuemos creciendo dicha amistad.

A mi gran amiga Arlette Violeta Richaud, porque nos hemos arriesgado a compartir momentos muy padres y momentos muy difíciles, porque tú, junto con otros amigos, me enseñaron que los amigos dan pequeños empujoncitos en los momentos más difíciles para que uno pueda emprender el vuelo. Ojalá y tú puedas emprender el vuelo y que sigamos arriesgándonos y apoyarnos como lo hemos hecho hasta ahora.

A mi amiga Gioconda, por escucharme cuando necesitaba ser escuchado y tratar de dar lo mejor de ti para ayudarme en los tiempos más difíciles. Lograste ayudarme mucho con ese esfuerzo. Ojalá y no te aburras de escucharme y que sigamos divirtiéndonos y aprendiendo uno del otro.

A mi amiga Tatiana, por tu amistad "virtual" en donde a pesar de la distancia y de las diferentes nacionalidades, tenemos una gran amistad y una cita con el correo varias veces a la semana y donde tengo una mejor comunicación contigo que con muchas personas que viven aquí y tú lo sabes.

Al Dr. Roberto Amador (†), por ayudarme a salir más rápido de las circunstancias que tuve que enfrentar. Por mostrarme lo interesante que es conocer a un filósofo, psicoanalista y una de las personas más inteligentes y cultas que he conocido. Por ayudarme a conocerme mejor y a desear el mejor desarrollo de mis potencialidades y desarrollo que pueda.

A mi familia, principalmente mi tía Estela Carrera Martínez. Por estar conmigo en las buenas y en las malas. Por hacerme saber que no estoy solo. Por todos los recuerdos y por los momentos futuros donde estoy seguro que ustedes estarán ahí.

Al Dr. Azrak por permitirme dar asesorías en la materia de Estadística. Porque me permitió conocer a muchos buenos amigos y saber lo padre que es dar clases.

A las universidades públicas, en particular la UNAM y la UAM. Por darme la oportunidad de estudiar becado al no pagar colegiatura o pagar una colegiatura muy pequeña. Porque gracias a estas instituciones pude seguir estudiando después de la crisis del 96 y de no poder pagar la colegiatura de la Universidad La Salle. A la DGSCA, por permitirme cursar el plan de becarios de supercómputo y mostrarme el mundo del cómputo científico. A todos los que conocí en la DGSCA (Enrique, Luis, Eduardo, Marisol, etc.) por aguantarme y por todo el esfuerzo que hacen por mejorar el área.

Y a todos los profesores que me ayudaron a crecer académica y personalmente, al Dr. Jorge Garza, Dr. Cedillo y al Dr. Marcelo por sus comentarios y a todos mis cuates que estuvieron conmigo de una u otra forma (Berenice, Mónica, Adriana, Rosana, Edgar, Alejandro, Sandra, Miroslawa, Abigail, Alberto, etc.)

A mis padres (†), a quienes les dedico el siguiente pensamiento:

Con la mayor gratitud por los esfuerzos realizados para que lograra terminar mis Carreras Profesionales, siendo para mi una de las mejores herencias. A quienes me han heredado el tesoro más valioso que pueda dársele a un ser: Amor. A quienes sin escatimar esfuerzo alguno han sacrificado gran parte de su vida para formarme y educarme. A quienes la ilusión de su vida ha sido convertirme en una persona de provecho. A quienes nunca podré pagar todos sus desvelos y sacrificios ni aún con las riquezas más grandes del mundo y a quienes son un modelo a seguir

Gracias por todo lo que me han dado...

...Siempre estaré orgulloso de ser su hijo.
Con Amor, Respeto y Admiración.
Por ello a Dios y a Ustedes, Gracias.

Edmundo Segundo Carrera Martínez

PRÓLOGO

En 1998, el premio Nóbel de química fue compartido por Walter Kohn y John A. Pople. El comité remarcó que la química cuántica computacional estaba "revolucionando el mundo de la química". Esta revolución ha permitido que investigadores de diversos campos de las ciencias puedan hacer cálculos muy sofisticados. Sin embargo, esta "democratización" ha provocado que se use a la computadora de manera muy ineficiente y como una caja mágica generadora de datos sin el menor análisis, lo que provoca altos costos tanto en recursos como en tiempo.

Para un país como México, donde se necesita aprovechar al máximo los pocos recursos que se tienen para la ciencia y las oportunidades que el cómputo paralelo actualmente ofrece, es muy importante eficientar el uso de la química cuántica computacional y para ello, se necesita un análisis y desarrollo metódico y sistemático de los proyectos de química cuántica computacional. Actualmente, esto no sucede y como Licenciado en computación y egresado del plan de becarios en Supercómputo de la DGSCA-UNAM (Dirección General de Servicios de Cómputo Académica) me parece muy importante atacar este problema.

Debido a esto, se pensó en esta tesis, que está enfocado principalmente a los químicos. Por ello, en el capítulo 2, se ofrece una introducción de las principales variables que se deben de tomar en cuenta en el cómputo paralelo, el cual es el más atractivo actualmente para los cálculos de la química cuántica computacional. En dicho capítulo se hace un pequeño análisis de las principales ventajas y limitaciones que existen actualmente en el cómputo paralelo aplicado a la química cuántica computacional y cómo enfrentarlas. Es muy recomendable profundizar estos temas con la ayuda de la bibliografía, principalmente si se piensa tomar alguna decisión con respecto al hardware, software o humanware.

En el tercer capítulo se hace un recuento de las principales aproximaciones y errores que pueden ocurrir en los cálculos de la química cuántica computacional, cómo enfrentarlas y la importancia de tomarlas en cuenta al momento de hacer las conclusiones. También se menciona la importancia de mejorar la precisión y aumentar el tamaño de los sistemas químicos, algunas implementaciones y programas de la química cuántica computacional y por último se explica cómo llevar a cabo un proyecto de química cuántica computacional.

Para aplicar lo visto en los capítulos 2 y 3, se discuten dos ejemplos, uno que se enfoca en cómo hacer un análisis, administración y documentación necesaria para el uso eficiente de una máquina paralela de bajo costo (cluster); y otro ejemplo donde se enfoca hacia los cálculos, sus implicaciones y las

conclusiones a las que se pueden llegar con éstos en un problema químico actual.

Con estos ejemplos y sus discusiones (capítulos 4 y 5) se puede observar la necesidad de hacer un buen análisis, diseño y uso eficiente de las herramientas que existen para la química cuántica computacional. Dichas observaciones y algunas recomendaciones se encuentran en el capítulo 6.

Por último, en los apéndices, se da una tabla del costo computacional que generan los diferentes métodos, programas más comunes, una pequeña historia de la química cuántica computacional donde se puede observar el avance de los métodos y de la tecnología computacional y al final una lista de los acrónimos más utilizados en la química computacional.

Edmundo

Capítulo 1

INTRODUCCIÓN

A menudo se dice que el desarrollo de las computadoras digitales ha transformado el alcance de la ciencia, porque ello ha originado una tercera metodología de hacer ciencia, detrás de las dos metodologías tradicionales: la experimental y la teórica; ahora se debe añadir la metodología computacional. En el área de cómputo científico se ha hecho énfasis en el estudio sistemático de la utilización de la computadora para resolver una amplia gama de problemas. Las simulaciones por computadora tienen varias ventajas como son:

1. Las simulaciones por computadora por lo regular son mucho más baratas y rápidas que los experimentos físicos.
2. Las computadoras pueden resolver un margen mucho más amplio de problemas que los que podrían resolverse con equipos de laboratorio específicos o tecnología actual
3. Las posibilidades de cálculo sólo están limitadas por la velocidad de la computadora y la capacidad de memoria de ésta, mientras que los experimentos físicos tienen muchas restricciones prácticas y de seguridad

Los científicos teóricos y experimentales son usuarios de los grandes códigos de programas suministrados por los científicos computacionales. Los códigos deben generar resultados precisos y/o señalar el alcance de dichos resultados, con un mínimo esfuerzo por parte del usuario; por lo tanto utilizan principalmente ambientes gráficos. Los científicos computacionales deben aplicar tecnologías avanzadas a la modelación numérica (métodos numéricos), la ingeniería del hardware y del software así como el desarrollo de éstos. Para usar eficientemente una computadora es necesario optimizar el programa de aplicación de acuerdo a las características de la computadora, como una herramienta en el alcance de metas específicas y locales.

Algunas de las principales preguntas que surgen en la búsqueda de esta meta son:

- ¿Cómo debe ser formulado un problema científico o tecnológico de tal manera que se facilite su tratamiento computacional?
- ¿Cómo puede el dominio de un problema ser representado mediante estructuras formales para su procesamiento computacional?
- ¿Qué tipos de arquitecturas de computadoras son las más adecuadas para la solución de un problema específico?
- ¿Qué algoritmos proveen el mejor equilibrio entre exactitud, velocidad y estabilidad computacional?
- ¿Qué herramientas de software existentes ofrecen las mejores expectativas?

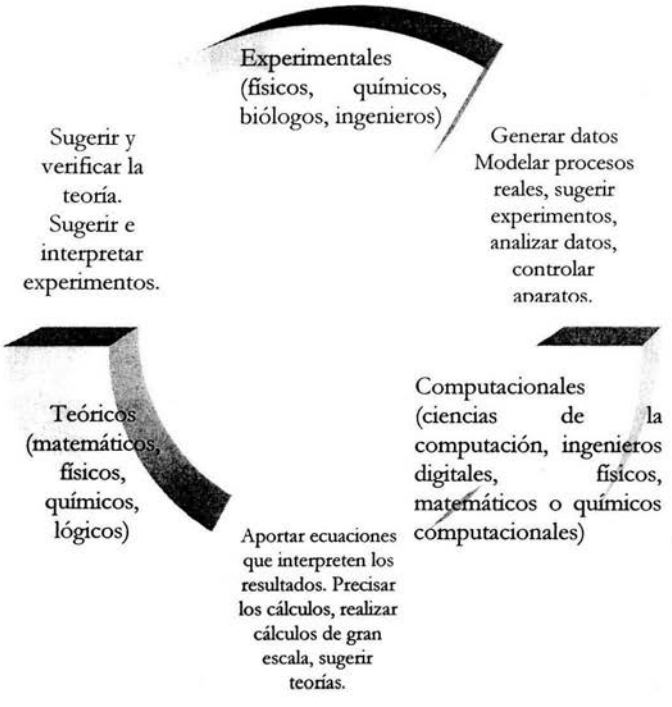


Figura 1.1 Interacción entre experimentos, teorías y el cómputo científico en la resolución de problemas de gran reto.

Las investigaciones científicas más actuales involucran el modelado, la simulación y el control de sistemas del mundo real. Una parte importante del cómputo científico se puede clasificar como cómputo de alto rendimiento (HPC→*High Performance Computing*), es decir, aquel cómputo que requiere de gran demanda de procesamiento de datos en procesadores, memoria y otros recursos de hardware y cuya comunicación entre ellos debe ser muy rápida. Sólo aplicando el cómputo de alto rendimiento, se pueden resolver problemas de gran reto, los cuales son aquellos proyectos difíciles de investigar debido a su gran complejidad y alto número de variables y datos. Las principales ventajas que ofrece el cómputo de alto rendimiento son:

- La posibilidad de poder cambiar los parámetros de una simulación para estudiar tendencias emergentes.
- Repetición de un evento particular de una simulación.
- Estudio de sistemas en los cuales no existe una teoría exacta (métodos heurísticos, aproximados, etc.)

Un campo de la ciencia que utiliza el cómputo de alto rendimiento es la química computacional, la cual ha tenido un gran desarrollo en las últimas décadas y ha afectado diversos campos dentro de las ciencias químicas como:

- la bioquímica (bioinformática)

- fisicoquímica (simulaciones)
- educación química (*software* educativo)
- química cuántica (estructuras electrónicas, propiedades moleculares y dinámicas),
- química analítica (*software* para el manejo de datos químicos)
- cristalografía

Debido al amplio mundo de las ciencias químicas, en esta tesis, solamente se enfocará a la química cuántica (química cuántica computacional), donde existe un amplio desarrollo y relación con las matemáticas, la física, la computación y donde los resultados obtenidos son utilizados por otros campos como la bioquímica, la fisicoquímica, desarrollo de materiales, etc.

Al recordar los inicios de la química cuántica, en 1929 Dirac escribió:

"Las leyes de las física que subyacen en la teoría matemática de ...la química en su totalidad, son, pues, completamente conocidas, y la única dificultad existente es que la aplicación de estas leyes conduce a ecuaciones demasiado complicadas de resolver."

Lo que muestra que en la primera mitad del siglo pasado se creía que era imposible realizar cálculos ab initio significativos de propiedades moleculares, salvo para moléculas muy pequeñas. Actualmente, se pueden hacer cálculos muy sofisticados con el uso computadoras muy poderosas lo que ha provocado superar, en cierto grado, las dificultades señaladas por Dirac y que los cálculos mecanocuánticos sean una herramienta válida para ayudar a responder una amplia variedad de interrogantes de interés químico.

Estos avances han generado una "democratización" de los cálculos mecanocuánticos, de tal manera que investigadores no especialistas en química teórica, ciencias de la computación y mucho menos en química computacional, puedan generar cálculos muy sofisticados utilizando a la computadora como una caja negra que da resultados fiables o exactos. Este es uno de los peores errores que un químico puede cometer ya que los trabajos de la química cuántica computacional están basados en aproximaciones que pueden ser muy crudas y dar resultados que no se acerquen a la solución o pueden ser tan precisas que ni un experimento con la tecnología actual puede mejorarlo. Para obtener resultados que se aproximen lo más posible a la solución, es necesario entender las diferentes aproximaciones que se utilizan, qué método será el más adecuado para el cálculo requerido y cómo la herramienta (computadora) puede ayudar a dichos propósitos.

Para ello, es necesario hacer diferentes análisis. Comúnmente, un investigador sólo hace el análisis de los resultados obtenidos lo que provoca que muchas veces se publiquen datos erróneos al no considerar los errores que generan las aproximaciones, el redondeo, el desbalanceo y mala descripción de las bases, la mala programación, introducción de datos de inicio erróneos, etc. Debido a esto, es muy importante conocer las variables que afectan nuestros cálculos, los cuales tienen su origen en las ciencias de la computación y en las ciencias químicas. El utilizar las herramientas que ofrecen dichos campos del conocimiento para aprovechar al máximo las bondades que ofrecen y conocer las limitaciones para evitarlas, es un paso muy importante para hacer investigaciones de una manera metódica y sistemática y obtener así resultados con el menor tiempo de cómputo e investigador.

Otro punto importante es que a pesar de las poderosas herramientas que existen actualmente para hacer un cálculo mecanocuántico, la química cuántica computacional todavía se encuentra muy lejos de poder hacer cálculos que tomen en cuenta todos los átomos involucrados en una reacción química (número de Avogadro), obtener resultados precisos para sistemas que sean influenciados por campos externos como radiación, métodos que puedan simular de manera precisa fenómenos dependientes del tiempo o de relajación en una escala de tiempo mayor a los femtosegundos, etc. Además, los químicos, requieren de la precisión química (1 kcal/mol) ya que esta precisión es suficiente para explicar las fuerzas de Van der Waals, puentes de hidrógeno y todas las interacciones débiles que afectan un sistema químico y que son muy importantes en la diferencia relativa entre, por ejemplo, dos confórmeros.

Para que la química cuántica computacional pueda lograr estos objetivos, es necesario que se tomen en cuenta todas las herramientas del cómputo de alto rendimiento, desarrollar nuevos métodos e implementarlos de tal manera que puedan explotar lo mejor posible el hardware y software que se tiene y que se podría tener, utilizar los algoritmos y estructuras que las ciencias de la computación ha desarrollado, nuevas tecnologías como los Grids, metodologías de análisis y desarrollo de software así como la vinculación con los nuevos datos que se van obteniendo de la química. De aquí la importancia de ver la evolución que ha tenido la química cuántica computacional ya que se podrá observar los avances que han existido y el porque se cree que métodos como Monte Carlo Cuántico o el Método Finito serán los que más se desarrollarán en el futuro al utilizar el cómputo masivamente paralelo y el uso de las nuevas tecnologías.

Por ello, el objetivo de esta tesis es dar a conocer las variables más importantes que se deben de tomar en cuenta al tener un proyecto de química cuántica computacional, cómo manejarlas y aprovechar al máximo todas las herramientas que los diferentes campos de la ciencia nos puedan dar, en especial la química cuántica y el cómputo paralelo ya que este último es la técnica más atractiva, principalmente por el costo, para poder resolver problemas de gran reto en la química cuántica computacional. Y en particular el uso de clusters, los cuales han tenido últimamente un gran desarrollo por el bajo costo, en comparación con las supercomputadoras, pero donde dicho gasto económico en hardware se tiene que invertir en tiempo y costo de administración, mantenimiento y puesta a punto como se explicará en más detalle en este trabajo.

Capítulo 2

LA QUÍMICA CUÁNTICA COMPUTACIONAL, EL CÓMPUTO PARALELO Y SUS IMPLICACIONES

2.1 Los avances en las ciencias de la computación

La eficiencia de una computadora depende directamente del tiempo requerido para ejecutar una instrucción básica y del número de instrucciones básicas que pueden ser ejecutadas concurrentemente. Esta eficiencia puede ser incrementada por avances en la arquitectura y por avances tecnológicos. Avances en la arquitectura incrementan la cantidad de trabajo que se puede realizar por ciclo de instrucción como por ejemplo el uso de memoria bit-paralela, aritmética bit-paralela, múltiples unidades funcionales, ejecución especulativa, ejecución fuera de orden, *pipelining* de instrucciones y de datos, la memoria *caché*, etc. Una vez incorporados estos avances de la arquitectura, mejorar la eficiencia de un procesador implica reducir el tiempo de los ciclos mediante los avances tecnológicos.

Hace un par de décadas, los microprocesadores no incluían la mayoría de los avances de arquitectura que ya estaban presentes en las supercomputadoras. Esto ha causado que en los últimos años, el adelanto visto en los microprocesadores haya sido significativamente más notable que el de otros tipos de procesadores: supercomputadoras, *mainframes*, etc. El crecimiento en la eficiencia para las supercomputadoras y mainframes ha estado por debajo del 20% por año, mientras que para los microprocesadores ha sido de un 35% anual en promedio; lo que ha provocado que en la evolución del cómputo serial y paralelo exista cada vez un menor costo, observándose una comercialización de bajo costo principalmente debido a los clusters; aunque hay que observar que el menor costo en hardware de un cluster representa un mayor costo humano en su mantenimiento y administración.

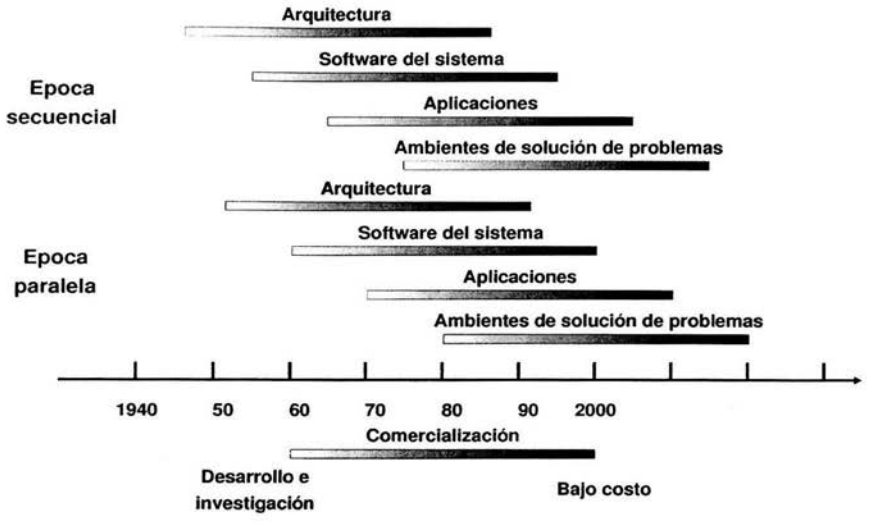


Figura 2.1 Épocas del cómputo.

La ley de Moore (1965) dice que el número de transistores por pulgada en los circuitos integrados, se duplicaría cada 18 a 24 meses, logrando así una mayor capacidad y un menor costo; lo que describe un crecimiento exponencial en la densidad de los transistores. Aunque esta ley se aplica principalmente al microprocesador; también se aplica a la memoria y al ancho de banda de las redes (aunque con una proporción menor) y ha permanecido vigente a pesar de las dificultades físicas; gracias al desarrollo de nuevos materiales, sistemas de enfriamiento y de nuevas tecnologías de producción.

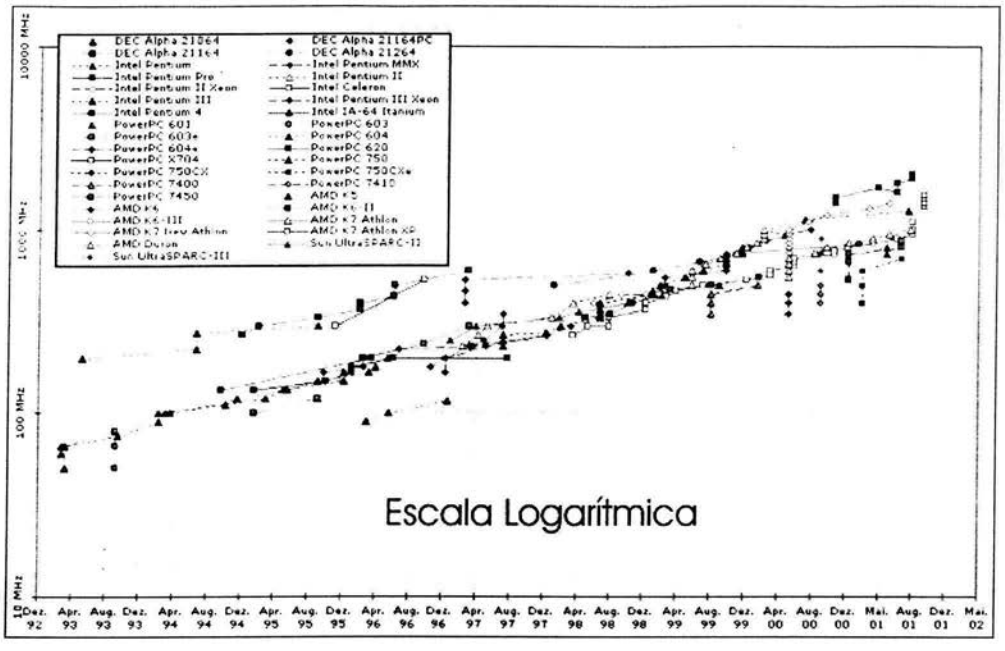


Figura 2.2 Historia en Mhz.

Sin embargo y a pesar de los avances que se han tenido, parece que se están alcanzando límites físicos como la velocidad de la luz; por lo tanto, no se puede depender completamente de procesadores más rápidos para obtener mayor eficiencia. Dadas estas dificultades en mejorar la eficiencia de un procesador y del *hardware* en general, a la convergencia relativa en eficiencia entre microprocesadores y las supercomputadoras tradicionales y al relativo bajo costo de los microprocesadores, ha habido un gran desarrollo de computadoras paralelas viables comercialmente con decenas, cientos y hasta miles de procesadores. Esto ha logrado que para la química computacional, el cómputo paralelo sea actualmente una solución atractiva para los problemas de gran reto y es por ello que en este capítulo, se enfocará casi exclusivamente a sus alcances y limitaciones, dándose una breve introducción de los conceptos más importantes y las implicaciones de éste; las cuales serán aplicadas en los capítulos 4 y 5.

Otro punto importante a tomar en cuenta es el desarrollo del *software*. El análisis y diseño de *software* no es una tarea trivial y es más difícil cuando se trata de programas paralelizados; ya que existen una mayor cantidad de variables que se deben de tener en cuenta para un buen análisis, diseño y mantenimiento del software creado, por lo tanto, éste es un proceso altamente creativo y se debe de seguir una buena metodología para obtener así el software que mejor satisfaga las expectativas y/o necesidades.

Pero antes que nada, es importante saber las implicaciones que trae el paralelismo a un problema y *software* determinado. El cómputo paralelo parece directo; aplicar múltiples procesadores a un problema para resolverlo más rápido, más realista y complejo (con más variables), más grande (mayores datos) o con resolución más fina. Desgraciadamente, esto no ocurre así en la gran mayoría de las

ocasiones ya que el cómputo paralelo involucra una curva de aprendizaje muy empinada, un esfuerzo intensivo del programador para pensar nuevas maneras de resolver el problema de manera paralela lo que puede conllevar a que se escriba totalmente el código; es decir, el código que corre de manera serial ya no serviría, a tomar en cuenta el ambiente de ejecución (arquitectura de la máquina paralela, balance de carga), etc. Además, las técnicas usadas para depurar y mejorar el rendimiento de un programa serial no se extienden fácilmente en el mundo paralelo lo que puede provocar que se trabaje meses en paralelizar una aplicación sólo para encontrar que da resultados incorrectos o que corre más lentamente.

2.2 Análisis del problema: Precondiciones para el paralelismo

Entonces, ¿qué hacer? El propósito y la naturaleza de la aplicación son los indicadores más importantes para saber que tan exitoso puede ser la paralelización. La máquina paralela en la que se trabaja así como el plan de ataque para la paralelización tendrán un significativo impacto en el rendimiento del programa y en el esfuerzo que se hará para ello. En la siguiente figura se muestran las precondiciones necesarias para saber si la paralelización es necesaria y viable y obviamente los tiempos dependen en gran medida del tipo de problema a resolver.

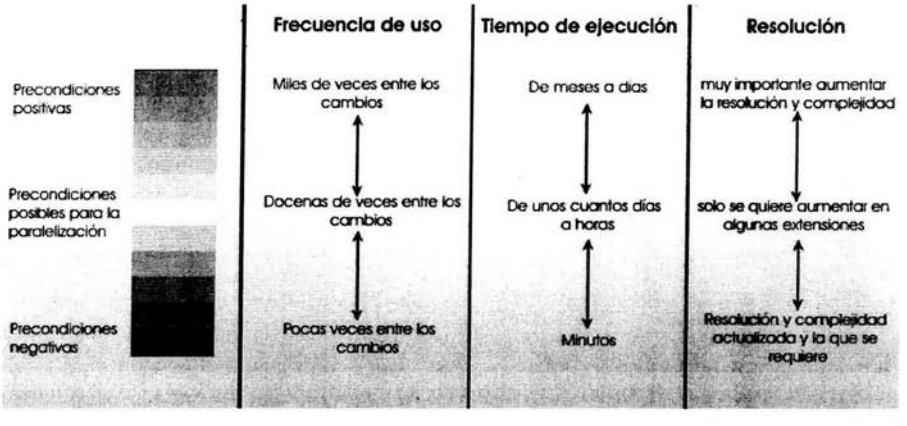


Figura 2.3 Las precondiciones para el paralelismo. ¿Qué rendimiento se necesita?

2.3 Análisis del problema: Arquitecturas del problema

El siguiente punto a analizar son las "arquitecturas del problema" los cuales son categorías que se tienen con base en las características de las aplicaciones. Estas son:

1. **Embarazosamente paralelos:** Son aquellas aplicaciones que pueden ejecutarse como una sucesión de programas seriales, los cuales no necesitan comunicación entre sí; es decir, los cálculos en cada conjunto de datos (programa serial) son totalmente independientes. Son los programas más fáciles de tratar y el programa paralelo ideal.

2. **Paralelismo pipeline:** Son aquellas aplicaciones donde después de determinados cálculos, se necesita que exista comunicación entre los procesos para seguir con los cálculos y así sucesivamente. Lo importante de éste tipo de aplicaciones es que los datos calculados; inmediatamente son requeridos para los siguientes cálculos después de la comunicación y ya no se necesitarán más. Esto simula una especie de tubería o cauce de la información la cual puede ser paralelizada cuando distintos procesos hacen las diferentes etapas siguiendo el modelo de procesamiento pipeline. No todas las etapas tienen la misma complejidad y en consecuencia el mismo tiempo de cómputo; por eso es importante hacer un buen balanceo entre las etapas y el poder de cómputo que se tiene; por lo tanto el diseño del programa paralelo requiere más esfuerzo que el paralelismo embarazoso.

3. **Paralelismo totalmente síncrono:** Son aquellas aplicaciones donde no existe un flujo único de la información y los datos; es decir, los cálculos de una región del problema afectan en cierto grado todo el espacio de éste, esperando así que terminen todos los cálculos en todas las regiones para continuar con la siguiente iteración. Lo importante de éste tipo de aplicaciones es que los futuros cálculos o decisiones dependen de todos los cálculos hechos anteriormente. El paralelismo se introduce cuando varios procesos participan en una iteración haciendo cálculos para diferentes subconjuntos de datos. Aquí es muy importante la sincronización de los procesos así como un buen balanceo de carga de trabajo para que no existan tiempos donde varios procesos no hacen cálculos esperando que terminen otros procesos; es por ello que se necesita un mayor esfuerzo al momento de paralelizar para obtener un buen rendimiento en las aplicaciones que en el paralelismo pipeline.

4. **Paralelismo flojamente síncrono:** En éstas aplicaciones los cálculos hechos afectan las decisiones y cálculos futuros como en el tipo de paralelismo anterior; pero además, no existe una sincronización determinada ya que la cantidad de cómputo necesario depende fuertemente de los valores iniciales y a la frontera del problema y del tiempo en que se lleva a cabo la aplicación ya que pueden aparecer nuevas variables y variar la complejidad conforme avanza el cálculo provocando muchas veces que el avance del cálculo sea irregular e imprevisible. El paralelismo se introduce dividiendo el trabajo entre varios procesos por cada tiempo de cómputo. Este es el tipo de aplicación que necesitará mayor esfuerzo para su paralelización ya que es muy importante la sincronización, el balanceo de carga y las comunicaciones que deben de existir entre los diferentes procesos para

intercambiar información vital y continuar con el cálculo. Es muy común encontrar que un proceso produce subprocesos para mejorar la sincronización y el balanceo de carga; lo que genera mayor comunicación para que cada proceso pueda determinar si los valores obtenidos pueden ser o no sobrescritos. La distribución de trabajo es lo más difícil de lograr ya que la carga de trabajo varía espacial y temporalmente.

Desgraciadamente la mayoría de los problemas de gran reto actuales caen en el paralelismo flojamente síncrono y la química computacional no es la excepción; es por ello, que si realmente se necesita hacer el esfuerzo para paralelizar con este tipo de arquitectura de problema, se debe tomar en cuenta cómo el cómputo (así como los datos) se comportan durante la ejecución de la aplicación y con esto analizar que tipo de máquinas ayudará a tener un mejor rendimiento.

2.4 Análisis del problema: Arquitecturas de las máquinas paralelas

La arquitectura de la máquina que se utilice afectará en gran medida el rendimiento y comportamiento de nuestra aplicación. En las máquinas SIMD (*Simple Instruction, Multiple Data*→clasificación de Flynn) la llave es la unidad de control ya que se pueden lograr grandes rendimientos de programas con el procesamiento pipelined, con el estilo de arreglos de operaciones de Fortran o que se hagan llamadas a librerías optimizadas para arreglos; de tal manera que el compilador automáticamente genere el código paralelo optimizado. Para problemas con paralelismo *pipeline* y totalmente síncronos, esta arquitectura funciona bien; aunque existen algunos detalles; como por ejemplo, los arreglos generalmente no tienen el mismo tamaño que CPUs por lo tanto hay que utilizar técnicas de optimización para lograr que todos los CPUs trabajen al momento de hacer las operaciones con los subarreglos que se generen; otro ejemplo son las operaciones condicionales donde se hacen todas las operaciones para un arreglo, y donde si en un dato de los arreglos no se cumple la condición, entonces todo el esfuerzo se concentrará en un CPU, que será el que defina la condición. Otros problemas son la dependencia de los datos, que puede provocar que los resultados finales sean erróneos.

Para máquinas MIMD (*Multiple Instructions, Multiple Data*→clasificación de Flynn) con memoria compartida los ciclos intensivos pueden ser paralelizados y optimizados de tal manera que tendrán un gran rendimiento, convirtiendo estos ciclos en una colección de ciclos para un subconjunto de datos para cada CPU aprovechando así la memoria compartida y que los procesos no se preocupen en las actividades de los otros CPUs; hay que seguir las técnicas de optimización para que el compilador y el programador pueda generar código optimizado en los ciclos y aprovechar la arquitectura de la máquina. Aquí es muy importante que el programador se preocupe por la dependencia de los datos y salvaguardar los datos compartidos para lograr así la coherencia en los datos. En el caso del paralelismo totalmente síncrono, a veces no es tan sencillo optimizar la aplicación para este tipo de máquinas ya que el acceso a datos es esporádico y muy interdependientes. Sin embargo, para aplicaciones embarazosamente paralelas y *pipeline* puede ser muy útil.

Para máquinas con memoria distribuida, la paralelización, depuración y optimización suele requerir un gran esfuerzo; sin embargo, este tipo de máquinas son las que mas se están desarrollando y las que mejor costo/beneficio tienen. En estas máquinas suelen duplicarse los datos en cada memoria del

CPU que lo requiera siendo muy importante la coherencia de los datos durante la comunicación entre los procesos vía mensajes. También es muy importante tener en cuenta los problemas que surgen con la comunicación y protección de datos compartidos (como mensajes corruptos o perdidos, bloqueos mutuos, etc.). Además, el balance entre la velocidad de CPU (altas) y la velocidad de las comunicaciones (relativamente bajas y costosas) es crítico y por ello, las métricas de rendimiento juegan un papel muy importante en las aplicaciones que se ejecutan en estas máquinas. Además, es conveniente reducir los tiempos de comunicación (o las comunicaciones) y mejorar la sincronización para mantener los CPUs ocupados. Las aplicaciones totalmente síncronas son impropias para este tipo de arquitecturas y las aplicaciones flojamente síncronas y *pipeline* pueden lograr un buen rendimiento si las comunicaciones entre los procesos se dan con datos pequeños y/o lapsos de tiempo relativamente largos entre las comunicaciones.

Para máquinas SMP (*Symmetric Multiprocessor*), cada nodo tiene un número par de microprocesadores (por lo regular 4 y 8) que tienen memoria compartida, pero entre cada nodo hay memoria distribuida. Se dice que son simétricos ya que dentro de un nodo, cada procesador puede acceder a una locación de memoria con la misma latencia. El mejor rendimiento se obtiene por lo regular cuando se tratan estas máquinas como una colección de distintos sistemas de memoria compartida en pequeña escala; es decir, aprovechar al máximo cada nodo y evitar las comunicaciones entre los nodos. Por ejemplo generar hilos (*threads* con OpenMP) que aprovechen al máximo un nodo y varios procesos paralelos que vean a la máquina como una arquitectura con memoria distribuida. Aplicaciones flojamente síncronas y *pipeline* pueden funcionar bien en esta arquitectura. Para lograr el mayor aprovechamiento de la arquitectura de la máquina, la programación, el lenguaje y las bibliotecas utilizadas son muy importantes.

El cómputo paralelo ha tenido un gran avance en los últimos años, principalmente debido a lo barato y rápido que están siendo los microprocesadores, logrando que, por ejemplo los *clusters*[1] sean una herramienta alcanzable para la mayoría de los investigadores. Sin embargo, nunca hay que olvidar que en un *cluster* no se tienen las ventajas de una supercomputadora, por ejemplo, el rendimiento de éste baja drásticamente si se tiene más de un trabajo corriendo al mismo tiempo y que dichos trabajos compitan por los recursos del *cluster*; por ello, lo mejor es tener subdividido el *cluster* de tal manera que se puedan correr varios programas; pero sin que éstos compitan por los recursos del sistema y muy comúnmente esto se logra con *software* de administración como las colas y también mediante el trabajo de un buen administrador que sepa mantener el *cluster* en producción; el cual genera un costo mayor en el personal, aunque se ahorra en costo de *hardware*. Otro problema, además de la administración, es la optimización, ya que ésta es más difícil en un *cluster* que en una supercomputadora, comenzando desde las bibliotecas numéricas, compiladores, herramientas de análisis, etc.

2.5 Análisis del problema: Lenguajes, librerías y utilerías

Definitivamente, una vez que se ha analizado la arquitectura del problema y de la máquina, el lenguaje de programación, las capacidades del compilador, las utilerías de depuración y las utilerías para ver el

[1] Un cluster son un conjunto de máquinas de arquitecturas homogéneas o heterogéneas conectadas en red y usadas como un recurso de cómputo unificado para atacar problemas de cómputo paralelo o distribuido a bajo costo.

comportamiento del programa durante la ejecución, afectarán en gran medida el esfuerzo que se tenga que hacer para paralelizar y optimizar la aplicación. Básicamente, existen dos modelos de programación paralela: MPMD (*multiple program, multiple data*) donde se crean diferentes ejecutables por cada CPU y SPMD (*single program, multiple data*) donde todas las instrucciones para todos los CPUs son combinados desde un solo ejecutable.

Para el modelo SPMD, cada CPU ejecuta el mismo código objeto. Puede ejecutar diferentes instrucciones en diferentes CPUs con la ayuda de una operación condicional pero por lo regular ejecuta el mismo programa en diferentes datos. En máquinas SIMD, quizá es la única opción y para máquinas MIMD el programador sólo tiene un programa para depurar y ver su rendimiento, algo que puede ser una ventaja. Pero puede haber un costo, todos los datos e instrucciones deben ser obtenidos por todos los CPUs eficazmente lo que aumenta considerablemente la memoria requerida y a menudo el tiempo de acceso a la memoria se vuelve un cuello de botella para el rendimiento de la aplicación.

