Reduciendo el Retardamiento Crítico con Algoritmos de Clúster

Simulaciones de Física Computacional. Proyecto de Curso

Rubert Martín Pardo ▸ Facultad de Física de la Universidad de La Habana ▸ 03/01/2013

El Algoritmo de Metrópolis Monte Carlo presenta un Retardamiento Crítico cuando se utiliza para simular el Modelo de Ising en las cercanías de la Temperatura de Curie T_c= 2.23. En el presente trabajo se estudian las causas de este Retardamiento y los principales algoritmos diseñados por la Física Computacional para evadirlo. Se presentan comparaciones entre estos "Algoritmos de Clúster" y el clásico algoritmo de Metrópolis. Se puede arribar que tanto el Algoritmo de Swendsen-Wang como el de Wolff son más eficientes que Metrópolis en las proximidades de Tc dado su carácter no local.

Introducción

El Modelo de Ising bidimensional es una aproximación a un imán compuesto por $L \times L = N$ momentos magnéticos a los que se conoce como spines. Cada spin sólo podrá tener uno de dos valores $s_i = +1, -1$. La energía de este modelo puede ser aproximada por

$$E = -J \sum_{<i,j>} s_i s_j - H \sum_i s_i$$

donde J es la fuerza de interacción y H representa el campo externo que se acopla con la magnetización $M = \sum_i s_i$. La primera sumatoria sólo se realiza entre spines vecinos.

Este sistema puede encontrarse en dos fases distintas de magnetización en dependencia de la temperatura. Para bajas temperaturas, el sistema se encuentra permanentemente magnetizado (fase ferromagnética). Si la temperatura es suficientemente alta, la magnetización del sistema es cero (fase paramagnética). Existe una temperatura crítica, conocida en la literatura como *Temperatura de Curie* Tc, a la cual ocurre el cambio de fase.

En 1941, H.A. Kramers y G.H.Wannier calcularon exactamente la *Temperatura de Curie* para el modelo de Ising bidimensional:

$$\frac{k_B T_C}{J} = \frac{2}{\log(1+\sqrt{2})} \approx 2.269 \ldots$$

Luego en 1944, Lars Orsager resolvió el modelo de Ising bidimensional en el *límite termodinámico* $N \to \infty$ con $H = 0$ y demostró que la magnetización por spin:

$$m = \begin{cases} \left[1 - \left\{\sinh\left(\frac{2J}{k_B T}\right)\right\}\right]^{-\frac{4}{8}} & si\ T \le T_C \\ 0 & si\ T > T_c \end{cases}$$

Las simulaciones de Monte Carlo son ampliamente usadas para simular sistemás como el de Ising. Una simulación de Monte Carlo implica generar un subconjunto de configuraciones posibles del sistema, según una función de distribución de probabilidades. Luego los observables son calculados usando las configuraciones generadas. Para el caso del Modelo de Ising, debe utilizarse la función de distribución de Boltzmann:

$$p(s_1, s_2 \ldots s_N) = \frac{e^{-E(s_1, s_2 \ldots s_N)/k_B T}}{\sum_{configs} e^{-E/k_B T}}$$

El algoritmo más eficiente para generar configuraciones distribuidas de acuerdo al factor de Boltzmann es el descubierto por Metrópolis y otros en 1953. Aunque este algoritmo destaca por su eficiencia e ingenio, no es objetivo de este trabajo estudiarlo. Basta conocer que este algoritmo sufre de un fenómeno conocido como *Retardamiento Crítico* cuando se intenta simular el sistema en las cercanías de Tc.

Este fenómeno implica una divergencia en el cálculo de los observables (magnetización, energía, capacidad, etc.) y hace que sea necesario buscar nuevos algoritmos para calcularlos en la cercanía de Tc.

Puede demostrarse que para evadir el *Retardamiento Crítico* deben usarse algoritmos de clúster. Esto es realizar la simulación volteando grupos correlacionados de spines (clústers) entre configuración y configuración, en vez de voltear un spin por vez como con Metrópolis.

Los algoritmos de clúster son muy útiles en aplicaciones que envuelven grafos o redes. En general resultan muy interesantes de estudiar.

¿Qué es el *Retardamiento Crítico*?

Los algoritmos de Monte Carlo generan sucesivas configuraciones de spines para el modelo de Ising, pero esto no representa la evolución temporal real del sistema. Ello implica que el modelo de Ising no cuente con una dinámica.

