Análisis y Diseño de Algoritmos

Técnicas de Aleatorización

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

Cálculo de Proposiciones

Ejemplo: $(P \wedge (Q \rightarrow R)) \rightarrow S$

$$
\begin{aligned}
(P \wedge (Q \rightarrow R)) \rightarrow S &= \\
&= (P \wedge (\sim Q \vee R)) \rightarrow S \\
&= \sim(P \wedge (\sim Q \vee R)) \vee S \\
&= (\sim P \vee \sim(\sim Q \vee R)) \vee S \\
&= (\sim P \vee (\sim(\sim Q) \wedge \sim R)) \vee S \\
&= (\sim P \vee (Q \wedge \sim R)) \vee S \\
&= ((\sim P \vee Q) \wedge (\sim P \vee \sim R)) \vee S \\
&= S \vee ((\sim P \vee Q) \wedge (\sim P \vee \sim R)) \\
&= (S \vee (\sim P \vee Q)) \wedge (S \vee (\sim P \vee \sim R)) \\
&= (S \vee \sim P \vee Q)) \wedge (S \vee \sim P \vee \sim R)
\end{aligned}
$$

Variables y Literales

☞ Sea $X = [X_j]_{j=1, \ldots, n0}$ una lista de *variables*. Utilizaremos los siguientes símbolos constantes:

- ← 1: Valor *verdadero*.
  - 👉 Es la unidad de "$\wedge$": $F \wedge 1 = 1 \wedge F = F$
- ← 0: Valor *falso*.
  - 👉 Es la unidad de "$\vee$": $C \vee 0 = 0 \vee C = C$
- ← **Literales**: Variables o negaciones de variables:

  L es *literal* si $\exists X \in \mathbf{X}$: $(L = X) \vee (L = \neg X)$

Cláusulas

☞ Sea X = $[X_j]_{j=1, ..., n0}$ una lista de *variables*. Utilizaremos los siguientes símbolos constantes

← **Cláusulas**: Disyunciones de literales.

☞Para cada e ∈ {-1, 0, 1}^{n0} hacemos

$$Claus(e) = \bigvee_{1 \le j \le n} X_j^{\downarrow e_j}, \text{ donde}$$

$$\forall j = 1, ..., n: X_j^{\downarrow e_j} = \begin{cases} X_j & \text{si } e_j = 1 \\ 0 & \text{si } e_j = 0 \\ \neg X_j & \text{si } e_j = -1 \end{cases}$$

$$e_j = \begin{cases} 1 & \text{si } X_j \text{ aparece en } Claus(e) \\ -1 & \text{si } \neg X_j \text{ aparece en } Claus(e) \\ 0 & \text{si ni } X_j \text{ ni } \neg X_j \text{ aparecen en } Claus(e) \end{cases}$$

Formas Proposicionales

← **Frases**: Conjunciones de literales: Para cada e ∈ {-1, 0, 1}^{n0} hacemos

$$Frase(\mathbf{e}) = \bigwedge_{1 \le j \le n} X_j^{\uparrow e_j}, \text{ donde}$$

$$\forall j = 1, ..., n: X_j^{\uparrow e_j} = \begin{cases} X_j & \text{si } e_j = 1 \\ 1 & \text{si } e_j = 0 \\ \neg X_j & \text{si } e_j = -1 \end{cases}$$

Formas Proposicionales

← **Formas Conjuntivas**: Conjunciones de cláusulas: Si

$$FC = \bigwedge_{1 \le i \le m} C_j = \bigwedge_{1 \le i \le m} \bigvee_{1 \le j \le n} L_{ij}$$

entonces representamos a *FC* por la matriz $[L_{ij}]_{i,j}{}^{C}$

← **Formas Disyuntivas**: Disyunciones de frases: Si

$$FD = \bigvee_{1 \le i \le m} F_i = \bigvee_{1 \le i \le m} \bigwedge_{1 \le j \le n} L_{ij}$$

entonces representamos a *FD* por la matriz $[L_{ij}]_{i,j}{}^{D}$

Análisis y Diseño de Algoritmos *Randomization*-7

Formas Proposicionales

☞ **Formas Proposicionales**:

← 0, 1 son formas proposicionales

← Las variables son formas proposicionales

← Las negaciones de formas proposicionales son formas proposicionales

← Las conjunciones y disyunciones de formas proposicionales también son formas proposicionales.