Para el modelo MPMD, la cual por lo regular sólo aplica a máquinas MIMD, utiliza el espacio de memoria más eficientemente y los requerimientos de espacio de código son reducidos para aplicaciones con paralelismo *pipeline* y flojamente síncrono. El espacio de datos puede ser reducido cuando se trabajan con grandes arreglos, si el programador los subdivide en porciones que sean accesibles sólo para los CPUs que los requieran. Para la depuración y optimización del rendimiento, el programador verá los programas independientemente o como componentes de otros programas, lo que nos ayudará con la modularidad y división del trabajo. Sin embargo, esto no siempre ayuda ya que pueden surgir ciertos tipos de problemas para la puesta a punto del programa que son difíciles de conceptualizar; por ejemplo, cómo las actividades de los CPUs independientes pueden influirse entre sí.

La gran mayoría de las máquinas paralelas, imponen el modelo SPMD debido a que sus sistemas operativos y utilerías ven al programa paralelo como una sola entidad y no pueden reportar información de múltiples ejecutables que se encuentren relacionados. Por lo tanto, el programador, rara vez tiene varias opciones. Los lenguajes, librerías y utilerías por lo regular están limitados a tipos particulares de máquinas y quizás fabricantes. Las librerías de envío de mensajes (principalmente MPI y PVM) son las más portables; pero con ello puede inhibir optimizaciones y capacidades de detecciones de errores de los compiladores. También, la opción de los lenguajes a utilizar, deben de estar encaminados al compilador más robusto, a la especialización que se tenga sobre dicho lenguaje y colegas que los han usado, a los reportes de rendimiento que se tengan sobre las bibliotecas matemáticas en la máquina a utilizar, etc.

Para máquinas vectoriales (SIMD), el mejor rendimiento se obtiene para aplicaciones con arquitecturas *pipeline* y totalmente síncronos utilizando Fortran ya que este lenguaje maneja muy bien los arreglos. Para aplicaciones con arquitecturas flojamente síncronos puede ser más importante ver la especialización de los programadores en un determinado lenguaje. Por lo regular, los estudiantes de las ciencias de la computación, tienen una mejor especialización en C y C++; mientras que los investigadores manejan Fortran, una solución es que partes del programa sean escritos en Fortran (las rutinas que tengan que ver con las bibliotecas matemáticas principalmente) y otras en C y C++ (las rutinas que tengan que ver con el control del programa, interfaces, etc.) para una mejor comunicación entre los especialistas en la computación y los especialistas en el problema. La ingeniería de software no puede ser olvidada para programas que se piensan paralelizar y para ver el balance entre la ingeniería de software y la optimización, no hay que olvidar que el 20% del código de las aplicaciones de

problemas de gran reto ocupan el 80% del tiempo de cómputo. Es por ello que el análisis y diseño del programa debe ser tomada con toda seriedad.

Existen algunos compiladores que pueden paralelizar automáticamente un software serial, sin embargo, el rendimiento obtenido por éstos es muy pobre e incluso menor al del programa serial. También los compiladores pueden optimizar automáticamente un programa, sin embargo, siempre es muy importante que el programa se estructure de tal manera que le compilador no tenga problemas en optimizar el software, como por ejemplo, evitar la dependencia de datos, las decisiones dentro de los ciclos, etc; principalmente si los programas son escritos en C y C++ donde los apuntadores juegan un papel muy importante. Quizá el compilador más robusto para la optimización es Fortran y casualmente se puede obtener un buen rendimiento al hacer un programa con éste lenguaje; sin embargo, este lenguaje no permite muchos de las bondades que permiten lenguajes como C y C++; principalmente en el manejo de memoria, plantillas (templates), etc. Con estos lenguajes, un buen rendimiento es más difícil de obtener; pero con un buen análisis y diseño del programa, éste puede ser tan eficiente como si fuera escrito en Fortran pero con las facilidades de que se puede tener un mejor mantenimiento, mayor robustez, mayor reutilidad y tener un ciclo de vida mayor. Por ello, es muy importante que el investigador que utilice programas de química computacional, entienda el funcionamiento y rendimiento del *software* paralelo de tal manera que se pueda eficientar su trabajo.

Es muy importante tomar en cuenta la precisión numérica ya que esta es especialmente importante para aplicaciones de aritmética de punto flotante como ocurre en la química cuántica computacional con el uso del álgebra lineal. Para la aritmética interna, el truncamiento por default, contrario a la opinión popular, es usualmente el truncamiento o redondeo hacia abajo, a menos que un tipo diferente se especifique y se requiera. Mientras más grande sea la palabra para la representación de los números, los cálculos serán más precisos, es decir, las aproximaciones de los diversos eventos simulados se acercarán más al valor correcto. Por lo tanto, el error tenderá al épsilon de la máquina (principalmente al usar el método variacional), es decir, el menor valor numérico al que la computadora puede hacerle una división y que es muy importante tomar en cuenta para los errores de redondeo.

2.6 Diseño del programa

En la metodología que se verá para el diseño de programas paralelos, durante sus dos primeras etapas, contempla un análisis que es independiente de los aspectos específicos de la máquina en donde se ejecutará la aplicación y en las últimas dos etapas sí se tomarán en cuenta. Las etapas son: partición, comunicación, agrupación y asignación. En las dos primeras etapas se enfocará en la concurrencia y escalabilidad que mejor se pueda encontrar; en las últimas dos etapas la atención recaerá en la localidad y en el funcionamiento relacionado a la máquina. Aunque las etapas se presentan como secuenciales, en la práctica no lo son, ya que por lo regular se dan de manera paralela y además las últimas etapas afectan a las primeras. A continuación se hace un resumen de las cuatro etapas:

1. Partición: El cómputo y los datos sobre los cuales se opera se descomponen en pequeñas tareas. Se ignoran aspectos como el número de procesadores de la máquina a usar y se concentra la atención en encontrar oportunidades de paralelismo. La partición se puede dar por el dominio (datos) o funcional (cálculos).

2. Comunicación: Se determina la comunicación requerida para coordinar la ejecución de las tareas. Se definen estructuras y algoritmos de comunicación.

3. Agrupación: El resultado de las dos etapas anteriores es evaluado en términos de eficiencia y costos de implementación. De ser necesario, se agrupan tareas pequeñas en tareas más grandes para mejorar el rendimiento y reducir los costos de desarrollo y comunicación.

4. Asignación: Cada tarea es asignada a un procesador tratando de maximizar la utilización de los procesadores y de reducir el costo de comunicación. La asignación puede ser estática o en tiempo de ejecución mediante algoritmos de balanceo de carga.

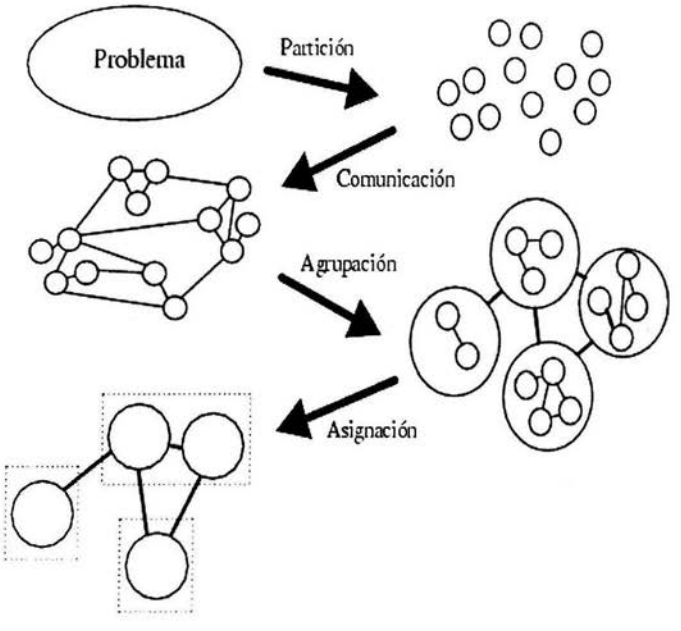


Figura 2.4 Etapas para el diseño de un programa paralelo.

En el diseño y análisis de un programa, por lo regular éste se divide para tener pequeños módulos que permiten encapsular complejos aspectos del programa, reutilizar código lo cual es muy importante para el mantenimiento, etc, así como permitir atacar el problema de tal manera que se enfrente con pequeños problemas sencillos en lugar de un problema grande y complejo, reduciendo costos y aumentando la fiabilidad del programa. Las técnicas del diseño modular nos ayudan a lograr todo esto, creando módulos que cumplan con las siguientes características:

- interfaz simple el cual reducirá el número de interacciones que deben ser consideradas cuando se verifique el buen funcionamiento del sistema y facilitará el reuso de este componente en diferentes circunstancias
- buen encapsulamiento de la información el cual reducirá el costo de subsecuentes cambios de diseño así como facilitará la tarea del entendimiento del problema en general
- uso de herramientas apropiadas, es decir, uso de un paradigma y lenguaje de programación que permita hacer todo ello y que se adecuen al tipo de análisis, diseño y metodología que se están haciendo. Los lenguajes modernos como Fortran90, C, C++ y Ada lo permiten hacer, la decisión también dependerá de la experiencia en programación que se tenga.

Con un buen diseño modular se obtienen módulos bien definidos, los cuales tienen un propósito claramente definido y cuyas interfaces son simples y lo suficientemente abstractas que encapsulen bien la información de tal manera que no sea necesario pensar en cómo se programó el módulo para entenderlo. Obviamente no debe haber módulos que repliquen funcionalidad y el reuso de módulos debe ser algo sencillo.

En la programación paralela, además se necesita considerar el número de procesos creados por módulos, la manera en que cada estructura de datos es particionada y asignada a los procesadores y el tipo de comunicación que existirá. También se debe pensar cómo se ejecutarán los módulos, si los módulos correrán como si fuera un programa secuencial donde diferentes componentes del programa se ejecutan en secuencia en todos los procesadores (llamándose composición secuencial), permitiendo así la distribución de los datos siendo muy útil para el modelo de programación paralela SPMD y que ha sido muy utilizado para la gran mayoría de los proyectos de bibliotecas matemáticas paralelas como ScaLAPACK; si los módulos correrán concurrentemente en diferentes procesadores (llamándose composición paralela) donde se mejorará la escalabilidad y localidad; y por último si los módulos correrán concurrentemente en los mismos procesadores (llamándose composición concurrente), con la ejecución de un módulo particular habilitada por la disponibilidad de los datos, lo que puede reducir la complejidad del diseño y puede permitir la superposición del cómputo y la comunicación.

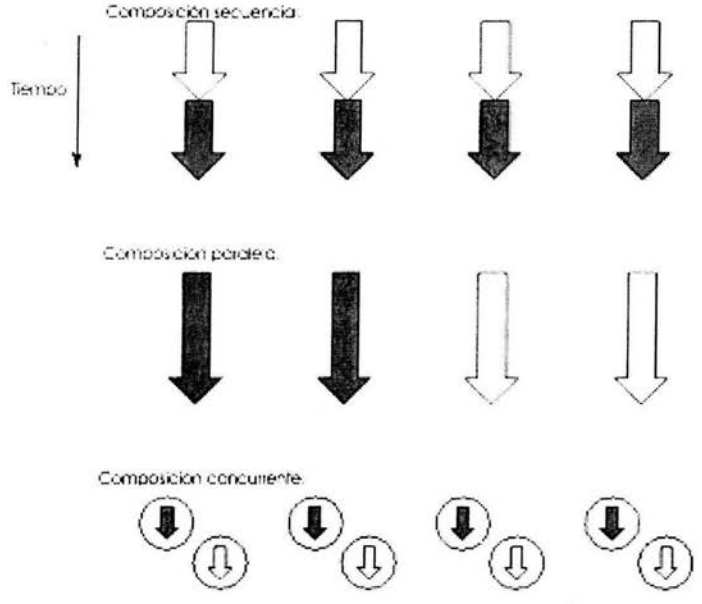


Figura 2.5 Ejecución de los módulos de un programa paralelo.

2.7 Métricas de rendimiento.

En la programación paralela, como en otras disciplinas de ingeniería, la meta del proceso de diseño sirve para lograr un óptimo tiempo de ejecución, menores requerimientos de memoria, de costos de implementación, de costos de mantenimiento, etc. Para ello, se toman en cuenta factores como simplicidad, rendimiento, portabilidad, etc. Todo ello se logra con varios modelos matemáticos de rendimiento, los cuales dan cierta información con las métricas de rendimiento, como la eficiencia de diferentes algoritmos, evaluar la escalabilidad, identificar cuellos de botella y otras ineficiencias, etc; y con ellos se puede ver dónde se necesita trabajar más para optimizar la aplicación y todo ello sin hacer un costo sustancial de implementación.

Por rendimiento de una computadora se entiende la efectividad del desempeño de una computadora, usualmente sobre una aplicación o un benchmark en particular. Esta noción de rendimiento incluye velocidad, costo y eficiencia. Algunos de los factores que afectan el rendimiento de una computadora son:

- Tiempo de ejecución de CPU para:
  a) operaciones de registros.
  b) operaciones de enteros.
  c) operaciones de punto flotante.
  d) operaciones en cadena.
- Tiempo de acceso a memoria para leer o escribir datos (recordando la jerarquía de la memoria donde se empiezan con memorias muy rápidas pero de tamaño pequeño en el mismo

microprocesador; hasta memorias muy grandes pero muy lentas). La memoria siempre es más lenta que la velocidad del microprocesador. La jerarquía de la memoria es:

a) en caché (dentro del microprocesador)
b) en memoria principal (RAM)
c) en memoria auxiliar (en discos duros, externos, etc.).

- Tamaño y divisiones de la memoria.
- Sistemas de archivos.
- Compiladores.
- Entrada y salida de datos.
- Comunicación, especialmente respecto a las computadoras paralelas, etc.

Dependiendo del tipo de aplicación y los requisitos de rendimiento, el buen funcionamiento de la aplicación dependerá del tiempo de ejecución, escalabilidad, mecanismo por el cual los datos son generados, guardados, transmitidos en las redes de interconexión, obtenidos y guardados del disco, etc; además de que se deben considerar costos que ocurren en las diferentes fases del ciclo de vida del software, incluyendo diseño, implementación, ejecución, pruebas, mantenimiento, etc. Por lo tanto, las métricas para medir el rendimiento de nuestro programa pueden ser tan diversas como: tiempo de ejecución, eficiencia de paralelismo, requerimientos de memoria, latencia, *throughput* (cantidad de datos que se pueden transmitir por segundo), relación *inputs/outputs*, costos de diseño, implementación, pruebas, potencial de reuso, requerimientos de *hardware*, costos de *hardware*, portabilidad, escalabilidad, etc. Por lo regular, sólo se necesita trabajar en las métricas que más afectan el rendimiento y muy comúnmente las más importantes son el tiempo de ejecución y la escalabilidad.

2.8 Algoritmia: Análisis asintótico

Al resolver un problema, es posible que estén disponibles varios algoritmos adecuados. Evidentemente, se desea seleccionar el mejor. Esto plantea la pregunta de cómo decidir entre varios algoritmos cuál es preferible y esta pregunta es muy común en la química computacional. Si solamente se tiene que resolver uno o dos casos pequeños de un problema más bien sencillo, quizá no importe demasiado qué algoritmo utilizar, en este caso se podría decidir seleccionar sencillamente el que sea más fácil de programar o uno para el cual ya exista un programa, sin preocupaciones por sus propiedades teóricas. Sin embargo, si se tienen que resolver muchos casos, o si el problema es difícil, como ocurre en la química computacional, se tiene que seleccionar de forma más cuidadosa.

El enfoque empírico (a posteriori) para seleccionar un algoritmo consiste en programar las técnicas competidoras e ir probándolas en distintos casos con ayuda de una computadora. El enfoque teórico (a priori) consiste en determinar matemáticamente la cantidad de recursos necesarios para cada uno de los algoritmos como función del tamaño de los casos considerados. Los recursos que más interesan son el tiempo de cómputo y el espacio de almacenamiento, siendo el primero normalmente el más importante. En el apéndice A se tiene dicho análisis para los diferentes métodos de la química computacional. En caso de un algoritmo paralelo, otro recurso muy importante es el número de procesadores que se necesitan. La ventaja de la aproximación teórica es que no depende ni de la computadora que se esté utilizando, ni del lenguaje de programación; ni siquiera de las habilidades del programador. Se ahorra, tanto el tiempo que se habría invertido innecesariamente para programar un

algoritmo ineficiente, como el tiempo de máquina que se habría desperdiciado comprobándolo. Lo que es más significativo, se permite estudiar la eficiencia del algoritmo cuando se utiliza en casos de todos los tamaños y número de procesadores, es decir, determinar la actuación en una región mayor; de lo que es un espacio multidimensional grande y complejo. Esto no suele suceder con la aproximación empírica, en la cual las consideraciones prácticas podrían obligar a comprobar los algoritmos sólo en un pequeño número de ejemplares arbitrariamente seleccionados y de tamaño moderado.

También resulta posible y muy útil analizar los algoritmos utilizando un enfoque híbrido, en el cual la forma de la función que describe la eficiencia del algoritmo se determina teóricamente, y entonces se determinan empíricamente aquellos parámetros numéricos que sean específicos para un cierto programa y para una cierta máquina, lo cual suele hacerse mediante algún tipo de regresión. Empleando este enfoque se puede predecir el tiempo que necesitará una cierta implementación para resolver un ejemplar mucho mayor que los que se hayan empleado en las pruebas. Sin embargo, hay que tener cuidado cuando se hacen estas extrapolaciones basándose solamente en un pequeño número de comprobaciones empíricas y anulando toda consideración teórica. Las predicciones hechas sin apoyo teórico tienen grandes probabilidades de ser imprecisas, si es que no resultan completamente incorrectas.

Un algoritmo ordinario, secuencial, se suele considerar eficiente si su tiempo de ejecución para un problema de tamaño n está en $O(n^k)$ para alguna constante k. Por otra parte, para que se considere eficiente un algoritmo paralelo, se espera normalmente que satisfaga dos restricciones, una con respecto al número de procesadores y la otra que concierne al tiempo de ejecución, las cuales son:

- el número de procesadores necesarios para resolver un caso de tamaño n debe estar en $O(n^a)$ para alguna constante a, y
- el tiempo requerido para resolver un caso de tamaño n debe estar en $O(\log^b n)$ para alguna constante b.

Se dice que un algoritmo paralelo eficiente requiere un número polinómico de procesadores, y un tiempo polilogarítmico.

Un algoritmo paralelo se dice óptimo si es más eficiente con respecto al mejor algoritmo secuencial posible. A veces se puede denominar óptimo si es más eficiente con respecto al mejor algoritmo secuencial conocido. En este caso, sin embargo, es preferible decir que el problema correspondiente tiene aceleración óptima. Hay muchos problemas para los cuales no se conoce ningún algoritmo secuencial eficiente (esto es, de tiempo polinómico y el ejemplo mas común en un sistema químico sin ninguna aproximación y tomando en cuenta todas las moléculas e interacciones). Para tales problemas, no se puede esperar hallar una solución paralela eficiente (esto es, una que utilice un número polinómico de procesadores y un tiempo polilogarítmico). Por otra parte, hay muchos problemas para los cuales se conoce un algoritmo secuencial eficiente, pero para los cuales todavía no se ha descubierto un algoritmo paralelo eficiente. Se cree, aunque no se ha demostrado, que algunos problemas que se pueden resolver mediante un algoritmo secuencial eficiente no poseen una solución paralela eficiente.

Encontrar el algoritmo óptimo es muy importante para lograr resolver problemas de gran reto, independientemente del avance del *hardware*, ya que sólo así se podrán resolver estos problemas con un tamaño considerable de datos y en tiempos aceptables. También hay que tener en cuenta que el análisis asintótico funciona muy bien para N y P grandes y no para tamaños de problema y procesadores reales que se tienen, donde además no toma en cuenta costos de comunicación, memoria, etc.

A modo de ejemplo considere dos algoritmos cuyas implementaciones en una cierta máquina requieren n^2 días y n^3 segundos respectivamente para resolver un caso del tamaño n. Solamente en casos que requieran más de 20 millones de años para resolverlos, el algoritmo cuadrático será más rápido que el algoritmo cúbico. Desde un punto de vista teórico, el primero es asintóticamente mejor que el segundo; esto es, su rendimiento es mejor para todos los casos suficientemente grandes. Sin embargo, desde un punto de vista práctico, se prefiere ciertamente el algoritmo cúbico. Otro ejemplo es en un cálculo HF con N orbitales donde al final del cálculo, las energías orbitales tienen que sumarse, por lo tanto, se tendrían N sumas. También habrá ciertas operaciones, que se denotarán por C, que serán independientes del tamaño del problema, como la inicialización de las variables, la asignación de memoria, etc. El algoritmo estándar para la inversión de la matriz requiere de N^3 operaciones. Calcular y manipular las integrales bielectrónicas de Coulomb y de intercambio requieren de N^4 operaciones. Por lo tanto, la complejidad total del cálculo HF será N^4+N^3+N+C. Del análisis asintótico, con una N suficientemente grande, el término N^4 será el único término que importará. Pero si se tiene N no tan grandes, quizá el término N^3 será el más importante. Afortunadamente, con las computadoras actuales, muy comúnmente, se tienen N lo suficientemente grandes para utilizar de manera aceptable el análisis asintótico en la química cuántica computacional; aunque se deben de tomar en cuenta en el análisis posterior las demás variables.

Un ejemplo práctico de este análisis sería que a un investigador le gustaría calcular cierta propiedad de un polímero. Primero examina la literatura para determinar que un cálculo *ab initio* con una base moderadamente grande le dará una precisión aceptable. Entonces el investigador decide correr un *single point* y una optimización de la geometría sobre un monómero, los cuales toman 2 y 20 minutos respectivamente. Como el cálculo escala a N^4, una optimización de la geometría para un trímero le tomará aproximadamente 3^4*20 minutos (27 horas). Al investigador, le gustaría modelar una celda con 15 monómeros, el cual requerirá 15^4*20 minutos (2 años). Obviamente, el uso de los métodos *ab initio* para la optimización de la geometría no es aceptable. Entonces, el investigador decide tener la celda sólo con 10 monómeros y optimizar la geometría con métodos de mecánica molecular, los cuales le toman como una hora y después obtiene los resultados que deseaba al correr un *single point* con métodos *ab initio*, que le tomará alrededor de 2 semanas lo que lo hace aceptable en tiempo para el investigador.

Un análisis más completo es para el siguiente ejemplo donde se tienen tres algoritmos, cuyos tiempos de ejecución se dan con las siguientes ecuaciones:

$$T_1 = N + \frac{N^2}{P}$$

$$T_2 = \frac{N + N^2}{P} + 100$$

$$T_3 = \frac{N + N^2}{P} + .6P^2$$

Estos algoritmos tienen un *speedup*[2] de aproximadamente 10.8 cuando P=12 y N=100. Sin embargo, se comportan completamente diferentes en otras situaciones como se muestra en la siguiente gráfica.

[2] La definición de *Speedup* se encuentra en la siguiente sección

Todos los algoritmos no tienen un muy buen rendimiento a P grandes y el algoritmo con T_3 claramente es el peor de todos. A N=100 el algoritmo con T_1 y T_2 se comportan casi igual; sin embargo si N=1000, el algoritmo con T_2 es muy bueno ya que casi logra un *speedup* perfecto. Este fenómeno es muy importante en la química computacional ya que como se verá en la discusión y resultados, los programas de química computacional y los diferentes métodos, tienen un rendimiento que varía conforme cambia el tamaño del sistema químico.

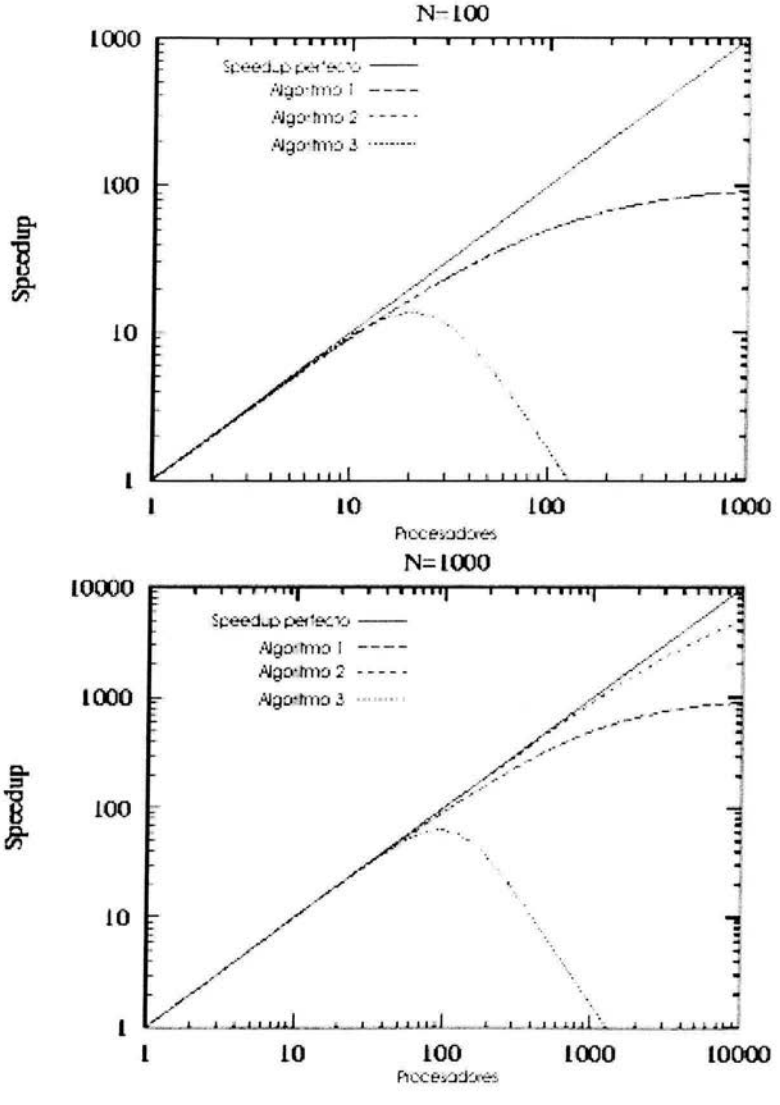


Figura 2.6 Ejemplos de diferentes algoritmos al variar el tamaño del problema

2.9 Rendimiento de computadoras paralelas: Ley de Amdahl.

La ganancia (incremento) de velocidad que puede conseguir una computadora paralela con n procesadores idénticos trabajando concurrentemente en un solo problema es como máximo n veces superior a la de un procesador único (ideal). En la práctica la ganancia es mucho menor, ya que algunos procesadores permanecen inactivos en algunos instantes debido a conflictos en los accesos a memoria, las comunicaciones, uso de algoritmos ineficaces en la explotación de la concurrencia natural del problema, etc. Existen varios modelos que pueden decir cómo son los rendimientos de una computadora paralela y éstos pueden ser desde muy sencillos hasta muy complejos Un modelo simple muy utilizado es la ley de Amdahl. Esta ley menciona que durante el proceso paralelo, siempre existe un componente secuencial que limitará el *speedup* que puede lograrse en una computadora paralela. El *speedup*, es la proporción entre tiempo de ejecución de un solo procesador y el tiempo de ejecución en los n procesadores de la máquina paralela. La ley de Amdahl dice que si el componente secuencial de un algoritmo es 1/s del tiempo de ejecución total del programa, entonces el posible *speedup* máximo que puede lograrse es s. Por ejemplo, si el componente secuencial es 5%, entonces el *speedup* máximo que puede lograrse es 20. Matemáticamente, esto es:

$$T_p=[1-\alpha+\frac{\alpha}{p}]*T_1 \quad .$$

Donde la ejecución o tiempo de CPU en los p procesadores paralelos depende de la fracción paralela α que es proporcional al tiempo de un procesador T_1 (que por lo regular es utilizando los mejores algoritmos seriales). El *speedup*, o el factor de aceleración, se define matemáticamente así:

$S_p=\frac{T_1}{T_p}=\frac{1}{1-\alpha+\frac{\alpha}{p}}=\frac{1}{s+\frac{\alpha}{p}}$ donde $s=1-\alpha$. Es decir, s es el componente secuencial del algoritmo.

De ésta forma, si el número de procesadores masivamente paralelos es muy grande, entonces en el límite $p\rightarrow\infty$ tenemos que $S_p=\frac{1}{1-\alpha}=\frac{1}{s}$.

Otro caso interesante es cuando el porcentaje paralelo es 100%, entonces el S_p tendrá un *speedup* máximo de infinito y regresamos al caso ideal. Los aspectos importantes de la ley de Amdahl son que no toma en cuenta la sincronización de los procesos ni la sobrecarga originada por dicha sincronización, es decir, omite los tiempos de las intercomunicaciones existentes entre procesadores en el procesamiento paralelo. Descartando u omitiendo estas intercomunicaciones es imposible obtener un buen rendimiento y en caso de no tener en cuenta esto y que en ocasiones las comunicaciones son más tardadas que el cálculo mismo, no se obtendrá un buen rendimiento de los procesos. Entonces es importante conocer una manera óptima o fiable de llevar a cabo la repartición de los datos (el esquema de paralelización) para tener el menor número de intercomunicaciones posibles. El *Speedup* decrece aproximadamente a:

$$S_p = \frac{T}{\frac{T}{p} + T_{comunicación}} < p$$

y para que la aceleración no sea afectada por el tiempo de comunicación es necesario que:

$$\frac{T}{p} >> T_{comunicación} \Rightarrow p << \frac{T}{T_{comunicación}}$$

Esto significa que a medida que se divide el problema en partes más y más pequeñas para poder ejecutar el problema en más procesadores, llega un momento en que el costo de comunicación se hace muy significativo y desacelera el cómputo; por lo tanto es muy importante minimizar la relación comunicación/cómputo.

Otra implicación de la ley de Amdahl es que el rendimiento es sensitivo al porcentaje paralelizable y al número de procesadores y que con esta ley se pueden hacer predicciones teóricas del comportamiento del programa en un número de procesadores con las que actualmente no se cuentan. Por lo tanto, además de la minimización de la relación tiempo de comunicación/tiempo de cómputo, es necesario hacer un esfuerzo para mejorar el algoritmo reduciendo el porcentaje del componente secuencial. Con la ley de Amdahl también es posible ver hasta dónde conviene utilizar "n" número de procesadores; ya que al acercarse al *speedup* máximo (la asíntota en la gráfica), no vale la pena correr el programa con más procesadores y entonces utilizar éstos para otros procesos. Como se ve en la tabla, incluso aplicar un número infinito de procesadores, sólo logrará un *speedup* de 30. En las gráficas y en la tabla se asume un α=96.77. Por lo tanto, es muy recomendable tener un algoritmo o aplicación que tenga un alfa mayor o igual a 95% para tener un buen *speedup*, recordando que por lo regular éste será una cota superior; ya que el *speedup* real, será menor debido a la carga de la máquina, las comunicaciones, la sincronización, etc.

Número de CPUs	Speedup Teórico
1	1.000
2	1.937
3	2.818
4	3.647
5	4.428
6	5.167
7	5.863
8	6.525
9	7.152
10	7.752
...	...
∞	30.959

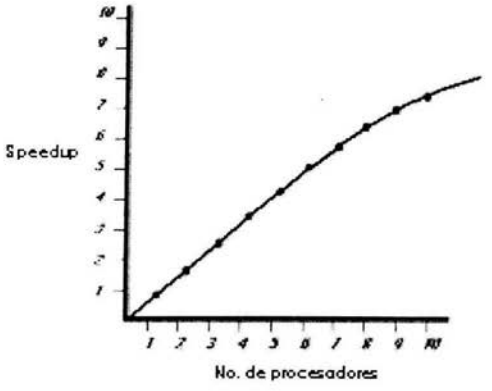


Figura 2.7 La aceleración y la ley de Amdahl

Las curvas cambian conforme cambia el tamaño del problema como muy comúnmente ocurre en la química computacional. Si se aumenta el tamaño del problema de tal manera que el porcentaje paralelizable aumenta (E/S y las comunicaciones se utilizan en menor tiempo en relación al tiempo total y existe un mayor tiempo de cómputo etc), entonces el *speedup* mejorará; pero sucederá exactamente lo contrario si el porcentaje paralelizable disminuye (la sincronización mata el paralelismo) y este fenómeno se verá claramente en la discusión y resultados por lo tanto es muy importante considerar la variación del *speedup* con respecto al tamaño del problema.

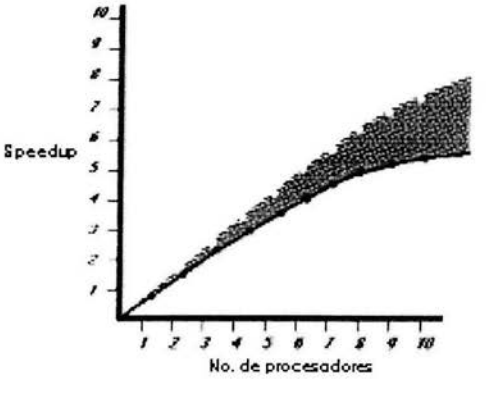


Figura 2.8 Variación del *speedup* conforme al cambio del tamaño del problema.[3]

[3] Gráfica muy importante ya que muestra la escalabilidad del problema (isoeficiencia). Se explicará a mas detalle en la sección 2.12

2.10 Aceleración superlineal

En teoría, no es posible obtener una aceleración superior a la lineal. Esto está basado en el supuesto de que un único procesador siempre puede emular procesadores paralelos. Dado que usualmente los algoritmos paralelos tienen asociados costos adicionales de sincronización y comunicación, es muy probable que la aceleración sea menor a la lineal. Sin embargo, hay circunstancias algorítmicas especiales que provocan que la aceleración pueda ser mayor a la lineal. Por ejemplo, cuando se esta resolviendo un problema de búsqueda, un algoritmo serial puede perder una cantidad de tiempo considerable examinando estrategias que no llevan a la solución. Sin embargo, un algoritmo paralelo puede revisar muchas estrategias simultáneamente y se puede dar el caso de que una de ellas dé con la solución rápidamente. En este caso el tiempo del algoritmo secuencial comparado con el paralelo es mayor que el número de procesadores empleados.

También existen circunstancias de arquitectura que pueden producir aceleraciones superlineales. Por ejemplo, considere que cada CPU en una máquina paralela tiene cierta cantidad de memoria caché. Comparando con la ejecución en un solo procesador, un grupo de p procesadores ejecutando un algoritmo paralelo tiene p veces la cantidad de memoria caché. Es fácil construir ejemplos en los que la tasa colectiva de hits de caché para los p procesadores sea significativamente mayor que los hits de *caché* del mejor algoritmo secuencial corriendo en un solo procesador, reduciendo los costos de accesos a memoria. En estas condiciones el algoritmo paralelo puede correr más de p veces más rápido. Estas circunstancias son especiales y se dan raras veces y en la discusión y resultados se observará un ejemplo de este fenómeno debido precisamente al manejo de la memoria.

2.11 Modelos más realistas

En los modelos anteriores se tiene la ventaja de que son sencillos para hacer predicciones rápidas; sin embargo, no toman en cuenta muchos aspectos que suelen ser importantes para los tiempos de ejecución reales. El tiempo de ejecución es una función multidimensional que depende del tamaño del problema, número de procesadores, número de procesos y otras características del algoritmo y del hardware, es decir:

$$T = f(N, P, U, ...)$$

Se define el tiempo de ejecución de un programa paralelo como el tiempo que transcurre desde que el primer procesador comienza la ejecución del problema hasta que el último procesador completa la ejecución. Durante la ejecución, cada procesador se encuentra haciendo cómputo, comunicando o está ocioso como se ilustra en la siguiente figura.

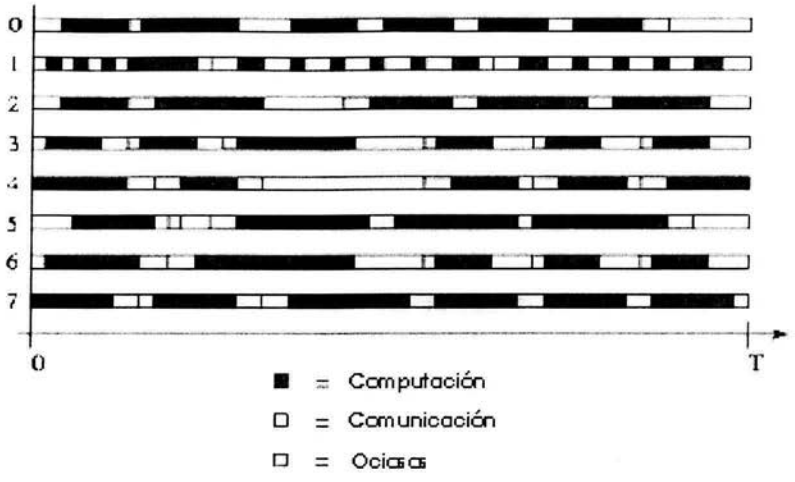


Figura 2.9 Actividad de 8 procesadores durante la ejecución de un programa paralelo. T es el tiempo total de ejecución.

La gráfica anterior es muy importante y utilizada ya que ayuda a ver el comportamiento del programa, sus comunicaciones, sincronización y la eficiencia de éste. El tiempo de ejecución total se define como la suma de los tiempos de comunicación, cómputo y tiempo de ocio de cada procesador divididos por el número total de los procesadores, es decir:

$$T = \frac{1}{P}\left(\sum_{i=0}^{p-1} T^{i}_{computación} + \sum_{i=0}^{p-1} T^{i}_{comunicación} + \sum_{i=0}^{p-1} T^{i}_{ocio}\right)$$

Esta ecuación se puede hacer más complicada si se toman en cuenta las jerarquías de memoria, la topología de red de la interconexión, costos de inicialización de comunicación o cómputo, etc. También se pueden tomar en cuenta estudios empíricos para calibrar esta ecuación en lugar de usar un modelo más complejo de primeros principios.

