El problema de SAT no tiene un algoritmo de resolución eficiente, por tanto, ningún problema de la clase NP tiene un algoritmo de resolución eficiente con computación discreta

Carlos Barrón Romero

Universidad Autónoma Metropolitana, unidad Azcapotzalco

La complejidad de la resolución de los problemas NP es clasificado como uno de los más trascendentes del siglo XX por la relevancia de dar solución a los problemas clasificados como NP. Tales problemas tienen gran importancia e influencia en muchas aplicaciones industriales, tecnológicas y sociales. Por ejemplo, la planificación de rutas de distribución de entrega de productos o de unas vacaciones recorriendo un circuito de ciudades, tienen relación con el problema NP conocido como el problema del agente viajero [D. Applegate and Chvatal, 1998], [Applegate et al., 2007], [Barrón-Romero, 2011], problemas de como acomodar sus bienes o productos en los almacenes de una compañía o industria, se relacionan con el problema NP conocido como el problema de la mochila, el diseño molecular y el estudio de estructuras de partículas, se relacionan con el problema de cúmulos de partículas bajo un buen potencial a pares [Barrón-Romero et al., 1996], [Barrón-Romero et al., 1997], [Barrón-Romero et al., 1999], [Romero et al., 1999b], [Romero et al., 1999a] y [Barrón-Romero, 2005].

Ocurre también que hay infinidad de problemas de la ciencia, la tecnología, el entretenimiento, las comunicaciones y de la industria que tienen a un problema de la familia NP como parte de un problema mayor, por lo que los ingenieros, científicos, administradores, economistas, meteorólogos, productores de cine e incluso los niños (para sus videojuegos) demandan cada vez mayor rendimiento para los supercomputadores, las computadoras, las laptops, las tablets, los teléfonos, los celulares, las consolas de juego, los equipos de producción multimedia, las salas de cine, las redes de internet y de telefonía.

Las industrias de electrónicos son felices con tales demandas y entre ellas tiene feroces competencias para ofrecer al público componentes más eficientes, sin embargo el daño al medio ambiente con el desecho de los equipos obsoletos o fuera de mnoda desata una controversia a nivel mundial de lo que persiguen sin tomar en cuenta los efectos. Incluso hay una gran falta de conciencia ecológica, en compañías de desarrollo de software, de tecnología y en las investigaciones de las universidades de las consecuencias al medio ambiente por el desarrollo de la tecnología de cómputo, la información, de las comunicaciones. Hay una negación de como enfrentar el daño al medio ambiente y el desperdicio de materiales y energía por obsolescencia o modas de los fabricantes de equipos. Por ejemplo, Windows y Apple sugieren con cada supuesta mejora de sus sistemas operativos la compra de nuevos equipos, en este mismo rubro de supuestas mejoras y necesidad de cambio de equipos también participan los fabricantes de teléfonos celulares (Nokia, Alcatel, LG, Sony, etc.), los fabricantes de consolas de juegos, sin faltar, los dos principales fabricantes de procesadores de cómputo Intel y AMD. Huelga decir, que el impacto económico, científico, tecnológico y de las ganancias relacionadas con la computación es una de las razones por las cuales Estados Unidos lo considera el quinto pilar para sostener nuestra civilización, su seguridad nacional y su supremacía como pais líder del mundo [Benioff and Lazowska, 2005]. La alternativa es crear equipos con compatibilidad y adaptabilidad, para que en lugar de comprar un equipo nuevo, se cambien partes. Tal solución no es una panacea, pero disminuiría los cambios totales de equipo por obsolecencia o moda.

A un lado de las controversias, el quid del asunto es que hay pocas personas en el mundo, por la falta de inversiones y condiciones para hacer teoría en Computación que entiendan y además investiguen como resolver el problema de determinar si existe un algoritmo eficiente, es decir capaz de resolver cualquier problema NP por medio de un algoritmo programado para una computadora o supercomputadora en poco tiempo. El asunto tiene intereses tanto teóricos, como personales acerca de que tipo de cientificos son los que deben resolverlo. Mención especial, es la propuesta del Instituto Clay de la ciudad de Nueva York que ofrece un millón de dólares por la respuesta de si es o no es posible resolver un problema en NP en un tiempo eficiente.