En un sistema real, las variables dinámicas son funciones del tiempo. Se conoce como tiempo de relajación el tiempo aproximado en que el sistema llega al equilibrio. Si $A(t)$ es una magnitud que tiende al valor de equilibrio $\bar{A}$, el tiempo de relajación correspondiente a esta magnitud se define teóricamente como:

$$\tau = \frac{\int_0^\infty dt\ t[A(t) - \bar{A}]}{\int_0^\infty dt\ [A(t) - \bar{A}]}$$

Cerca de la temperatura crítica, el tiempo de relajación se torna demásiado grande. Puede demostrarse que el mismo diverge para un sistema infinito:

$$\tau \sim \frac{1}{|T - T_C|^z}$$

donde z es el exponente crítico dinámico asociado a la variable observable A. Este fenómeno es conocido como *Retardamiento Crítico*.

Este fenómeno se debe al aumento de la correlación spin-spin cuando T se aproxima a Tc. Con correlación nos referimos a cómo influye el cambio del valor de un spin (voltearlo) en los spines de su vecindad.

Se define la longitud de correlación ξ en el modelo de Ising como la distancia entre dos spines completamente independientes. El tiempo de realajación previamente definido, da igualmente una medida de otra magnitud: el tiempo de correlación τ. Esta cantidad mide el número de pasos de Monte Carlo entre dos configuraciones independientes y se comporta según:

$$\tau \sim \xi^z$$

donde z=2.1 para el algoritmo de Metrópolis, es el exponente dinámico crítico.

El algoritmo de Metrópolis es un Algoritmo local, o sea, se prueba un spin por vez. Cerca de Tc, el sistema desarrolla grandes dominios de spines altamente correlacionados, que son difíciles de romper, por lo tanto, el cambio más probable en la configuración es el movimiento de un dominio entero de spines. Este argumento sugiere que una manera de acelerar un algoritmo de Monte Carlo cerca de Tc es usando un algoritmo no-local.

Algoritmo de Swendsen-Wang

Este algoritmo, sugerido por R.H. Swendsen y J.S. Wang en 1987 se basa en identificar clústers de spines afines (con igual valor), y tratar a los mismos, como "spines gigantes", esto es, voltearlos como un todo:

1. Congelar/Eliminar enlaces:

La red de $L \times L = N$ spines con fronteras periódicas tiene 2N enlaces.

Primero debe construirse una red de enlaces como sigue:

-Si el enlace conecta spines opuestos, eliminarlo, o sea, desacoplar temporalmente ambos spines. Debe tenerse en cuenta que spines opuestos tienen una mayor energía de enlace +J si J>0, y por ende una temperatura mayor. Por lo tanto si J es grande, estamos "derritiendo" el enlace.

-Si el enlace conecta spines iguales (ambos arriba o ambos abajo), entonces elimina el enlace con una probabilidad $e^{-2J/(k_BT)}$. Notar que un par de spines iguales tiene una energia de interaccion de –J luego el cambio de nergia al cambiar uno de los spines del par es 2J. Decimos que los spines que pasan esta prueba se quedan "congelados". Esto ocurre con una probabilidad de $1 - e^{-2J/(k_BT)}$. Si T=0 todos los spines iguales se quedan congelados, si T= infinito, todos los spines se derriten.

Construir la red de enlaces toma un tiempo O(n), pues hay 2n enlaces.

2. Descomposición en clústers.

Después de formar las redes de enlaces de la manera descrita arriba, los spins son divididos por clústers. Un clúster no es más que un dominio de spines conectados unos a otros por enlaces congelados. Esa descomposición es unívoca. La descomposición en clústers consume muyo tiempo, por lo que es esencial usar un algoritmo lineal con respecto a L.

3. Actualización de spines:

Ahora se cambian todos los spines de cada clúster con una probabilidad de 0.5. Note que en este proceso la Temperatura no juega papel alguno. Este algoritmo se comporta según el número de clústers, que es siempre menor que N.

Swendsen y Wang tiene un $z = 0.35$, por lo que el tiempo de la simulación es de $N\tau \sim N\xi^{0.35} \sim NL^{0.35} \sim N^{1.175}$ lo cual es mucho mejor que el O(N^2) correspondiente a Metrópolis.

Algoritmos para la Descomposición en Clústers

Una red con huecos y enlaces puede ser vista como un grafo, y el problema de encontrar todos los clústers equivale al problema de identificar los componentes conectados en la teoría de grafos.

Existen dos algoritmos populares para identificar clústers, que tienen la propiedad de escalar linealmente (o casi linealmente) con el tiempo:

Algoritmo de Retroceso:

Se establece una matriz de tamaño N que mantiene como datos si el i-ésimo spin ha sido visitado o no por el algoritmo:

1. Marcar todos los spines como no visitados.

2. Llamar una función f(i) para cada spin s_i de la red. Esta función hace lo siguiente:
   - Si el spin s_i ya ha sido visitado previamente, termina la llamada.
   - Si no, marca s_i como visitado.
   - Repetir los dos pasos anteriores para cada uno de los cuatro vecinos s_j de s_i: Si el enlace i-j está congelado, entonces llama a f(j). Este es el paso recursivo.