El tiempo de cómputo depende del tamaño del problema, número de procesos o procesadores, tipos de procesadores, jerarquía de la memoria, etc. Por lo tanto, es muy común obtener este tiempo con base en estudios empíricos y/o mediciones indirectas. El tiempo de comunicación es el tiempo que los procesos utilizan para enviar y recibir los mensajes que se hacen durante la ejecución. La comunicación puede ser interprocesador (entre diferentes procesadores) e intraprocesador (en el mismo procesador). El modelo mas utilizado para tomar el tiempo de comunicación es aquel que toma en cuenta el tiempo de inicio del mensaje t_s que es el tiempo necesario para comenzar la comunicación y el tiempo de traslado por palabra t_w, el cual está determinado por el ancho de banda de la red de interconexión. Entonces, el tiempo que se necesita para enviar un mensaje de L palabras de tamaño es:

$$T_{msg} = t_s + t_w L$$

Estos datos por lo regular se obtienen mediante un programa que mide el tiempo que se tarda en mandar un mensaje de un procesador a otro. Entre más grandes sean los mensajes, entonces ésta ecuación funcionará mejor y asintóticamente para L muy grandes, solamente el término t_w será el importante. En cambio, cuando los mensajes son muy pequeños, entonces el término más importante será el t_s. Cuando una aplicación genera muchos mensajes, entonces es necesario refinar este modelo y esto se logra desarrollando un modelo más detallado y que tome en cuenta la red de interconexión.

En cuanto al tiempo de ocio, por lo regular, es el más difícil de determinar y se hace comúnmente con mediciones indirectas. Un procesador puede estar ocioso a falta de cómputo o de datos. En el primer caso, se puede evitar con un balanceo de carga eficiente y en el segundo caso si el programa es lo suficientemente robusto para que los procesadores realicen otro cómputo o comunicación mientras esperan por datos remotos. Esta técnica se llama traslape de cómputo y comunicación (*overlapping computation and communication*). Esto se puede lograr ya sea que se generen múltiples tareas o procesos en un mismo procesador pero sólo es eficaz si el costo de programar una nueva tarea es menor al costo del tiempo de ocio; y la otra manera se logra explícitamente programando otros cálculos mientras se espera la comunicación como se muestra en la siguiente figura. Muchas bibliotecas permiten este tipo de comunicación, por ejemplo en MPI la comunicación puede ser bloqueante o no bloqueante para permitir el solapamiento de la comunicación y cómputo.

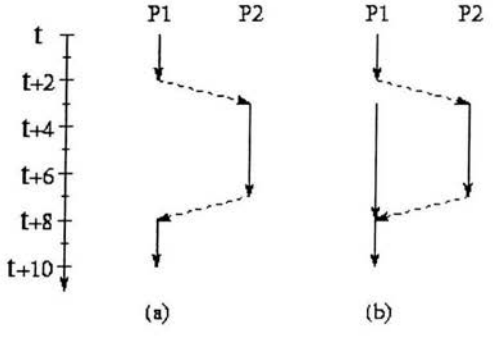


Figura 2.10 Técnica de *overlapping computation and communication*

2.12 Eficiencia

No siempre el tiempo de ejecución es la métrica más conveniente para evaluar el rendimiento de un algoritmo paralelo, debido a que éste varía fácilmente con el tamaño del problema como ocurre en la química computacional. Una manera de normalizar el tiempo de ejecución al tamaño del problema es con la eficiencia, el cual es la fracción de tiempo que los procesadores utilizan para hacer el trabajo de cómputo. La eficiencia relativa se define como: $E_{relativo} = \frac{T_1}{PT_p}$ donde T_1 es el tiempo de ejecución en un procesador y T_p es el tiempo en P procesadores. Entonces el *speedup* relativo se define como

$S_{relativo}=PE_{relativo}$. Estos valores son relativos debido a que son definidos con respecto al algoritmo paralelo que se ejecuta en un procesador (y son las que comúnmente se reportan) y estas métricas son usuales porque exploran la escalabilidad del algoritmo. La eficiencia y *speedup* absolutos se obtienen cuando se compara con respecto al tiempo de ejecución mejor conocido en un solo procesador, donde muchas veces este tiempo se debe a un programa o algoritmo serial.

Debido a todo esto, un programa o algoritmo serial optimizado, robusto y bien diseñado es muy importante, ya sea para un posterior paralelización o para obtener buenas métricas con respecto al programa paralelo obtenido.

La escalabilidad de un algoritmo, se puede cuantificar también con la eficiencia y *speedup* relativos incrementando los procesadores para un tamaño de problema fijo. Por lo regular, se llegará a un número máximo de procesadores para un tamaño fijo de problema. Un dato importante es variar el tamaño del problema (la carga de cómputo y en el caso de la química computacional el número de funciones base como se recomienda en los resultados y la discusión) con respecto al número de procesadores para mantener la eficiencia relativa constante. A esta función que se obtiene se le llama función de isoeficiencia; en donde un algoritmo con una función de isoeficiencia de orden $O(p)$ es altamente escalable, no así una cuadrática o exponencial. Recordando que $E_{relativo} = \frac{T_1}{PT_p} = \frac{T_1}{\sum_{i=0}^{p-1} T^i_{computación} + \sum_{i=0}^{p-1} T^i_{comunicación} + \sum_{i=0}^{p-1} T^i_{ocio}}$, entonces para mantener la eficiencia constante se tiene que cumplir que el tiempo que transcurre en un solo procesador, debe aumentar a la misma proporción al aumento del tiempo total de cómputo, comunicación y ocio que ocurren en el cómputo paralelo.

2.13 Datos empíricos

Con los análisis que se han hecho hasta ahora, se puede tener un perfil más o menos amplio sobre el comportamiento y rendimiento del algoritmo ya sea en diferentes máquinas, tamaños de problema, etc. Sin embargo, todavía no se tiene una base suficiente para responder las siguientes preguntas:

- ¿el algoritmo cumple con los requerimientos (tiempo de ejecución, requerimientos de memoria, etc.) en la máquina paralela designada?
- ¿qué tan adaptable es mi algoritmo? Es decir, ¿qué tanto se afecta con los aumentos del tamaño del problema o con los parámetros dependientes de la máquina como t_s y t_w?
- ¿Qué diferencia en tiempo de ejecución puede esperarse con los diferentes algoritmos?

Para poder responder bien estas preguntas, se necesitan de estudios empíricos ya que una vez que el algoritmo es implementado, se pueden validar los modelos e inclusive mejorarlos con respecto a los recursos que se tienen. La programación paralela, todavía es por encima de todo una disciplina experimental.

El primer paso en un estudio experimental es la identificación de los datos que se desean obtener, como por ejemplo t_s t_w, etc., y éstos se varían con respecto a otras variables, como tamaño de

problema, numero de procesadores, etc, obteniendo así un rango de datos, el cual dirá en qué región mejor se adaptará el modelo; un método común, es el de mínimo cuadrados. Maximizando el número de datos que se obtienen, se reduce el impacto de errores de medición. El segundo paso, es el diseño de los experimentos que darán los datos que se necesitan obtener. El problema crítico aquí es asegurar que los experimentos realmente midan lo que se necesita medir, además de que sean datos reproducibles y lo más precisos posibles. Siempre deben repetirse los experimentos para verificar que los resultados sean reproducibles y generalmente, los resultados no deben variar más del 2% del valor. Las posibles causas de variación pueden ser que el algoritmo es no determinístico, que se use un cronómetro inexacto, los costos de las variables de inicio y terminación debidos principalmente al estado del sistema y la interferencia de otros programas. Estos puntos son muy importantes y por ejemplo en la metodología se hará un estudio de este tipo y donde es muy importante asegurar que los datos obtenidos son reproducibles, miden exactamente lo que se necesita medir y que los experimentos son de la utilidad que se requieren.

Los estudios de variabilidad en los resultados experimentales pueden ayudar a identificar fuentes de error o incertidumbres en las medidas. Sin embargo, incluso cuando los resultados son reproducibles, todavía no se tiene la certeza de que sean correctos; por ello, es importante medir la misma métrica mediante diferentes métodos y verificar que los resultados de estas redundantes mediciones sean consistentes.

2.14 Entradas/Salidas

Un importante determinante del comportamiento de muchos programas paralelos es el tiempo requerido para mover los datos entre las diferentes jerarquías de memoria, es decir, el tiempo requerido para las entradas/salidas (input/output→I/O). Aplicaciones con requerimientos sustanciales de I/O son los checkpoints, uso de memoria virtual, datos de la simulación, análisis de datos como sucede en la química computacional con los análisis QSAR, etc. Es difícil proporcionar una discusión general de I/O paralelo debido a que las diferentes máquinas paralelas tienen arquitecturas y mecanismos de I/O radicalmente diferentes. Sin embargo, algunos puntos que se deben tratar son:

- se puede pensar en los I/O como comunicaciones entre los procesadores de tal manera que se puede determinar un costo de inicio y de transferencia de palabra por tiempo para medir el costo de los I/O en la aplicación. Obviamente, el costo de inicio en I/O es mucho mayor al de las comunicaciones interprocesador. Entonces, se pueden utilizar las mismas técnicas para minimizar los costos, inicios, etc.
- en la mayoría de las computadoras paralelas, existen diferentes caminos para que los procesadores puedan hacer I/O concurrentemente, por lo tanto se buscará organizar los I/O para que diferentes procesadores lean y escriban concurrentemente (obviamente manteniendo la coherencia) usando diferentes estrategias, rutas o caminos.
- es importante conocer el modo de trabajo del sistema de archivos de la máquina para conocer sus ventajas y aprovecharlas además de evitar sus desventajas y obtener así un mejor rendimiento.
- recordando la jerarquía de la memoria de la máquina, es a menudo mejor realizar una redistribución de datos explícita en la memoria, es decir, ocupar mucho más las comunicaciones entre los procesadores que los I/O, debido al alto costo que tienen.

2.15 Resumen

El ciclo de la aplicación es:

- primero se hace el análisis del problema para determinar qué tan eficiente puede ser un algoritmo paralelo para nuestras necesidades creando así un diseño que se caracteriza por los requerimientos de cómputo y comunicaciones que se tienen.
- después se analizan las diferentes alternativas que tenemos para identificar las áreas de problema como los cuellos de botella y para verificar que los algoritmos reúnen los requisitos de rendimiento y actuación.
- después se refinará la actuación y conducta de los algoritmos escogidos mediante la obtención de diferentes métricas que mostrarán que tan bien se ha cumplido con los requisitos planteados tales como tiempo de ejecución, costos de comunicación, etc.
- por último durante la implementación, se comparará la actuación real del programa paralelo con su modelo de actuación ideal. Esto puede ayudar a identificar los errores de la implementación y mejorar así la calidad del modelo.

El último paso, se logra al obtener un perfil del programa obtenido durante la ejecución. Este perfil es muy importante ya que sólo así se podrán atacar los cuellos de botella de la aplicación y obtener así un programa optimizado. Todos los pasos durante el ciclo de la aplicación, afectan o pueden afectar pasos anteriores, generando así un proceso de *software* tipo espiral.

La optimización es un factor básico para obtener un alto desempeño en equipos de cómputo, por ello, el desarrollo de software procura alcanzar este objetivo, donde los resultados deben ser los esperados, que la escritura del código sea adecuada y se realice en el tiempo requerido, resultando que cada vez sea más productivo realizarlo. Para obtener un buen rendimiento del programa paralelo se requieren de varios factores, los principales factores para un buen paralelismo y comportamiento de los programas paralelos son:

1. el grado de paralelismo inherente de la aplicación
2. la arquitectura de la máquina paralela en donde la aplicación se ejecutará
3. lo eficaz que el lenguaje y el sistema de ejecución explote la arquitectura (disponibilidad y velocidad del CPU, instrucciones de entrada y salida, tamaño y disponibilidad de memoria, etc.)
4. lo eficaz que el código explote al lenguaje, sistema de compilación y arquitectura (errores→*bugs*, usar apropiados algoritmos y estructuras de datos, bibliotecas matemáticas y/o de comunicación optimizadas, etc.)
5. el ambiente de ejecución en el momento en que se dé (herramientas para el análisis de rendimiento y actuación, colas, etc.)

La optimización de una aplicación consta de una serie de etapas a realizar, iniciando por el código fuente, mejorando las líneas del código, de modo que resulte un código más rápido de ejecutar. El tiempo de ejecución de un programa siempre dependerá en gran medida de la arquitectura de la computadora en la que se esté ejecutando, por ello no es lo mismo hablar de ejecución en máquinas cuyo procesador es RISC a que sea CISC, que tenga niveles de caché intermedio a que no los tenga, etc. El vendedor de las máquinas paralelas, por lo regular, ofrecerá documentos que permitirán conocer las bondades de dicha máquina y cómo utilizarlas de la mejor manera para lograr optimizar los códigos que

se utilizarán; con excepción de los *clusters* hechos en "casa", donde el "*humanware*" disponible toma un papel sumamente importante. Sin embargo, siempre es recomendable hacer diferentes *benchmarks* en diferentes máquinas para conocer sus comportamientos en los códigos que más se utilizarán y tomar así una mejor visión de las necesidades de *hardware* y *software* ya que los datos que dan los vendedores siempre son los picos del comportamiento de la máquina, es decir, en condiciones ideales para su mejor funcionamiento; por lo tanto, para la mayoría de las aplicaciones paralelas donde la química computacional no es la excepción, la actuación será entre el 10% y el 20% del comportamiento máximo de la máquina.

Capítulo 3

LA QUÍMICA CUÁNTICA COMPUTACIONAL, LAS CIENCIAS QUÍMICAS Y SUS IMPLICACIONES

3.1 Aproximaciones

La química teórica, es la descripción matemática de la química y en particular, la química cuántica, es la descripción de la química generada por la mecánica y la dinámica cuántica. Una vez que un método matemático se desarrolla lo "suficientemente bien" para poder ser automatizado por una implementación en la computadora; entonces, la química computacional se encargará de dicha implementación y su desarrollo. La mecánica cuántica nos da una descripción matemática exacta del ambiente de los electrones; sin embargo, las ecuaciones que produce nunca se han resuelto exactamente para ningún sistema químico (excepto el átomo de hidrógeno y el catión de la molécula de hidrógeno mediante el uso de coordenadas elípticas) debido a la complejidad de éstos y a que nos enfrentamos a un problema de muchos cuerpos, que se ha demostrado que no tiene solución analítica exacta. Por lo tanto, el campo de la química cuántica computacional se construye sobre resultados aproximados. Sin embargo, se pueden tener soluciones numéricas y/o aproximaciones que se acerquen a la solución exacta. Para ello, se puede por ejemplo, calcular sólo una parte del problema omitiendo las partes que no afecten demasiado nuestro resultado, un promedio en lugar de una solución exacta o usar métodos de aproximaciones comunes como el método de variaciones, perturbaciones, simplificación de funciones y ajuste de parámetros.

Pero existen varias implicaciones a las aproximaciones, algunas pueden ser muy crudas y dar resultados que no se acerquen tanto a la solución y otras pueden ser tan precisas que ni un experimento de hoy puede mejorarlo. Por lo tanto, se debe tener conocimiento de cada aproximación que se usará y cómo afecta ésta la precisión del dato obtenido. Por eso, las teorías, modelos y aproximaciones, son herramientas poderosas que ayudan a entender y lograr las metas de la investigación. El precio de tener estas herramientas, es que no son perfectas y es por ello que es muy aconsejable que se desarrolle una comprensión de la naturaleza de las aproximaciones y limitaciones en cuanto a la precisión que dan.

La ecuación que describe la química en su totalidad, es la ecuación de Dirac, la cual es muy compleja y tiene un costo computacional muy alto por lo que sólo se ha podido resolver para muy pocos casos de manera muy precisa. Cuando nos acercamos al límite no relativista ($c \rightarrow \infty$) se obtiene la ecuación de Schrödinger dependiente del tiempo. Al separar la función de onda en dos funciones, una dependiente de la posición y otra del tiempo, obtenemos la ecuación de Schrödinger independiente del tiempo donde al obtener algún observable o valor esperado, será independiente del tiempo; a pesar de que la función de onda no lo sea, llamando al comportamiento descrito por la función de onda obtenida como estado estacionario ya que las propiedades y "estados" que se obtienen son independientes del tiempo aunque la partícula que describe no lo sea.

La ecuación de Schödinger independiente del tiempo también es sumamente complicada para resolver, ya que genera una serie de ecuaciones integrodiferenciales que no tienen solución analítica exacta. Sin embargo, como los núcleos son mucho más pesados que los electrones (1/1836), estos últimos se

mueven mucho más rápidamente y por lo tanto responden prácticamente instantáneamente al movimiento de los núcleos. La aproximación de Born-Oppenheimer (aproximación adiabática), desacopla los movimientos de los electrones y de los núcleos logrando que se pueda resolver la ecuación de Schrödinger fijando la posición de los núcleos y que el hamiltoniano sea separable en una parte nuclear y una parte electrónica. Cuando se resuelve la ecuación de Schrödinger con esta aproximación, se obtiene una energía que sólo depende de la parte electrónica y una energía que sólo depende de la parte nuclear y la suma de ambas sería la energía total del sistema.

Al resolver la parte electrónica del hamiltoniano se obtendrá una función de onda que describirá la estructura electrónica del sistema y la energía electrónica. Dichos resultados sólo serán válidos para la estructura en la que fijamos los núcleos. La función de onda electrónica depende paramétricamente de las coordenadas nucleares y no de su momento; por lo tanto si los movemos, entonces se volverá a resolver la parte electrónica del hamiltoniano obteniendo así otra función de onda y otra energía y así sucesivamente para construir la superficie de energía potencial (PES→*Potential Energy Surface*). Encontrar el punto mínimo global de esta superficie, que sería el de mínima energía y por lo tanto la estructura más estable, es uno de los problemas más difíciles de resolver y se puede demostrar que no existe un método que nos asegure que se ha encontrado el mínimo global.

Esta aproximación es sumamente confiable en el estado basal y por lo tanto, prácticamente todo el trabajo de la química cuántica computacional se aplica sobre esta aproximación. Sea que se use un punto simple (single point) o una optimización de la geometría molecular, siempre se utilizará ésta aproximación ya que siempre estarán los núcleos fijos en el espacio al momento de resolver la ecuación de Schrödinger y una vez resuelta; entonces se podrán mover los núcleos para ir minimizando la energía o buscar otras propiedades. Monte Carlo Cuántico es uno de los pocos métodos que pueden no utilizar esta aproximación; sin embargo el costo computacional es muy alto.

Otra aproximación que se tiene es darle una estructura matemática bien conocida a las soluciones utilizando los spin-orbitales y los determinantes de Slater donde en lugar de tener una función muy complicada con 3N variables espaciales; se tendrá a una función que es un producto de N funciones, cada una con 3 variables espaciales dependientes solamente de cada electrón. Esta aproximación, la utiliza el método de Hartree-Fock (HF); sin embargo, existen otros métodos multideterminantales que minimizan el error de esta aproximación los cuales entran en la categoría de métodos de correlación.

$$\Psi_{el}(1,2,\ldots,N)=\left(\frac{1}{N!}\right)^{\frac{1}{2}}\begin{vmatrix}\phi_a(1) & \phi_b(1) & \ldots & \phi_N(1)\\ \phi_a(1) & \phi_b(2) & \ldots & \phi_N(2)\\ \vdots & \vdots & \ddots & \vdots\\ \phi_a(N) & \phi_b(N) & \ldots & \phi_N(N)\end{vmatrix}$$

Figura 3.1 Determinante de Slater de N partículas (electrones)

El método óptimo dentro de la aproximación orbital es el método de campo autoconsistente de Hartree-Fock (SCF→*self-consistent field*). Éste consiste en considerar una función de onda tipo determinante de Slater y buscar el conjunto de orbitales ϕ_i que minimice el valor esperado del hamiltoniano electrónico utilizando el método variacional. Como consecuencia de esto, se obtiene la

llamada ecuación de Hartree-Fock. Es importante señalar que cada método generará sus ecuaciones, los cuales serán los que se resolverán para obtener los datos que se necesitan y no se resolverá directamente la ecuación de Schrödinger. La ecuación de Hartree-Fock producirá un sistema de ecuaciones integro-diferenciales acoplados muy difíciles de resolver, ya que se necesita conocer las funciones orbitales ϕ_j de los otros electrones para obtener ϕ_i. De igual forma, es necesario conocer ϕ_i para obtener cualquier de las ϕ_j. Para resolverlo, se usa el método del campo autoconsistente, el cual es sumamente complicado de hacer y como ejemplo, Hartree y su padre (el cual se encontraba jubilado) durante la década de los 30, realizaron multitud de cálculos que consumieron muchos años. Para evitar esta complejidad y como se ha observado que en el estado basal, las ϕ_i se parecen a los orbitales hidrogenoides en el comportamiento; entonces a las ϕ_i se les ha dado una estructura de combinación lineal de éstos y es por ello que la siguiente aproximación y quizá la mas importante es la aproximación de la combinación lineal de orbitales atómicos (LCAO→*Linear Combinations of Atomic Orbitals*):

$$\phi_i = \sum_r c_{ir} \chi_r$$

donde los orbitales atómicos usados en esta expansión constituyen el conjunto de base del cálculo. En principio, para que la igualdad se cumpla, se requiere de un conjunto completo de bases del espacio de Hilbert, lo que requiere un número infinito de funciones de base. En la práctica, se tiene que usar un número finito de funciones de base lo que genera un error que puede ser mínimo si se escogen bien y que el conjunto sea lo suficientemente grande. Sin embargo y a pesar del error que se comete, la ventaja es que en lugar de resolver un problema de ecuaciones integrodiferenciales, sólo se resolverá un problema de álgebra lineal que fácilmente puede ser trasladado a una computadora y donde los métodos numéricos y computacionales han sido muy desarrollados; y esto se logra con las ecuaciones de Hartree-Fock-Roothaan, donde los parámetros variacionales son coeficientes y no funciones. Otra consecuencia es que solamente se obtendrá un número de soluciones igual a la dimensión de la base, obteniendo así los llamados orbitales ocupados y los virtuales, donde estos últimos son una primera aproximación a los estados excitados; a diferencia de la función de onda exacta que tendría un número infinito de soluciones.

Entonces las aproximaciones que se hacen para sistemas químicos en el estado estacionario son:

1. la omisión de efectos relativistas
2. la aproximación de Born-Oppenheimer
3. la incompleta correlación electrónica
4. y el uso de un conjunto de base incompleto.

De estas aproximaciones, la 3 y la 4 suelen ser las que mayor error introducen al cálculo y la 4 es la más importante ya que de ella depende si se obtendrá éxito en los cálculos o serán totalmente erróneos a pesar de que la metodología sea la correcta. El escoger correctamente la base y el método de cálculo para el sistema químico puede ahorrar muchas horas de cómputo y de trabajo y de ahí la importancia de conocer suficientemente bien estos temas.

3.2 Bases

3.2.1 Tipos y mejoras

Para sistemas químicos pequeños y altamente simétricos, las ecuaciones de Hartree-Fock se pueden resolver mapeando los orbitales en un conjunto de puntos de una malla (método del elemento finito); éstos son los métodos numéricos de Hartree-Fock; aunque también se pueden utilizar estos métodos para resolver otras ecuaciones generadas por los diferentes métodos. Computacionalmente, estos métodos numéricos son muy costosos y por lo tanto, el cálculo de los orbitales moleculares se hace a partir de conjuntos de bases conocidas; sin embargo, debido al avance de la tecnología en las ciencias de la computación y al paralelismo inherente de estos métodos, se piensa que en el futuro, estos métodos serán tan utilizados como lo son actualmente los métodos que usan las bases centradas en los núcleos. Las bases pueden ser cualquier tipo de funciones; sin embargo, deben representar lo mejor posible el problema físico y cumplir con todos los requisitos de una función de onda para que el teorema variacional se cumpla. Un buen ejemplo de ello son los orbitales hidrogenoides centrados en el núcleo ya que se conocen muy bien su comportamiento además de que con la experiencia, se ha visto que pueden representar muy bien los orbitales moleculares.

Slater propuso en 1950 sus orbitales (STO→*Slater Type Orbitals*), los cuales no tienen nodos radiales pero pueden ser introducidos con una combinación lineal de STOs. Estos orbitales generan una gran cantidad de integrales multicéntricas (hasta de 4 centros, las cuales no se pueden resolver analíticamente) que son muy difíciles de resolver y computacionalmente muy costosas; por lo tanto se ha optado por otras funciones que reducen las integrales de 4 centros hasta de 2 centros aunque en comparación tienen una peor descripción de la estructura electrónica que las STOs. Estas funciones son las gausianas (GTO→*Gaussian Type Orbitals*), las cuales pueden ser escritas en coordenadas polares o cartesianas y donde las gausianas cartesianas son las de menor costo computacional y por lo tanto las más utilizadas. Hay que señalar que en los orbitales d, f, ... siempre se tendrán mas combinaciones con las bases GTO cartesianas que con las polares, por ejemplo en los orbitales d se tienen 6 combinaciones en las GTO cartesianas (x^2, y^2, z^2, xy, xz, yz) y 5 en las GTO polares ($Y_{2,2}$, $Y_{2,1}$, $Y_{2,0}$, $Y_{2,-1}$, $Y_{2,-2}$). Sin embargo, se pueden transformar las 6 funciones cartesianas en las 5 d-funciones esféricas y una función s ($x^2+y^2+z^2$). Lo mismo con las 10 funciones cartesianas f que son transformadas en las 7 f-funciones esféricas y 3 p-funciones. Por lo regular se recomienda hacer los cálculos con las funciones cartesianas y después transformarlas en polares para una mejor descripción de los orbitales al hacer la combinación lineal de los GTOs para describir un STO.

Las bases gausianas tienen una descripción más pobre que las STO debido a que estas últimas tienen una cúspide que provoca una discontinuidad en la primera derivada cerca del núcleo y las GTO tienen por tanto problemas para representar el ambiente cerca del núcleo, además de que descienden muy rápido al alejarse del núcleo; por lo tanto se requieren mas GTOs para tener una precisión cercana a la STO.

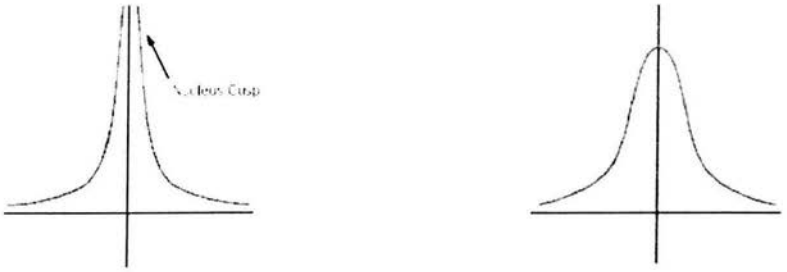


Figura 3.2 Comparación de una GTO y una STO.

Se ha visto que al menos se requieren 3 GTO por cada STO para obtener un grado de precisión aceptable y a pesar de que se necesitan más bases al utilizar las GTO, éstas se utilizan más, ya que el costo computacional del cálculo de las integrales multielectrónicas es muy alto. Por lo regular, las bases se centran en los núcleos; sin embargo, también se pueden centrar por ejemplo en el centro del enlace, dependiendo del cálculo que se necesite.

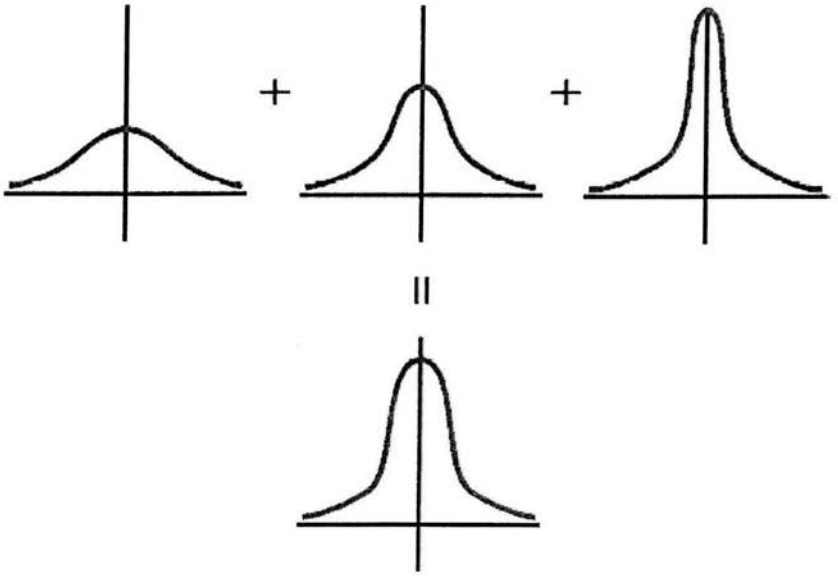


Figura 3.3 Tres GTO describen aceptablemente una STO.

Una vez que se ha decidido qué tipo de funciones utilizar (GTO/STO) y dónde centrarlas, el factor más importante es el número de funciones a usar, debido a que el truncamiento de la base es una de las

aproximaciones que más error puede provocar. El menor número de funciones posibles es la base mínima, ya que sólo se emplean las funciones suficientes para contener todos los electrones de los átomos. La siguiente mejora en las bases consiste en poner el doble de funciones que la base mínima produciendo así las bases doble zeta (DZ). La importancia de las DZ se observa cuando la distribución electrónica alrededor de un núcleo es muy diferente entre los diferentes enlaces que tiene con sus átomos cercanos. Duplicando el número de bases, se obtendrá una mejor descripción del hecho de que la distribución electrónica es diferente en las diferentes direcciones del espacio.

Sin embargo, todo químico sabe que en los enlaces químicos, participan principalmente los orbitales de valencia; mientras que los orbitales internos (*de core*) se comportan prácticamente como los orbitales atómicos; por lo tanto, comúnmente solamente se duplican los orbitales de valencia, produciendo las bases de valencia dividida (split Valence), las cuales pueden ser VDZ (Valence Double Zeta), TZ (Triple Zeta), QZ (Quadruple Zeta), 5Z (quintuple zeta), etc.

En los sistemas químicos (principalmente donde participan partículas cargadas como aniones) la interacción entre las partículas suele romper la simetría de los orbitales y para una mejor descripción de ello se mezclan orbitales con diferentes simetrías, los cuales son llamados funciones de polarización.

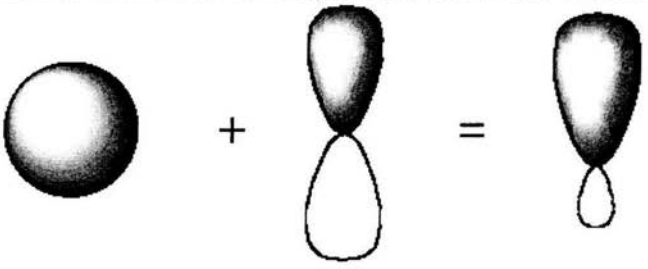


Figura 3.4 Funciones de polarización

Los orbitales p introducen polarización a los orbitales s, los orbitales d introducen polarización a los orbitales p y así sucesivamente. Se podría lograr el mismo resultado, por ejemplo, en el caso del hidrógeno adicionando 2 orbitales s, uno centrado en el núcleo y el otro orbital variacionalmente se podría centrar en diferentes partes del espacio, formando así los denominados orbitales flotantes; sin embargo, computacionalmente este procedimiento es muy costoso y con los orbitales de polarización se han logrado buenos resultados.

Cuando se agregan funciones de polarización a todos los átomos excepto a los hidrógenos, por ejemplo a una base doble zeta, la nueva base se llamará doble zeta mas polarización (DZP). Esto se hace comúnmente porque la mayoría de los hidrógenos tienen un rol "pasivo" y por lo tanto no afectan la propiedad de nuestro interés. El error introducido por no incluir las funciones de polarización en los hidrógenos por lo regular es constante y como los químicos están interesados regularmente en las diferencias de energías, este error tiende a cancelarse y debido a que la mayoría de las moléculas orgánicas tienen una gran cantidad de átomos de hidrógeno, el no calcular estas funciones de polarización reduce el costo computacional significativamente. Sin embargo, cuando los hidrógenos

tienen un papel importante en la propiedad de nuestro interés (puentes de hidrógeno), entonces sí es importante tomar en cuenta las funciones de polarización en los hidrógenos.

Al igual que antes, se pueden agregar múltiples conjuntos de polarización con diferentes exponentes y así obtener por ejemplo bases triple zeta más doble polarización (TZ2P), etc. Para el nivel de HF, es usual utilizar a lo más bases TZ2P ya que incluso las bases DZP usualmente dan buenos resultados, en comparación con el límite de HF. Sin embargo, los métodos de correlación requieren más funciones de base, con más funciones de polarización y altos momentos angulares para obtener el mismo nivel de convergencia.

3.2.2 Balanceo de las bases

En principio, se pueden introducir todos los conjuntos de bases de polarización sin alterar el número de bases con momentos angulares menores; sin embargo, esto puede provocar artificios ya que se podrían utilizar funciones con altos momentos angulares para compensar las malas descripciones de los momentos angulares menores. Como regla general, se deben de utilizar un número mayor de funciones con momentos angulares menores que con las de mayores momentos angulares, al menos con una diferencia de uno y por lo regular sólo se utilizan las funciones con el siguiente momento angular al ocupado; aunque depende del nivel de teoría utilizado y del átomo. En cuanto las bases, la experiencia y el conocimiento del comportamiento de éstas para problemas similares es muy importante para tener una base bien balanceada y se obtengan resultados aceptables generando así sólo el costo computacional necesario para los resultados que necesitamos.

Otro aspecto en el balanceo de bases es el ocasionado por el uso de conjuntos de bases mixtas, por ejemplo, el uso de una DZP en átomos que se encuentran en la parte de la molécula de nuestro interés y una base mínima para los demás átomos; o el uso de funciones de polarización solo en algunos átomos de hidrógeno que se cree pueden ser importantes. Esto también puede predisponer los resultados y crear artefactos. Por ello es recomendable usar para la mayoría de los átomos el mismo nivel de calidad de las bases incluso quitando si es necesario las bases de polarización y/o de difusión en los hidrógenos. Muy importante es tomar en cuenta un balanceo de bases por número de electrones en los átomos ya que se podría tener un sistema que esté bien descrito en algunos átomos y con mala descripción en otros átomos provocando así un desbalanceo en el cálculo. Éste efecto disminuye conforme se aumentan el tamaño de las funciones de base; por ello, el uso de pequeños conjunto de bases para sistemas que contienen elementos con una diferencia sustancial en el número de electrones de valencia, puede producir artefactos. En caso de que sea necesario el uso de conjunto de bases mixtas, éstas se tienen que usar sólo después de consideraciones muy cuidadosas.

Es claro que, excepto para sistemas químicos muy pequeños, es impráctico saturar los conjuntos de bases para disminuir el error absoluto en la energía y obtener la precisión química (1 kcal/mol). El punto importante, es escoger un conjunto de bases balanceado para mantener dicho error constante tanto como sea posible y obtener dicha precisión en las energías relativas donde los errores absolutos se cancelen.

3.2.3 Optimización y convergencia de las bases

La optimización de los exponentes en las funciones de base es un ejemplo de optimización altamente no lineal y para lograr un estudio de la convergencia de éstas, se utilizan diferentes técnicas; principalmente las bases bien temperadas. Otro caso es, por ejemplo, por el método variacional se optimiza el exponente para una función de polarización y los exponentes para funciones múltiples de polarización se fijan simétricamente alrededor del valor óptimo de la función de polarización simple con un factor, tomado típicamente entre 2 y 4.

Un análisis del conjunto de bases que han sido optimizadas por métodos variacionales, revela que la proporción entre dos exponentes sucesivos es aproximadamente constante. Tomando esta proporción constante, se reduce el problema de optimización a dos parámetros por cada tipo de función de base independientemente del tamaño de éstas, generando así las bases "*even-tempered*". La ventaja de estas bases es que se pueden generar fácilmente secuencias de conjunto de bases, las cuales tienen la garantía de converger hacia un conjunto completo de bases, lo cual es muy útil cuando se intenta extrapolar una propiedad hacia el límite del conjunto de base. La desventaja es que convergen muy lento y las bases optimizadas explícitamente por lo regular dan una mejor respuesta que las bases "*even-tempered*" para un mismo tamaño de bases. Es por ello que generaron las funciones de bases "*well-tempered*", donde se tuvo más cuidado en la región de valencia que en la región del núcleo y donde los exponentes son generados por una fórmula que contiene sólo algunos parámetros para ser optimizados.

Por lo regular, los químicos no tienen que preocuparse por la optimización del conjunto de bases, ya que se encuentran disponibles ya sea en forma de tablas o se construyen transparentemente para el usuario en los programas de química computacional. El usuario, solamente tiene que seleccionar el conjunto de base a utilizar; sin embargo, si la propiedad de interés requiere de una buena precisión de la densidad electrónica cerca del núcleo o muy lejos de él; entonces los exponentes de las bases requerirán de cierta manipulación. La optimización de las funciones de base y el tipo de conjunto de base que se utilicen, por lo regular se hace a través del estudio de una propiedad del sistema químico y no en términos de la energía como se explicará a continuación.

3.2.4 Bases contraídas

Una de las desventajas de usar métodos variacionales para optimizar el conjunto de bases en términos de la energía es que ésta depende en gran medida de los electrones de "*core*", es decir, los electrones internos; sin embargo, las propiedades químicas dependen en gran medida de los electrones de valencia y de la región que se encuentra lejana al núcleo de la función de onda (como polarizabilidad), la cual no es energéticamente importante; por lo tanto en la optimización, se obtendrán funciones óptimas para los electrones internos y no tan óptimas para electrones de valencia. Para lograr una buena descripción de la región química, se podrían utilizar una gran cantidad de funciones de base tanto para los electrones internos como para los de valencia además de las funciones de polarizabilidad; sin embargo, para no perder el balanceo, utilizaríamos una gran cantidad de funciones de base lo que haría el cálculo impráctico. Una solución a ello es la utilización de funciones de base difusas (funciones s o p) que tienen un exponente muy pequeño. Las funciones difusas son muy útiles cuando se tienen electrones de enlace con interacciones muy débiles como sucede en los aniones y en los estados excitados o cuando la propiedad de interés dependen en gran medida del comportamiento de los electrones muy

lejos del núcleo; como la polarizabilidad o en la esfera de Watson. Por lo regular, dichos exponentes son manipulados por el investigador.

