Análisis y Diseño de Algoritmos

Simulated Annealing
&
Tabu Search

Arturo Díaz Pérez
Sección de Computación
Departamento de Ingeniería Eléctrica
CINVESTAV-IPN
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. CP 07300

Tel. (5)747 3800 Ext. 3755
e-mail: `adiaz@cs.cinvestav.mx`

Estrategias Generales

- Estrategias generales para resolver problemas de optimización sobre espacios de búsqueda exponenciales
  - Búsqueda Local
  - Recocido Simulado (Simulated Annealing)
  - Búsqueda Tabú
  - Algoritmos Genéticos
  - Redes Neuronales
  - Técnicas de Aleatorización

Revisión de Búsqueda Local

- La idea básica de los algoritmos de búsqueda local es:
  - Para una instancia x de un problema de minimización se mueve de una solución actual s a una solución vecina s' siempre y cuando $m(x,s') < m(x,s)$.
  - Esto conduce a un óptimo local el cual no es necesariamente un óptimo global
  - No se toma en cuenta la posibilidad que un incremento temporal de la medida de la solución actual puede eventualmente conducir a una mejor solución.

Análisis y Diseño de Algoritmos — *Annealing*-3

¿Cómo superar el problema?

- Sean t_1, t_2, ..., con $t_i \geq 0$ una secuencia (posiblemente infinita) de umbrales
  - Se permite mover de s a s' en el k-ésimo paso del algoritmo si y solamente si $m(x,s') - m(x,s) < t_k$.
  - Nótese que si $t_k > 0$, entonces, s' puede ser seleccionado aún si representa una peor solución que s.
  - Nótese que si $t_i = 0$, $i \geq 1$, entonces, el algoritmo coincide con el de búsqueda local

Análisis y Diseño de Algoritmos — *Annealing*-4

Condiciones de Paro

- En general, se puede asumir que la secuencia de umbrales se decrementa monótonamente a 0
  - $t_i < t_{i-1}$, $i \geq 2$
  - Para cualquier $e > 0$, existe $i \geq 1$, tal que, $t_i < e$.
- Regla de paro
  - El algoritmo se detiene después de k movimientos en la solución s si, para todos las soluciones vecinas s', la siguiente desigualdad se satisface

$$m(x,s') - m(x,s) < t_k$$

Análisis y Diseño de Algoritmos — *Annealing*-5

Recocido Simulado

- Simulated Annealing: corresponde a una versión *aleatorizada* del enfoque anterior
  - La elección para moverse de s a una peor solución s' se realiza con una probabilidad inversamene proporcional a $m(x,s') - m(x,s)$
  - No obstaculiza de forma determinista una dirección de búsqueda
  - La probabilidad de seleccionar una solución que no mejora se decrementa sobre el tiempo
- Annealing (Recocido) es el proceso físico en el cual un sólido que ha sido fundido por altas temperaturas se enfría lenta y cuidadosamente al encontrar un estado de baja energía para el conjunto de partículas en el sólido.

Análisis y Diseño de Algoritmos — *Annealing*-6

Recocido Simulado

- Ejemplo: la producción de un cristal a partir de su estado líquido
  - En la fase líquida todas las partículas se arreglan aleatoriamente, debido a la alta energía en el sistema
  - En la configuración de energía mínima se presenta una estructura regular en la distribución espacial de todas las partículas, lo que corresponde al estado cristalino.
  - El sólido puede tomar una de un conjunto de configuraciones localmente mínimas, lo que corresponde a cristales defectuosos
  - El enfriamiento se debe hacer lentamente para evitar que el sólido entre en alguna de esas configuraciones localmente mínimas y conducirlo a una configuración de energía mínima global.

Análisis y Diseño de Algoritmos *Annealing-7*

Algoritmo de Metropolis

- Simula el enfriamiento del sólido generando una secuencia de estados siguiente:
  - Dado un estado i con energía asociada E_i, se genera un nuevo estado j, aplicando una perturbación y calculando la energía correspondiente E_j.
    - Ej. cambiando el desplazamiento de una partícula individual
  - Sea $d = E_j - E_i$, si $d > 0$, entonces, la transición del estado i al estado j es aceptada.
  - De otra forma, la transición es aceptada con probabilidad

$$p_T(d) = e^{-\frac{d}{k_B T}}$$

  - donde T es la temperatura y
  - k_B es la constante de Boltzmann

Análisis y Diseño de Algoritmos *Annealing-8*

Algoritmo de Metropolis

$$p_T(d) = e^{-\frac{d}{k_B T}}$$

- Si el algoritmo de Metropolis se aplica en una situación de un decremento lento de la temperatura
  - permitiendo que el sólido alcance un equilibrio térmico para cada temperatura, entonces,
  - el estado de energía mínima del sistema se aproxima conforme la temperatura se aproxima a 0.