Análisis y Diseño de Algoritmos *Randomization*-8

Satisfactibilidad

- Toda forma proposicional F determina una función

$$F: B^{n0} \rightarrow B.$$

  - Para cada $x \in B^{n0}$, $F(x)$ es el valor de verdad que asume F bajo la asignación x

- Dos formas son equivalentes si definen a una misma función

Satisfactibilidad

- **Una forma proposicional F es**
  - **Satisfactible** si hay una asignación que la satisface, es decir, si $F \neq 0$
  - **Tautología** si es satisfecha por todas las asignaciones posibles, es decir, si $F = 1$
  - **Refutable** si hay una asignación que la refuta, es decir, si $F \neq 1$
  - **Insatisfactible** si es refutada por todas las asignaciones posibles, es decir, si $F = 0$

SAT

- Problema SAT
  - Consiste en decidir si acaso una forma proposicional dada es satisfactible. Formalmente el problema se especifica como sigue:
    - **Instancia**: Una forma proposicional *F*
    - **Solución**: 1 si *F* es satisfactible, 0 si *F* es insatisfactible

MaxSAT

- Problema SAT
  - Consiste en decidir el número máximo de cláusulas que pueden satisfacerse simultáneamente con una interpretación.
    - **Instancia**: Una forma proposicional *F*
    - **Solución**: El número máximo de cláusulas satisfactibles
- ¿Cuál es el tamaño del espacio de búsqueda para SAT y MaxSAT?

Aleatorización

- Motivación
  - En muchos aplicaciones, un algoritmo aleatorio es
    - más simple ó
    - más rápido
  - que los algoritmos deterministas

- Un algoritmo aleatorio es un algoritmo en el cual durante sus pasos de ejecución toma decisiones aleatorias.
  - Al ejecutar un algoritmo aleatorio para la misma instancia del problema no necesariamente se obtienen los mismos resultados
  - En problemas de optimización se pueden encontrar soluciones con valores radicalmente diferentes.

Aleatorización

- Dada una instancia de un problema de optimización combinatoria, el valor de una solución encontrado por un algoritmo aleatorio es una variable aleatoria.

  - Cuando se estima el valor esperado de esta variable aleatoria, se tiene que considerar el comportamiento del algoritmo promedio sobre todas las posibles ejecuciones.

Satisfactibilidad Aleatoria

- Entrada: Sea C un conjunto de cláusulas sobre un conjunto de variables V.
- Salida: Un asignamiento de verdad
  - ← f: $V \rightarrow \{1, 0\}$
- Método:

```
begin
    foreach v ∈ V do
        Hacer f(v) = 1 con probabilidad 1/2;
    return f
end.
```


Satisfactibilidad Aleatoria

- Sea $m_{RS}(x)$ la variable aleatoria que denota el valor de la solución encontrada con el algoritmo anterior con la entrada x.
- Sea C un conjunto de cláusulas con al menos k literales
- Dada una instancia x del problema de máxima satisfactibilidad en el cual todas las c cláusulas tienen al menos k literales, la medida esperada $m_{RS}(x)$ de la solución encontrada por el programa anterior satisface la siguiente desigualdad:

$$E[m_{RS}(x)] \geq \left(1 - \frac{1}{2^k}\right) c$$

Satisfactibilidad Aleatoria

- La probabilidad de que cualquier cláusula con k literales no se satisfaga con el asignamiento encontrado con el algoritmo aleatorio es 2^{-k}.
  - La cual es la probalidad que todas las literales de la cláusula se hagan falsas.
  - Por tanto, la probabilidad de que una cláusula con al menos k literales sea satisfecha es al menos $1- 2^{-k}$.
  - De aquí se sigue que la contribución esperada de una cláusula a $m_{RS}(x)$ es al menos $1- 2^{-k}$.
  - Sumando todas la cláusulas se obtiene

$$E[m_{RS}(x)] \geq \left(1 - \frac{1}{2^k}\right) c$$

Satisfactibilidad Aleatoria

- Dada una instancia x del problema de máxima satisfactibilidad, la medida esperada $m_{RS}(x)$ de la solución encontrada por el programa anterior satisface la siguiente desigualdad:

$$\frac{m^*(x)}{E[m_{RS}(x)]} \leq 2$$

MaxSAT con Pesos

- Problema MaxSAT con Pesos
  - **Instancia**: Una forma proposicional *F* dada como un conjunto de cláusulas las cuales tienen asociado un peso no negativo.
  - Consiste en encontrar un asignamiento de valores de verdad a cada variable de la fórmula que maximice el peso de las cláusulas satisfechas.
  - **Solución**: El peso máximo de cláusulas satisfactibles