Sin embargo, lo que más se utiliza para resolver dicho problema es la utilización de bases contraídas. Como los orbitales *de core* cambian muy poco conforme cambian las interacciones químicas (enlaces); entonces los coeficientes de la expansión de los orbitales moleculares de estas funciones cambian muy poco, por lo tanto, se puede suponer que son constantes, logrando así que los orbitales *de core* sean descritos por una combinación lineal fijas de funciones de base donde el único parámetro variacional será el coeficiente de dicha combinación lineal. La combinación de todo el conjunto de funciones de bases, conocidas como primitivas (PGTOs) en un conjunto más pequeño de funciones de bases formadas por una combinación lineal fija se le llama contracción y forman los llamados funciones de bases contraídas (CGTOs):

$$\chi(CGTO) = \sum_{i}^{k} a_i \chi_i(PGTO)$$

Esta "contracción" es especialmente útil para los orbitales que describen los electrones *de core* ya que por lo regular requieren de un gran número de funciones para ser representados y por lo regular son independientes del ambiente químico. La contracción de bases siempre aumenta la energía ya que existe una restricción en los parámetros variacionales lo que provoca que los conjuntos de bases sean menos flexibles; pero también reduce en gran medida el esfuerzo computacional; por lo tanto la decisión de cuánta pérdida de precisión sea aceptable depende en gran medida del ahorro del costo computacional que se tiene.

El grado de contracción es el número de PGTOs que se encuentran en la CGTO y típicamente varía de 1 a 10. Existen dos maneras para contraer las GTOs: la contracción segmentada y la general. La contracción segmentada, que fue el primer método utilizado, es cuando se particiona un conjunto de PGTOs en subconjuntos de CGTOs y en donde cada primitiva es usada en sólo un subconjunto de PGTOs. Solamente en algunos casos será necesario duplicar una o dos primitivas en dos subconjuntos adyacentes de PGTOs. En la contracción general, todas las primitivas de un átomo y de un momento angular determinado entran en todas las funciones contraídas con el momento angular determinado pero con diferentes coeficientes de contracción. Un ejemplo de esta contracción general son los llamados Orbitales Atómicos Naturales (ANO→*Atomic Natural Orbitals*).

Existen muchas bases contraídas disponibles en la literatura y construidas en los programas y por lo regular, el usuario solamente necesitará seleccionarlas dependiendo de la calidad de las bases que necesite para el cálculo. Los conjuntos de bases mas utilizados son las bases tipo Pople y DH (Dunning-Huzinaga) [24] y la principal razón de ello es la extensiva calibración que se tienen de estos conjuntos lo que ha provocado que existan una gran cantidad de cálculos reportados con estas bases, por lo que es posible tener una buena idea de la precisión que se obtendrá al utilizar estos conjuntos de bases. Por lo tanto, entre más cálculos reportados existan de un conjunto de bases, más comunes y populares serán y la calibración de estas será cada vez mayor.

Para moléculas orgánicas, el conjunto de funciones 3-21G, 6-31G y 6-311G, con sus funciones extra de polarización y difusión son las más utilizadas para cálculos cuantitativos; los conjuntos de "*correlation consistent*", los cuales recuperan la energía de correlación de los electrones de valencia y pueden servir

para estudios de convergencia al límite del conjunto de base, han sido los más usados últimamente para cálculos con alta precisión. Los métodos CBS, G2 y G3 tienen un amplio uso con resultados con alta precisión ya que estos métodos han sido calibrados con 125 propiedades atómicas y moleculares utilizando una corrección empírica para tratar de anular los errores sistemáticos de las bases y los métodos. Los resultados de estos métodos son comparables a cálculos hechos con un nivel de CCSD(T)/cc-pVTZ pero con un costo computacional mucho menor [25]. Los conjuntos Wachters y Hay son los más populares para los metales de transición y los conjuntos de potenciales efectivos, particularmente Hay-Wadt, LANL2DZ, Dolg y SBKJC son usados para elementos pesados: del Rb en adelante. Es importante señalar que los potenciales *de core* deben ser usados junto con sus conjuntos de bases de valencia para los que fueron creados [26].

3.2.5 Error de sobreposición de la base (Basis Set Superposition Error)

Las bases que se utilizan principalmente, están centradas en los núcleos y tienen un error debido a que el conjunto de las bases no está completo, además del error que nos da el límite de la teoría utilizada. Sin embargo, los químicos al estar interesados en las diferencias entre las energías relativas y por lo tanto necesitar que los errores sean lo más constantes posibles; utilizan bases balanceadas y por lo regular las mismas bases al calcular las energías. Pero, al momento de hacer esto, estamos comparando energías con diferentes geometrías lo que provoca que exista un error, ya que cambiamos el arreglo espacial de los núcleos y la calidad de los conjuntos de bases no es la misma en todas las geometrías debido al hecho de que la densidad electrónica alrededor de un núcleo puede ser descrita por funciones centradas en otro núcleo. Si se utilizaran conjuntos de bases completas o una solución numérica (utilizando una malla) no existiría dicho error que puede ser muy importante cuando se calculan efectos pequeños como energías de Van der Waals y puentes de hidrógeno. De hecho, al publicar cálculos que impliquen dichos efectos, se debe de tomar en cuenta este error el cual es llamado el error de sobreposición de la base (*Basis Set Superposition Error*→BSSE). Dicho error, sobreestima los puentes de hidrógeno al reducir variacionalmente la energía ya que se utilizan más bases para la descripción del sistema en por ejemplo un dímero que en un monómero.

Una manera de reducir dicho error es, obviamente, aumentar el tamaño de la base; sin embargo, esto es computacionalmente muy costoso y de hecho no es viable en nuestro días. Por lo tanto, se estima el BSSE mediante la corrección de contrapeso (*counterpoise correction*→CP) [27] el cual, estima el BSSE como la diferencia entre las energías de los monómeros con una base regular y las energías calculadas con un conjunto de base completo para el complejo. La corrección CP es definida como:

$$E_{CP}=E(A)_{ab}+E(B)_{ab}-E(A)_{a}-E(B)_{b}$$

Donde $E(A)_{ab}$ es la energía de monómero A calculado con el conjunto completo de bases utilizado para el dímero ab (es decir, se tiene el conjunto de bases como si se calculara el complejo ab pero no se tiene el núcleo ni los electrones de B, llamándoseles orbitales fantasmas), $E(B)_{ab}$ es la energía de monómero B calculado con el conjunto completo de bases utilizado para el dímero ab, $E(A)_{a}$ es la energía del monómero A calculado con el conjunto de bases del monómero A y $E(B)_{b}$ es la energía del monómero B calculado con el conjunto de bases del monómero B, todos utilizando la geometría obtenida para el dímero.

El BSSE siempre está presente, de hecho, el llamado efecto del conjunto de base, donde se tiene una mejora al aumentar el tamaño de la base y al cambiar las geometrías, puede ser considerado como BSSE intramolecular; sin embargo, este error es difícil de definir y por lo regular es ignorado.

3.3 Métodos

Una vez que se ha visto la importancia de las bases, es necesario conocer los diferentes métodos, recordando que al resolver éstos, resolvemos las ecuaciones que estos generan y no la ecuación de Schrödinger en sí. En el método de HF, al darle una estructura a las funciones de onda utilizando un determinante de Slater, se puede suponer que para sistemas de capa cerrada cada par de electrones apareados pueden ser descritos por la misma función orbital espacial y sólo se cambian la función de spin a lo que se llama Hartree-Fock restringida (RHF→*Restricted Hartree-Fock*) y para sistemas de capa abierta se puede suponer que cada par electrónico comparte la misma función espacial a lo que se le llama Hartree-Fock restringida de capa abierta (ROHF→*Restricted Open shell Hartree-Fock*) o que cada electrón tiene una función espacial diferente a lo que se llama Hartree-Fock no restringida (UHF→*Unrestricted Hartree-Fock* generando las ecuaciones de Pople-Nesbet). Para sistemas de capa cerrada; excepto para algunos casos raros como biradicales, tanto RHF como UHF dan la misma función de onda y por lo regular se usa RHF ya que el cálculo es más fácil de implementar y requiere menos tiempo de cómputo. Para sistemas de capa abierta, UHF da una energía variacional y un costo computacional menor con respecto a RHF; sin embargo, esta técnica introduce un error llamado contaminación de spin ya que la función de onda generada por este método, no es una función propia del operador de spin; mientras que la verdadera función de onda y la función de onda ROHF sí son funciones propias de dicho operador. Por lo tanto, lo que se hace es calcular el valor esperado del operador de spin para la función de onda generada por UHF y se compara con el verdadero valor esperado para el estado basal; si la discrepancia no es significante, entonces el método UHF generó una función de onda aceptable; en caso contrario se puede usar la técnica ROHF u otros métodos que no generen tanta contaminación de spin.

Hasta ahora, los resultados obtenidos con el método de Hartree-Fock son muy cercanos a los experimentales, alrededor de 1% arriba de ellos; sin embargo, para los químicos es muy importante llegar a la precisión química al comparar las diferencias de energía entre dos diferentes estados y justamente esta precisión química es del 1% de las energías totales, por ello, pequeños errores en el cálculo de energías totales conducen a grandes errores en diferencias de energías. Por esta razón, muchas veces resulta imprescindible ir más lejos que la aproximación de Hartree-Fock. Debido a esto y a lo generalizado del método de Hartree-Fock, se ha llamado al límite de Hartree-Fock a aquella energía obtenida utilizando una base completa pero que no toma en cuenta la energía de correlación, la cual se define como la diferencia entre el eigenvalor exacto del hamiltoniano y su valor esperado en la aproximación de Hartree-Fock. Los métodos que comienzan con un cálculo HF y después incluyen la correlación electrónica son los llamados métodos de correlación.

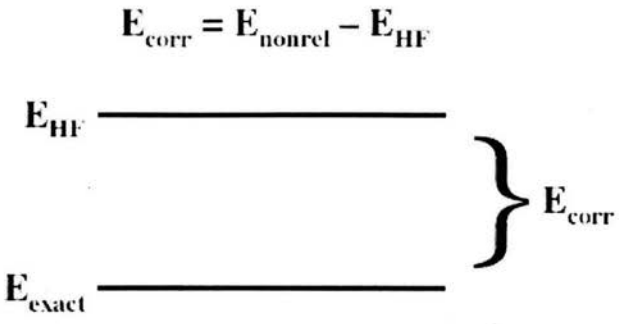


Figura 3.5 Definición de energía de correlación

3.3.1 Métodos de correlación

Para recuperar la energía de correlación, se han utilizado varias técnicas. La primera es el uso de una función de onda representada por muchos determinantes de Slater, llamada función de onda multideterminantal, la cual tiene la siguiente forma:

$$\psi = C_0\psi_0 + \sum_{a,p} C_a^p \phi_a^p + \sum_{\substack{a<b \\ p<q}} C_{ab}^{pq} \phi_{ab}^{pq} + \sum_{\substack{a<b<c \\ p<q<r}} C_{abc}^{pqr} \phi_{abc}^{pqr} + \cdots$$

donde,

$$\phi_0 = |\phi_1\phi_2\cdots\phi_a\phi_b\cdots\phi_n|$$
$$\phi_a^p = |\phi_1\phi_2\cdots\phi_p\phi_a\cdots\phi_n|$$
$$\phi_{ab}^{pq} = |\phi_1\phi_2\cdots\phi_p\phi_q\cdots\phi_n|$$

es decir, se cambian los orbitales ocupados a, b, etc, por los orbitales virtuales p, q, etc; de tal manera que se tienen excitaciones simples (primera sumatoria), excitaciones dobles (segunda sumatoria), excitaciones triples (tercera sumatoria) y así sucesivamente. A este método se le llama interacción de configuraciones (CI→*Configuration Interaction*) y dependiendo del número de excitaciones usadas, se puede llamar CIS (*Configuration Interaction Single-excitation*), CISD (*Configuration Interaction Single and Double excitation*), CISDT (*Configuration Interaction Single, Double and Triple excitation*) y solamente para cálculos que requieran una gran precisión se utiliza CISDTQ (*Configuration Interaction Single, Double, Triple and Quadruple excitation*). Si sólo se utiliza por ejemplo las excitaciones dobles entonces se llamará CID (*Configuration Interaction Double excitación*).

En estos cálculos, se optimizan variacionalmente los coeficientes de los determinantes y no los coeficientes que se encuentran en las bases de los orbitales; es por ello que, el cálculo de referencia de HF ψ_0 es muy importante hacerlo con una base lo suficientemente completa para que el cálculo CI tenga una buena precisión y pueda converger. Un cálculo CI que incluya todas las funciones de

configuración posibles con la simetría apropiada, se denomina un cálculo full CI. La diferencia de la energía obtenida por el cálculo HF y por un full CI con la misma base se llama energía de correlación de la base. Desgraciadamente, debido al crecimiento descomunal de determinantes de Slater que se pueden dar; por lo regular, se trunca la lista de los determinantes a las dobles (las cuales son muy importantes en la contribución de la función de onda y las monoexcitaciones son muy importantes para una buena descripción de las propiedades monoelectrónicas), triples o cuádruples excitaciones; aunque muchas veces se utilizan aproximaciones para estimar la contribución de cuádruples excitaciones, como la corrección de Davidson. También se puede usar la aproximación *de core* congelado (FC→*frozen-core*) donde no se incluyen las excitaciones de los orbitales de la capa *de core* de la molécula. Los problemas al utilizar un cálculo CI limitado es que se deben de escoger bien las excitaciones que no se incluirán; de lo contrario, se puede provocar que el cálculo sea menos preciso que un UHF. Además, los cálculos de CI limitados no serán consistentes con el tamaño, es decir, el error en la energía de un cálculo, crece de una manera no proporcional con el tamaño de la molécula. Debido a esto y a la lenta convergencia de los cálculos, existen otros métodos de correlación que se usan más comúnmente como son MRCI (*Multireference Configuration Interaction*), CC (*Coupled Cluster*) y DFT (*Density Functional Theory*).

Además de optimizar los coeficientes de los determinantes, se puede optimizar los coeficientes de los orbitales moleculares, dando así al método del campo autoconsistente multiconfiguracional (MCSCF→*Multiconfiguration Self-Consistent Field Method*) que al tener más parámetros variacionales, logrará que la energía obtenida sea menor además de ser un método consistente con el tamaño; pero con el problema de tener un costo computacional mucho mayor. Sin embargo, se pueden obtener muy buenos resultados con este método inclusive si utilizamos pocas funciones de configuración (determinantes). El tipo de método MCSCF más comúnmente utilizado es el método del espacio completo activo del campo autoconsistente (CASSCF→*Complete Active-Space Self-Consistent Field Method*) el cual requiere de mucha manipulación por parte del usuario al momento de escoger los orbitales activos.

Otro método ampliamente usado es el método de interacción de configuraciones multireferencial (MRCI→*Multireference Configuration Interaction*), el cual combina el MCSCF y los métodos CI convencionales. En el método MRCI, primero se efectúa un cálculo MCSCF para producir la función de onda multideterminantal de referencia, después se mueven los electrones fuera de los orbitales ocupados a orbitales virtuales y se hace un cálculo CI normal. Por lo regular, solamente se toman en cuenta las mono y dobles excitaciones; las cuales ya incluyen algunas excitaciones triples y cuádruples dentro de los determinantes. Las funciones CASSCF se usan a menudo como punto de partida para cálculos MRCI y este método reduce drásticamente la inconsistencia en el tamaño del cálculo CI y tanto el método CASSCF como MRCI convergen mucho más rápido que los cálculos CI.

Otra alternativa para adicionar la energía de correlación es agregándola como una perturbación, generando así cálculos que son consistentes con el tamaño pero que al no ser variacionales, se pueden obtener resultados tanto mayores como menores del valor exacto. La aplicación de la teoría de perturbaciones a muchas partículas interactuantes generalmente es llamada como la teoría de perturbaciones de muchos cuerpos (MBPT→*Many-Body Perturbation Theory*). Para los químicos, se puede utilizar el hamiltoniano de orden cero; es decir, sin perturbar, a aquel que se obtiene del operador de Fock; a esta forma de MBPT se le llama teoría de perturbaciones de Moller-Plesset (MP). Por ejemplo, para la corrección de segundo orden (MP2) se obtiene una serie de integrales de 4 centros que por lo regular son un poco difíciles de evaluar y sólo se necesitará evaluar excitaciones dobles; sin

embargo, el costo computacional es mucho menor que los métodos variacionales antes vistos. También se tienen las fórmulas para MP3 y MP4 donde se tienen que evaluar excitaciones simples y dobles; doble, triple y cuádruple respectivamente. En los cálculos MP4, la evaluación de los términos que incluyen los determinantes triplemente sustituidos consume mucho tiempo, de forma que estos términos a veces se desprecian, dando la aproximación MP4-SDQ (MP4-*Simple Double Quadruple excitations*). Muchas veces también se usa la aproximación del core congelado, donde se omiten los términos que incluyen las excitaciones que sean de los orbitales *de core*. El nivel MP mas usado es el MP2 seguido del MP4. Se pueden hacer cálculos con multiconfiguraciones y después aplicar MP2 formando así los métodos CASPT2. La precisión de un cálculo MP4 puede ser equivalente a un cálculo CISD.

Otro método muy comúnmente utilizado es el llamado cluster acoplados (CC→*Coupled Cluster*), el cual, es muy parecido a los cálculos CI ya que la función de onda también es una combinación lineal de muchos determinantes; aunque más complicada, y se debe tener cuidado principalmente al escoger los determinantes para el cálculo. Como en CI, existen varios órdenes de expansión de CC, llamados CCSD (*Coupled Cluster Simple and Double excitations*), CCSDT (*Coupled Cluster Simple, Double and Triple excitations*), etc. Además existen métodos como CCSD(T) donde la triple excitación es incluida perturbativamente. CC es un método variacional si las excitaciones se van incluyendo sucesivamente. La gran ventaja de CC es que es un método consistente con el tamaño, además de que sus resultados son más precisos que CI debido a que por ejemplo, cuando se incluye la excitación doble en CC, también se están incluyendo sus excitaciones triples y cuádruples equivalentes en CI. Un cálculo full CC es equivalente a un cálculo full CI. También se han desarrollado (Pople y colaboradores) otros métodos que son intermedios entre CC y CI llamados método de interacción de configuraciones cuadráticos (QCI→*Quadratic Configuration Interaction*). El cálculo QCISD es una aproximación del cálculo CCSD. Estos cálculos son populares ya que dan una precisión muy alta en moléculas orgánicas con un menor costo computacional en comparación con los métodos CC. Los métodos más populares para cálculos de alta precisión son QCISD(T) y CCSD(T) ya que han demostrado ser muy precisos.

En general, la precisión relativa de los métodos vistos son:

$$HF << MP2 < CISD \cong MP4 \cong CCSD < CCSD(T) < CCSDT < FullCI$$

En la siguiente figura se puede ver una pequeña jerarquía de los cálculos *ab initio* donde lo más recomendable es seguir la línea inclinada para obtener mejores resultados.

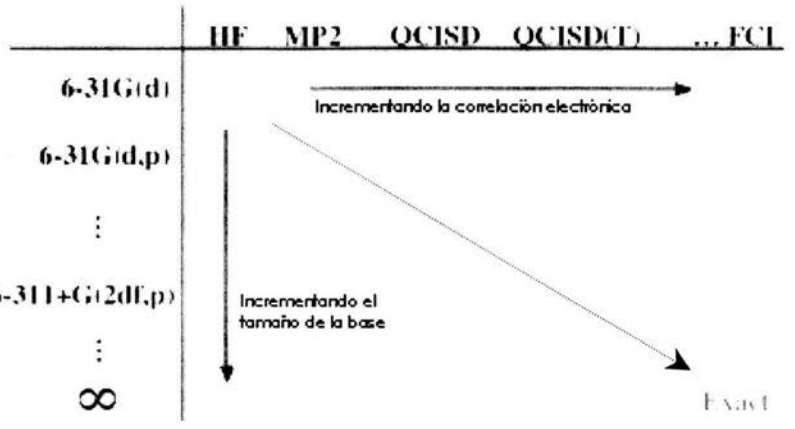


Figura 3.6 Jerarquía de los cálculos *ab initio*

Un método que es hasta ahora es el más preciso de todos; pero cuyo costo computacional es muy alto es el llamado método de Monte Carlo Cuántico (QMC→*Quantum Monte Carlo*). Estos métodos resuelven numéricamente la ecuación de Schrödinger al transformar ésta de una representación diferencial a una representación integral. En este método, no se dependen de expansiones de conjuntos de base además de que la función de onda puede tener cualquier estructura (en el caso del método de Monte Carlo Cuántico variacional) donde la energía de correlación se encuentra explícitamente. Los métodos de Monte Carlo Cuántico de difusión y de función de Green usan una función de onda numérica en donde se tiene que tener cuidado en las propiedades nodales de la función antisimétrica. Por lo regular, se utilizan los nodos de la función de onda obtenida por HF; pero existen cálculos más sofisticados donde las superficies nodales son "relajadas" para poder optimizar la función de onda. Debido a la gran precisión que se puede alcanzar con éstos métodos y al paralelismo inherente de los algoritmos, es muy posible que en un futuro tengan una gran importancia debido a que cada vez serán más rápidos y a que son altamente paralelizables. Actualmente, la mayoría de los investigadores que usan cálculos de QMC, usan sus propios programas e inventan los métodos que estos contienen.

3.3.2 Teoría del funcional de la densidad (DFT→Density Functional Theory)

La energía electrónica de un sistema químico es un funcional de la densidad electrónica y en 1964, Pierre Hohenberg y Walter Kohn [28] demostraron que para moléculas con un estado fundamental no degenerado, la energía molecular del estado fundamental, la función de onda y todas las demás propiedades electrónicas, están determinadas unívocamente por la densidad de probabilidad electrónica del estado fundamental, la cual es una función de solo tres variables independientemente del número de electrones. Basado en este teorema, la teoría del funcional de la densidad intenta calcular la energía y otras propiedades moleculares del estado fundamental a partir de la densidad electrónica del estado fundamental; aunque actualmente, también existen métodos para calcular estados excitados. Kohn y Sham [29] idearon un método muy similar en estructura al método de HF para calcular la energía. En éste método, la densidad electrónica es expresada como una combinación lineal de funciones de base (orbitales de Kohn-Sham) similares matemáticamente; aunque no equivalentes, a la forma de los

orbitales HF y éstos pueden ser obtenidos por un procedimiento autoconsistente como un cálculo HF; pero ahora para resolver las ecuaciones de Kohn-Sham. En la actualidad existe un debate sobre cómo asignar similitudes e interpretar físicamente las diferencias entre estos tipos de orbitales.

La energía electrónica total exacta puede ser escrita como una suma de términos, las cuales dependen de la densidad. Los términos son debidos a la energía cinética de un sistema con electrones no interactuantes con la misma densidad que la del sistema real, a una energía potencial debido a las interacciones electrónicas con un potencial externo debido típicamente al núcleo, a una energía de repulsión electrón-electrón, conocida también como energía electrostática de Hartree y a una energía de correlación e intercambio que contiene no sólo las contribuciones de correlación y de intercambio; sino que también las contribuciones de la diferencia entre la verdadera energía cinética del sistema y la energía cinética de los electrones no interactuantes. Sin embargo, para este último funcional, no existe una forma conocida exacta, debido a esto y a pesar de que la teoría DFT es variacional, no existe un método para señalar cómo se puede, de manera sistemática, diseñar un funcional que minimice la energía o se aproxime a la energía exacta; lo que genera que los cálculos dependan en gran medida del funcional y del conjunto de base.

Existen varias aproximaciones para dicho funcional, entre las que más se usan están la aproximación de densidad local (LDA→*Local Density Approximation*) donde se supone que la densidad varía de forma extremadamente lenta con la posición y donde los electrones constituyen un gas de electrones uniforme y da resultados aceptables para el estudio de estructura de bandas en sólidos, la aproximación de densidad de spin local (LSDA→*Local Spin Density Approximation*), la cual es utilizada para sistemas de capa abierta y sistemas próximos a la disociación y sería un análogo a UHF y para sistemas donde la aproximación LDA no es aceptable, como sucede en las moléculas, existe la aproximación del gradiente generalizado, corregido o no locales los cuales persiguen utilizar la variación de la densidad electrónica con la posición, introduciendo dentro del funcional los gradientes de las densidades. En éstos casos, cualquier funcional de intercambio se puede combinar con cualquier funcional de correlación. También existen funcionales híbridos que mezclan ambas aproximaciones vistas y piezas del cálculo de HF.

Los funcionales pueden ser desarrollados desde los fundamentos de la mecánica cuántica o pueden tener funciones parametrizadas para reproducir datos experimentales, de tal manera que sólo los cálculos DFT se pueden clasificar como *ab initio* o semiempíricos, dependiendo del funcional que utilicen.

3.3.3 Métodos semiempíricos

Los cálculos semiempíricos han sido establecidos con la misma estructura general de un cálculo HF en donde se tiene un Hamiltoniano y la función de onda pero cierta información es aproximada o completamente omitida, como por ejemplo, usualmente los electrones *de core* no son incluidos en el cálculo y se usa un conjunto de base mínima; también son omitidas algunas integrales de traslape (ya que un alto porcentaje del costo de los cálculos *ab initio* es calcular y manipular integrales) y para corregir todos estos errores, los cálculos son parametrizados. En los cálculos *ab initio*, tenemos un Hamiltoniano electrónico exacto pero aproximamos la función de onda, en los cálculos semiempíricos, podemos aproximar el Hamiltoniano para hacerlo más fácil en cuanto a los cálculos necesarios para resolverlo. Los parámetros por lo regular se estiman de datos experimentales o de cálculos *ab initio*. La

ventaja de éstos cálculos es que son mucho más rápidos que los cálculos *ab initio* pero se tiene un costo de precisión en los resultados y sólo muy pocas propiedades pueden ser predecidas confiablemente. Si se tiene una molécula que es similar a moléculas que se tienen en una base de datos para parametrizar el método; entonces, es muy probable que los resultados sean aceptables; de lo contrario, los resultados pueden ser muy pobres. Los métodos semiempíricos, han sido parametrizados para reproducir varios resultados, entre ellos, geometría y energía (calor de formación, por lo tanto, no se tiene que hacer la corrección del punto cero debido a que las correcciones termodinámicas están implícitas en la parametrización). Algunos investigadores se han extendido a momentos bipolares, calores de reacción, potenciales de ionización y algunos métodos han sido parametrizados para reproducir una propiedad específica como espectros electrónicos.

Cálculos CIS de una función de onda semiempírica pueden ser usadas para calcular estados electrónicos excitados. Los cálculos semiempíricos han sido muy exitosos para la descripción de la química orgánica; sin embargo, existen algunos métodos semiempíricos que han sido desarrollados específicamente para la descripción de la química inorgánica.

Los métodos semiempíricos más ampliamente usados son AM1 y PM3 debido a que tienen una precisión aceptable; sin embargo, estos métodos no se pueden usar actualmente para sistemas químicos que contengan más de 1000 átomos; a menos que se utilicen algunos métodos que sólo pueden ser aplicados a algunas moléculas o se combinen cálculos para regiones diferentes de la molécula. Para sistemas químicos con una cantidad mayor de moléculas se utiliza la mecánica molecular.

3.3.4 Mecánica Molecular

La mecánica molecular no es un método mecanocuántico, ya que no trata con un Hamiltoniano, con una función de onda o con la densidad electrónica. En lugar de ello, el método usa un modelo de una molécula compuesta por átomos que se mantienen unidos por enlaces y usa parámetros como constantes de fuerza de tensión de enlace y flexión de enlace y permite interacciones entre los átomos no enlazados. El método construye una expresión de la energía potencial (llamada energía estérica, aunque también se puede parametrizar para obtener calores de formación) que es una colección de las posiciones atómicas; ya que una de las suposiciones más importantes es la transferibilidad de los parámetros entre los diferentes átomos, es decir, por ejemplo un enlace simple carbón-carbón tendrá el mismo comportamiento no importando en qué molécula se encuentre y por lo tanto, un parámetro puede ser usado en una gran cantidad de moléculas; aunque sí habrá diferencia entre las diferentes hibridaciones del carbono e incluso algunos métodos harán diferencia entre los diferentes grupos funcionales, es decir, un CO tendrá un parámetro diferente en un ácido carboxílico que en una cetona.

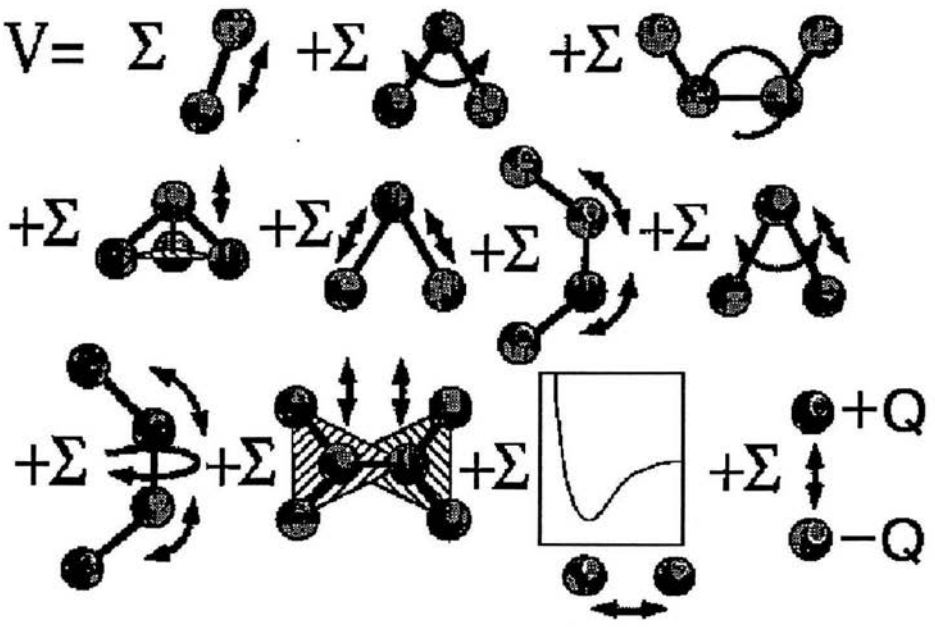


Figura 3.7 Expresión de la energía potencial entre los diferentes términos

El rendimiento de estos métodos depende de cuatro factores:

- la forma de la expresión de la energía
- los datos usados para parametrizar las constantes
- la técnica usada para optimizar las constantes de los datos
- la habilidad del usuario para aplicar el método de una manera consistente con sus virtudes y limitantes.

Las expresiones explícitas usadas para cada uno de los términos definen a lo que se llama campo de fuerza de la mecánica molecular, ya que las derivadas de la función energía potencial determinan las fuerzas sobre los átomos. Los diferentes campos de fuerzas, difieren en el número de términos en la expresión de la energía, la complejidad de éstos y la manera en que las constantes son obtenidas. Con estos métodos, no se pueden modelar procesos electrónicos, ya que los electrones no están explícitamente incluidos. Los términos (figura 3.7) en los campos de fuerzas son de valencia (que modelan el cambio de energía por el ángulo de enlace, torsión, el estiramiento del enlace, etc), términos cruzados (que modelan como un movimiento de la molécula, afectan otra), electrostáticas (por lo regular el potencial de Coulomb), de Van der Waals (por lo regular el potencial de Lennard-Jones), etc.

El problema con la mecánica molecular, es el no poder confiar en los resultados que da. Quizá la mayor aplicación son dinámicas y diferencias de energías entre confórmeros. Tampoco existen muchas propiedades químicas que puedan ser definidas por éste método al no poder modelar procesos

electrónicos, ni tampoco modelar reacciones químicas ya que no puede haber una manipulación matemática que examine la ruptura y formación de enlaces. La gran ventaja de estos cálculos es que pueden modelar moléculas muy grandes, como ADN, proteínas, etc; y por lo regular, los programas de computadoras que las acompañan, tienen una interfaz gráfica muy fácil de usar. Por lo tanto, para escoger el mejor campo de fuerza una vez que se ha escogido usar la mecánica molecular es ver que existan estudios similares en la literatura y validar dichos estudios con datos experimentales.

3.4 Propiedades

Con los diferentes métodos que se pueden utilizar para la química cuántica computacional, es importante conocer las propiedades moleculares que se pueden calcular con éstos métodos y cuáles de éstos funcionan mejor para una propiedad determinada. Varias de estas propiedades se pueden obtener de la función de onda o de la densidad electrónica, y otras, se obtienen dependiendo del método utilizado, mediante algunas correcciones. También existen algunos métodos como QSPR (*Quantitative Structure Property Relationships*) y QSAR (*Quantitative Structure Activity Relationships*) que mediante bases de datos, pueden ayudar a obtener propiedades a partir de la estructura molecular. A continuación se verán los métodos más utilizados en la química cuántica computacional y las propiedades en las que mejor y menor precisión tienen.

Para cálculos HF, se sabe que dan buenos resultados sólo para la primera energía de ionización, momento dipolar molecular (aunque para momentos dipolares pequeños se requieren cálculos CI) y geometrías aceptables para moléculas que no tienen metales de transición. HF por lo regular da malos resultados en la disociación de los enlaces; pero con el método UHF se pueden obtener resultados cualitativos aceptables. Los cálculos para las distancias de los enlaces covalentes por lo regular son cortos debido a que no existe una repulsión suficiente, de igual manera las interacciones electrostáticas comúnmente son más grandes y esto se debe a que no se toma una suficiente interacción. Se obtienen intensidades IR y Raman aceptables, por lo regular da frecuencias vibracionales armónicas 10% más grandes; pero se pueden escalar. Se espera que HF tenga una menor precisión en sistemas excitados y cuando exista reducción u oxidación (ionización).

Los cálculos de métodos de correlación tienen una mayor precisión en energías y geometrías y son muy importantes para el cálculo de propiedades como puentes de hidrógeno, fuerzas de Van der Waals, etc. Los cálculos CID y CISD son razonablemente factibles para un número grande de electrones y de funciones de base al ser métodos variacionales; pero no son consistentes en el tamaño y esta deficiencia aumenta conforme aumenta el tamaño del sistema. Para moléculas que contienen átomos de la primera fila, el método CISD da para moléculas con 20 electrones alrededor del 82-90% de la energía de correlación, 68-78%, para 50 electrones y 55-67% para 100 electrones. En el caso de los métodos QCI, que son consistentes con el tamaño pero no son variacionales, el método QCISD no da una buena descripción de los efectos que generan las configuraciones triplemente excitadas, las cuales son importantes para algunos sistemas; sin embargo, el método QCISDT que no tiene este problema, no es realizable para sistemas grandes pero se puede usar el método QCISD(T). Por lo regular, los métodos QCISD y QCISD(T) son muy confiables y precisos, por lo tanto se utilizan como benchmark para cálculos teóricos; aunque se sabe que han tenido fallas para un par de sistemas. Para los métodos MCSCF y MRCI, es deseable usarlos cuando la función de onda de referencia obtenida por el método

de HF da una descripción pobre del sistema. Desafortunadamente, estos métodos requieren de una alta sofisticación por parte del usuario para que den buenos resultados debido a que se debe de escoger el espacio activo; por ello se puede usar el método CASSCF que incluirá a todos los orbitales activos. Estos métodos son muy precisos; pero tienen un costo computacional muy alto; por lo tanto, se debe tener en cuenta la propiedad, la precisión y el costo computacional y obviamente tener entendimiento del método para que los resultados sean los esperados.

Para los métodos MP, los más usados son MP2 y después MP4. Son menos costosos computacionalmente que los métodos CI y ambos requieren de bases "suficientemente" grandes (de 6-31G(d) en adelante). No requieren tanta sofisticación del usuario como los métodos CI y son consistentes con el tamaño. Estos métodos han sido muy usados y se conocen bien sus virtudes y limitantes, los cuales están muy documentados. Estos métodos no son variacionales (aunque actualmente, la consistencia en el tamaño es visto como más importante que el ser variacional). Se ha establecido que el uso de bases más completas es muy importante para estos métodos, incluso más importante que si se usa la misma base y se hace un cálculo con MP2 o con MP4. En cálculos MP con una función de referencia dada por UHF, la contaminación de spin puede ser importante y siempre hay que tenerlo en cuenta. Los cálculos MP no dan buenas geometrías fuera del equilibrio y no son aplicables generalmente a estados electrónicos excitados. Sin embargo, los métodos más utilizados para obtener propiedades de sistemas que incluyen los efectos de correlación en moléculas basales son MP2 y DFT, aunque suele sobreestimar los efectos de la correlación electrónica, las distancias de enlace suelen ser más largas y las barreras de reacción suelen ser más altas.

Para los métodos CC, pasa exactamente lo mismo que los métodos CI en cuanto a las excitaciones triples y por lo tanto también existen los métodos CCSD(T). Los métodos CC son consistentes con el tamaño, también son tomados como benchmark para cálculos teóricos convencionales, incluso sobre resultados QCI. Estos métodos junto con DFT tienen una contaminación de spin menor que CI y MP y se tienen algunas teorías CC para el tratamiento de estados excitados. Los métodos CC no son variacionales y su costo computacional es muy alto (incluso en *I/O*). Por lo regular, éstos métodos de correlación, no pueden usarse para sistemas con más de 30 átomos pesados.