Análisis y Diseño de Algoritmos *Annealing*-9

Optimización Combinatoria

- Estado del sistema físico → la solución factible
- Energía asociada al estado → la medida de la solución
- El algoritmo de Metropolis puede ser adaptado a cualquier problema de optimización.
  - Para hallar la solución óptima el algoritmo puede requerir un número de transiciones mayor que el tamaño del espacio solución del problema
  - El algoritmo práctico utiliza un número finito de pasos para encontrar una solución.

Análisis y Diseño de Algoritmos *Annealing*-10

Implementación

- La secuencia de temperaturas, denotada como el calendario de enfriamiento, se genera por la función

$$\tau = r\,\tau$$

  - donde τ es la variable de temperatura y
  - $0 < r < 1$ es un parámetro de control, el rango de enfriamiento
  - $t_1 = t$, $t_2 = rt$, $t_3 = r^2t$,...
- A una temperatura dada, t_k, el movimiento de s a s' se selecciona con la siguiente probabilidad

$$P(s,s',k) = \begin{cases} 1 & \text{si } m(x,s') < m(x,s) \\ e^{-\frac{m(x,s')-m(x,s)}{t_k}} & \text{en caso contrario} \end{cases}$$

Análisis y Diseño de Algoritmos *Annealing*-11

Algoritmo de Simulated Annealing

- Parámetros de Entrada
  - $l > 0$, número de transiciones realizadas en cada temperatura
  - *N*, una vecindad
  - s_0, una solución inicial
  - r, factor de enfriamiento
  - t, temperatura
  - FROZEN, regla de paro
- Datos de Salida
  - s, solución

Análisis y Diseño de Algoritmos *Annealing*-12

Algoritmo de Simulated Annealing

```
function SimulatedAnnealing()
begin
    τ := t;
    s := solución inicial factible s0;
    repeat
        for i := 1 to l do begin
            selecciona cualquier estado no visitado s' ∈ N(s);
            if m(x,s') < m(x,s) then
                s := s'
            else begin
                δ := m(x,s') - m(x,s);
                s := s' con probabilidad e^(-δ/τ)
            end
        end;
        τ := r τ
    until FROZEN;
    return s
end.
```

Análisis y Diseño de Algoritmos *Annealing*-13

Algoritmo de Simulated Annealing

```
function SimulatedAnnealing()
begin
    τ := t;
    s := solución inicial factible s0;
    repeat
        for i := 1 to l do begin
            selecciona cualquier estado no visitado s' ∈ N(s);
            if m(x,s') < m(x,s) then
                s := s'
            else begin
                δ := m(x,s') - m(x,s);
                if e^(-δ/τ) > random(0,1) then s := s'
            end
        end;
        τ := r τ
    until FROZEN;
    return s
end.
```

Análisis y Diseño de Algoritmos *Annealing*-14

Estrategias Generales

- Estrategias generales para resolver problemas de optimización sobre espacios de búsqueda exponenciales
  - Búsqueda Local
  - Recocido Simulado (Simulated Annealing)
  - Búsqueda Tabú
  - Algoritmos Genéticos
  - Redes Neuronales
  - Técnicas de Aleatorización

Análisis y Diseño de Algoritmos *Annealing*-15

Búsqueda Tabú

- Motivación:
  - Generalizar la búsqueda local simple para evitar los riesgos de restringir los movimientos solo a mejores soluciones.
- Permite moverse de una solución *s* a una solución vecina peor *s*'.
  - Simulated annealing: aleatorización
  - Tabú search: enfoque completamente determinista

Análisis y Diseño de Algoritmos *Annealing*-16

Búsqueda Tabú

- Para cualquier solución s, denotemos como $\Delta(s)$ el conjunto de movimientos que se pueden aplicar en s.
  - La función δ^* se define tal que, para cualquier $S \subseteq \Delta(s)$, $\delta^*(S)$ regresa el mejor movimiento.
- Se mantiene una lista $\boldsymbol{T}$ de movimientos tabú
  - movimientos que son excluidos dado que se considera que conducen a óptimos locales malos
    - Ej. movimientos que son el inverso de los movimientos realizados recientemente
  - $\boldsymbol{T}$ tiene una tamaño máximo t.