Satisfactibilidad Aleatoria con Pesos

- Entrada: Sea C un conjunto de cláusulas sobre un conjunto de variables V con un peso asociado cada cláusula
- Salida: Un asignamiento de verdad
  - f: $V \rightarrow \{1, 0\}$
- Método:

```
begin
    foreach v_i ∈ V do
        Hacer f(v_i) = 1 con probabilidad p_i;
    return f
end.
```

Máxima Satisfactibilidad Pesada

- Sea $m_{GRWS}(x)$ la variable aleatoria que denota el valor de la solución encontrada con el algoritmo anterior con la entrada x.
- Se puede ver que

$$E[m_{GRWS}(x)] = \sum_{c \in C} w(c) \left(1 - \prod_{i \in V_c^+} (1 - p_i) \prod_{i \in V_c^-} p_i \right)$$

- donde V_c^+ y V_c^- denotan el conjunto de índices de las variables que aparecen en forma positiva y negativa, respectivamente, en la cláusula c.

MaxSAT: Programación Lineal

- Sea $C = \{c_1, c_2, ..., c_t\}$ un conjunto de cláusulas definidas sobre las variables $v_1, v_2, ..., v_n$ y una función de peso w.

$$\text{maximizar} \sum_{c_j \in C} w(c_j) z_j$$

$$\text{sujeto a} \sum_{i \in V_{c_j}^+} y_i + \sum_{i \in V_{c_j}^-} (1 - y_i) \geq z_j, \forall c_j \in C$$

$$y_i \in \{0,1\} \quad 1 \leq i \leq n$$
$$z_j \in \{0,1\} \quad 1 \leq j \leq t$$

- $y_i = 1$ si y solamente si x_i es verdadera
- $z_j = 1$ si y solamente si la cláusula C_j es verdadera

MaxSAT: Programación Lineal

- Sea *LP-SAT*(x) el programa lineal que se obtiene relajando las restricciones de integridad de *IP-SAT*(x).
  - ← Sea $y^*=(y_1^*, y_2^*, ..., y_n^*\}$ y $z^*=(z_1^*, z_2^*, ..., z_t^*\}$ la solución óptima de *LP-SAT*(x).
  - ← Claramente, $m^*_{LP\text{-}SAT(x)} \geq m^*_{IP\text{-}SAT(x)}$

MaxSAT: PL + Aleatorización

- Entrada: Instancia x. Sea C un conjunto de cláusulas sobre un conjunto de variables V. Una función w: $C \rightarrow \mathbf{N}$
- Salida: Un asignamiento de verdad
  - ← f: $V \rightarrow \{1, 0\}$
- Método:

```
begin
    encuentre el valor óptimo (y*,z*) de LP-SAT(x);
    foreach v_i ∈ V do begin
        p_i = g(y_i*) {para una función adecuada g};
        Hacer f(v) = 1 con probabilidad p_i
    end;
    return f
end.
```

Si g puede ser calculada en tiempo polinomial, el algoritmo toma tiempo polinomial

MaxSAT: PL + Aleatorización

- El factor de aproximación de la solución regresada depende de la elección de la función *g*.
  - Supongamos que existe un número real α, 0 < α< 1, tal que, para cada cláusula c_j

$$\left(1-\prod_{i\in V_c^+}(1-p_i)\prod_{i\in V_c^-}p_i\right)\geq \alpha z_j^*$$

Ya que $\sum_{j=1}^{t} w(c_j)z_j^* = m_{LP-SAT}^*(x)$

y $E[m_{GRWS}(x)] = \sum_{c\in C} w(c)\left(1-\prod_{i\in V_c^+}(1-p_i)\prod_{i\in V_c^-}p_i\right)$

  - La solución regresada por el programa tiene un factor de aproximación de a lo más 1/ α.

MaxSAT: PL + Aleatorización

- Todo depende de cómo elegir *g*.
  - La primera opción es hacer $g(y_i) = y_i^*$, para $i = 1, 2, ..., n$.
  - En otras palabras cada variable v_i se hace verdadera de forma independiente con probabilidad y_i^*.