El método DFT, tiene la ventaja de permitir que se incluyan los efectos de correlación en los cálculos que consumen aproximadamente el mismo tiempo que los cálculos HF que no incluyen correlación. Debido a su menor costo computacional, es una opción muy utilizada para sistemas químicos grandes que requieren de precisión. A pesar de que incluyen la correlación, no siempre dan buenos resultados en sistemas donde importan los puentes de hidrógeno y las fuerzas de Van der Waals. La contaminación de spin en la aproximación LSDA es menor a la que se da en los métodos UHF y MP. Para cálculos LDA, los enlaces tienden a ser cortos y fuertes. Los cálculos DFT pueden alcanzar la precisión de métodos de correlación; aunque la mayor parte de las propiedades moleculares con los funcionales disponibles actualmente, no pueden igualar la precisión de los métodos CCSD(T) y QCISD(T); pero DFT puede tratar moléculas más grandes. Como esta teoría ha sido desarrollada recientemente en comparación con las demás, todavía existen muchos sistemas químicos que no se han explorado con DFT, por lo tanto es crucial comparar los resultados que se obtengan con datos experimentales o *ab initio*. Los cálculos DFT al igual que los CI requieren de bases grandes para tener buenas precisiones. Los cálculos DFT por lo regular, dan barreras energéticas muy altas y PES muy gordas y aunque el principal problema es que no existe una manera metódica de obtener funcionales, con DFT se han desarrollado definiciones cuantitativas de conceptos químicos y los llamados

parámetros de reactividad que nos pueden ayudar a explicar la reactividad química como la electronegatividad, dureza, función de Fukui, etc.

Para los métodos semiempíricos, si tenemos una molécula similar a las que se tienen en la base de datos para parametrizarla, es muy probable que los resultados sean aceptables (aunque no son tan sensibles a la parametrización como la mecánica molecular). Se sabe que por ejemplo MNDO (*Modified Neglect of Diatomic Overlap*) sobreestima las energías de los estados excitados, tiende a dar las barreras de activación muy altas, muchas veces el confórmero correcto no es el de menor energía obtenido, tiende a hacer pequeñas las barreras de rotación y falla en sistemas hipervalente. Para ZINDO (*Zerner's Intermediate Neglect of Differential Overlap method*), predice bien las transiciones UV, con excepción de metales con electrones desapareados y da muy pobre resultados para la optimización de geometrías. Para el método AM1 (*Austin Model 1*), mejora las barreras de activación dadas por MNDO, falla en algunas barreras de rotación, predice pobres geometrías para moléculas con fósforo, tiene errores sistemáticos con la predicción de energías de elementos del grupo alkilo, los grupos nitro tienden a ser muy positivos en energía, da mala orientación para los enlaces de hidrógeno aunque da buena distancia de enlace y consistentemente tiende a dar entalpías de enlace bajos. PM3 (*Parameterization method 3*) tiende a mejorar las energías y geometrías que AM1 pero tiene algunas limitantes, sobre todo en la barrera de rotación del enlace C-N en los péptidos es bajo, falla para compuestos con germanio, falla con las afinidades protónicas, tiende a hacer enlaces de hidrógeno mas cortos por alrededor de 0.1 A°; pero da buena orientación. En general PM3, predice mejor las energías y distancias de enlace que AM1 y MNDO. Un método que da mejores resultados que PM3 y AM1 es SAM1 (*Semi-ab initio method 1*) pero tenemos un costo computacional que debemos tener en cuenta. Los métodos semiempíricos en general dan pobres resultados para fuerzas de Van der Waals y fuerzas de dispersión molecular, debido a que carece de bases difusas.

En el caso de los métodos de la mecánica molecular, la mejor técnica para escoger el campo de fuerza a usar es buscar estudios similares en la literatura y validar los resultados con datos experimentales. Una generalización de resultados para los diferentes campos de fuerzas nos dice que los campos MM2 (*Molecular Mechanics 2*), MM3 y MMFF (*Merck Molecular Force Field*) han tenido los mejores resultados para un amplio rango de moléculas orgánicas, los campos de fuerzas AMBER (*Austin Model Building with Energy Refinement*) y CHARMM (*Chemistry at Harvard Macromolecular Mechanics*) han dado los mejores resultados para proteínas y ácidos nucleicos y para el caso de moléculas inorgánicas, se requiere de una muy cuidadosa adaptación de los parámetros del campo de fuerzas, donde UFF (*Universal Force Field*) ha sido el más utilizado. Para estudios de dinámica molecular, los mejores estudios se han dado con los campos de fuerzas diseñados para dicho propósito, lo mismo para estudios con carbohidratos.

3.5 Precisión vs. tamaño

Los químicos, como todos los científicos de las ciencias básicas y la ingeniería, necesitan simular procesos cada vez más complejos. Se requiere simular sistemas químicos cada vez más grandes (millones de átomos) con una precisión química. Desgraciadamente, esto no es posible en nuestros días, ya que los cálculos *ab initio* y DFT pueden calcular moléculas con alrededor de 200 átomos pesados en computadoras muy poderosas; pero en muchos sistemas químicos, se requiere tomar en cuenta los efectos de solvatación, a una temperatura dada, etc; lo que provoca una mayor complejidad y

reduce la precisión obtenida. En el caso de métodos semiempíricos y de la mecánica molecular, a pesar de que se pueden hacer cálculos con millones de átomos, es difícil saber la precisión que tendrán debido a la fuerte dependencia de los parámetros que tienen.

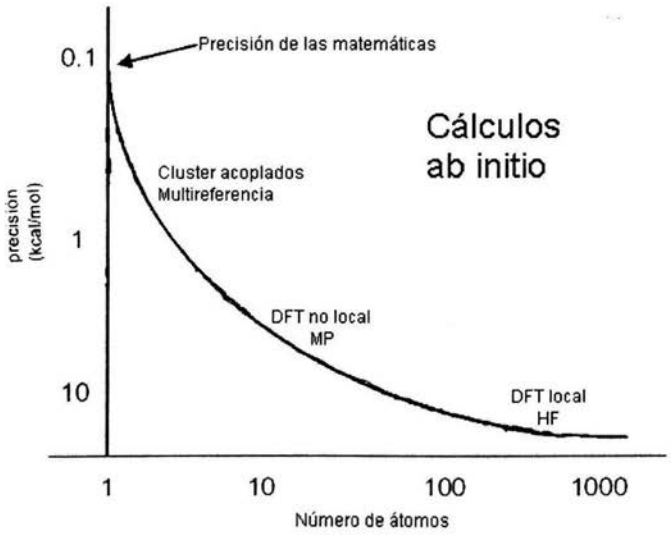


Figura 3.8 Precisión vs. tamaño en los cálculos *ab initio*

Se necesita aumentar el tamaño en el caso de la química cuántica computacional debido a que en general, la química involucra moléculas muy grandes, sobre todo en bioquímica a la que se le ha llamado la ciencia del siglo XXI, como el ADN; y exceptuando experimentos en fase gaseosa a muy baja presión; incluso sistemas con moléculas pequeñas, requieren de la interacción de disolventes y de otras moléculas con estado electrónico diferentes, como en el caso de la fotoquímica, lo que involucra una gran cantidad de moléculas o campos externos. La precisión es muy necesaria debido a las diferencias relativas en energías entre confórmeros; aunque también se requiere una gran precisión en moléculas que se encuentren en el estado de transición para poder planificar una síntesis orgánica como por ejemplo con una barrera energética de 0.25 kcal/mol y definir mejores PES para obtener cálculos dinámicos aceptables y simulaciones que nos permitan ver diferentes fenómenos como el efecto túnel en una reacción química.

Es por ello, que es muy importante aumentar la precisión de los cálculos y el tamaño de los sistemas ya sea con nuevos métodos para obtener una mayor precisión; desarrollando nuevos métodos con base en las ciencias químicas y mejores algoritmos que implementen dichos métodos para explotar mejor el nuevo hardware y software que se está desarrollando disminuyendo así los tiempos de cómputos, etc. Debido a esto es necesario utilizar todas las herramientas posibles que las ciencias de la computación y

las ciencias químicas puedan dar y para ello, se requiere de tener cierto conocimiento de las virtudes, limitaciones de los cálculos y las herramientas que se utilizan.

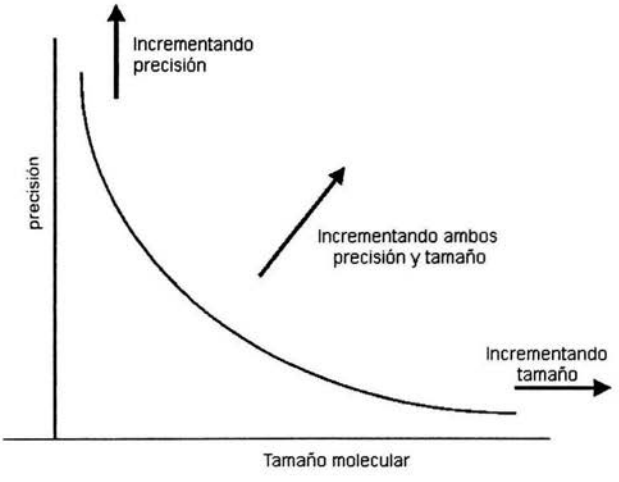


Figura 3.9 Química computacional: mayor precisión y tamaño

La manera actual de atacar estos problemas y utilizar las ventajas de los métodos es mediante una combinación de ellos utilizando QM/MM (*Quantum Mechanics/Molecular Mechanics*) el cual combina cálculos *ab initio* o DFT con cálculos de la mecánica molecular; ya que con los cálculos *ab initio* se pueden calcular muchas propiedades y modelar una reacción química y con la mecánica molecular se pueden modelar sistemas muy grandes rápidamente. En este método, una pequeña región de interés será modelada usando QM y lo demás será modelado con MM, como el solvente o la parte no activa de una proteína. Existen una gran variedad de esquemas de cálculos propuestos, la mayoría de ellos pueden ser expresados como:

$$E=E_{QM}+E_{MM}+E_{QM/MM}+E_{polarización(solvente)}+E_{fronteras}$$

Para los métodos QM/MM existen varias técnicas como MMOM (*Many Methods Of Morokuma*), ONIOM, etc. También existen métodos para simular los efectos de solvatación (recordemos que la mayoría de los métodos descritos hasta ahora, son simulaciones en fase gaseosa), los cuales pueden ser los métodos de disolvente explícito y del disolvente continuo.

3.6 Implementación y programas de la química computacional

Existen muchas maneras de implementar un algoritmo o método en las computadoras y dependiendo del tipo de computadora, se utilizarán las diferentes herramientas para aprovechar las ventajas y cómo

disminuir las limitaciones de determinada arquitectura. Un ejemplo muy importante es cuando se observa que la mayoría del tiempo que se dedica en un cálculo *ab initio*, se hace manipulando y calculando integrales, principalmente las integrales de traslape o multielectrónicas, las cuales también ocupan una gran cantidad de memoria. Los algoritmos de evaluación de integral convencional, calculan todas las integrales y las guardan en memoria para que una vez que se requieran, no se tengan que volver a calcular y sólo se buscarán en memoria. Debido a la gran cantidad de integrales y dependiendo del sistema químico con el que se esté trabajando, toda esta información no se puede guardar en la memoria *RAM*, e incluso se puede dar el caso de que no quepan ni en el disco duro. Para estos casos, existen los algoritmos de evaluación de integrales directas en donde las integrales son recalculadas conforme sea necesario, provocando así que se requiera menos memoria; aunque se requiere un mayor costo de cómputo para lograrlo. Con los avances que se han tenido, el microprocesador es muchísimo mas rápido que la memoria, sobre todo si se tiene que buscar la información en el disco duro y que una de las técnicas para lograr la aceleración superlineal es precisamente la replicación de cómputo y por lo tanto, muchas veces es mucho más rápido rehacer el cálculo que guardar la información en memoria. Además, si se tiene una computadora paralela, dependiendo del tipo de memoria y del balance de trabajo que se tiene, se puede tener un menor tiempo de ocio de los procesadores y ahorrar el tiempo de fallo de memoria y comunicaciones, lo que mejorará en mucho el tiempo de cómputo total.

Existen algoritmos que pueden recalcular algunas integrales, las más sencillas, y guardar los datos al calcular las integrales más costosas computacionalmente, para no tener que volverlas a calcular. Estos algoritmos se llaman semidirectos y tratan de obtener un balance para tratar de mejorar el tiempo de cómputo total. Los algoritmos in-core, son aquellos que guardan todas las integrales exclusivamente en la memoria *RAM* y de esa manera, evitar que se busque en la memoria del disco duro, que es la más lenta dentro de la jerarquía de la memoria; pero solamente se puede lograr, si se tiene una computadora con una gran cantidad de memoria *RAM*. Nuevamente, depende del tipo de computadora, para saber qué tipo de algoritmo sería el conveniente. De hecho, algunos programas como *Gaussian* y *NWChem*, prueban cuánta memoria accesible se tiene, para decidir que tipo de algoritmo utilizar y así mejorar el rendimiento del cálculo. También se pueden utilizar diferentes métodos para calcular las integrales de una manera más rápida que la convencional. Algunos de estos métodos son el método pseudoespectral (utilizado en el programa Jaguar), método rápido de multipolo, código del árbol, la aproximación integral multiplicativa, etc.

Los cálculos de la optimización de la geometría toman más tiempo que un cálculo simple de energía. Esto se debe principalmente a dos razones: una es que se deben de hacer muchos cálculos cada vez que se cambia la geometría y la segunda es que en cada iteración, se deben de calcular los gradientes de energía. El tiempo de cómputo que se requiere para hacer una optimización de la geometría, T_{opt}, depende del número de grados de libertad, D, las cuales son las variables geométricas a ser optimizadas (distancias de enlace, ángulos, etc). Como una regla general, la cantidad de tiempo de cómputo que se requiere para una optimización de la geometría puede ser estimada a partir del tiempo de cómputo de un single point, T_{single}, por la siguiente ecuación:

$$T_{opt} \cong 5 \times D^2 \times T_{single}$$

Un costo adicional que se tiene que tomar en cuenta es el costo del investigador para poder utilizar el programa. Un ejemplo de ello, es el programa *SPARTAN* o *Hyperchem*, los cuales son extremadamente

fáciles de usar; pero suelen ser muy ineficientes computacionalmente. Muchos de los investigadores comienzan con programas fáciles de usar pero cuando necesitan resolver problemas cada vez más sofisticados, requieren de programas donde puedan hacer cálculos más sofisticados y tener más control de los cálculos y por lo tanto requerirán de programas mas complicados y robustos, y tendrán que correrlos en supercomputadoras donde la optimización y la eficiencia computacional del programa es muy importante.

La gran mayoría de los algoritmos de la química computacional tienen un paralelismo flojamente síncrono lo que los hace muy difíciles de paralelizar eficientemente. Los algoritmos más difíciles de paralelizar son los métodos de correlación y la optimización de la geometría, debido a la gran variedad de diferentes tipos de cálculos que requieren y al alto porcentaje serial que para muchos sistemas químicos existe. Hasta ahora, los programas se han paralelizado al momento de hacer los cálculos del álgebra lineal, como la linearización e inversión de las matrices, calcular recurrentemente varias integrales o mediante el manejo de datos. La mayoría de la paralelización ha sido por grano grueso, donde no se debe de tener una gran cantidad de comunicaciones, aunque esto no siempre se cumple. Para una máquina SMP, convendría una paralelización de grano grueso acompañada de una paralelización local en cada nodo de grano fino.

Hasta ahora los programas que han hecho gran énfasis en la paralelización son: *NWChem*, *DFT++* y en menor medida *GAMESS*. *NWChem* está siendo desarrollado para que pueda correr eficientemente en computadoras masivamente paralelas, *DFT++* ha sido programado utilizando el análisis y diseño orientado a objetos tipo *UML* lo que permite que la paralelización, mantenimiento, desarrollo y optimización sea más fácil de lograr y *GAMESS* ha utilizado algunas herramientas desarrolladas para *NWChem* para lograr cierta paralelización de manera eficiente. En el apéndice B se da una lista de los programas de química computacional más usados.

3.7 Uso eficiente de las computadoras, los programas y las metodologías

Cuando se usa la química computacional para responder cuestiones químicas, el problema obvio es que el investigador necesita saber utilizar las herramientas necesarias para lograrlo, como el software. Actualmente con los grandes avances en las ciencias de la computación y las metodologías de las ciencias químicas, se pueden hacer grandes y sofisticados cálculos de química computacional utilizando la computadora sólo como una caja negra, lo que ha llevado a que la química computacional se "democratice" de tal manera que una persona no tiene que ser especialista para poder hacer dichos cálculos y obtener conclusiones pero, por otro lado, ha provocado que se puedan obtener artificios y malas conclusiones y un pésimo manejo de las computadoras.

Para poder llevar a cabo un buen proyecto de química computacional, lo primero que se debe preguntar es: ¿Qué es lo que se quiere saber? y ¿Qué precisión es necesaria para lograrlo? Para poder responder estas preguntas, se necesita conocer bien el sistema químico que se estudiará y la información que se requiere obtener. Al responder estas preguntas, se podrá saber el tipo de cálculo necesario para lograrlo y muchas veces es recomendable hacer un pequeño estudio para verificar la viabilidad del proyecto.

Al tener toda esta información, se tendrá una idea del tipo de cómputo que se necesitará, así como el software necesario para obtener los resultados y se podrá comparar entre los diferentes tipos de máquinas y programas y ver las ventajas y limitaciones que cada uno tiene y con base en ésto, escoger el mejor para las circunstancias. En esta parte, es muy importante utilizar los benchmark y poder tener acceso a diferentes arquitecturas de computadoras para obtener mayor información. Sin embargo, existen muchos datos en la literatura o de personas que pueden ayudar y lograr así la mejor decisión. Es muy recomendable, conocer el comportamiento de los benchmark y del programa antes de tomar cualquier decisión.; por lo tanto, se tienen que hacer pequeños estudios tipo benchmark donde se observará no sólo el comportamiento de la ejecución; sino también los resultados obtenidos, verificando así, los archivos de entrada del programa. Con esto, se tendrá la suficiente información del comportamiento que tendrá el cálculo y habrá una mayor seguridad del buen funcionamiento de éste.

Una vez que se ha tomado la decisión y que se han hecho los cálculos, entonces sí se podrá analizar los resultados obtenidos; pero ahora no sólo desde el punto de vista químico, sino también desde el punto de vista de la computación para ver si se puede mejorar de alguna manera el tiempo que toma el cálculo, en caso de que sea necesario, o la memoria que se requiere, etc. Para ello, es muy importante conocer las limitaciones de los cálculos obtenidos y el comportamiento del sistema químico que tenemos. De esta manera, no se desperdiciará mucho tiempo (tanto el tiempo de cómputo como del investigador) y se podrán hacer las investigaciones de manera más sistemática y ordenada.

Capítulo 4

METODOLOGÍAS

4.1 Benchmark de un cluster dedicado a la química cuántica computacional

Los *benchmarks*, son estándares de comparación que ayudan a evaluar el rendimiento de una computadora, proveen de un método para hacer un análisis comparativo y así medir por ejemplo, en un *cluster* la escalabilidad de un nodo para aplicaciones de cómputo y/o comunicaciones intensivas, proveyendo de información que permitirá ver cuál será el mejor sistema de cómputo para las aplicaciones. En esta tesis se verá un ejemplo del comportamiento de dos programas de química computacional (*Gaussian 98* y *GAMESS-US*) en un *cluster* tipo *Beowulf*[4]; aunque comúnmente se usan programas de mecánica molecular en clusters con cientos de procesadores. Este estudio se tomó de un artículo[5][22], de donde se tomaron los datos para nuestro análisis y discusión. Los *clusters* han sido una atractiva opción para obtener cómputo de alto rendimiento a bajo costo y tener así una buena relación de precio/rendimiento. El objetivo de los autores fue hacer un estudio con las diferentes arquitecturas de procesadores, redes y software para obtener el *cluster* tipo *Beowulf* que mejor relación beneficio/costo les podría dar como *cluster* dedicado a la química cuántica computacional.

Las variables más importantes al diseñar un *cluster*, son el balance entre la velocidad del procesador, la red de conexión y la memoria. En un *cluster*, es mejor tener programas que no tengan demasiada comunicación entre ellos, ya que por lo regular, la red es el cuello de botella; pero donde la memoria que se tiene en cada nodo, puede ayudar mucho. El sistema operativo usado es *Linux* y para que *Gaussian* pueda correr en una máquina con memoria distribuida como el *cluster*, es necesario la instalación de *Linda*, el cual le dará el ambiente de programación paralela, aunque no es muy eficiente. Para las comunicaciones de *GAMESS-US* solamente se requiere de las librerías de *MPI*.

Al momento de diseñar el *cluster*, es necesario tener en cuenta, el costo, los tipos de procesadores, las diferentes configuraciones posibles, la cantidad de memoria necesaria en cada nodo, el tipo de discos que se tendrán en cada nodo y la red de interconexión. El nivel de demanda de cada uno de estas variables por parte de los programas de la química computacional, ayudará al diseño ya que se debe encontrar un balance entre cada uno de ellos. Algunos de los programas de la química computacional están siendo desarrollados de tal manera que tomen la mayor cantidad de ventajas del *hardware* más barato de tal manera que extienden su mercado en el uso de *clusters*. Para los *clusters*, los algoritmos que mejor funcionan son los directos y en los que se duplican los datos entre algunos nodos, de tal manera que la red de interconexión, no se vea afectada por las integrales ni los datos.

Los procesadores que escogieron fueron los *Pentium* debido a su bajo costo y la arquitectura del nodo fue SMP de tal manera que tuvieron sólo dos procesadores por nodo y lo hicieron así debido al costo y por el tráfico que se puede generar en el bus de *I/O*; ya que todos los procesadores que se encuentran

---

[4] El cluster tipo Beowulf es un cluster de componentes *commodity* de cómputo dedicados a un problema paralelo. Los nodos son conectados por medio de una red privada y solo el "nodo maestro" es visible desde el exterior. [11]

[5] Internet Journal of Chemistry, 2000, 3, 4. http://www.dhpc.adelaide.edu.au/reports/073/html/dhpc-073.html

en un nodo la comparten. Por lo tanto, los procesadores tendrán memoria compartida por nodo que se puede utilizar para el paralelismo de grano fino y se verá, muy útil para *Gaussian*; aunque este programa suele tener un buen comportamiento con el cómputo paralelo sólo para problemas químicos grandes; ya que para problemas pequeños, suele ser mas lento utilizando los 2 procesadores del nodo que solamente uno, debido al tráfico que se genera en el bus de *I/O*. Entre más grande sea el sistema químico, mayor tiempo de cómputo tendrá y menores comunicaciones, lo que será más conveniente para este tipo de arquitectura y en general para el cómputo paralelo. Por lo que para sistemas químicos pequeños, la gran mayoría de las veces es mejor hacer los cálculos en un solo procesador. Además, *Gaussian 98* tiene muy pocos cálculos paralelizados o sólo una parte de ellos están paralelizados lo que provoca que este programa no sea tan eficiente, de tal manera que sólo escala bien entre 8 y 16 procesadores, dependiendo del cálculo.

En cuanto a la red de interconexión, es importante tomar en cuenta la topología para conocer qué tipo de tecnología sería la necesaria y la escalabilidad que se podría tener. Es importante utilizar al menos una *Gigabit Ethernet* para conectar los nodos para prevenir que la red sea un cuello de botella en la infraestructura y aunque los programas *Gaussian* y *GAMESS-US* no utilizan todo el ancho de banda de una *Gigabit*, si es importante la latencia que tienen dichas redes de interconexión y en caso de que ésta sea un cuello de botella, se podrá utilizar una red más cara pero con menor latencia.

La opción de la arquitectura del *cluster* es quizá el aspecto más difícil en el diseño y requerirá de consideraciones muy cuidadosas. Para los temas como la configuración de discos, sistema de archivos, donde comúnmente se usan varios al mismo tiempo y se busca que sean visibles en todos los nodos para que el usuario no se preocupe en migrar ejecutables, datos, etc. entre los nodos servidor y los nodos de cómputo; sistema operativo y herramientas de instalación como *Oscar* donde por lo regular sólo se instalan las partes necesarias para lograr así un óptimo rendimiento del *cluster* dedicado; compiladores, donde es recomendable hacer un gasto en comprar los PGI ya que tienen el mejor rendimiento para optimizar las aplicaciones y una mayor robustez y donde de hecho, sería el único gasto para el sistema (a menos que se necesiten bibliotecas numéricas comerciales o en el caso de *Gaussian*, la compra de *Linda*); la administración del *software* para lograr una administración, configuración, depuración y monitoreo del *cluster* de manera aceptable, se recomienda ver la discusión de estos temas y cómo afectan el rendimiento del *cluster* que existe en la bibliografía [9, 10] que se da en el capítulo 2.

En el caso del artículo, construyeron un *cluster* tipo Beowulf con procesadores Pentium II a 400 MHz con 256 MBytes de memoria RAM y 4GBytes de disco duro. Un *cluster* funciona eficientemente sólo si los trabajos que corren en él no compiten por recursos, es decir, pueden correr varios procesos pero cada uno tendrá sus recursos exclusivos y reservados. Por lo tanto, el uso de *software* que administre los trabajos es muy importante ya que sólo así se podrá obtener un buen provecho del *cluster* sin que los usuarios se afecten unos a otros al momento de ingresar sus trabajos; obviamente, si se tiene una buena política de administración y un buen administrador; además al momento de hacer los experimentos y de obtener las métricas, se debe estar seguros de obtener las mediciones que realmente se necesitan y que éstas sean reproducibles, tal y como se mencionó en el capítulo 2. Sin embargo, además de obtener métricas y mejorar el rendimiento del *cluster*, el mantenimiento, documentación, puesta a punto para producción y buen uso de éste es muy importante para su buen funcionamiento, para obtener los resultados esperados y que el *cluster* sea una herramienta que dure un lapso de tiempo aceptable y para

ello es recomendable utilizar las herramientas explicadas en el capítulo 2 y 3 y en la bibliografía [5, 6, 7, 8, 9, 10].

En cuanto a los *benchmarks* en la química computacional, es un poco difícil escoger un conjunto coherente de programas que puedan servir como *benchmarks* genéricos debido a la gran diversidad de programas que existen y a los diferentes métodos, algoritmos y problemas que se pueden simular y que tienen una gran cantidad de diferentes variables. Los *benchmark* que se escogieron en el artículo fueron para dos programas de química computacional muy usados y que ellos requerían para sus investigaciones, *GAMESS-US* y *Gaussian 98*, donde se estudiará la eficiencia y escalabilidad de las versiones paralelas de estos programas, así como su comportamiento cuando se tienen muchos programas seriales ejecutándose al mismo tiempo y de esta manera, asignar mejor las colas y darle a sus usuarios información para que ellos puedan saber hasta dónde y cómo se puede utilizar eficientemente el *cluster*.

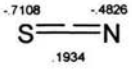
El programa *GAMESS-US* fue diseñado para correr eficientemente en una máquina con memoria distribuida usando *MPI* mientras que *Gaussian* ha sido enfocado principalmente hacia las computadoras con memoria compartida y sólo puede correr en máquinas con memoria distribuida utilizando el ambiente de programación paralela que ofrece *Linda*, lo que provoca que exista poca eficiencia en más de 8-16 nodos. Los problemas de química computacional que escogieron como *benchmark* fueron tomados de los *benchmarks ab initio* de *EMSL* usando estructuras moleculares, nivel de teoría y conjuntos de bases estándar. Los sistemas que escogieron fueron hacer un cálculo *single point* con un nivel de teoría RHF usando la base tipo Pople de 6-311+G**(6d) para el etileno, y 6-31G**(6d) para el eter 18-crown-6 ($C_{12}H_{24}O_6$) y para el benceno y la morfina fueron hechos los cálculos de *single point*, cálculo del gradiente y del hesiano usando un nivel de teoría de RHF y MP2 usando la base 6-31G(d,p) (aunque utilizaron bases mas grandes como 6-311++G(3d,2p) pero sus resultados tuvieron las mismas tendencias). Para la mayoría de los casos, utilizaron los métodos directos y convencionales en el manejo de las integrales y la simetría fue de C1.

Desgraciadamente en el artículo, no mencionan la metodología utilizada para hacer los *benchmark*, sólo mencionan la metodología utilizada para escoger el *cluster* y que sus datos obtenidos son reproducibles y aseguran que sus mediciones son correctas. Sin embargo, los datos que se muestran, son congruentes con diversos documentos de Gaussian, GAMESS y *clusters*. Los resultados y la discusión de éstos serán presentados en el siguiente capítulo en la sección 5.1

4.2 Síntesis de tiocianatos

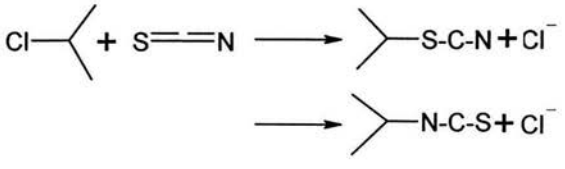
El sistema químico a calcular fue la síntesis de tiocianatos orgánicos[6] [23] Este sistema tiene un interesante comportamiento químico, ya que el anión tiocionato es ambivalente y a tener resonancia, su estructura es híbrida y su distribución de carga es como la que se muestra a continuación:

[6] The Chemistry of Cyanates and their thio derivatives. Saul Patai. J. Wiley 1977.



El anión tiocianato puede atacar nucleofílicamente compuestos orgánicos produciendo tiocianatos si el anión ataca por el sulfuro o isotiocianatos si el anión ataca por el nitrógeno. El producto más común es el tiocianato ya que es el producto más estable termodinámicamente pero en muchas reacciones el producto cinéticamente más estable es el isotiocianato. Estos productos tienen una importancia fisiológica ya que son tóxicos produciendo en la mayoría de los casos dermatitis, pueden ser utilizados para la obtención de productos con actividad fisiológica como N- o S-heterociclos, etc. Existe mucha información en la literatura sobre estos compuestos; sin embargo, muchas veces es contradictoria y confusa y todavía existe un gran debate sobre su mecanismo de reacción ya que es muy dependiente del tipo de carbono al que se va a unir (primario, secundario o terciario), solvente, catalizador utilizado, concentración, temperatura, grupo saliente y del contraión y por lo tanto obtener la mayor información del mecanismo de reacción sería muy útil para poder diseñar síntesis que tengan un mayor rendimiento y estereoselectividad.

Existe cierta tendencia en las reacciones que es congruente con el principio de ácidos y bases duros y blandos (HSAB→*Hard and Soft Acids and Bases*) tanto para un mecanismo SN_1 como SN_2. Con este principio se puede decir que el anión tiocianato es una base más suave por el lado del azufre y que preferirá ácidos más suaves como son los carbonos primarios y que el anión tiocionato es una base más dura por el lado del nitrógeno y que preferirá ácidos más duros como son los carbonos secundarios y terciarios. Debido a todo ello, este sistema es muy interesante para poder obtener resultados cualitativos y observar el comportamiento de las simulaciones y las conclusiones a las que se pueden llegar. La reacción que se simulará será la del cloruro de isopropilo con el tiocianato, el cual ataca tanto por el lado del azufre como por el lado del nitrógeno y donde según la teoría de HSAB el producto más estable será el isotiocianato.



El programa que se utilizó fue *HyperChem Professional Release 7.5 for Windows* debido a su fácil manejo, a que no se necesita hacer cálculos demasiado sofisticados donde se tiene que tener demasiado control sobre el tipo de cálculos (como escoger el espacio activo en un cálculo MCSCF o utilizar alguna base especial para un cálculo de hiperpolarizabilidad, etc) y que éste trabajo tiene un objetivo de resultados cualitativos, mas que cuantitativos y de investigación.

Primero se optimizaron las geometrías de los reactivos y de los productos por separado. Estas optimizaciones se hicieron con un nivel de teoría RHF y con una base mínima. Un punto muy importante aquí es el control de las iteraciones, donde se tiene que seleccionar el límite de convergencia para la energía (.0000001) y el límite de iteración (32767, el mayor que permite el programa). Se debe de considerar el épsilon de la máquina para no tener errores de redondeo y verificar que el cálculo logró la convergencia ya que de lo contrario, los resultados no darán la precisión que se requiere. De hecho,

para la optimización de la geometría de los productos, fue necesario hacer tres cálculos seguidos ya que no se había alcanzado la convergencia.

En el cálculo de las integrales, los cálculos fueron convencionales al no tener un sistema muy grande y las integrales bielectrónicas fueron calculadas hasta un valor mínimo de $1X10^{-10}$ con un tamaño de buffer de 32000. Los orbitales moleculares iniciales se obtuvieron del Hamiltoniano y no de un cálculo semiempírico. Se utilizaron 5 orbitales d, es decir, representación polar. El algoritmo utilizado en la optimización de la geometría fue de gradiente conjugado de Polak-Ribiere. Una vez que fueron optimizadas todas las geometrías, se hizo un análisis de frecuencias para asegurar que era un mínimo, es decir, que todos los modos normales de vibraciones eran positivos (figura 4.1).

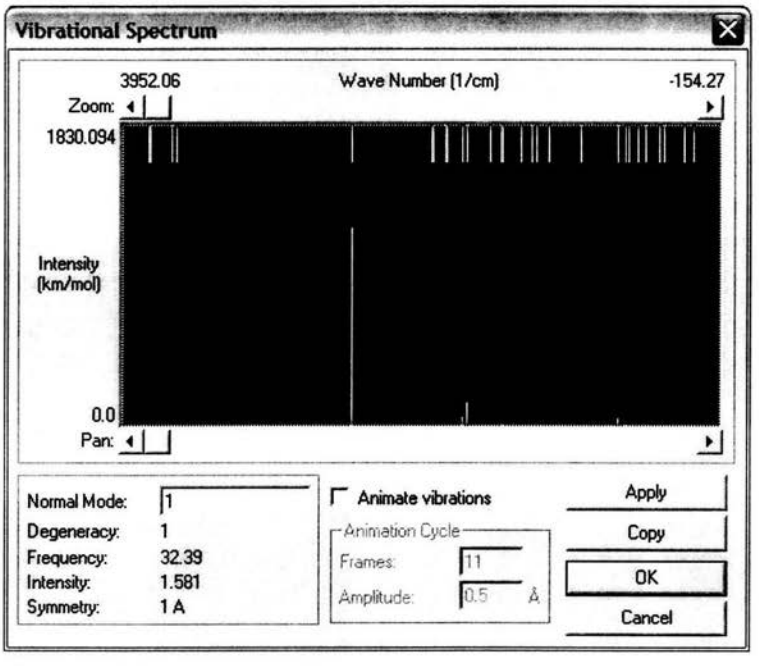


Figura 4.1 Análisis vibracional. Todas las frecuencias deben ser positivas para encontrarnos en un mínimo.

Al obtener todos los confórmeros, se construyó la supermolécula tanto de la reacción 1 (ataque nucleofílico por el lado del azufre, figura 4.2) como de la reacción 2 (ataque nucleofílico por el lado del nitrógeno, figura 4.3). En ambas reacciones, se puso el tiocianato a una distancia de 10 A° del isopropilo y con un ángulo de 180°. Poco a poco se acercaron las moléculas, utilizando un mecanismo de reacción SN_2. En cada paso se calculó la energía y se calculó la isosuperficie de densidad electrónica a un valor de 0.01, mapeando los valores del potencial electrostático a dicha isosuperficie; donde los valores en rojo son valores del potencial negativos y los valores en azul son valores del potencial positivos (figura 4.4).