1 Resolver un problema por computadora

El calcular una solución de un problema cualesquiera por medio de una computadora, es un problema teórico de la Ciencia de la Computación, cuando decimos computadora o supercomputadora, en realidad nos referimos a cualesquiera de los modelos de máquinas de computación: Máquina de Turing, Máquina de Von Newman, Máquina de Post, Máquinas de arquitectura No Von Newman, Máquina de Harvard, etcétera. En realidad no importa ni la máquina físicamente, ni la marca de CPU (Intel, HP, Cray, AMD, etc.) sino el que

el programa o algoritmo se entienda y seamos capaces de contar los pasos o iteraciones que le toma resolver o llegar a la solución.

Mas precisamente, supongamos que debemos sumar 20 números con una calculadora de mano de contabilidad. Copiamos el primer número, oprimimos sumar, copiamos el segundo número, oprimimos sumar hasta llegar al último número y oprimimos la tecla de totalizar (que en muchas sumadoras de escritorio es el mismo de sumar). Llegar al resultado nos tomo 20 pasos y si cada paso lo realizamos en medio segundo, nos tardamos 10 segundos en realizar la suma de 20 números. Se puede ser muy puntilloso, y contabilizar el proceso con más detalle, por ejemplo contabilizar el leer cada dígito y el tiempo en oprimir la tecla correspondiente al copiarlo a la calculadora. De esta forma la complejidad de la suma de 20 números es el número de digitos de los numeros al copiarlos más el tiempo de oprimir 19 veces la tecla de suma y una vez la tecla de totalizar. No hay gran perdida conceptualmente, si tomamos como una operación o un paso de nuestro procedimiento como el copiar el número a sumar y oprimir la tecla de opración, sin tomar en cuenta la longitud de dichos números. Es decir, suponemos que los números a sumar tiene similar y razonable longitud.

En el ejemplo anterior, convertimos a tiempo los 20 pasos que tomo la suma multiplicando 20 por nuestra velocidad de medio segundo (constante) para realizar la copia de un número y el oprimir la tecla de suma o de total. Por muy sofisticada que sea una computadora, lo mismo que hicimos para medir el tiempo que nos toma hacer las sumas en forma manual, es posible hacerlo para cualquier máquina o modelo de cómputo.

En el ejemplo de la suma de números, si en lugar de 20 números vamos a sumar n, lo que podemos decir (sin tomar en cuenta que nos podemos cansar o aburrir de hacer sumas) es que la complejidad de obtener la suma de n números es kn, donde $k = 1/2$ segundo por operación.

Si hacemos caso omiso de k que es una constante (irrelevante desde el punto de vista de que lo importante son los n pasos que se deben de realizar), decimos que la complejidad de las suma de n números es n, y en símbolos lo escribimos como $\mathbf{O}(n)$. Notemos que n es el número de datos de entrada y que el número de operaciones por realizar está en función de n como un polinomio de la forma n^e, donde el exponente e es uno.

2 Clasificación de problemas por el número de pasos para resolverlos

La clase P es la de los problemas que se resuelven en tiempo polinomial, es decir su complejidad es de tipo un polinomio y lo escribimos como $\mathbf{O}(n^e)$, donde el exponente e es un entero pequeño. A los problemas de esta clase se les dice los problemas resolubles, ya que teóricamente se obtiene su solución o respuesta en poco tiempo. El problema que describimos de sumar n números de la sección anterior es de la clase polinomial.

Y los problemas que mencionamos al inicio, como el problema del agente viajero y de la mochila son de la clase NP.

La mayoría de las personas, aún con falta de conocimientos teóricos de computación y matemáticas, entiende el concepto de resolución por similitud.

Por ejemplo, comprar un auto y una casa. La manera de proceder es buscar la información, conocer los bienes. Tomar la decisión de adquirir cualquiera de estos objetos estará basada en razones objetivas y subjetivas que bajo la finalidad de obtener la mayor satisfacción personal y costo-beneficio, la gran mayoría de las personas encontraran un solo método (o sea un algoritmo) que resuelva ambos a su gusto.

De la misma forma, un problema NP, por ejemplo, el problema de satisfacción de una fórmula normal conjuntiva (SAT) y el problema de la mochila) se pueden resolver en forma similar, porque todos los problemas de la clase NP tienen dos propiedades básicas:

1. Todos los miembros de NP, pueden traducir sus problemas entre ellos mediante un algoritmo de tiempo polinomial.

2. Todo miembro de NP tiene un algoritmo eficiente que determina si un candidato o entrada de datos es o no solución del problema.

Bajo estas dos propiedades, supongamos que existe un algoritmo S de tiempo polinomial capaz de resolver cualquier problema SAT.