- Dada una instancia x del problema de Máxima Satisfactibilidad Pesada. Sea (y^*,z^*) la solución óptima de LP-SAT(x). Entonces, para cualquier cláusula c_j en x con k literales se tiene que

$$\left(1-\prod_{i\in V_c^+}(1-y_i^*)\prod_{i\in V_c^-}y_i^*\right)\geq a_k z_j^* \quad \text{donde} \quad a_k = 1-\left(1-\frac{1}{k}\right)^k$$

MaxSAT: PL + Aleatorización

- Sin pérdida de generalidad, supongamos que cada variable en la cláusula c_j es positiva.
  - ← Se debe probar que

$$\left(1-\prod_{i=1}^{k}(1-y_{j_i}^{*})\right)\geq a_k z_j^{*}$$

- Se puede ver que dado un conjunto de números no negativos $\{a_1, a_2, ..., a_k\}$, se cumple que

$$\frac{a_1+\ldots+a_k}{k}\geq \sqrt[k]{a_1\cdots a_k}$$

MaxSAT: PL + Aleatorización

- Aplicando la fórmula anterior y recordando que

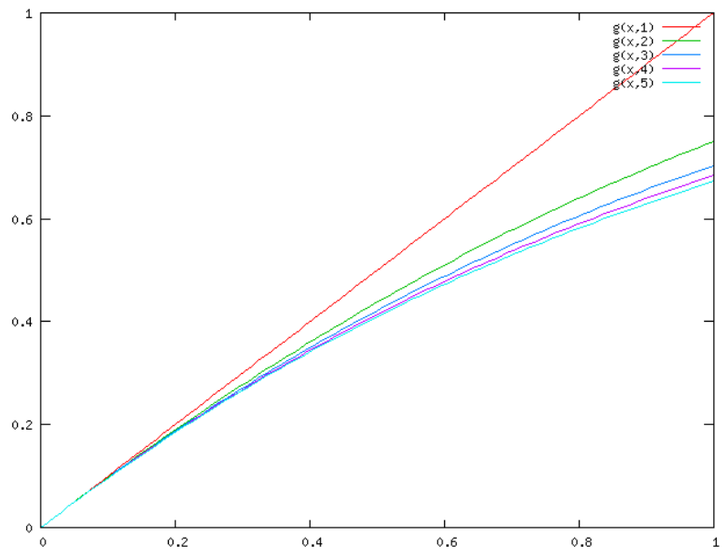
$$\sum_{i=1}^{k} y_{j_i}^{*}\geq z_j^{*} \qquad \text{pues} \sum_{i\in V_{c_j}^{+}} y_i+\sum_{i\in V_{c_j}^{-}}(1-y_i)\geq z_j, \forall c_j\in C$$

$$\left(1-\prod_{i=1}^{k}(1-y_{j_i}^{*})\right)\geq 1-\left(\frac{\sum_{i=1}^{k}(1-y_{j_i}^{*})}{k}\right)^{k}\geq 1-\left(1-\frac{\sum_{i=1}^{k} y_{j_i}^{*}}{k}\right)^{k}$$

$$\geq 1-\left(1-\frac{z_j^{*}}{k}\right)^{k}\geq a_k z_j^{*}$$

- La última desigualdad se debe a que $f(z_j^{*})=1-\left(1-\frac{z_j^{*}}{k}\right)^{k}$ es una función cóncava en el intervalo [0,1] y $f(z_j^{*})\geq a_k z_j^{*}$ en los extremos de ese intervalo.

$f(z_j^*)$



MaxSAT: PL + Aleatorización

$$a_k = 1 - \left(1 - \frac{1}{k}\right)^k$$

- Ya que α_k es una función que decrece con respecto a k,
  - ← dada una instancia del MaxSAT Pesado, tal que, cada cláusula tiene a lo más k literales,
  - ← el resultado anterior implica que si g es la identidad se produce un algoritmo aleatorizado cuyo cociente de rendimiento es a lo más $1/\alpha_k$.
- En particular,
  - ← si $k \leq 2$, el cociente de rendimiento está acotado por 4/3.
  - ← si $k \leq 3$, el cociente de rendimiento está acotado aproximadamente por 1.582.

α_k, $1/\alpha_k$