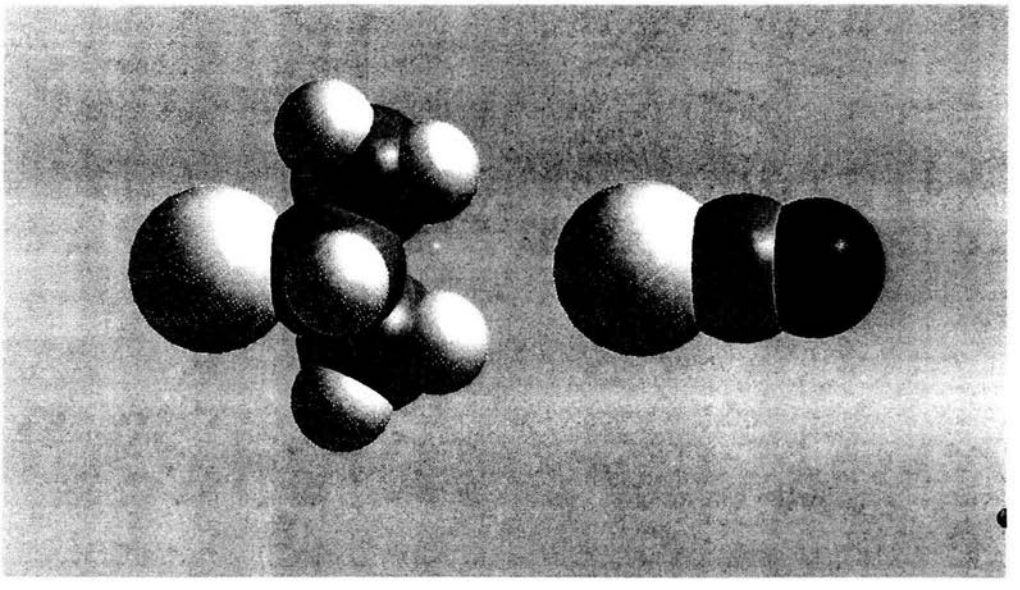


Figura 4.2 Supermolécula de la reacción 1

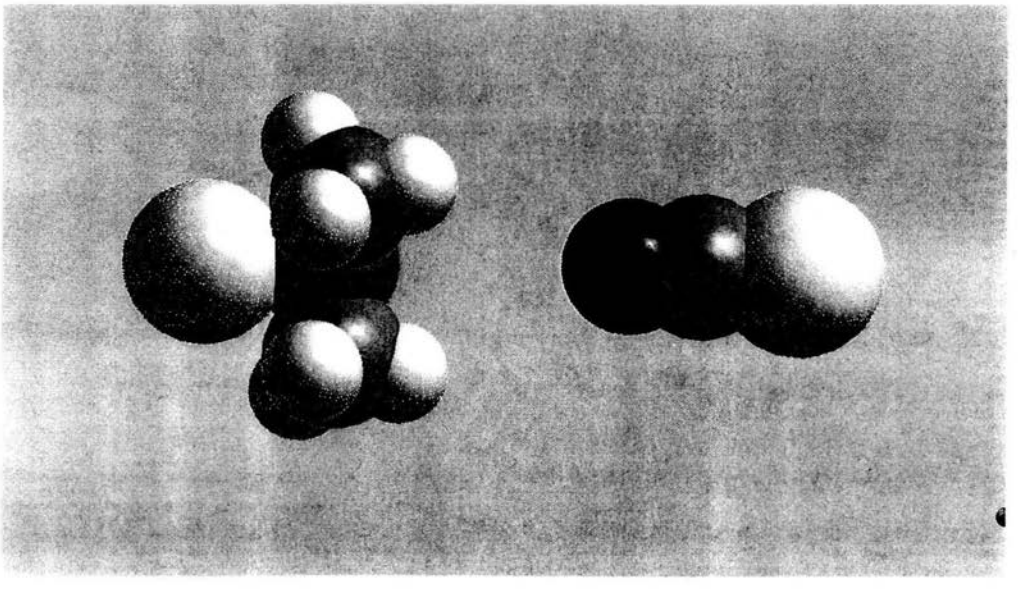


Figura 4.3 Supermolécula de la reacción 2

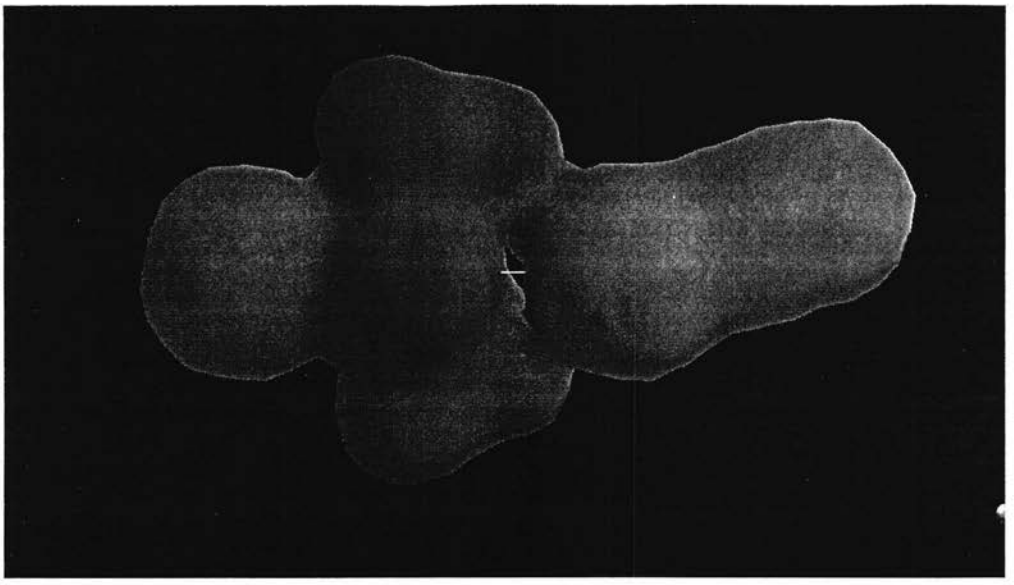


Figura 4.4 Isosuperficie de densidad electrónica mapeado con valores del potencial electrostático

Una vez que se han realizado los cálculos, se modificaron las diferentes distancias y ángulos que sean necesarios, dependiendo del análisis de los resultados que se obtuvieron y con base en ellos, proseguir con los siguientes cálculos necesarios para obtener así el mecanismo de la reacción de una manera cualitativa y siguiendo la experiencia química y un mecanismo de reacción SN_2.

Los resultados y su discusión serán presentados en el siguiente capítulo en la sección 5.2.

Capítulo 5

RESULTADOS Y DISCUSION

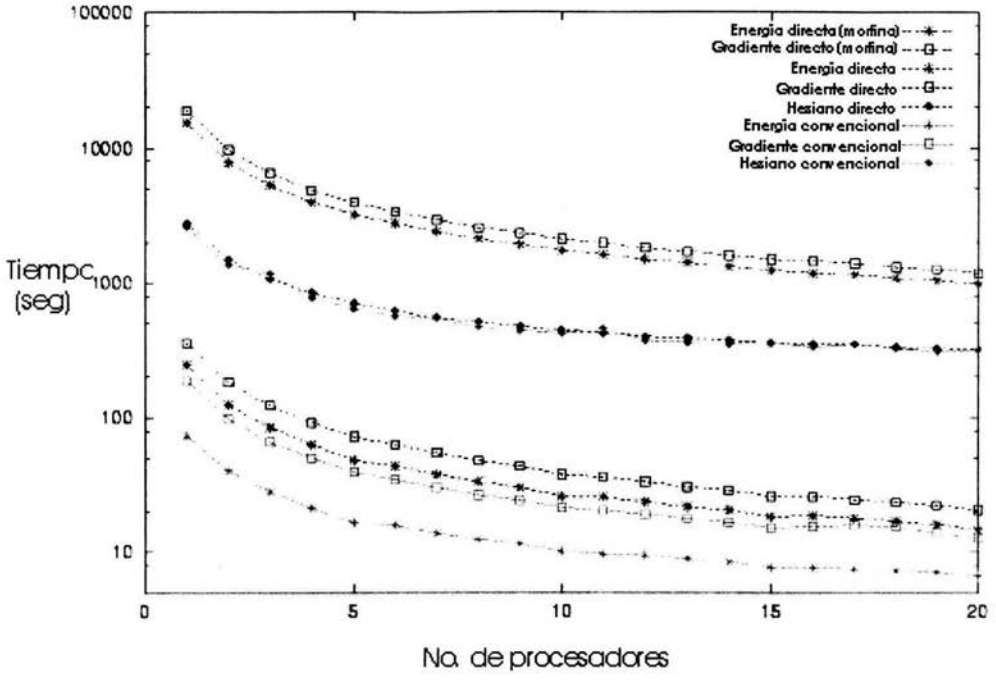
5.1 Benchmark de un cluster dedicado a la química cuántica computacional

A continuación se presentarán los resultados que se obtuvieron del artículo mencionado en el capítulo anterior en la sección 4.1 y su discusión. En la siguiente tabla, se observa los tiempos que tomaron los cálculos usando las versiones paralelas de *Gaussian 98* y *GAMES-US* en un nodo con dos procesadores con memoria compartida.

	Gaussian 98			GAMESS-US		
	1 Procesador	2 Procesadores	Speedup	1 Procesador	2 Procesadores	Speedup
Etileno RHF Conventional	13.3	13.7	0.97	4.6	3.0	1.53
Etileno RHF Direct	28.4	20.9	1.36	19.9	10.7	1.86
Etileno MP2 Conventional	19.3	19.6	0.99	9.4	5.4	1.74
Etileno MP2 Direct	33.8	27.5	1.23	24.9	13.2	1.89
Benceno RHF Conventional	141.8	146.2	0.97	73.5	40.8	1.80
Benceno RHF Direct	135.8	78.6	1.73	4777.7	2433.6	1.96
Benceno MP2 Conventional	251.1	250.1	1.00	214.1	112.1	1.91
Benceno MP2 Direct	340.0	238.2	1.43	389.5	198.2	1.97
Crown Eter RHF Direct	3430.2 (57.2 min)	1914.3 (31.9 min)	1.79	7306.5 (121.8 min)	3811.7 (63.5 min)	1.92
Morfina RHF Direct	5261.8 (87.7 min)	2871.5 (47.9 min)	1.83	15284.5 (254.7 min)	7790.8 (129.9 min)	1.96

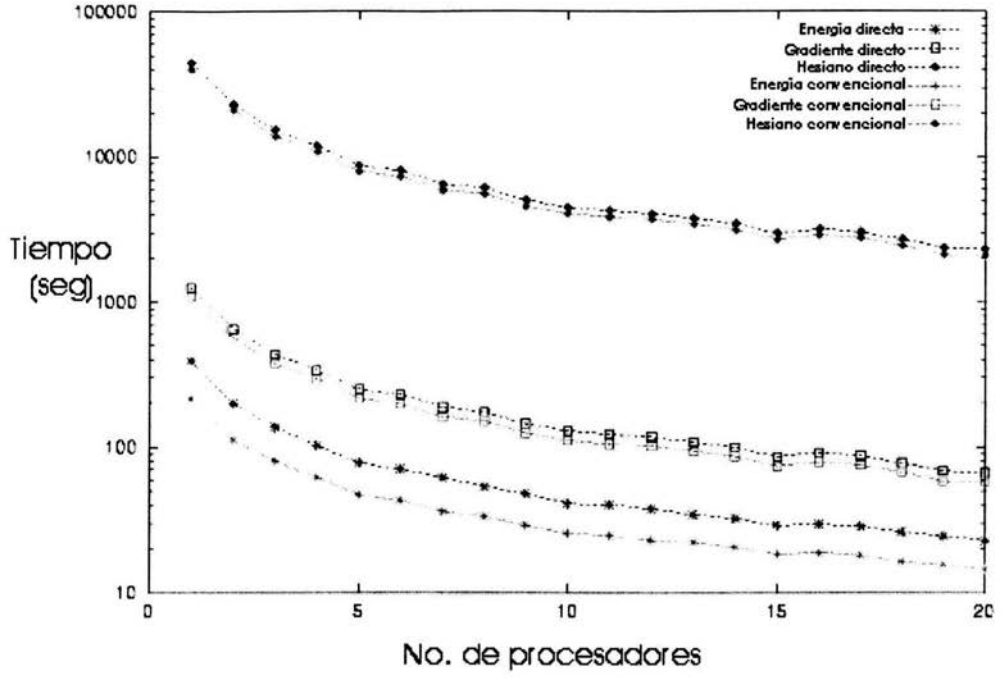
GAMESS-US tiene un *speedup* y eficiencia mayor que *Gaussian 98*; sin embargo, *Gaussian* mejora mucho en sus tiempos para moléculas grandes tanto en los cálculos seriales como los que sí están paralelizados (no todos los cálculos en *Gaussian 98* están paralelizados como se puede ver con los cálculos MP2, sólo están paralelizadas cálculos de álgebra lineal y RHF). El cómputo directo da mejores *speedups* que el cómputo convencional para ambos paquetes. El *speedup* mejora conforme aumenta el tamaño del sistema químico y con *GAMESS-US* se muestra prácticamente un *speedup* perfecto para moléculas grandes. La escalabilidad de *Gaussian* es mucho más dependiente del tamaño del problema químico y muestra buenos *speedups* para sistemas químicos grandes usando cálculos directos. Por lo tanto, usar un nodo dual con procesadores *Pentium*, utilizará mejor la implementación paralela de *Gaussian* (la cual no requiere de *Linda* en el caso de un nodo dual) con una extensión limitada, las cuales pueden ser muy útiles para cómputos que involucran sistemas químicos grandes ya que se obtendrá mucho más tiempo de cómputo que de comunicación, como se prefiere en el cómputo paralelo.

Los siguientes datos se dan con *GAMESS-US*, ya que como se explicó en la metodología, *Gaussian* sólo tiene buena eficiencia para un rango de 8 a 16 nodos dependiendo del cálculo. En la gráfica 5.1 se puede observar el comportamiento que dio un cálculo de energía, gradiente y cálculo del hesiano del benceno con RHF/6-31G(d,p) usando las técnicas del cómputo directo y convencional. También se muestran algunos cálculos de la morfina, la cual tarda aproximadamente 100 veces más. Fue imposible hacer el cálculo de la morfina con la técnica convencional ni del hesiano debido a que carecían del espacio suficiente en disco. En la gráfica se puede ver cómo la técnica convencional tiene un mejor rendimiento que el cálculo directo tanto para el cálculo de la energía como el del gradiente. Inversamente, en el caso del cálculo del hesiano, el cómputo directo compite con el cálculo convencional. Este resultado es importante debido a que los nodos pequeños con pequeñas capacidades de disco pueden ser usados para cálculos del hesiano usando métodos directos, incluso para sistemas químicos grandes.



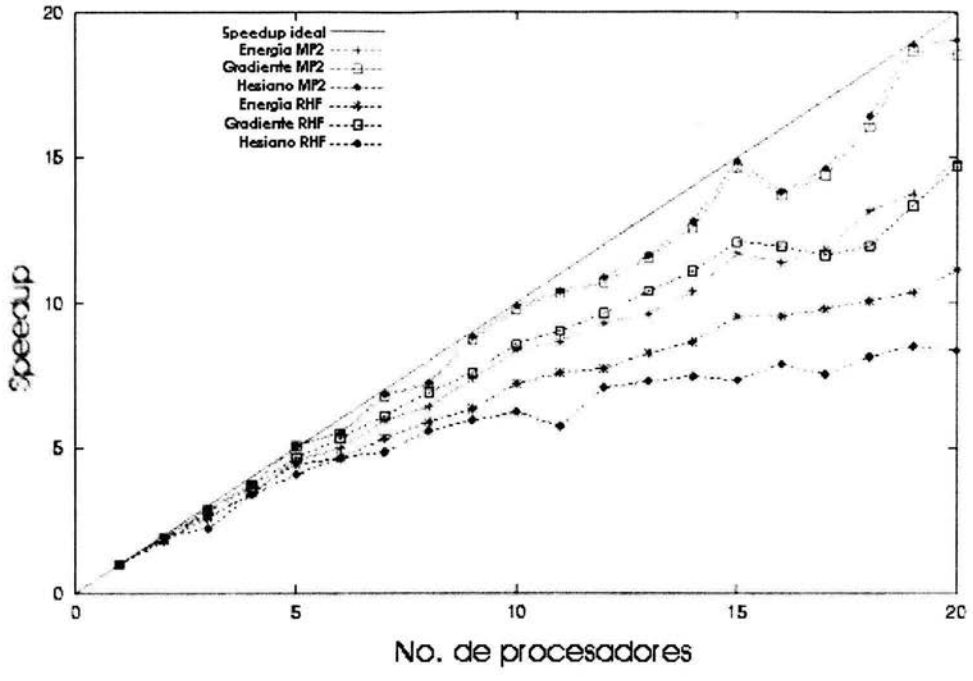
Gráfica 5.1

En la gráfica 5.2, se puede observar el comportamiento de un cálculo de energía, gradiente y hesiano para el benceno con MP2/6-31G(d,p) usando tanto las técnicas de cómputo directas como convencionales. Notar que el cálculo MP2 toma mucho mas tiempo que el cálculo RHF mostrado en la gráfica 5.1. Como se vio en la gráfica anterior, los cálculos convencionales tuvieron un rendimiento mejor que el cálculo directo para el cálculo de energía y de gradiente; sin embargo y de manera muy interesante, la técnica convencional fue un poco más rápida para los cálculos del hesiano que el método directo. Para el caso de la morfina y debido a la poca memoria que se tiene, los cálculos MP2 sólo se pudieron llevar a cabo hasta 16 procesadores; por lo tanto, no incluyeron dichos resultados.



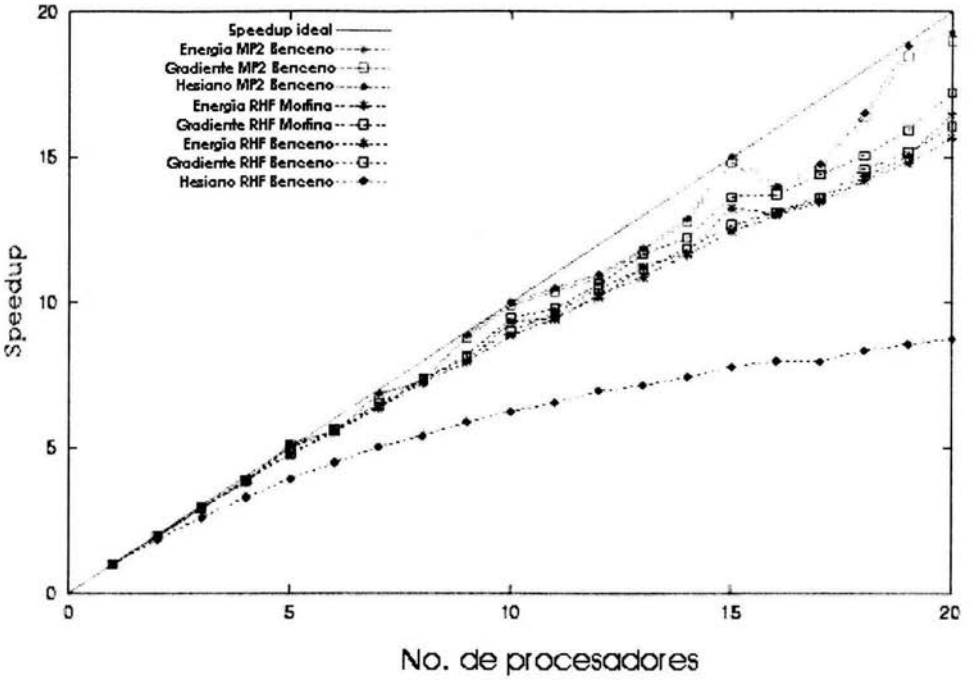
Gráfica 5.2

La gráfica 5.3 muestra el *speedup* de los diferentes cálculos para el benceno usando los métodos convencionales. Como se puede ver, los cálculos RHF no escalan particularmente muy bien y sólo dan eficiencias razonables hasta alrededor de 10-15 procesadores. En contraste, en un cálculo computacionalmente más intensivo como el MP2, se obtienen un *speedup* muy bueno. Como se puede ver, para los cálculos del hesiano con RHF, no tiene caso hacer el cálculo con mas de 20 procesadores ya que comienza a verse que las ganancias por utilizar mas procesadores son mínimas, como señala la ley de Amdahl. Un fenómeno extraño es que aparentemente en los cálculos MP2, conviene hacerlos en un número de procesadores que sean múltiplos de 5, este fenómeno es reproducible y se puede relacionar con cómo los arreglos son distribuidos para el cálculo y el tamaño del buffer para los mensajes.



Gráfica 5.3

La gráfica 5.4 muestra las curvas de *speedup* para los diferentes cálculos del benceno y de la morfina usando los métodos directos. Por lo regular, los *speedup*s son generalmente mejores que los mostrados en la gráfica 5.3, como se esperaba; ya que los cálculos directos requieren de mucho mas cómputo pero reducen significativamente los requerimientos de I/O. Nuevamente el cálculo que no escala bien es el hesiano con RHF teniendo una eficiencia menor al 50% para 20 procesadores. En contraste el cálculo del hesiano con MP2 muestra un *speedup* muy bueno y como en la gráfica 5.3, se observan mejores rendimientos en los cálculos cuando se utilizan un número de procesadores múltiplos de 5.



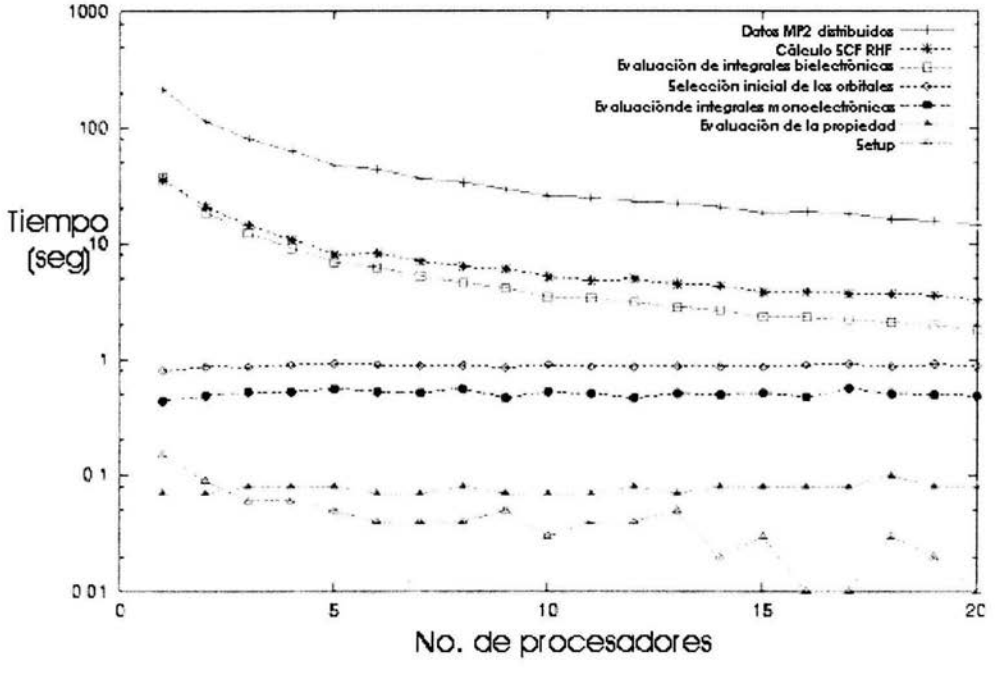
Gráfica 5.4

De las gráficas anteriores se puede observar que el cálculo MP2 usando una base 6-31G(d,p) toma más tiempo de cómputo que el cálculo RHF con la misma base, como se esperaba. Sin embargo, conforme el número de procesadores aumenta, el rendimiento del cálculo MP2 se acerca al rendimiento del cálculo RHF debido a la mejor eficiencia y escalabilidad del cálculo MP2, a la relación cómputo/comunicación y al esfuerzo por paralelizar eficientemente las simulaciones con este nivel de teoría en *GAMESS-US*. Las versiones actuales de *GAMESS*, han paralelizado eficientemente muchos cálculos y en este programa como en *NWChem* ha habido gran desarrollo en este sentido.

Cada single-point usando un nivel de teoría MP2 envuelve los siguientes pasos:

1. organización de los cálculos iniciales
2. selección inicial de los orbitales moleculares
3. evaluación de las integrales monoelectrónicas
4. evaluación de las integrales bielectrónicas
5. rutina de convergencia de RHF
6. distribución de los datos del cálculo MP2
7. evaluación final de la propiedad molecular

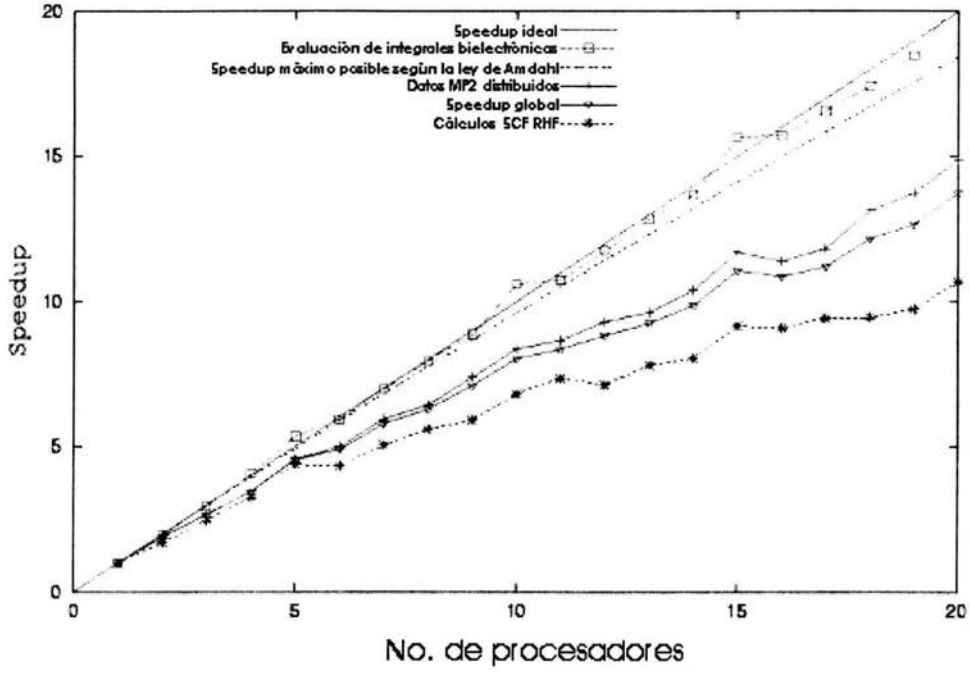
En gráfica 5.5 se puede ver el tiempo que toma cada uno de los pasos en el cálculo MP2/6-31G(d,p) para el benceno; para poder observar así la eficiencia del paralelismo en cada fase de un cálculo *ab initio* completo. De estos pasos, 4 son secuenciales: organización de los cálculos iniciales, selección inicial de los orbitales moleculares, evaluación de las integrales monoelectrónicas y la evaluación final de la propiedad molecular (como claramente se puede observar en la gráfica). Conforme mas procesadores se usen para el problema, el tiempo que se tome para estos pasos secuenciales para el cómputo, permanecerá prácticamente sin cambio; mientras que el tiempo para las secciones paralelizadas continúan decreciendo hasta que el cuello de botella sea la parte secuencial, como la ley de Amdahl lo predice. Como se puede observar en la gráfica 5.5, el cálculo RHF para el benceno, comienza a mostrar deficiencias en la paralelización alrededor de los 20 procesadores; sin embargo, el cálculo MP2 tiene aún buena escalabilidad con este número de procesadores, debido a que el paso de distribución de datos del cálculo MP2, toma más tiempo que la parte secuencial; pero con la ley de Amdahl, podremos predecir que alrededor de 50 o 60 procesadores, el cuello de botella será el cálculo RHF y por lo tanto el cálculo MP2 ya no será escalable a más procesadores para este sistema.



Gráfica 5.5

Por último, en la gráfica 5.6, se puede observar el *speedup* de un cálculo *single-point* para el benceno con los pasos de la evaluación de las integrales bielectrónicas, el cálculo RHF y la distribución de datos del cálculo MP2. El paso de la evaluación de las integrales bielectrónicas paralelizan muy bien, prácticamente dando un *speedup* casi perfecto, lo que muestra porque el cálculo MP2 escala tan bien, al

menos con este sistema químico y hasta con 20 procesadores, ya que la parte secuencial todavía no limita la parte paralela. Nuevamente se puede ver cómo mejora el rendimiento del cálculo cuando se utiliza un número de procesadores múltiplo de cinco lo que nos generaría incluso un *speedup* superlineal y que este comportamiento se debe al cálculo de las integrales bielectrónicas y cómo se manejan en memoria. Esto muestra la importancia de incrementar la memoria caché y RAM para almacenar los datos distribuidos.



Gráfica 5.6

Con base en estos resultados, el *cluster* que ellos decidieron fue uno que tiene nodos duales con procesadores *Pentium*, con una red *Fast Ethernet* (aunque convendría una *Gigabyt* por las circunstancias ya mencionadas), y donde obtendrían un pico en el rendimiento de 100 Gflops[7] con un costo muy menor a las computadoras comerciales y que puede ser muy útil para grupos de investigadores que no puedan invertir en una computadora comercial muy cara; pero si tiempo y en *humanware* para mantener el *cluster* en producción. El *cluster* fue desarrollado para tener varios programas corriendo al mismo tiempo; pero donde cada programa tendrá reservada sus recursos de tal manera que no exista competencia entre ellos. Es muy conveniente que se tenga una parte del *cluster* que sólo pueda correr

[7] Flop→Operaciones de punto flotante por segundo (por sus siglas en inglés)

muchos programas programas seriales, los cuales utilizarían el poder de cómputo sin afectar en gran medida el tráfico de la red y otra parte que tenga los programas que correrán de manera paralela pero que serán usados de manera eficiente y utilizando sólo los recursos necesarios dependiendo del cálculo que se tiene, es decir, para un cálculo RHF con la base vista y para el benceno, no conviene usar mas de 20 procesadores usando el programa *GAMESS-US* y en el caso de *Gaussian*, para sistemas químicos, conviene usar sólo un nodo.

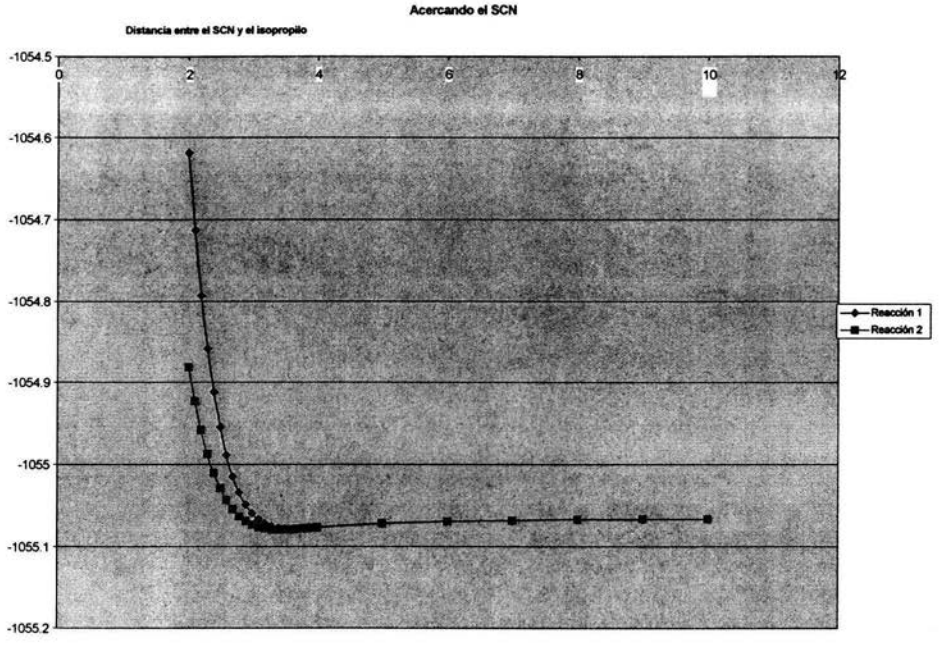
La versión paralela de *GAMESS-US* se desempeña bien en el *cluster* para la mayoría de los cálculos, particularmente para los métodos directos y MP2 para sistemas químicos grandes. *Gaussian 98* no fue diseñado para máquinas con memoria distribuida como los *cluster*; sin embargo, tiene *speedups* razonables para cálculos directos, sistemas químicos grandes (donde la relación cómputo/comunicación es grande) y para nodos con varios procesadores (máquinas SMP) que tienen memoria compartida. Por lo tanto, la velocidad de la red (comunicaciones), la memoria y los procesadores son las variables más importantes para estos tipos de cómputo y depende del comportamiento de los programas, saber a qué variable necesitaremos invertirle más para obtener un óptimo comportamiento. También se puede observar como es muy necesario utilizar el cómputo paralelo pero sólo para sistemas químicos grandes, ya que de otra manera, el cálculo se puede hacer fácilmente en un solo nodo, es decir, el cómputo paralelo funciona bien para el cómputo intensivo.

Sin embargo, es difícil comparar estas tendencias con otras que se pueden obtener de otros *bechmark* ya que dependen del sistema químico, el nivel de teoría, la base, el algoritmo usado y su implementación y el ambiente de ejecución; además las computadoras van mejorando sus rendimientos día con día. Un detalle con los cálculos que hicieron fue que RHF no escaló tan bien a pesar de que es el nivel de teoría que mayor esfuerzo de paralelización ha tenido. Esto se debe a que los sistemas químicos utilizados no fueron lo suficientemente grandes para obtener una relación de cómputo/comunicación aceptable y que nos pueda mostrar el comportamiento de una aplicación de cómputo intensiva. En esta tesis, se recomienda que para obtener buenos parámetros en las computadoras actuales, es necesario correr sistemas químicos que duren alrededor de 50,000 segundos con unas 300 funciones de base y que sería muy interesante, útil e importante obtener el *speedup* de nuestra computadora variando el número de funciones de base (tamaño del sistema) y el número de procesadores para un mismo sistema (obteniendo la isoeficiencia), de tal manera que un investigador podría fácilmente observar hasta qué número de procesadores su problema con determinados números de funciones de base, escala bien.

Por ello, es muy importante, tener siempre *benchmarks* que sean parámetros para el tipo de problemas que se tienen y la mejor métrica será el tiempo total de ejecución, mediante algunos modelos y experimentos se podrá ver la escalabilidad y rendimiento en diferentes máquinas y también ver la escalabilidad de la máquina; además de las herramientas. También es recomendable ver el avance que se tiene en velocidad de procesadores, redes, etc, para que la computadora tenga el mayor ciclo de vida posible.

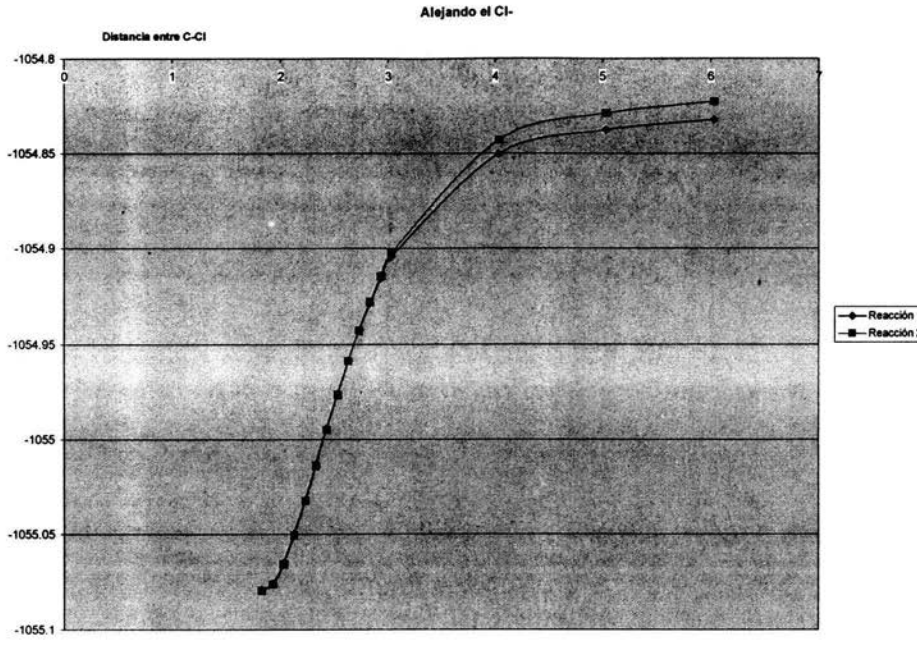
5.2 Síntesis de tiocianatos

A continuación se mostrarán los resultados y su discusión en la síntesis de tiocianatos. Se realizaron cálculos de energía (*single point*) de la reacción 1 (ataque nucleofílico por el lado del azufre) y de la reacción 2 (ataque nucleofílico por el lado del nitrógeno) simplemente acercando el anión al isopropilo de 10 A° hasta los 4 A° variando 1 A° entre cada cálculo. De los 4 A° hasta los 2 A° sólo se varió 0.1 A° entre cada cálculo ya que se sabe que a estas distancias, la variación de energía será muy importante y en ese rango se encontrará el primer mínimo. La gráfica que se obtiene es la 5.7:



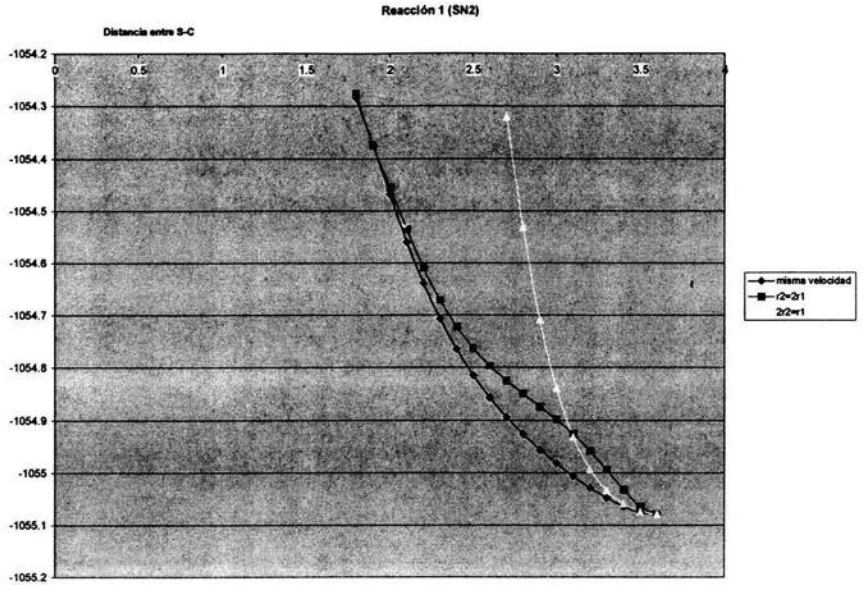
Gráfica 5.7

donde el mínimo encontrado para la reacción 1 fue a 3.6 A° (-661727.7201949 kcal/mol) y para la reacción 2 fue de 3.4 A° (-661727.0588209 kcal/mol). Estos mínimos tienen una diferencia muy pequeña, por lo que no podemos concluir nada con estos datos. Una vez encontrado este primer mínimo, se sabe que la interacción entre ambas moléculas es fuerte, provocando que todos los ángulos de enlace y distancias entre los diferentes átomos sufran modificaciones para disminuir la energía. A continuación se probó de que manera los átomos tenderán a moverse y para ello, primero se movió el contraion de tal manera que se alejaba del isopropilo. La gráfica que se obtiene es la 5.8, en donde claramente se puede ver como aumenta la energía, lo que hace concluir que este mecanismo no es el adecuado para la reacción. Esto no significa que el mecanismo SN_1 no se encuentre en la reacción ya que en este mecanismo, primero se forma un carbocatión, principalmente con la interacción del solvente, y después el anión atacará al carbocatión formado, el cual es un mecanismo diferente al que se está estudiando.