Entonces, cualquier problema de la mochila se traduce a un problema SAT, el traducido se resuelve con el algoritmo S y este SAT con su solución se traduce de nuevo al problema de la mochila. Regresamos con un problema de la mochila resuelto. La complejidad de resolver el problema dado de la mochila, es debida a la suma de la complejidad del algoritmo S y la complejidad de realizar dos traducciones, lo cual bajo la

hipótesis de la existencia de S, corresponde a la suma de polinomios, es decir la complejidad total es un polinomio y por tanto la complejidad de resolver el problema de la mochila es polinomial.

En otras palabras, de existir tal algoritmo S para alguno de los problemas NP, en particular, digamos para SAT, este podrá resolver cualquier problema de NP en tiempo polinomial. Lo anterior significaría que la clase NP es equivalente a la clase P, en el sentido de que resolver cualquier problema NP se puede realizar en corto plazo. Más aún, cualquiera de los importantes problemas o subproblemas mencionados al principio, no necesitaría de un supercomputador o de la mejor computadora porque teóricamente lo que se necesita es traducirlo y resolverlo con el algoritmo S. A los magnates de la industria de las partes y de las computadoras no les alegraría tal resultado.

Una exploración y clasificación de los problemas de la clase NP, permite dividirlos y clasificarlos en problemas en suaves (soft) y duros (hard). Los problemas suaves son aquellos en donde se toma una pregunta de decisión por ejemplo dado un problema SAT, ¿este tiene solución? Un problema duro es aquel donde se trata de determinar por ejemplo un circuito cerrado que recorra todas las ciudades con un costo mínimo (problema del agente viajero).

Podemos establecer la siguiente relación desde el punto de la complejidad NP-Soft $\preceq$ NP-Hard, que significa que la complejidad de los algoritmos NP-Soft es menor a la de los algoritmos NP-Hard.

En el artículo [Barrón-Romero, 2010] se presenta un estudio de diferentes problemas NP, se introduce el procedimiento de reducción polinomial y presenta un problema que da un marco general a los problemas NP, se define el problema de GAP (*General Assign Problem*).

La idea del procedimiento de reducción polinomial es porque para resolver un problema NP de forma eficiente el número máximo de operaciones debe ser polinomial. La reducción significa la reducción del espacio de candidatos a ser la solución. Dicha transformación no cambia el problema en sí, sino que propiamente trata de descartar muchos posibles casos que no son o que no están cerca de la solución. Por ejemplo, para el problema del agente viajero, supongamos una serie de ciudades ubicadas alrededor de un lago circular. Todos los circuitos para recorrer las ciudades incluyen aquellos en donde se puede cruzar en bote de una ciudad de la costa a otra. Una reducción seria solo considerar recorridos sin cruzar el lago. Tal reducción es valida si se considera que el recorrido entre las ciudades debe ser en un tiempo mínimo y si el ir en bote es más lento que el ir en auto. Sorprendentemente, tal reducción elimina muchas alternativas y el espacio de las posibles soluciones se reduce muchísimo. Sin embargo, las condiciones de tal reducción no son generales desde el punto de vista de un problema GAP, sino particulares a la distribución de ciudades alrededor de un lago circular de un tipo particular de problemas del agente viajero.

Cuando hay una reducción del espacio de candidatos a un número polinomial, el caso de verificar cual de ellos es la solución, es polinomial, ya que los siendo polinomial el número de candidatos, la verificación de la solución corresponde a una suma de tiempos polinomiales, es decir la verificación corresponde a una suma de tiempos polinomiales y, por tanto, es de tiempo polinomial. Esto es muy diferente de buscar o construir la solución cuando no hay reducción del espacio de candidatos porque al no poder garantizar por la falta de propiedades generales, no es posible verificar que candidato o grupo candidatos representan la solución respecto de los otros candidatos del espacio de candidatos, por lo que se tendrían que realizar verificaciones del orden del tamaño del espacio de candidatos, lo cual lleva a número exponencial de verificaciones (este es el resultado de la proposición 6.12 de [Barrón-Romero, 2010]: *An arbitrary and large GAP_n of the NP-Class has not an polynomial algorithm for checking their solution.* Lo anterior conduce a que no hay forma de verificar eficientemente cuando se tiene una solución de un problema NP-Hard.