Análisis y Diseño de Algoritmos *Annealing*-17

Búsqueda Tabú

- En cada paso, se mantiene la mejor solución encontrada hasta el momento
  - Dado una solución actual s, se selecciona el vecino s' con el mejor movimiento no tabú, $\delta^*(\Delta(s) - \boldsymbol{T})$
- El método se detiene cuando todos los movimientos posibles son tabú, o
- Si k movimientos se han realizado sin mejorar la solución actual
  - k es un parámetro definido por el usuario
    - A valores grandes de k se espera tener mejores soluciones
    - A valores grandes de k se espera tener tiempos de ejecución mayores

Análisis y Diseño de Algoritmos *Annealing*-18

Búsqueda Tabú

- Para cualquier solución s y para cualquier subconjunto S de $\Delta(s)$, $\delta^*(S)$ se define como el movimiento en S que conduce al vecino s', tal que,

  $m(x,s') = \min\{ m(x,y) \mid y$ se obtiene aplicando un movimiento $\delta \in S\}$

  - esto no evita la posibilidad de que, en el caso de s sea un óptimo local, se mueva a una solución peor

- ***T*** se define de tal forma que se evite visitar soluciones que ya han sido consideradas
  - Usualmente, se guardan los últimos t movimientos

Análisis y Diseño de Algoritmos — *Annealing*-19

Cubierta de Vértices Mínima

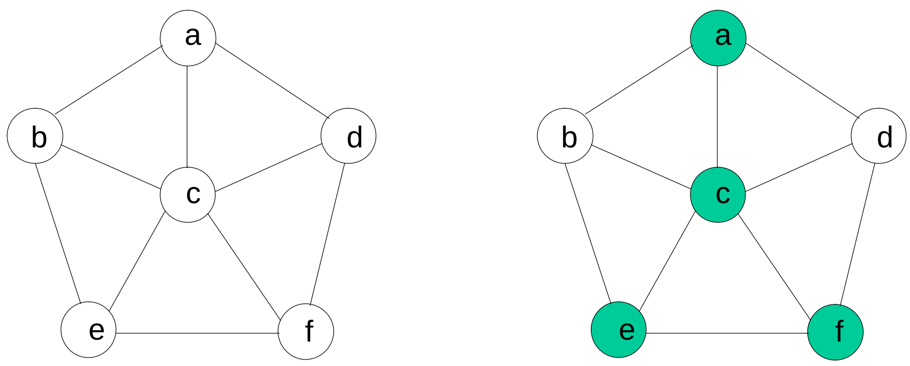
- Sea $G=(V, E)$ una gráfica.
  - Una cubierta para G es un subconjunto $V' \subseteq V$, tal que,
    - Para cada $(u, v) \in E$, $u \in V'$ o $v \in V'$

- El problema es hallar V^* tal que
  - $V^* = \min_{|V|}\{ V$ es una cubierta$\}$

- ¿Cuántas cubiertas posibles existen?
  - $2^{|V|}$

Análisis y Diseño de Algoritmos — *Annealing*-20

Cubierta de Vértices Mínima



Cubierta Mínima

Análisis y Diseño de Algoritmos *Annealing*-21

Cubierta de Vértices Mínima

- Dada una instancia del problema $G=(V, E)$,
  - Sea n el número de vértices
- Cualquier solución U tiene n posibles movimientos
  - Los primeros $|U|$ movimientos eliminan un vértice de U.
  - Los siguientes $|V-U|$ movimientos agregan un vértice a U.
    - Note que las $|U|$ supresiones pueden conducir a soluciones no factibles; éstas se pueden evitar definiendo adecuadamente a δ^*.
- T es implementada como una cola de t movimientos, en donde se indica
  - el vértice que causa el movimiento y
  - si es una inserción o supresión

Análisis y Diseño de Algoritmos *Annealing*-22

Cubierta de Vértices Mínima

- Se define $A(U, m) = |W|$ en donde U es la cubierta actual y W es la cubierta obtenida de $|U|$ al aplicar el movimiento m.

- Se permite un movimiento tabú, m, aplicado a la solución U, si $A(U, m) < |V^*|$, donde V^* es la cubierta más pequeña encontrada hasta el momento
  - Esto es, se permite un movimiento tabú si la cubierta resultante mejora la medida de la mejor solución encontrada hasta el momento.
  - $A(U, m)$ se conoce como niveles de aspiración

Análisis y Diseño de Algoritmos *Annealing*-23