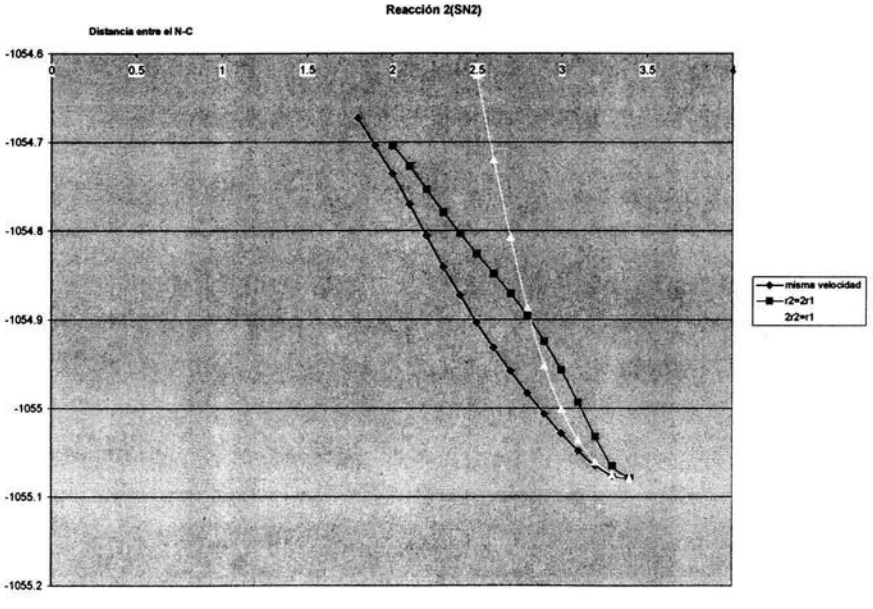


Gráfica 5.8

Con la gráfica 5.8, se puede concluir que mientras el contraión se aleja del isopropilo, el anión debe de acercarse a éste para formar el producto de manera concertada. Pero no se puede concluir de que manera sucede; por lo tanto, las siguientes pruebas, fueron observar cómo se comporta el sistema energéticamente si se aleja mas, menos o exactamente igual al contraión que de lo que acercamos el anión al isopropilo, obteniendo las gráficas 5.9 y 5.10; donde claramente se puede observar que en las dos reacciones, la mejor tendencia para la reacción es que exactamente igual mientras se aleja el contraión, se acercará el anión al isopropilo.



Gráfica 5.9



Gráfica 5.10

Con estos resultados, se puede concluir que el mejor mecanismo que se ha encontrado hasta ahora, es que exactamente mientras el anión se acerca al isopropilo, el contranión se alejará de éste y que cada uno de las distancias y ángulos de enlace, variarán concertadamente hasta formar los productos. Con estos datos se planeó los posibles mecanismos para las dos reacciones de tal manera que de los valores de las distancias y ángulos de enlace de los reactivos, se llegara a los valores de distancia y ángulos de enlace de los productos, dividiendo los valores exactamente entre el número de pasos que toma acercar el anión hasta llegar a la distancia del producto final. Los valores obtenidos se encuentran en las siguientes tablas:

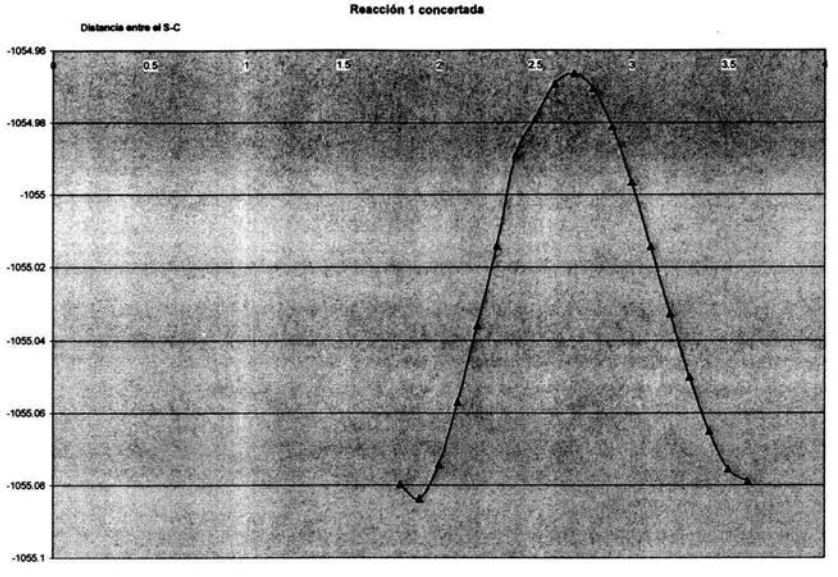
Para la reacción 1

C-S(A°)	C-Cl(A°)	S-C(A°)	C-N(A°)	CSC(°)	Metilo 1(°)	Metilo 2(°)	H(°)
3.5	1.9	1.65723	1.16615	175.5	72.9211	72.6193	76.6868
3.4	2	1.66073	1.16565	171	75.22	74.6193	78.5468
3.3	2.1	1.66423	1.16515	166.5	77.52	76.6193	80.4068
3.2	2.2	1.66773	1.16465	162	79.82	78.6193	82.2668
3.1	2.3	1.67123	1.16415	157.5	82.12	80.6193	84.1268
3.0	2.4	1.67473	1.16365	153	84.42	82.6193	85.9868
2.9	2.5	1.67823	1.16315	148.5	86.72	84.6193	87.8468
2.8	2.6	1.68173	1.16265	144	89.02	86.6193	89.7068
2.7	2.7	1.68523	1.16215	139.5	91.32	88.6193	91.5668
2.6	2.8	1.68873	1.16165	135	93.62	90.6193	93.4268
2.5	2.9	1.69223	1.16115	130.5	95.92	92.6193	95.2868
2.4	3.0	1.69573	1.16065	126	98.22	94.6193	97.1468
2.3	3.1	1.69923	1.16015	121.5	100.52	96.6193	99.0068
2.2	3.2	1.70273	1.15965	117	102.82	98.6193	100.8668
2.1	3.3	1.70623	1.15915	112.5	105.12	100.6193	102.7268
2.0	3.4	1.70973	1.15865	108	107.42	102.6193	104.5868
1.9	3.5	1.71323	1.15815	103.5	109.72	104.6193	106.4468
1.8	3.6	1.71623	1.15765	99	112.02	106.6193	108.3068

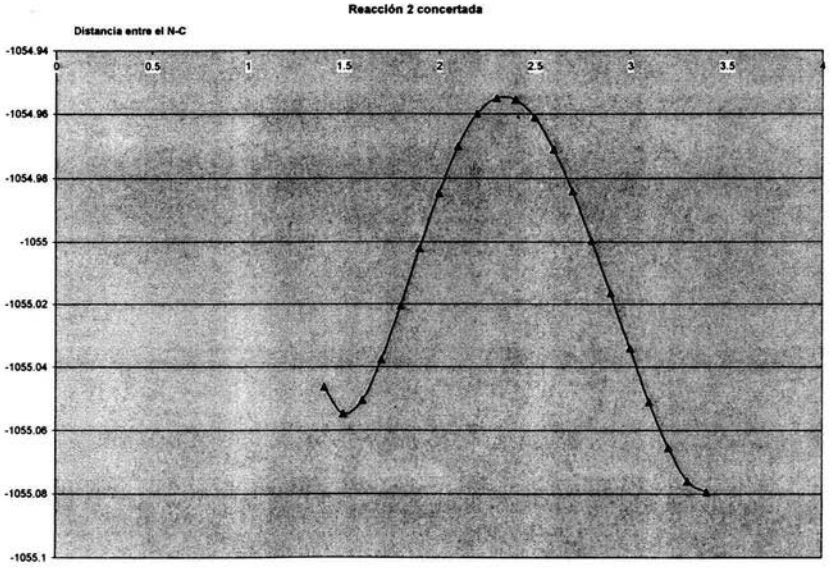
Para la reacción 2

C-S(A°)	C-Cl(A°)	N-C(A°)	C-S(A°)	CNC(°)	Metilo 1(°)	Metilo 2(°)	H(°)	NCS(°)
3.3	1.9	1.16885	1.64843	177.6	72.72	72	76.556	179.61
3.2	2.0	1.17105	1.64313	175.2	74.82	74	78.286	179.22
3.1	2.1	1.17325	1.63783	172.8	76.92	76	80.016	178.83
3.0	2.2	1.17545	1.63253	170.4	79.02	78	81.746	178.44
2.9	2.3	1.17765	1.62723	168	81.12	80	83.476	178.05
2.8	2.4	1.17985	1.62193	165.6	83.22	82	85.206	177.66
2.7	2.5	1.18205	1.61663	163.2	85.32	84	86.936	177.27
2.6	2.6	1.18425	1.61133	160.8	87.42	86	88.666	176.88
2.5	2.7	1.18645	1.60603	158.4	89.52	88	90.366	176.44
2.4	2.8	1.18865	1.60073	156	91.62	90	92.126	176.10
2.3	2.9	1.19085	1.59543	153.6	93.72	92	93.856	175.71
2.2	3.0	1.19305	1.59013	151.2	95.82	94	95.586	175.32
2.1	3.1	1.19525	1.58483	148.8	97.92	96	97.316	174.93
2.0	3.2	1.19745	1.57953	146.4	100.02	98	99.046	174.54
1.9	3.3	1.19965	1.57423	144	102.12	100	100.776	174.15
1.8	3.4	1.20185	1.56893	141.6	104.22	102	102.506	173.76
1.7	3.5	1.20405	1.56363	139.2	106.32	104	104.236	173.37
1.6	3.6	1.20625	1.55833	136.8	108.72	106	105.966	172.98
1.5	3.7	1.20845	1.55303	134.4	110.52	108	107.696	172.59

Las gráficas obtenidas fueron la 5.11 y 5.12, donde claramente se ha encontrado un máximo, el cual será el estado de transición encontrado y los mínimos dados por los reactivos y los productos, es decir, se ha encontrado una coordenada de reacción y un mecanismo válido para la reacción con cálculos que nos dan resultados cualitativos. Es interesante observar como el producto 1 (el tiocianato) tiene una energía menor al primer mínimo encontrado (-662075.0828549 kcal/mol) dando una diferencia de 347.36266 Kcal./mol; mientras que el producto 2 (el isotiocianato) tiene una energía mayor al primer mínimo encontrado (-662056.9404314 kcal/mol) dando una diferencia de 329.881611 kcal/mol; aunque el primer mínimo de la reacción 2 es menor al primer mínimo de la reacción 1 y ambos mínimos son menores a las sumas de las energías de los reactivos. También se puede observar como según estos resultados, el producto 1 sería más estable que el producto 2 al tener el menor mínimo encontrado tal y como se esperaba con una diferencia de 18.142423 kcal/mol que es el 2.74% de la energía total. Estos valores, nos pueden asegurar que obtendremos termodinámicamente el producto 1 en el vacío y a una temperatura de 0°K. Sin embargo, para cálculos más sofisticados que puedan dar mucha mayor información, es necesario tomar en cuenta el disolvente, efectos dinámicos, etc. Obviamente, los resultados obtenidos, sólo pueden dar una visión cualitativa de la reacción, la cual ayuda para preparar cálculos con bases mucho más completas que una base mínima, ya que se está tratando con un sistema químico que contiene un anión bivalente y que sufre de resonancia; por lo tanto es muy importante recuperar la energía de correlación con métodos de correlación y/o utilizando bases grandes o bases de "correlation consistent" como una cc-pV6Z. Estos cálculos tardarán muchísimo más tiempo que los cálculos realizados, pero se pueden hacer cálculos primero para puntos que se consideren muy importantes y que estén cercanos a los mínimos y al punto máximo que se ha encontrado y observar si siguen o varían la tendencia encontrada.



Gráfica 5.11



Gráfica 5.12

Dos fenómenos donde claramente se puede ver que la base mínima no es suficiente para el sistema, es que en las densidades de carga total, a pesar de que el anión por parte del azufre es blando de tal manera que es muy polarizable, esto no se observa al momento de que ambas moléculas se acercan. El otro fenómeno es el error BSSE ya que la energía obtenida del producto mas el cloruro fue de -1055.096578 u. a. y -1055.06139 u. a. respectivamente; mientras que el mínimo encontrado para los productos fue de -1055.098899 u. a. y -1055.063787 u. a. respectivamente.

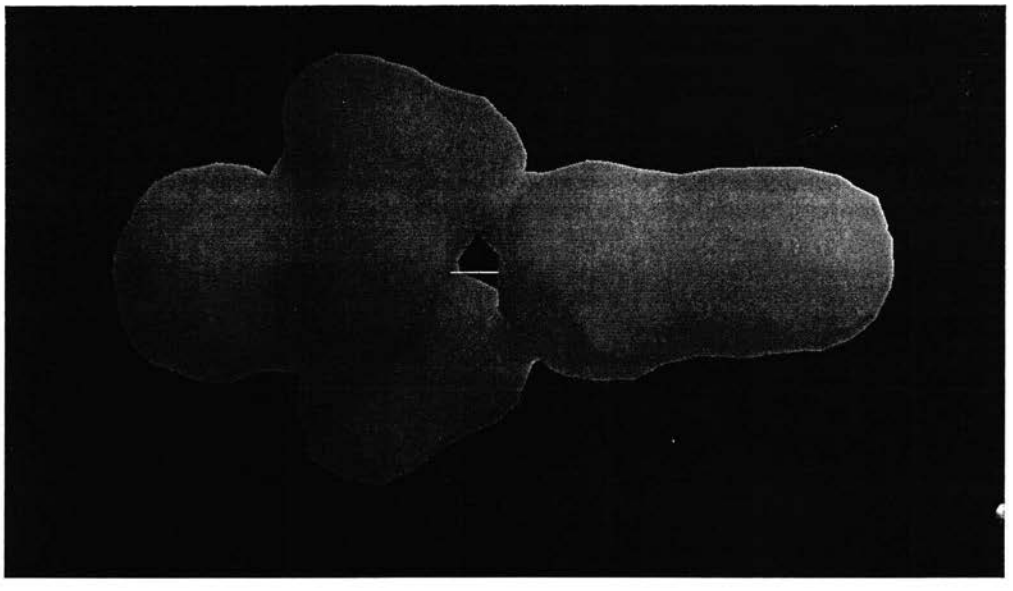


Figura 5.1 Densida muy rígida por parte del anión debido a la falta de funciones de polarización

Con esto, se ha podido ver claramente cómo se pueden obtener resultados interesantes utilizando incluso una base mínima con un nivel de teoría HF y con éstos comenzar realmente el estudio; ya que este sistema químico necesita de un nivel de teoría mucho mayor y una base mas completa para poder obtener resultados con una precisión confiable; y esto, sólo se logra entendiendo que información se necesita y el sistema químico en el que se trabaja. Obviamente se podían realizar cálculos IRC, búsqueda del estado de transición, dinámica molecular, etc para poder encontrar el mínimo y el mecanismo de la reacción; sin embargo, el objetivo al ser didáctico, era hacer y entender los pasos a seguir y no sólo utilizar la caja negra que haría los cálculos de manera automática y sin enfrentar las diferentes interrogantes que surgen con el mecanismo de reacción.

Capítulo 6

CONCLUSIONES Y RECOMENDACIONES

A lo largo de este trabajo, se ha hecho énfasis en las variables que más afectan los cálculos en la química cuántica computacional, donde dichas variables son estudiadas por las ciencias de la computación y las ciencias químicas. El estudio, entendimiento y buen manejo de dichas variables y herramientas permitirán el uso eficiente, metodológico y sistemático de la química cuántica computacional de tal manera que se minimicen los costos, el tiempo de cómputo y de investigador obteniendo así los mejores resultados posibles. Desgraciadamente esto no ha ocurrido así, provocando que se haga un pésimo uso de los recursos computacionales y obteniendo resultados erróneos o con artificios, los cuales muchas veces son publicados y que provienen de fuentes de error de la computadora o del cálculo mismo al ignorar las aproximaciones que se usan.

Las variables que más afectan son:

- perfecto entendimiento del sistema a estudiar e información a obtener (métodos, bases, propiedades, etc.)
- precondiciones del paralelismo
- arquitectura del problema y el grado de paralelismo inherente de la aplicación en caso necesario
- arquitectura de las máquinas (procesadores, red de interconexión, memoria, etc.)
- eficacia de las diferentes herramientas para la buena ejecución de los cálculos (software, compiladores, bibliotectas, etc.)
- documentación completa para un uso eficiente de la computadora (análisis asintótico, *benchmarks*, etc.)

En caso de que se inicie un proyecto de química cuántica computacional, es necesario tomar en cuenta todas estas variables. Cuando ya se han tomado las decisiones y ya se cuenta con el *hardware*, *software* y *humanware* necesario; entonces como usuario, sólo es necesario seguir las políticas de buen uso, la documentación y el perfecto entendimiento del sistema químico a estudiar y la información que se obtendrá para obtener resultados que se puedan asociar y racionalizar con una gran cantidad de moléculas generando así un modelo; ya que ésta es la fuerza real de la química cuántica computacional y no simplemente la obtención de simples datos crudos los cuales rápidamente serán mejorados por el avance de las computadoras.

A pesar de que un usuario, sólo necesita el perfecto entendimiento del sistema químico y la documentación necesaria para el buen uso del cómputo intensivo, es necesario que tenga algo de conocimiento de las variables que afectan los cálculos inherentes a las ciencias de la computación (errores por redondeo, mala programación, etc.) para evitar los errores, las limitaciones, aprovechar las ventajas y entender el porqué de la documentación y políticas de administración para el buen uso de la computadora y esto se hace más evidente cuando se requiere de la actualización de las herramientas para los cálculos; donde nuevamente se buscará que las nuevas decisiones ofrezcan herramientas que tendrán el mayor ciclo de vida posible.

Con los estudios mostrados, se verificó la necesidad de hacer un buen análisis desde el punto de vista de las ciencias de la computación y de las ciencias químicas. En el estudio con el *cluster*, se recomienda tenerlo particionado de tal manera que se puedan correr programas seriales y paralelos y que todos los programas se ejecuten de tal manera que no compitan por los recursos del *cluster* mediante el uso de colas. Para los programas estudiados, se recomienda sólo ejecutar los cálculos en un solo nodo para *Gaussian* y para *GAMESS-US* utilizar el cómputo paralelo para sistemas químicos que lo requieran. Se hacen varios estudios tipo *benchmark* con estos programas. Sin embargo, debido a la dificultad de trasladar estos estudios a otros programas, sistemas químicos, cálculos, etc; se recomienda hacer un *benchmark* donde para un mismo sistema químico y cálculo, se obtenga el *speedup* variando el número de funciones base y el número de procesadores (isoeficiencia) de tal manera que un investigador fácilmente podría con estos datos saber hasta qué número de procesadores su cálculo escalará bien y qué cola será la conveniente a usar. También se recomienda hacer *benchmark*s donde la ejecución dure aproximadamente 50,000 segundos y quizá con unas 300 funciones de base para obtener datos del cómputo intensivo en las computadoras actuales.

En cuanto a la síntesis de tiocianatos, se mostró la importancia de hacer estudios cualitativos para ver las tendencias y la viabilidad de la investigación, así como la importancia de tomar en cuenta los tipos de cálculos de tal manera que no se pueden hacer más conclusiones que a las que se llegaron. Con estos datos, se puede hacer una mejor planeación de los futuros cálculos (nivel de teoría mayor y conjunto de bases más completas) donde se podrían incluir efectos de disolvente, dinámicas, diferente contraión, etc., para ver el porqué de la reactividad del ión tiocianato, su mecanismo de reacción y cómo encontrar un rendimiento mayor para un producto deseado; sin el desperdicio de tiempo de cómputo y de investigador al diseñar mal el proyecto de investigación.

Por último, para el avance de la química cuántica computacional y para que los recursos que se obtengan, tengan el mayor ciclo de vida posible, es necesario ver cómo ha evolucionado la tecnología en las ciencias de la computación y en las ciencias químicas, como evoluciona actualmente y entender así el porqué se dice que métodos como Monte Carlo Cuántico y del Elemento Finito, serán los más utilizados en el futuro. Para poder hacer cálculos en sistemas cada vez más reales (número de Avogadro) y con una mayor precisión, es necesario utilizar todas las herramientas que las ciencias químicas (desarrollo de nuevos métodos y teorías) y las ciencias de la computación (implementaciones de dichos métodos que aprovechen de la mejor manera el hardware que existe, nuevos algoritmos y metodologías de programación, desarrollo de software) pueden dar y para ello es de vital importancia, conocer un poco de estos campos de la ciencia.

Un ejemplo muy claro de ello es el desarrollo de los *cluster*s y de los *Grids*, tecnologías de los 90 y actuales, que permiten a grupos de investigadores sin muchos recursos económicos tener acceso al cómputo de alto rendimiento; sin embargo, es importante señalar que el menor costo económico implica un mayor costo en tiempo y en *humanware* (administración y puesta a punto).

Capítulo 7

BIBLIOGRAFÍA

Libros para la arquitectura de las computadoras y de las computadoras paralelas:

[1] Arquitectura de computadoras y procesamiento paralelo Kai Hwang, Fayé A. Briggs. Mc. Graw-Hill. 1988

[2] Organización y arquitectura de computadoras. Diseño para optimizar prestaciones. William Stallings. Prentice-Hall. Cuarta edición. 1997

[3] Organización de computadoras. Andrew S. Tanenbaum. Prentice-Hall. Tercera edición. 1992

[4] Advanced Computer Arquitecture. Parallelism, Scalability, Programmability. Kai Hwang. Mc. Graw-Hill. 1992

Libros para el cómputo paralelo y de alto rendimiento:

[5] Foster, I. Designing and Building Parallel Program: Concepts and Tools for Parallel Software Engineering. Addison-Wesley, New York, 1995. http://www.mcs.anl.gov/dbpp

[6] Is Parallelism For You? Rules-of-Thumb for Computational Scientists and Engineers. Cherri M. Pancake. Computacional Science and Engineering. Vol. 3, No.2 (Summer, 1996). pp. 18-37. http://cs.oregonstate.edu/~pancake/papers/IsParall/index.html

[7] An Introduction to Parallel Computing: Design and Análisis of Algorithms. 2nd edition. Ananth Grama. Pearson Addison Wesley. 2003

[8] Software Optimization for High Performance Computing: Creating Master Applications. Isom Crawford. Prentice Hall. 2000

Libros para cluster:

[9] High Performance Cluster Computing: Architectures and Systems. Vol. 1. Rajkumar Buyya. Prentice Hall. 1999

[10] High Performance Cluster Computing: Programming and Applications. Vol. 2. Rajkumar Buyya. Prentice Hall. 1999

[11] http://www.clusters.unam.mx

Libros principales para la química computacional:

[12] Computational Chemistry: A Practical Guide for Applying Techniques to Real World Problems. David Young. Wiley Interscience. 2001

[13] Introduction to Computational Chemistry. Frank Jensen. John Wiley & Sons. 1998

Otros:

[14] Quantum Chemistry. Ira N. Levine. 5th edition. Prentice Hall. 1999

[15] Molecular Quantum Mechanics. P. W. Atkins. 3rd edition. Oxford University Press. 1999

[16] Modern Quantum Chemistry. Introduction to Advanced Electronic Structure Theory. Attila Szabo. Dover Publications, Inc. 1996

[17] http://qcldb2.ims.ac.jp

[18] http://www.osc.edu/PET/CCM/skeleton/training/courses/foundations/foundations.html

[19] Impact of Advances in Computing and Communications Technologies on Chemical Science and Technology: Report of a Workshop. http://www.nap.edu/openbook/0309065771/html/20.html Software Development for Computational Chemistry: Does Anything Remain to Be Done? Peter R. Taylor. San Diego Supercomputer Center.
[20] Impact of Advances in Computing and Communications Technologies on Chemical Science and Technology: Report of a Workshop. http://www.nap.edu/openbook/0309065771/html/109.html A Computer Science Perspective on Computing for the Chemical Sciences. Susan L. Graham. University of California, Berkeley.
[21] SciFinder Scholar, 2004 Edition. American Chemical Society.
[22] Internet Journal of Chemistry, 2000, 3, 4. http://www.dhpc.adelaide.edu.au/reports/073/html/dhpc-073.html
[23] The Chemistry of Cyanates and their thio derivatives. Saul. Patai. J. Wiley 1977.
[24] T. H. Dunning, Jr., K. A. Peterson, D. E. Woon, Encycl. Comput. Chem. 1, 88 (1998)
[25] J. M. L. Martin, J. Chem. Phys., 100 (1994), 8186.
[26] M. Klobukowski, S. Huzinaga, Y. Sakai, Computational Chemistry Reviews of Current Trends Volume 3 49, J. Leszczynski, Ed., World Scientific, Singapore (1999).
[27] F. B. van Duijneveldt, J. G. C. M. van Duijneveldt-van de Rijdt and J. H. van Lenthe, Chem. Rev., 94 (1994), 1873.
[28] P. Hohenberg, W. Kohn, Phys, Rev., 1958, 136, B864.
[29] W. Kohn, L. J. Sham, Phys, Rev., 1965, 140, A1133.

APENDICE A

ANÁLISIS ASINTÓTICO EN LA QUÍMICA COMPUTACIONAL

Existen muchos algoritmos, técnicas y métodos en la química computacional que pueden ser programadas de una manera eficiente para determinadas arquitecturas de computadoras y obtener prácticamente el mismo resultado. Debido a esto, un método dado, tendría algunas diferencias en el tiempo de ejecución e incluso en el análisis asintótico de un programa a otro. A continuación se da una tabla de un análisis asintótico de los métodos más utilizados, donde M es el número de átomos, N es el número de orbitales y puede aumentar al incluir más átomos o al utilizar una base más grande.

Método	Escalamiento de tiempo (donde N es el número de orbitales)	Comentarios
DFT	N	Para moléculas muy grandes con algoritmos que escalen linealmente.
	N^3	
MM	M^2	
MD	M^2	
Semiempíricos	N^2	Para moléculas pequeñas y medianas (cuello de botella→integrales)
	N^3	Para moléculas muy grandes (cuello de botella→inversión de matrices)
HF	N^2-N^4	Dependiendo del uso de la simetría y quitando el cálculo de integrales con valores muy pequeños.
	N^3	Método pseudos espectral.
QMC	N^3	Con la inversión de la matriz de Slater
	N!	Sin la inversión de la matriz de Slater
MP2	N^5	
CC2	N^5	
MP3,MP4(SDQ)	N^6	
CCSD	N^6	
CISD	N^6	
MP4	N^7	
CC3,CCSD(T)	N^7	
MP5	N^8	
CISDT	N^8	
CCSDT	N^8	
MP6	N^9	
MP7	N^{10}	
CISDTQ	N^{10}	
CASSCF	A!	A es el número de orbitales activos.
Full CI	N!	

Para los cálculos ab initio, semiempíricos y de dinámica molecular, la cantidad de tiempo de cómputo necesario para el cálculo es generalmente el factor más importante. Sin embargo, para sistemas muy grandes, el uso de memoria comienza a ser un factor importante para los cálculos de la mecánica molecular. A continuación se da una tabla con los requerimientos de memoria de una lista de algoritmos para la optimización de la geometría:

Algoritmo	Uso de memoria
Gradiente conjugado	D
Fletcher-Reeves	D
Polar-Ribiere	D
Simples	D^2
Powell	D^2
Quasi-Newton	D^2
Fletcher-Powell	D^2
BFGS	D^2

En general, cuando se tiene un método que requiere un tiempo de cómputo grande; también requerirá de mucha memoria, aunque obviamente, estos valores dependen de la implementación que se dio en el programa que se utiliza. Por ejemplo, para cálculos HF, por lo regular se requiere un orden de N^2 para el uso de la memoria, excepto para algoritmos in-core donde se requiere una orden de N^4. Para cálculos QMC, que requieren de un tiempo de cómputo muy alto, requieren de muy poca memoria en comparación con los demás métodos ab initio.

Computational Chemistry: A Practical Guide for Applying Techniques to Real World Problems. David Young. Wiley Interscience. 2001

APENDICE B

SOFTWARE PARA LA QUÍMICA COMPUTACIONAL

Este apéndice sólo intenta proveer una lista de los softwares más utilizados o que tienen alguna particularidad importante; sin embargo, existen una gran cantidad de programas que están siendo desarrollados y que han sido desarrollados pero que sólo se utilizan para un propósito muy particular. Como la funcionalidad y costo cambian rápidamente, la lista no puede ser extensiva. Por ello, sólo se dan algunos detalles de los programas además de la página donde se puede obtener mayor información y en cuanto a los costos, se dividirán en las siguientes categorías:

Precio (en dólares americanos)	Categoría
0	Libre
1-100	Estudiantil
101-300	Individual
301-1000	Producción
1001-5000	Departamental
>5000	Institucional
Contactar	Contactar con el vendedor para el precio

1- **Alchemy:** es una interface gráfica para hacer cálculos de mecánica molecular y semiempíricos. Los cálculos pueden ser hechos con TRIPOS que se encuentra incorporado en el programa o llamando a MM3 o MOPAC que se encuentran incluidos. Alchemy es diseñado por Tripos y vendido por SciVision donde el costo es de producción. Las moléculas pueden construirse en dos dimensiones y el programa le dará una geometría tridimensional. Contiene librerías de grupos funcionales orgánicos, constructor de proteínas además de que existen varios programas que interactúan con Alchemy como SciQSAR, SciLoP y SciPolymer. El usuario puede cambiar la estereoquímica y angulos de conformación. La plataforma es PC. http://www.scivision.com/

2- **ADF (Amsterdam Density Functional):** programa DFT que puede usar conjunto de bases STO y GTO, puede hacer cálculos DFT relativistas y contiene funcionales LDA y de gradiente corregido. Tiene una interfaz gráfica llamada Cerius que es vendida por Molecular Simulations, Inc. También hay un programa aparte para cálculos de estructuras de bandas. Tiene buena documentación. Se pueden hacer cálculos para obtener la optimización de la geometría, optimización de la estructura de transición, cálculos de frecuencias, IRC, estados excitados usando el método TDDFT, efectos de solvatación con el método COSMO, simulaciones ESR y NMR, hiperpolarizabilidades, intensidades de Raman con el método TDDFT, efectos relativistas con el método ZORA e intensidades IR. El costo es de estudiante para arriba y se tiene para plataformas PC, Cray, SGI, DEC, Fujitsu, NEC, HP y RS/6000 bajo UNIX. La compañía que lo vende es Scientific Computing & Modelling NV http://www.scm.com/

3- **AMPAC:** programa para hacer cálculos semiempíricos. Tiene una interface gráfica incluida que es fácil de usar y tiene buena documentación. AMPAC soporta AM1, SAM1, SAM1/d, MNDO, MNDO/d, MNDOC, MINDO/3 y PM3. Para hacer cálculos de solvatación tiene los métodos COSMO y SM1-SM3. Puede hacer single-point, optimización de energías, cálculos de frecuencias e IRC. Tiene búsqueda conformacional, ESR y propiedades ópticas no lineales. El costo es de producción, departamental e institucional. Las plataformas son PC (bajo Windows y Linux), SGI, RS6000, Alpha, Sun e HP-UX. La compañía que lo vende es Semichem. http://www.semichem.com/

4- **Babel:** es una utilidad que traduce los archivos de entrada de un formato a otro de varios programas de química computacional, alrededor de 50. Su costo es libre y corre bajo UNIX, PC y Macintosh. http://www.ccl.net/pub/chemistry/software/UNIX/babel/

5- **Chem3D:** programa para modelaje molecular para PC y Macintosh. Hace cálculos usando MM2 y Hückel extendido y puede servir como interfaz gráfica para MOPAC (que viene incluido) o Gaussian. Puede leer una gran variedad de archivos de otros programas como Gaussian, MacroModel, GAMESS, MDL, MOPAC, PDB y SYBYL. Las estructuras de dos dimensiones hechas con ChemDraw o ISIS/Iris pueden ser fácilmente convertidas en estructuras de 3 dimensiones. Los datos pueden ser exportados en una gran cantidad de formatos e imágenes. Con base en una serie de ventanas se pueden escoger el nivel de teoría, tipo de cálculos y propiedades como: optimización de geometría, optimización de la estructura de transición, momentos de dipolo, análisis de población, solvatación mediante COSMO, constantes de acoplamiento hiperfina y polarizabilidad. Se pueden desplegar superficies moleculares mostrando la densidad electrónica, densidad de spin, potencial electrostático y orbitales moleculares. Mientras los cálculos son ejecutados, la interface sigue siendo operacional logrando así que se puedan tener varios trabajos corriendo simultáneamente o serialmente al hacer una cola. Se pueden calcular varias propiedades físicas como área, volumen, peso molecular, momentos de inercia; propiedades químicas como punto de ebullición, de fusión, variables críticas, solubilidad, refractividad, etc. La calidad de las gráficas es muy buena y se pueden hacer películas con ellas al generar múltiples estructuras como una simulación de dinámica molecular. Estas películas sólo se pueden ver con Chem3D. Existen varias versiones de Chem3D y cada una difiere en el precio. La compañía que los vende es CambridgeSoft. http://www.camsoft.com/

6- **Cristal:** programa para cálculos ab initio y de estructura de bandas. Los cálculos pueden ser por HF o DFT con funcionales LDA, híbridos y de gradiente corregido. Se pueden hacer cálculos con spin restringido, no restringido y restringidos de capa abierta. También de all-electron o pseudopotencial. Se pueden calcular varias propiedades como análisis de población de Mulliken, densidad electrónica, multipolos, factores de estructuras de rayos X, potencial electrostático, estructuras de bandas, tensores hiperfinos, etc. Para la interfaz gráfica se puede usar Cerius y XcrySDen. El costo puede ser de producción, departamental e institucional. Las plataformas son PC (Windows y Linux) y UNIX. http://www.ch.unito.it/ifm/teorica/crystal.html

7- **CODESSA:** es un programa QSAR/QSPR convencional. Lee estructuras moleculares generadas por otros programas para comenzar el análisis QSAR. Puede importar resultados de

AMPAC, MOPAC y Gaussian. Las moléculas pueden categorizarse en constitución, topología, geometría, electrostática, química cuántica y termodinámicamente. Su costo puede ser de producción, departamental e institucional; las plataformas son PC (Windows y Linux), SGI, RS6000, Alpha, Sun e HP-UX. http://www.semichem.com/

8- **ChemSketch:** interface gráfica que se relaciona con el ACD (Advanced Chemistry Development) Software Suite. La documentación es buena. Tiene un constructor de péptidos y de ácidos nucleicos. Tiene una rutina de optimización de geometría que utiliza el campo de fuerzas CHARMM. Tiene un lenguaje interpretativo llamado ChemBasic, que es similar a BASIC que puede incorporar funciones adicionales. Este programa está principalmente orientado a moléculas orgánicas aunque también puede trabajar con estructuras inorgánicas. También puede calcular propiedades de los líquidos como refractividad, volumen molar, índice de refracción, tensión superficial, densidad y constante dieléctrica. Puede predecir espectros NMR para varios núcleos y de dos dimensiones, puntos de ebullición, pka, solubilidad, el parámetro sigma de Hammett, tiempo de retención cromatográfica y nombres sistemáticos basados en la IUPAC y CAS. La gran mayoría de las predicciones las hace mediante la comparación de bases de datos. El costo es libre excepto de diferentes módulos con diferentes precios. La plataforma es la PC. La compañía que lo vende es Advanced Chemistry Development, Inc. http://www.acdlabs.com/

9- **DFT++:** este programa es muy especial computacionalmente hablando ya que fue diseñado desde un principio como un programa paralelo, optimizado, modular, extensible, muy portable, al que se le pueda dar buen mantenimiento y el costo es libre. Esto se intenta con base a un buen análisis y diseño del programa utilizando la programación orientada a objetos. Está escrito en C++ con un rendimiento muy aceptable pero sobre todo con un buen análisis que permite entender y desarrollar de manera rápida y sencilla. Es un programa que hace cálculos complejos DFT principalmente en sistemas periódicos. En cuanto al paralelismo puede ser de grano grueso con MPI e hilos (threaded) y de grano grueso y fino combinando MPI e hilos. Puede hacer cálculos all-electron con wavelet (paquetes de ondas) o pseudopotenciales de ondas planas. Se puede instalar en cualquier plataforma que use UNIX y Windows. http://dft.physics.cornell.edu/

10- **GAMESS:** programa que puede hacer cálculos ab initio y semiempíricos. El hecho de que este programa sea libre y de alta calidad como software, lo hace el favorito para muchos investigadores. Los métodos ab initio son RHF, UHF, ROHF, GVB, MCSCF, MP2 y correcciones CI. También contiene los cálculos semiempíricos MNDO, AM1 y PM3. Puede calcular estructuras de transición, coordenadas de reacción, frecuencias de vibración, propiedades ópticas lineales y no lineales, incluye una aproximación de acoplamiento spin-orbital y puede calcular una trayectoria clásica en una PES ab initio. Existes varios modelos de solvatación, análisis de multipolo y de población. El programa macmolplt fue diseñado para desplegar las salidas de los cálculos de GAMESS. GAMESS fue diseñado para tener algoritmos robustos y dar un alto nivel de detalle y de control sobre las rutinas. GAMESS ha sido paralelizado usando algunas rutinas de NWChem. El costo es libre. http://www.msg.ameslab.gov/GAMESS/GAMESS.html