El asunto de la falta de entendimiento en la formulación de algoritmos y las propiedades de un problema para ser usadas dentro de un algoritmo para resolver problemas NP se presento en el artículo personal [Barrón-Romero, 2015b]. Este artículo es controversial, explica que se da gran importancia al juicio de los investigadores afines o pares, sin embargo, el dilema de si NP es no P, tiene diferentes grupos de interés y desafortunadamente las personas se ubican en matemáticas o en computación pero no en una forma adecuada y crítica por el progreso de la ciencia, sino bajo intereses de grupo o de una bandera de conveniencia (ad-hoc), porque en mi opinión es muy difícil resolver los problemas mediante algoritmos basados en propiedades. El artículo muestra ejemplos de propiedades que sirven para formular algoritmos eficientes, que sin embargo no son propiedades generales. Por ejemplo, para el problema del agente viajero bajo el objetivo de recorrer un circuito Hamiltoneano que minimiza la distancia Euclidiana, toda solución debe ser una curva simple de Jordan. Esta condición suficiente es bien conocida pero no ha sido utilizada en los algoritmos de resolución del problema del agente viajero. En [Barrón-Romero, 2015b] se muestra un algoritmo para usarla en la detrerminación de una situación de paro. La situación de paro es detener la determinación del recorrido del circuito de ciudades cuando se logra pintar su región dividida por el recorrido de las ciudades con exactamente dos colores.

3 El problema SAT

Los problemas SAT consisten en sistemas de formulas lógicas en forma conjuntiva. Por ejemplo: $(x_1 \vee x_0) \wedge (\overline{x}_1 \vee x_0)$. Los valores que las variables pueden tomar son 0 para falso y 1 para verdadero. Una asignación puede ser $x_1 = 1$ y $x_0 = 0$. Para ver si dicha asignación satisface al sistema anterior los sustituimos y resulta: $(1 \vee 0) \wedge (0 \vee 0) = (1) \wedge (0) = 0$. que significa que no es satisfactoria para el sistema dado. Por otro lado, la asignación $x_1 = 1$ y $x_0 = 1$ si es satisfactoria porque sustituyendo se tiene: $(1 \vee 1) \wedge (0 \vee 1) = (1) \wedge (1) = 1$. El problema SAT consiste en responder si un sistema dado de formulas lógicas en forma conjuntiva tiene solución. Es decir si existe al menos una asignación de valores de 0'1 y 1's para las variables del problema tal que al sustituir en el sistema de formulas, el resultado sea 1.

Es importante notar que no basta que un programa o persona nos diga si o no, sino que la forma de creer que hay solución es probándolo con un candidato. Por otro lado y en forma similar, tampoco podemos aceptar que un SAT no tiene solución sin una prueba o demostración adecuada. El asunto es que para creer que no hay solución debería ser posible aceptarlo sin probar todos los posibles candidatos. Notemos que, el número de asignaciones o candidatos posibles es del orden de 2^n donde n es el número de variables booleanas del problema. Este número se obtiene del número de valores que cada variable puede tomar. Así la cuestión de la complejidad de resolver SAT es cuantos pasos o intentos se deben realizar para aceptar si un problema SAT tiene o no solución.

Cuando hay solución, si además se da el candidato o testigo, verificar es trivial por sustitución y pasa a segundo termino la complejidad pues se resuelve antes de que se acabe el tiempo del universo. Pero si la respuesta es no, y el problema es de digamos 1,000,000 variables, el universo se acabará pero mas vale probar las más que podamos de las $2^{1,000,000-1} = 2^{999,999}$ posibles asignaciones para estar seguros. Desafortunadamente, moriremos con la duda. El porqué del número de posibles asignaciones es por ser el tamaño del espacio de candidatos y no hay forma de reducirlo más que por un factor de 2 (por eso el exponente tiene un menos 1). Cada intento de probar un número, elimina a su complemento como se explica con detalle más adelante y en [Barrón-Romero, 2015a].

En [Barrón-Romero, 2015a] se demuestra que el problema de decidir si un problema SAT tiene o no solución es equivalente a determinar si uno o varios números dentro del intervalo de 0 a $2^n - 1$ (dos a la potencia n), donde n es el número de variables booleanas del problema SAT lo satisfacen. Es decir, se puede construir un problema SAT con 20 variables, donde la única asignación que lo hace verdadero es el número binario 10000101001010101010 de los valores entre 0 y $2^{20} - 1$ (dos a la veinte menos 1).