Este algoritmo visitará cada spin en el clúster conectado por enlaces congelados. De esta manera, mientras la red es recorrida, los distintos clústers serán identificados.

Además, a la función recursiva se le puede dar un segundo argumento el cual puede ser un número único de clúster, usado para marcar todos los spines de un mismo clúster.

Este algoritmo no es el más eficiente ya que muchos sitios serán probados aunque ya se hayan visitado previamente.

Algoritmo Hoshen-Koppelman

Aunque más eficiente, este es un algoritmo más difícil de entender y programar. Las ideas esenciales:

Como los spines son visitados en orden, si nos movemos de izquierda a derecha y de arriba hacia abajo los vecinos superior y derecho de cada spin ya habrán sido visitados y los otros dos no.

A cada spin se le asigna un *número de clúster* de la manera siguiente: Si el spin está conectado a su vecino superior o a su vecino izquierdo por un enlace congelado, entonces se le asigna el menor de los dos números. De lo contrario, se le asigna un número mayor que todos los asignados hasta el momento.

En general, con este procedimiento los clústers tendrán más de un número asignado. Para resolver esto, se calificará a los varios números asignados a un clúster como propios o impropios (buenos o malos). Un clúster tendrá solamente un número propio y todos los demás serán impropios. El algoritmo mantendrá un arreglo A que clasificará cada número de clúster entre las dos categorías. El número "propio" de un clúster será el mínimo de todos los números que le están asignados. Cuando un nuevo número es creado, es marcado como "propio" haciendo $A[l] = l$. Cuando la red es escaneada puede ocurrir que se revise un spin cuyos vecinos superior e izquierdo tengan asignados diferentes números de clúster. Este "conflicto" se resuelve haciendo $A[l_{max}] = l_{min}$, o sea, el valor de A correspondiente al mayor de los números de clúster se iguala al menor. Al final, cualquier número de clúster l para el cual $A[l] \neq l$ será "impropio". Dado un número "impropio", su número "propio" correspondiente puede ser encontrado por la siguiente iteración: *Mientras* $A[l] \neq l$ *hacer* $l = A[l]$.

Algoritmo de Wolff

Este es un algoritmo aún más eficiente que el de Swendsen-Wang. Está basado en construir y cambiar un sólo clúster por vez.

A grosso modo, funciona de la siguiente manera:

1. Elegir un spin s_i aleatoriamente de la red, voltearlo, y finalmente marcarlo como "spin de clúster".
2. Formar un clúster usando este spin como "semilla" y chequeando cada uno de sus cuatro vecinos:
   a. Si el vecino ya es un "spin de clúster", revisar el siguiente vecino. Si los cuatro vecinos ya son "spines de clúster", termina el algoritmo.
   b. Si el vecino es "opuesto" al spin semilla, agregarlo al clúster con probabilidad $1 - e^{-2J/k_BT}$. En el caso de haber sido agregado al clúster, marcarlo como "spin de clúster" y luego chequear recursivamente cada uno de sus vecinos.

Debe notarse que la decisión de agregar o no un spin al clúster se realiza con el mismo criterio de probabilidad $r < 1 - e^{-2J/k_BT}$ con que se "congela" un enlace en el algoritmo de Swendsen-Wang. Por eso ambos algortimos tienen las mismas propiedades estadísticas.

Puede demostrarse sencillamente que el Algoritmo de Wolff es mucho más eficiente que el de Swendsen-Wang. Supongamos que la red está particionada en clústers según Swendsen-Wang. El

algoritmo de Wolff construye y voltea un clúster mediante la elección aleatoria de un spin que será usado como semilla. Esta elección aleatoria favorece claramente a los clústers más grandes. A su vez, voltear clústers más grandes, resulta en configuraciones no correlacionadas con mayor probabilidad. Por tanto, se consigue de una manera más eficiente la simulación del sistema.

Códigos en C++

Los algoritmos Wolff y Swendsen-Wang fueron implementados en C++. Sus códigos se encuentran adjuntos a este documento.

Conclusiones

En el trabajo se expuso brevemente el fenómeno de Retardamiento Crítico, y luego se estudiaron dos algoritmos de cluster que evitan las divergencias en el cálculo inherentes a Metropolis cuando la temperatura del sistema se acerca a la Temperatura de Curie.

Se pudo observar que es precisamente el carácter local del algoritmo de Metropolis el que causa estas divergencias. Se proponen entonces los algoritmos de cluster diseñados por Swendsen-Wang y Wolff, con la características que trabajan con grupos de spines afines, en vez de con spines individuales.