11- **Gaussian:** es un programa ab initio monolítico. Probablemente incorpora el más extenso rango de funcionalidades para cálculos ab initio. También incluye algunos métodos semiempíricos y de mecánica molecular que pueden ser usados solos o como parte de un cálculo QM/MM. Existen varias interfaces que pueden trabajar con Gaussian como GaussView, Cerius, UniChem, AMPAC GUI, Chem3D, Viewmol y Spartan. Tiene buena documentación con muchos ejemplos, principalmente en varios libros. De cálculos ab initio contiene HF, ROHF, MPn, CI, CC, QCI, MCSCH, CBS y G2. Contiene varios funcionales para cálculos DFT, varios conjuntos de bases y pseudopotenciales. Para cálculos semiempíricos contiene AM1, PM3 y ZINDO sólo para single point. Los métodos de mecánica molecular son Amber, Dreiding y UFF. Existen una gran cantidad de propiedades moleculares que se pueden calcular, además de propiedades ópticas no lineales, NMR, varios análisis de población, frecuencias vibracionales, etc. Cálculos QM/MM se pueden hacer usando el método ONIOM. También se pueden hacer IRC y búsquedas de estructuras de transición. Se puede escoger si la evaluaciones de las integrales se hacen directas, in-core y semidirectas. Puede correr de manera paralela. El precio de toda esta funcionalidad es que el usuario debe invertir tiempo en aprender la complejidad del programa para hacer los cálculos deseados. La salida del programa tiene una gran información, mucha de la cual puede ser no importante para el usuario y existen una serie de revisiones al programa que van generando varios parches que son necesarios incluir para una mayor robustez del mismo. Gaussian ha sido el programa que más se ha usado para el modelado de moléculas orgánicas; sin embargo, para moléculas inorgánicas, generalmente requiere cierta sofisticación técnica del usuario. El costo puede ser de producción, departamental e institucional. Las plataformas son PC (Windows y Linux), DEC, Cray, Fujitsu, HP-UX, RS/6000, NEC, SGI, Sun. La compañía que lo vende es Gaussian, Inc. http://www.gaussian.com/

12- **HyperChem:** programa de interface gráfica integrada, cálculos y visualización. Es el programa mas usado en PCs. Puede ser usado como interface gráfica para Q-Chem. Incorpora cálculos ab initio, semiempíricos y de mecánica molecular. Puede ser usado para calcular frecuencias vibracionales, búsqueda estados de transición, estados electrónicos excitados, QM/MM, dinámica molecular, cálculos DFT y simulaciones de Monte Carlo. Tiene un constructor de biomoléculas, polímeros y de cristales para sistemas periódicos. La interface gráfica es muy buena y completa. Tiene una serie de ventanas con opciones y menús para los cálculos, el nivel de teoría, bases y propiedades. Mientras se hacen los cálculos, ninguna otra acción se puede hacer, además de que los resultados sólo se pueden guardar en un archivo .log. Para la mecánica molecular, los campos de fuerzas disponibles son MM+, OPLS, BIO+ y AMBER. Las técnicas semiempíricas disponibles son EH, CNDO, INDO, MINDO/3, ZINDO, MNDO, AM1, ZINDO/S, MNDO/d PM3(TM) y PM3. Se pueden poner moléculas de solventes para la mecánica molecular y para cálculos biomoleculares se puede hacer una búsqueda conformacional y cálculos QSAR. El costo puede ser de estudiante, individual, producción y departamental y las plataformas en PC (Windows), SGI, HP-UX, Win CE. La compañía que lo vende es HyperCube Inc. http://www.hyper.com/

13- **Jaguar::** programa de cálculos ab initio diseñado, debido a los algoritmos escogidos y a la estrategia de optimización, para correr eficientemente con moléculas grandes. Jaguar utiliza un esquema de integración pseudoespectral que logra que los cálculos HF, GVB, DFT y MP2 sean más eficientes computacionalmente. A Jaguar se le llama formalmente PS-GVB. Se pueden hacer cálculos GVB-RCI, GVB-DFT y GVB-MP2. Las propiedades que se pueden calcular

son cargas electrostáticas, multipolos, polarizabilidades, hiperpolarizabilidades, varios esquemas de análisis de población e IR. Geometrías y cálculos, pueden ser importados de GAMESS, Gaussian y Spartan. Jaguar, por lo regular es utilizado para cálculos GVB o MP2 en grandes moléculas. El costo es de producción y mayor y las plataformas son SGI, RS/6000, HP-UX, Cray, Alpha y PC con Linux. La compañía que lo vende es Schrödinger, Inc. http://www.schrodinger.com/

14- **MacroModel:** es un programa poderoso para hacer cálculos de mecánica molecular. El programa puede correr con interfaz gráfica o con línea de comando. La utilidad XCluster permite el análisis y filtrado de un gran número de estructuras como en trayectorias dinámicas o en una simulación de Monte Carlo. La documentación es muy completa. Los campo de fuerzas disponibles son: MM2*, MM3*, AMBER*, OPLSA*, AMBER94 y MMFF. El asterisco indica que estos campos de fuerzas tienen modificaciones a las descripciones originales que se dieron en la literatura. Se pueden hacer minimizaciones con la mecánica molecular, simulaciones de Monte Carlo y simulaciones de dinámicas moleculares. El modelo GB/SA de solvatación también está disponible. Tiene constructores para moléculas orgánicas, péptidos, nucleótidos y carbohidratos. El precio es de producción o más alto, la compañía que la vende es Schrödinger, Inc. y las plataformas son SGI, RS/6000. http://www.schrodinger.com/

15- **MOE:** programa de modelaje molecular que incluye mecánica molecular, dinámica, condiciones de frontera periódicas, QSAR, un constructor combinatorio, de carbohidratos y proteicos y muchas funciones para modelaje proteico. También se pueden calcular propiedades de polímeros y parámetros de difracción. La interfaz gráfica es multitarea sin embargo, las imágenes no se pueden guardar como archivos de imágenes; a menos que se use otro programa para hacerlo. Los campos de fuerzas disponibles son: AMBER89, AMBER94, MMFF94 y PEF95SAC. Las utilidades QSAR incluyen funciones para la diversidad molecular, similaridades y agrupaciones además de que se pueden incluir nuevas bases de datos. Existen más de 130 descriptores pero el programa no puede calcular dichos descriptores, necesitará de cálculos semiempíricos o ab initio hechos por otro programa. Tiene el lenguaje SVL (Scientific Vector Language) que es una combinación de C++ y lenguaje de script que sirve para desarrollar autómatas y otras manipulaciones para la mecánica molecular, manipulación simbólica, desplegado gráfico, etc. Con este lenguaje es sencillo adicionar funciones propias y modificar rutinas existentes. Debido a todo esto existe gran portabilidad del programa, además de que la documentación es buena. Para el costo hay que hacer contacto, las plataformas son PC (Windows y Linux), SGI, Sun y HP-UX. La compañía que lo vende es Chemical Computing Group, Inc. http://www.chemcomp.com/

16- **MOLDEN:** programa para desplegado molecular. Puede desplegar geometrías moleculares de una gran cantidad de archivos de varios programas. Varias vistas de la función de onda pueden ser desplegados de los archivos de salida de Gaussian y GAMESS. Ciertas funcionalidades están disponibles de MOPAC y AMPAC. Programas de conversión están disponibles para importar funciones de onda de ADF, MOLPRO, ACESS II, MOLCAS, DALTON, Jaguar y HONDO. Tiene un editor de la matriz Z y puede optimizar la geometría pasando los datos al programa TINKER. Las funciones de onda pueden ser visualizadas como densidad electrónica total, densidad orbital, potencial electrostático, el laplaciano de la densidad

electrónica, etc. El costo puede ser libre o de producción y las plataformas son PC (Windows, Linux, free BSD), SGI, RS/6000, Alpha, Cray, HP-UX, Sun, open VMS y OS2. http://www.cmbi.kun.nl/~scharft/molden/molden.html

17- **MOLPRO:** programa para cálculos ab initio, diseñado para desempeñarse bien en cálculos que presenten dificultades técnicas o que son muy sensitivos a la correlación electrónica. Pequeñas partes del código han sido paralelizadas. Los cálculos que han sido desarrollados son HF, CI, MRCI, FCI, CC, DFT, MCSCF, CASSCF, ACPF, CEPA y varias variaciones de estos. Pueden hacerse cálculos con teoría de perturbaciones para un o múltiples determinantes. Los algoritmos MCSCF y CC han probado ser muy eficientes. Están disponibles funciones de onda para cálculos restringidos, no restringidos y restringidos de capa abierta. El usuario puede tener un gran control en la construcción de la función de onda. Se pueden hacer correcciones energéticas relativistas, acoplamientos spin-orbita, gradientes de campo eléctrico y multipolos. También están disponibles opciones para estados electrónicos excitados y cálculos de estructura de transición. Se pueden usar conjunto de bases con altos momentos angulares (spdfghi) y potenciales *de core* efectivos. Dentro de los archivos de entrada se pueden tener variables, ciclos y procedimientos lo que hace que se puedan hacer cálculos muy complejos y por lo tanto el usuario debe entender muy bien la teoría de los métodos ab initio para obtener todos los beneficios del programa ya que es muy poderoso pero no es para principiantes. Este programa es excelente para cálculos ab initio sofisticados y de alta precisión y que pueden tener dificultades técnicas como estados excitados, sistemas de capa abierta, metales de transición y correcciones relativistas. El costo es de producción, departamental e institucional. Las plataformas son Linux, Alpha, Cray, Fujitsu, AIX, SGI, Sun, HP-UX y NEC. http://www.tc.bham.ac.uk/molpro/

18- **MOPAC:** uno de los programas mas usados para cálculos semiempíricos y ha sido diseñado para ser robusto y tener un amplio rango de funcionalidad. Pueden usarse WinMOPAC, Alchemy, PC Model, Chem3D e HyperChem como interfaces gráficas. Existen versiones libres (hasta la sexta) que han sido incorporadas en otros programas, actualmente el costo del programa puede ser de producción, departamental y la compañía encargada de venderla es Schrödinger, Inc. La última versión comercial de MOPAC puede hacer cálculos semiempíricos con MNDO, MINDO/3, MNDO/d, AM1, AM1-d y PM3 y dichos cálculos se pueden hacer para sistemas con alto-spin y estados excitados usando interacción de configuraciones. Se pueden calcular estructuras de transición e IRC. Se pueden hacer cálculos para sistemas periódicos, para hiperpolarizabilidades, acoplamientos hiperfinos y análisis de población. Pueden hacerse simulaciones de dinámica clásica en una PES calculada semiempíricamente. Las plataformas son para PC con Linux y muchos sistemas UNIX. http://www.schrodinger.com/

19- **NWChem:** programa para cálculos ab initio, estructura de bandas, mecánica molecular y dinámica molecular. Este programa es único en el sentido de que su diseño está basado para correr eficientemente en máquinas paralelas. Existen varios bechmark en la página del programa que lo muestran. Puede correr de manera serial o paralela. Los método ab initio disponibles son HF, DFT, MPn, MCSCF, CI y CC. Existen una gran cantidad de bases disponibles, incluyendo ECP. Los cálculos dinámicos se pueden hacer con cálculos ab initio. El programa puede evaluar las integrales de manera convencional, in-core o directas. El campo

de fuerza disponible para la mecánica molecular es AMBER95 y CHARMM. Están disponibles la minimización y cálculos dinámicos mediante QM/MM. Su costo es libre y las plataformas son PC (Linux), SGI, Cray, Paragon, SP2, KSR, Sun, DEC, IBM. http://www.emsl.pnl.gov/pub/docs/nwchem

20- **PCModel:** es un programa de mecánica molecular con interfaz gráfica. Puede usarse como interfaz gráfica para AMPAC, MOPAC y Gaussian. Los campos de fuerzas disponibles son MMX, MM3 y MMFF94. Algunos de los cálculos disponibles son la optimización de geometría, dinámica molecular y un algoritmo para calcular las barreras rotacionales. Tiene constructores para aminoácidos, azucares y ácidos nucleicos. Tiene librerías para geometrías de estados de transición, grupos orgánicos funcionales y organometálicos. El costo es de producción y las plataformas son PC (Windows y Linux) y SGI. La compañía que lo vende es Serena Software. http://www.serenasoft.com/

21- **Q-Chem:** es un programa de cálculos ab initio diseñado para hacer cálculos eficientemente para moléculas grandes. Se pueden usar los programas HyperChem y UniChem como interfaces gráficas para Q-Chem. Incluye HF, ROHF, UHF y MP2 y una buena selección de funcionales para cálculos DFT. Están disponibles los métodos de análisis de población de Mulliken y NBO. Se tienen múltiples opciones para la convergencia de SCF, optimización de la geometría y funciones de onda iniciales (initial guess). También se pueden hacer cálculos IR y Raman. Tiene buena documentación. Tiene el método CFMM (Continuous Fast Multipole Method) para escalar linealmente cálculos DFT que lo hace más rápido en comparación con Gaussian FMM aunque requiere mas memoria. Los usuarios de Gaussian y GAMESS encontrarán muy familiar los formatos de Q-Chem. Para estados excitados, contiene los métodos CIS, RPA, XCIS y CIS(D). El precio es de producción, departamental e institucional. Las plataformas son PC (Windows y Linux), DEC, Cray, Fujitsu, RS/6000, SP2, SGI y Sun. La compañía que lo vende es Q-Chem, Inc. http://www.q-chem.com/

22- **SPARTAN:** programa para cálculos ab initio, DFT, semiempíricos y de mecánica molecular con una interfaz gráfica amigable. Este es uno de los programas favoritos para químicos experimentales e instituciones educacionales. Es un programa sencillo de usar y muy robusto; pero el costo de esta robustez es que no es un programa muy eficiente computacionalmente. Puede hacer cálculos para estados de transición, búsquedas conformacionales, herramientas combinatorias y visualización. El constructor incluye péptidos, grupos funcionales, nucleótidos, geometría de alta coordinación, etc. El único método de correlación es MP2. Los cálculos DFT sólo son con la aproximación LDA. Pocas propiedades se pueden obtener de la mecánica molecular. El costo es individual, producción o departamental y las plataformas son PC (Windows y Linux), Macintosh, SGI, RS/6000, Alpha y HP-UX. La compañía que lo vende es Wavefunction, Inc. http://www.wavefun.com/

23- **TINKER:** es una colección de programas desarrollados para hacer cálculos de mecánica y dinámica molecular. Se pueden usar los programas RasMol, ChemDraw, Chem3D, gOpenMol, MOLDEN y ReView como interfaz gráfica. Bases de datos de proteínas pueden ser importadas y los archivos de salida se pueden formar en formato para Civil, Insight II o Xmol. Los campos de fuerzas disponibles son AMBER95, CHARMM22, MM2(91), MM3(99), OPLS-AA, OPLS-UA y TINKER. Se pueden hacer optimización de geometría,

dinámica molecular, búsqueda conformacional, estados de transición, etc. El programa es libre y las plataformas son PC (Windows y Linux), SGI, Macintosh, RS/6000, Alpha, HP-UX y Sun. http://dasher.wustl.edu/tinker/

24- **UniChem:** interfaz gráfica hecha para correr cálculos en máquinas remotas. Puede servir como interfaz para Gaussian y Q-Chem. El servidor puede venir con MNDO, DGauss y CADPAC. Puede servir para crear interfaces para programas propios. Existen librerías para grupos funcionales. Muy útil al momento de encolar los trabajos con NQS y tener control de éstos. Para el costo es necesario hacer contacto y las plataformas son Cray, SGI, RS/6000. La compañía que lo vende es Oxford Molecular Group, Inc. http://www.oxmol.com/

25- **WebLab Viewer:** programa para desplegados moleculares. Puede ser usado como interface gráfica de MedChem Explorer (perfeccionamiento de fármacos), Diversity Explorer (química combinatoria) y Gene Explorer (bioinformática). Este programa da una calidad de desplegado muy buena que puede servir para publicaciones o representaciones. Se puede usar para crear animaciones con la ayuda de ViewerPro y tiene un lenguaje de script para diferentes tareas. Su costo es libre, individual o de producción. Las plataformas son PC o Mac. La compañía que la vende es Molecular Simulations, Inc. http://www.msi.com/viewer

26- Otros programas son: CHEOPS, gNMR, MedChem Explorer, POLYRATE, QCPE, SynTree, YAeHMOP.

Computational Chemistry: A Practical Guide for Applying Techniques to Real World Problems. David Young. Wiley Interscience. 2001

Para una mayor información sobre los programas consultar la página:
http://www.ccl.net/cca/documents/chamotlabs/Software.shtml

Para una mayor información sobre los acrónimos y los métodos consultar las siguientes páginas:
http://www.ccl.net/cca/documents/chamotlabs/Acronyms.shtml
http://www.ccl.net/cca/documents/chamotlabs/Methods.shtml
http://www.chamotlabs.com/Freebies/ModelRef.html

APENDICE C

HISTORIA

Año	Publicaciones	Software y Hardware
1925	Heisenberg publica su primer artículo sobre mecánica cuántica. (Z. Phys., 1925, 33, 879).	
1926	Schrödinger publica su primer artículo sobre la teoría de la mecánica cuántica. (Ann. Phys., 1926, 79, 361).	
	Aproximación de Born-Oppenheimer. (M Born y R. Oppenheimer, Ann. D. Phys. 84, 427, 1927). Slater propone sus orbitales. (J. C. Slater, Phys. Rev., 36, 57, 1930).	
1931	La teoría del electrón pi es postulado por Hückel. (Z. Phys., 1931, 70, 204).	
1934	Moller y Plesset proponen la teoría de perturbaciones MP.	
1943		Se construye la primera computadora, la ENIAC (Electronica Numerical Integrator and Computer).
1947		Se inventan los transistores en los laboratorios Bell.
1950	Roothaan publica la descripción del método LCAO-OM-SCF. Boys propone el uso de funciones base tipo gaussiana en lugar de las de tipo slater. (S.F. Boys, Proc. R. Soc. (London) A, 200, 542, 1950).	Se desarrolló la EDVAC (Electronic Discrete Variable Automatic Computer) que permitió el uso del concepto de programa almacenado. El desarrollo de la máquina de Von Neumann.
1951		La primera UNIVAC (Universal Automatic Computer) es comercializada.
1953	Se publican los detalles de la teoría PPP (Pariser-Pople-Parr). (J. Chem. Phys., 1953, 21, 466; J. Chem Phys., 1953, 21, 767). Metropolis y sus ayudantes describen la aplicación de la simulación de monte carlo para problemas de fisicoquímica. (J. Chem. Phys., 1953, 1087, 21).	
1954		Se construye la TRADIC (Transistorized Digital Computer), primera computadora con transistores.
1955	Primer cálculo all-electron de un sistema diferente al hidrógeno: el N_2 hecho por Scherr. Dicho cálculo tardó dos años.	
1956		Se desarrollaron los primeros lenguajes de alto nivel→Fortran.
1957	Pople publica los detalles de la aplicación de los métodos de SC-MO para los electrones pi. (J. Phys. Chem., 1957, 61, 6). B. J. Alder y T. E. Wainwright desarrollan la dinámica	

	molecular (J. Chem. Phys., 27, 1208).	
1958	Coester y Kümmel introducen el método de clusters acoplados.	El pirmer circuito integrado es desarrollado por Jack Kilby. La ARPA es formada. (Advanced Research Projects Agency).
1960	Pople publica su trabajo para el uso de UHF. (J. A. Pople y R.K. Nesbet, J. Chem. Phys. 22, 571, 1960) Roothaan publica su trabajo para el uso de RHF. (C. C. J. Roothaan, Rev. Mod. Phys. 32, 179, 1960).	
1963		Es formada la QCPE (Quantum Chemistry Program Exchange) para distribuir códigos de química cuántica. (Hall, K.; and Prosser, F.; Rev. Comp. Chem., 1963, 5:33)
1964	Hohenberg y Kohn demuestran su teorema surgiendo así la teoría del funcional de la densidad para la química. (P. Hohenberg, W. Kohn, Phys, Rev., 1958, 136, B864). Hansch y Fujita describen una nueva aproximación para analizar las acciones de los fármacos: QSAR (Quantitative Structure Activity Relationship). (Hansch, C y Fujita, T., J. Amer. Chem. Soc., 1964, 86, 1616).	IBM anuncia el sistema/360 y la DEC la PDP-8, primeras computadoras comerciales con transistores
1965	Kohn y Sham describen su método dentro de DFT (W. Kohn, L. J. Sham, Phys, Rev., 1965, 140, A1133). Pople publica el método CNDO y después el NDDO (Pople, J. A.; Santry, D.P.; Segal, G.A.; J. Chem. Phys., 1965, 43: S129). El programa de información química, es iniciado en el Chemical Abstracts Service.	
1966		Se crea el estándar Fortran IV.
1967	El método INDO es publicado por Pople, Beveridge y Dobosh.	El primer artículo proponiendo la ARPANET es publicado por Larry Roberts de la ARPA.
1969	Levitt y Lifson reportan el uso de campos de fuerzas para refirnar la conformación de proteínas derivados de datos experimentales. (Michael Levitt y Shneior Lifson; J. Mol. Biol, 1969, 46, 269-279). Se publican las reglas de Woodward-Hoffmann. (R.B. Woodward y R. Hoffmann, Angew. Chem. Intern. Ed., 1969, 8, 781).	La ARPANET es creada uniendo computadoras de Stanford, UCLA, UCSB y UTA. Dennis Ritchie y Brian Kernighan crean C en los laboratorios Bell. Ken Thompson, Dennis Ritchie y Joseph Ossanna desarrollan el sistema operativo UNIX para la DEC PDP-7.
1970	Warshel y Lifson publican la descripción del campo de fuerzas consistentes. (Warshal, A.; Lifson, S.; J. Chem. Phys., 1970, 53: 582-594).	Se desarrolla el lenguaje Prolog.
1971	Hendrickson describe un programa de síntesis assitido por computadora. (Hendrickson, J. B., J. Amer. Chem. Soc., 1971, 93, 6847).	Intel desarrolla el microprocesador 4004. El microprocesador había nacido.
1972		Ray Tomlinson describe lo que se convertirá en el correo electrónico (E-mail). Intel desarrolló el procesador 8008, el primer microprocesador de 8 bits.

1973	N. L. Allinger describe el modelaje de hidrocarburos con el nuevo campo de fuerzas MM1. (Allinger, N.L.,; Sprague, J.T.; J. Amer. Chem. Soc., 1973, 95:3893).	Robert Metcalfe recibe su Ph. D. de la Universidad de Harvard por su tesis donde describe la Ethernet.
1974	Clementi y Roetti publican los cálculos de Hartree-Fock para el estado fundamental y algunos excitados de los primeros 54 elementos de la tabla periódica. (E. Clementi, C. Roetti, At. Data Nucl. Data Tables, 1974, 14, 177). Robert Langridge publica el uso de visualizadores en 3-D para estructuras químicas. (Langridge, R.; Fed. Proc. Fed. Am. Soc. Exp. Biol., 1974, 33:2332).	Vint Cerf y Robert Kahn desarrollan el concepto de conección de computadoras mediante redes en la Internet y desarrollan el TCP (Transmission Control Protocol).
1975	Pople y Bartlett aplican por primera vez la teoría MP. Dewar publica los detalles del método MINDO. (R. C. Bingham, M.J.S. Dewar, D.H. Lo, J. AM. Chem. Soc. 1975, 97, 1285-1307). Dewar publica los detalles del método MINDO/3. (R. C. Bingham, M.J.S. Dewar y D.H. Lo, J. AM. Chem. Soc., 1975, 97, 1311).	Microsoft es fundado por Bill Gates y Paul Allen.
1976	N. L. Allinger libera el programa MM1 a través de QCPE.	
1977	N. L. Allinger publica la descipción del programa MM2 (Allinger, N.L.; J. Amer. Chem. Soc., 1977, 99:8127) Dewar publica los detalles del método MNDO. (Dewar, M.J.S.; Thiel, W.; J. Amer. Chem. Soc., 1977, 99:4899). Anthony Hopfinger publica la descripción del programa CAMSEQ (precursor de Chemlab). (Potenzone, R. Jr.; Cavicchi, E.; Weintraub, H. J. R.; Hopfinger, A. J.; J. Comput. Chem., 1977, 1: 187). Martin Karplus publica el primer studio de dinámica molecular para una proteina. (McCammon, J.A.,; Gelin, B.R.; Karplus, M.; Nature, 1977 267:585-590).	Se crea Fortran77
1978	Pople aplica MP3 y MP4 (R. Krishnan, J. A. Pople, Int. J. Quantum Chem., 1978, 14, 91).	DEC (Digital Equipment Corporation) introduce la computadora VAX 11/780. La NRCC (Nacional Resource for Computation in Chemistry) es establecida en el laboratorio nacional de Lawrence Berkeley.
1979	Martin Karplus describe un nuevo campo de fuerzas para proteinas. (Gelin, B.R.; Karplus, M.; Biochemistry, 1979,18:1256).	
1980	MM2/MMP2 con el campo de fuerzas 1977 es cedido a QCPE por Allinger. El primer número del Journal of Computational Chemistry es publicado.	GAMESS (General Atomic and Molecular Electronic Structure System) es desarrollado en NRCC por Michel Dupuis.
1981	Meter Kollman publica la descripción del campo de fuerza AMBER para cálculos de proteina/ADN. (Weiner, P.K.; Kollman, P.A.,; J. Comp. Chem., 1981,	IBM introduce su PC al mercado junto con el MS-DOS.. Se establece el CommonLISP.

	2:287-303).	
1982	El algoritmo del programa DOCK para el acoplamiento de pequeñas moléculas a los receptores es publicado por Irwin Kuntz. (Kuntz, I.D.; Blaney, J.M.; Oatley, S.J.; Langridge, R.; Ferrin, T.E.; J. Mol. Biol., 1982,161,269).	Intel desarrolla la 80286.
1983	Martin Karplus publica la descripción del programa CHARMM (Brooks, B.R., Bruccoleri, R.E.; Olafson, B.D.; States, D. J., Swaminathan, S. y Karplus, M.; J. Comp. Chem., 1983,4:187-217). Michael Connolly publica la descripción del programa para calcular y desplegar superficies de solventes de proteinas y ácidos nucleicos. (Connolly, Michael L.; Science, 1983, 221: 709-713).	El CD es lanzado al mercado.
1984	Meter Kollman publica la descripción del programa AMBER (Weiner, S.J., Kollman P.A., Case, D.A., Singh U. C., Ghio C., Alagona G, Profeta, S., Weiner, P; J. Amer. Chem. Soc., 1984, 106: 765-784).	Jon Postel's Domain Name System (DNS) es puesta en linea. La Macintosh es anunciada. Se desarrolla C++ y C concurrente.
1985	Dewar publica AM1. (M.J.S. Dewar, E.G. Zoebisch, E.F. Healy y J.J.P.Stewart, J. Am. Chem. Soc., 1985, 107, 3902). Parinello describe su método de dinámica molecular ab initio. (R. Car y M. Parinello, Phys. Rev. Lett., 1985, 55, 2741).	Intel comercializa el microprocesador 80386, microprocesador de 32 bits.
1987	El Journal of Computer-Aided Molecular Design es publicada.	Larry Wall realiza un nuevo lenguaje de programación PERL (Practical Extraction and Report Language).
1988		El primer gran virus que ataca la internet infecta 6000 computadoras de la milicia en EUA.
1989	Allinger publica la primer descripción del programa MM3 (Allinger N.L., Yuh Y.J., Lii, J.H.; J. Am. Chem. Soc., 1989, 111:8551,8566,8576). Stewart publica el método PM3 (J.J.P. Stewart, J. Comput. Chem. 1989, 10, 209, 221).	Intel comercializa el microprocesador 80486
1990		Se crea el ANSI C y Fortran90.
1991	Morokuma desarrolla varios métodos QM/MM para diversos estudios en química organometálica.	La CERN anuncia la creación del protocolo WWW. El programa Gopher es realizado es la universidad de Minnesota.
1993	Dewar publica el método SAM1 (M.J.S. Dewar, C. Jie y G. Yu, Tetrahedron, 1993, 23, 5003).	El Instituto de Supercómputo de la Universidad de Illinois (NCSA) crea Mosaic. Intel comercializa el microprocesador Pentium el primero de 32 bits para PC.
1994		Netscape Communications es fundada y desarrolla su Netscape Navigator. Se construye el primer cluster por Thomas Sterling y Don Becker en CESDIS (Center of Excellence in Space Data and Information Sciencies).
1995		Sun lanza Java. Netscape y Sun desarrollan JavaScript.

		Microsoft desarrolla el Internet Explorer.
1996	N.L. Allinger publica la primer descripción de MM4 (Allinger N.L., Chen K.S., Lii J.H., Nevins N.; J. Comp. Chem., 1996, 17: 642,669,695,730). Thiel y Voityuk publican el método MNDO/d. (W. Thiel y A.A. Voityuk, J. Am. Chem. Soc., 1996, 100, 616).	Se establece el ANSI C++.
1997		Se crea el estándar para el DVD. Intel comercializa el microprocesador Pentium II.
1998	El premio Nobel de química fue compartido por Walter Kohn y John A. Pople por sus aportaciones a la química cuántica computacional y donde el comité remarcó que la química cuántica computacional estaba "revolucionando el mundo de la química".	
1999	Se publica el método de pseudoenlace QM/MM (Y. Zhang, T.S. Lee y W. Yang, J. Chem. Phys. 1999, 110, 46).	Aparece Napster. Se comercializa la Pentium III. Nace Gridware (antes Genias software y fundado por el Dr. Wolfgang Gentzsch) y la comunidad de grid desarrollan proyectos como Globos, Legion, Punch y Cactus para los primeros Grids.
2000		

http://www.netsci.org/Science/Compchem/feature17a.html

The development of Computational Chemistry in the United States. John D. Bolcer and Robert B. Hermann. Reviews in Computational Chemistry, Volume V. Kenny B. Lipkowitz and Donald B. Boyd. VCH Publishers, Inc. 1994.

APENDICE D

ACRÓNIMOS

AIM→Atoms in molecules. Técnica de análisis de población.
AM1→Austin model 1. Método semiempírico.
AMBER→Assisted Model Building With Energy Refinement. Campo de fuerzas para la mecánica molecular.
ANO→Atomic Natural Orbital. Técnica para obtener bases con contracción general.
APW→Augmented plane wave. Método de cómputo para estructuras de bandas.
B3LYP→Becke 3 term, Lee Yang, Parr. Funcional híbrido.
B96→Becke 1996. Funcional de gradiente corregido.
BLYP→Becke, Lee, Yang, Parr. Funcional de gradiente corregido.
BSSE→Basis Set Superposition Error. Error introducido al usar un conjunto de bases incompleto centrado en los núcleos.
CASSCF→Complete Active-Space Self-Consistent Field Method. Método MCSCF.
CBS→Complete Basis Set. Método compuesto.
CC→Coupled Cluster. Método ab initio de correlación.
CCSD→Coupled Cluster Simple and Double excitation. Método ab initio de correlación.
CCSDT→Coupled Cluster Simple, Double and Triple excitation. Método ab initio de correlación.
CCSD(T)→Coupled Cluster Simple, Double and Triple excitation. Método ab initio de correlación que introduce la triple excitación perturbativamente.
CFF→Consistet Force Field. Campo de fuerzas para la mecánica molecular.
CFMM→Continuous Fast Multipole Method. Método para cálculos DFT rápidos para moléculas grandes.
CHARMM→Chemistry at Harvard Macromolecular Mechanics. Campo de fuerzas para la mecánica molecular.
CHEAT→Carbohydrate Hydroxyls represented by External Atoms. Campo de fuerzas para la mecánica molecular.
CI→Configuration Interaction. Método ab initio de correlación.
CID→Configuration Interaction Double excitation. Método ab initio de correlación.
CIS→Configuration Interaction Single excitation. Método ab initio de correlación.
CISD→Configuration Interaction Single and Double excitation. Método ab initio de correlación.
CISDT→Configuration Interaction Single, Double and Triple excitation. Método ab initio de correlación.
CISDTQ→Configuration Interaction Single, Double, Triple and Quadruple excitation. Método ab initio de correlación.
CNDO→Complete Neglect of Differential Overlap. Método semiempírico.
CVT→Canonical Variational Theory. Teoría variacional del estado de transición.
DFP→Davidson-Fletcher-Powell. Algoritmo de optimización de la geometría.
DFT→Density Functional Theory. Método computacional basado en la densidad electrónica total.
DHF→Dirac-Hartree-Fock. Método relativista ab initio.
DIIS→Direct Inversion of the Iterativa Subspace. Algoritmo usado para mejorar la convergencia del SCF.
DIM→Diatomics-in-molecules. Método semiempírico usado para representar PES.
DM→Direct Minimization. Algoritmo usado para forzar que los cálculos SCF convergan.

DREIDING→Campo de fuerzas para la mecánica molecular.
ECEPP→Empirical Conformational Energy Program for Peptides. Campo de fuerzas para la mecánica molecular.
ECP→Effective Core Potential. Potencial que representa los electrones de core en cálculos ab initio.
EF→Eigenvector Following. Algoritmo para la optimización de geometría.
EFF→Empirical Force Field. Campo de fuerzas para la mecánica molecular.
Fenske-Hall→Método semiempírico.
FP→Fletcher-Powell. Algoritmo para optimizar la geometría.
FMM→Fast Multipole Method. Método para cálculos rápidos DFT en moléculas grandes.
G1,G2,G3→Gaussian theory. Métodos compuestos que extrapolan los resultados de cálculos ab initio a una estimación de la energía exacta.
G96→Gill 1996. Método DFT.
GTO→Gaussian-type orbital. Función matemática que describe la función de onda.
GVB→Generalized Valence Bond. Método ab initio.
GROMOS→Gronigen Molecular Simulation. Campo de fuerzas para la mecánica molecular.
HF→Hartree-Fock. Método ab initio.
HFS→Hartree-Fock-Slater. Método DFT.
Hückel→Uno de los métodos semiempíricos más simples.
ICVT→Improved Canonical Variational Theory. Técnica del estado de transición variacional.
INDO→Intermediate Neglect of Differential Overlap. Método semiempírico.
IRC→Intrinsic Reaction Coordinate. Ruta de mínima energía entre los reactantes y los productos.
LCAO→Linear Combination of Atomic Orbitals. Construcción de la función de onda mediante combinación lineal de funciones de base atómicas.
LDA→Local Density Approximation. Aproximación usada en métodos DFT.
Level shifting→Algoritmo usado para mejorar la convergencia del SCF.
LMP2→Local Second-Order Moller-Plesset. Técnica perturbativa.
LSDA→Local Spin-Density Approximation. Aproximación usada en métodos DFT para sistemas con capa abierta.
MBPT→Many Body Perturbation Theory. Teoría de perturbaciones.
MCSCF→Multiconfigurational Self-Consistent Field. Método ab initio de correlación.
MEP→Minimum-Energy Path. Ruta de mínima energía entre los reactantes y los productos.
MIM→Molecules-in-Molecules. Método semiempírico usado para representar PES.
MINDO→Modified Intermediate Neglect of Differential Overlap. Método semiempírico.
MMFF→Merck Molecular Force Field. Campo de fuerzas para la mecánica molecular.
MMn→MM1, MM2, MM3, MM4, MMX, MM+. Nombres de las familias de similares campos de fuerzas para la mecánica molecular.
MMOM→Many Methods of Morokuma. Métodos QM/MM
MNDO→Modified Neglect of Diatomic Overlap. Método semiempírico.
MPn→Moller-Plesset. Método ab initio de correlación basado en la teoría de perturbación.
MOMEC→Campo de fuerzas para la mecánica molecular con un término semiempírico para la descripción de metales de transición.
MC→Monte Carlo. Técnica de simulación que incorpora movimientos aleatorios de átomos y moléculas.
MRCI→Multireference Configuration Interaction. Método ab initio de correlación.
Newton-Raphson→Algoritmo para la optimización de la energía.

OPW→Orthogonalized Plane Wave. Método de cómputo para la estructura de bandas.
P86→Perdew 1986. Método DFT de gradiente corregido.
PES→Potential Energy Surface. Espacio de energías correspondiente a la localización de los núcleos ignorando los movimientos de vibración.
PLS→Partial Least-Squares. Algoritmo usado para cálculos 3D QSAR.
PM3→Parameterization Method 3. Método semiempírico.
PPP→Pariser-Parr-Pople. Método semiempírico simple.
PRDDO→Partial Retention of Diatomic Differential Overlap. Método semiempírico.
PW91→Perdew, Wang 1991. Método DFT de gradiente corregido.
QCI→Quadratic Configuration Interaction. Método ab initio de correlación.
QCISD→Quadratic Configuration Interaction with Single and Double excitation.
QCISD(T)→Quadratic Configuration Interaction with Single, Double and Triple excitation. Método ab initio de correlación que introduce la tercera excitación perturbativamente.
QMC→Quantum Monte Carlo. Método ab initio de correlación explícita.
QM/MM→Técnica donde se combinan los cálculos basado en orbitales y basados en mecánica molecular.
QSAR→Quantitative Structure-Activity Relationship. Técnica para calcular propiedades químicas, principalmente para actividad biológica.
QSPR→Quantitative Structure-Property Relationship. Técnica para calcular propiedades químicas.
Random flight→Técnica para la simulación de polimeros.
RECP→Relativistic Effective Core Potencial. Potencial que representa los electrones de core en cálculos ab initio y que incluye efectos relativistas.
RIS→Rotational Isomeric State. Técnica para simular polimeros.
RHF→Restricted Hartree-Fock. Método ab initio para sistemas singuletes.
ROHF→Restricted Open-Shell Hartree-Fock. Método ab initio para sistemas de capa abierta.
SAC→Symmetry-Adapted Cluster. Una variación del método de clusters acoplados.
SACM→Statistical Adiabatic Channel Model. Método para calcular constantes de velocidad.
SAM1→Semi-ab initio Method 1. Método semiempírico.
SCF→Self-Consistent Field. Procedimiento para resolver las ecuaciones de Hartree-Fock.
SINDO→Symmetrically Orthogonalized intermediate neglect of differential overlap. Método semiempírico.
STO→Slater Typer Orbital. Función matemática que describe la función de onda.
TDHF→Time Dependent Hartree-Fock. Método ab initio usado para calcular propiedades opticas no lineales.
TST→Transition State Theory. Método para calcular constantes de velocidad.
UHF→Unrestricted Hartree-Fock.
UFF→Universal Force Field. Campo de fuerzas para la mecánica molecular.
VTST→Variational Transition State Theory. Método para predecir constantes de velocidad.
VWN→Vosko, Wilks y Nusair. Método DFT.
Xα→Método DFT.
YETI→Campo de fuerzas para la mecánica molecular.
ZINDO→Zerner's Intermediate Neglect of Differential Overlap, sinónimo de INDO/S. Método semiempírico.