Tomemos cualquier persona que desconoce la solución y le damos una caja cerrada con un circuito lógico, un teclado y una pantalla. Dicha persona puede escribir cualquier numero y la caja le responde a cada número si es o no la solución. Por ejemplo, si la persona escribe 00000000000000000100, la caja responde no. ¿A los cuantos intentos podrá deducir que el 10000101001010101010 es la solución?

La respuesta es que aún llevando una lista de los intentos fallidos es extremadamente difícil adivinar que 10000101001010101010 es la solución. En este caso se elige, tener un solo número como solución, pero se puede diseñar una caja para la cual ningún número sea satisfactorio, es decir no hay solución. Dado que cualquier número puede ser la solución, $1/2^{20}$ es la probabilidad de elegir la respuesta correcta al primer intento, este número es casi cero (o sea implica una probabilidad casi nula), se trata de un evento casi imposible de ocurrir el que se pueda adivinar el número solución en un solo intento.

La probabilidad después de m intentos es $1/(2^{20} - 2m)$ que es muy similar a la del primer intento. Nuevamente se tiene que es un evento casi imposible el de adivinar el número solución con un número m de intentos.

Aquí el valor de m juega un papel crucial, si se realizan muchos intentos, o sea si m es tan grande como 2^{20}, se tiene que no es eficiente adivinar la solución ya que se realizaron un numero exponencial de 2^{20} intentos o sea un orden exponencial de intentos u operaciones de adivinar. La única forma de adivinar la solución eficientemente, dado que la solución es un número cualquiera que no esta relacionado con ningún otro, es que el número de intentos m sea pequeño con respecto a 2^{20}. Pero como se mencionó la probabilidad se mantiene casi nula luego de m intentos, siendo m un número entero razonablemente pequeño.

Por otro lado, si suponemos que existe un algoritmo S capaz de resolver un tipo de problema NP, entonces SAT se puede traducir a ese tipo. Luego entonces el algoritmo S es capaz de adivinar el número solución de un problema SAT en un número de intentos pequeño. Pero esto es imposible, ya implicaría que los números no tienen una distribución uniforme de probabilidad dado que no importa que número sea la solución, este se adivina en pocos pasos, contradiciendo que su probabilidad es del orden de $1/2^n$, o sea que la probabilidad no es uniforme o de que un solo tenga número tenga siempre una probabilidad mayor a la de los demás números binarios de entre 0 y $2^n - 1$. Esto significa que tal algoritmo puede ganar siempre cualquier Lotería en el mundo.

Lo anterior se demuestra en la proposición 18 de [Barrón-Romero, 2015a]: *NP has not property or heuris-*

tic to build an efficient algorithm. Es decir, NP no tiene un algoritmo de resolución eficiente o polinomial, y por lo tanto la clase NP no es igual a la clase P. Lo cual es una buena noticia para los fabricantes de componentes electrónicos y de computadores porque se requiere por siempre mayor poder de cómputo.

En forma equivalente, para conocer la solución de un problema SAT se requiere al menos leer toda la lista de formulas booleanas de SAT porque la solución es un numero independiente y cualquiera. Si el número de fórmulas es cercano a 2^n, entonces el costo mínimo es $2^n - 1$, o sea leer o revisar al menos la mitad de las fórmulas, lo cual no es eficiente o polinomial.

Vale la pena aclarar porque se requiere al menos revisar la mitad de las fórmulas, esto es debido a que por cada intento si este no satisface las fórmulas del problema dado, es porque esta bloqueada por su fórmula complementaria, es decir, por cada candidato fallido se eliminan dos candidatos.

Note que hemos establecido una cota mínima exponencial de la complejidad que es $\mathbf{O}(2^{n-1})$ para los problemas de tipo SAT, lo cual es una cota mínima para los problemas NP-Soft, lo cuales cumplen que NP-Soft $\preceq$ NP-Hard. Por transitividad se tiene que $\mathbf{O}(2^{n-1}) \preceq$ NP-Hard, es decir tampoco existe un algoritmo eficiente de resolución para los NP-Hard.

Este es el resultado que tenemos en la proposición 19 de [Barrón-Romero, 2015a]: *A lower bound for the complexity of* $SAT_{n\times m}$ *for the extreme case when there is no solution or only one solution is* $2^{n-1} - 1$. *Therefore,* $\mathbf{O}(2^{n-1}) \preceq$ *NP-Soft* $\preceq$ *NP-Hard.*

Comentarios finales

La computación basada en el concepto de estados discretos, no permite la construcción de las máquinas de computo que demandan los problemas científicos y tecnológicos de hoy y del futuro.

Mí articulo [Barrón-Romero, 2015a], demuestra que con una computadora basada en la computación cuántica [Barrón-Romero, 2015a], o sea en el concepto de estados superpuestos o qbits [Feynman, 1996], se obtiene el sorprendente resultado de que el algoritmo 5 descrito en [Barrón-Romero, 2015a] resuelve cualquier problema SAT en a lo más n iteraciones donde n es el número de variables booleanas, no el tamaño de los datos de entrada del problema. Es decir la reducción de la complejidad de resolver un problema SAT pasa de ser exponencial $\mathbf{O}(2^n)$ a ser $\mathbf{O}(n)$, que es la complejidad de realizar n sumas por una persona o una computadora de nuestros días. Importante notar que tal resultado implica que todo problema NP usando Computación Cuántica se podrá resolver en tiempo eficiente.

El futuro es el estudio de la de la computación cuántica para la construcción de máquinas cuánticas para la resolución de problemas vitales de ciencia y tecnología de hoy y del mañana, entender sus limites, consecuencias, complejidades y propiedades de esta novel área de la computación. Existe abundante literatura e interés al respecto y se espera pronto un mayor interes e investigación. Siendo óptimistas es posible que en este mileneo las aplicaciones y dispositivos básados en computación cuánticas se vuelvan de uso cotidiano y común para cualesquier persona.

REFERENCIAS

[Applegate et al., 2007] Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA.

[Barrón-Romero, 2005] Barrón-Romero, C. (2005). Minimum search space and efficient methods for structural cluster optimization. *arXiv*, Math-ph:0504030-v4.

[Barrón-Romero, 2010] Barrón-Romero, C. (2010). The Complexity Of The NP-Class. *arXiv*, arxiv.org/abs/1006.2218.

[Barrón-Romero, 2011] Barrón-Romero, C. (2011). The complexity of euclidian 2 dimension travelling salesman problem versus general assign problem, NP is not P. *arXiv.org*, abs/1101.0160.

[Barrón-Romero, 2015b] Barrón-Romero, C. (marzo, 2015b). Complexity and Stop Conditions for NP as General Assignment Problems, the Travel Salesman Problem in $\mathbb{R}^2$. *Página Personal: Carlos Barrón Romero.*

[Barrón-Romero, 2015a] Barrón-Romero, C. (octubre, 2015a). Classical and Quantum Algorithms for the Boolean Satisfiability Problem. *ArXiv e-prints.*

[Barrón-Romero et al., 1996] Barrón-Romero, C., Gómez, S., and Romero, D. (1996). Archimedean Polyhedron Structure Yields a Lower Energy Atomic Cluster. *Applied Mathematics Letters*, 9(5):75–78.

[Barrón-Romero et al., 1997] Barrón-Romero, C., Gómez, S., and Romero, D. (1997). Lower Energy Icosahedral Atomic Cluster with Incomplete Core. *Applied Mathematics Letters*, 10(5):25–28.

[Barrón-Romero et al., 1999] Barrón-Romero, C., Gómez, S., Romero, D., and Saavedra, A. (1999). A Genetic Algorithm for Lennard-Jones Atomic clusters. *Applied Mathematics Letters*, 12:85–90.

[Benioff and Lazowska, 2005] Benioff, M. R. and Lazowska, E. D. (2005). REPORT TO THE PRESIDENT Computational Science: Ensuring Americas Competitiveness. Technical report, National Coordination Office for Information Technology Research and Development.

[D. Applegate and Chvatal, 1998] D. Applegate, R. B. and Chvatal, V. (1998). On the solution of traveling salesman problems. In *Documenta Mathematica Journal, Journal der Deutschen Mathematiker-Vereinigung, International Congress of Mathematicians*, volume III, pages 645–656.

[Feynman, 1996] Feynman, R. P. (1996). *Feynman Lectures on Computation.* Addison-Wesley.

[Romero et al., 1999a] Romero, D., Barrón-Romero, C., and Gómez, S. (1999a). The optimal configurations of LJ atomic clusters: 148-309. In *Siam Annual Meeting*, Atlanta, GA, USA.

[Romero et al., 1999b] Romero, D., Barrón-Romero, C., and Gómez, S. (1999b). The optimal geometry of Lennard-Jones clusters: 148-309. *Computer Physics Communications*, 123:87–96.