Aproximación De Problemas Combinatorios Con Optimización Semidefinida Y Redondeo Aleatorio

1. Resumen

Max cut es el problema de hallar el corte máximo sobre los vértices de un grafo en el cual se le ha asignado un valor no negativo y racional a cada arista. Este problema es NP-Hard y por tanto no existe un algoritmo que lo resuelva en tiempo polinomial a menos que P = NP. Goemans y Williamson diseñaron (ver [7]) un α-algoritmo de aproximación aleatorio para max cut con $\alpha = 0{,}87856...$ al cual en este trabajo llamaremos G-W, este aprovecha la equivalencia de max cut con el problema de programación entera

$$\underset{x_i,x_j\in\{1,-1\}}{max}\frac{1}{2}\sum_{i<j} w_{ij}(1-x_ix_j), \quad 1\leq i,j\leq n \qquad (Q)$$

y luego pasa a la relajación

$$\underset{v_i,v_j\in S^m}{max}\frac{1}{2}\sum_{i<j} w_{ij}(1-v_i.v_j), \quad 1\leq i,j\leq n \qquad (P)$$

sobre la cual se realizan los siguientes pasos:

1. Resolver (P) como un problema de programación semidefinida obteniendo un conjunto de n vectores $\{v_1,...v_n\}$ sobre la esfera $(m-1)$-dimensional para $m \geq 2$, estos vectores son los vertices de una realización del grafo G sobre S^{m-1}
2. generar un vector r aleatorio uniformemente distribuido sobre la esfera
3. $S = \{i|v_i.r \geq 0\}$. Donde S representa el corte máximo

En el presente trabajo Aplicamos el algoritmo G-W a otro problema NP-Hard conocido como Partition. Esto fue posible porque la función objetivo de partition se puede expresar en términos de la función objetivo de Max-cut.

2. Introducción

Desde el punto de vista computacional un problema se considera resuelto si existe un algoritmo que lo resuelve en tiempo polinomial. Existen muchos problemas para los cuales esto se ha conseguido como por ejemplo 2-Sat, el problema hallar el mínimo árbol generador sobre un grafo y el problema de hallar el camino mínimo sobre las aristas de un grafo, por eso estos problemas pertenecen a la clase de complejidad P.

Pero también existen problemas para los cuales no se conoce un algoritmo que los resuelva en tiempo polinomial, ejemplos de esta clase de problemas los mostramos en la lista de la sección 6.

Hay problemas NP que tienen aplicaciones muy importantes entre ellos podemos mencionar el problema Set cover utilizado por la empresa IBM en 1998 para construir algoritmos eficientes para encontrar viruses de computador.

Por otro lado para que un problema sea aplicable usualmente no se requiere que este se resuelva exactamente, es común que cierto porcentaje de la solución óptima del problema ya tenga utilidad y es por esto que tiene sentido buscar algoritmos que aproximen problemas de la clase NP-Hard. Aún existen problemas que se pueden resolver en tiempo polinomial pero para su aplicación práctica es preferible usar un algoritmo de aproximación, como por ejemplo con 2-Sat.

Uno de los trabajos más importantes en teoría de la optimización es el articulo de [7] de Goemans y Williamson, en este artículo ellos construyen lo que se denomina un ρ-algoritmo de aproximación aleatorio que en tiempo polinomial da una solución con una aproximación de más 87,8 % de la solución óptima, que para lo logrado hasta el momento fue un resultado más que sorprendente.

Este algoritmo lo llamamos algoritmo G-W y lo presentamos en la sección 9 y es el resultado que motivo el presente trabajo. Esencialmente buscamos aplicar este algoritmo a otros problemas de la clase NP, ya sea adecuando el algoritmo para poder resolver aproximadamente un nuevo problema o bien transformando el problema para que el algoritmo sea aplicable. En esta segunda forma contamos con el hecho afortunado de que el problema de decsición asociado a Max cut el cual llamamos Number-cut es NP - completo (ver definición 7.2) lo que significa que cualquier problema de la clase NP se puede reducir a el en tiempo polinomial.

Nuestro objetivo con el presente trabajo es comprender la naturaleza de los problemas de la clase NP para poder diseñar algoritmos que aproximen estos problemas eficientemente.

3. Preliminares

El objetivo de esta sección es establecer las convenciones, el significado de los símbolos que usaremos en el presente trabajo y revisar algunos conceptos básicos que luego nos serán de utilidad. Para comenzar $\mathbb{R}^{m\times n}$ representa el conjunto de las matrices de orden $m \times n$ con entradas reales.

De acuerdo con esta notación el conjunto de las matrices cuadradas de orden n lo representamos por $\mathbb{R}^{n\times n}$ y el subconjunto de $\mathbb{R}^{n\times n}$ formado por las matrices invertibles lo simbolizamos por $GL_n(\mathbb{R})$. En particular la matriz identidad multiplicativa de $GL_n(\mathbb{R})$ es I_n. Para la traza de A usamos $Tr(A)$ y para el determinante $det(A)$.

Dada una matriz $A \in \mathbb{R}^{n\times n}$ sabemos que tiene n valores propios en el campo de los complejos cada valor propio lo representamos con λ_i para $i = 1, ..., n$ y el espacio propio asociado al valor propio λ por $E_\lambda(A)$ y como es usual $ma(\lambda)$ representa la multiplicidad algebraica de λ así como $mg(\lambda)$ representa su multiplicidad gemétrica. Con $\delta_i(A)$ denotamos el determinate de la matriz $M_i(A)$ que es la matriz obtenida al eliminar de A las ultimas i filas y las ultimas i columnas.

$\mathbb{R}^n$ es el espacio vectorial formado por los vectores de n componentes reales, consideraremos los elementos de $\mathbb{R}^n$ como vectores columna.

Para nuestros propósitos el subespacio más importante de $\mathbb{R}^{n\times n}$ es el formado por las matrices simétricas, este espacio lo denotamos por $S\mathbb{R}^{n\times n}$. Las matrices A de $\mathbb{R}^{n\times n}$ que tienen la propiedad de que para todo $x \in \mathbb{R}^n$ $x^T Ax \geq 0$, se llaman positivas semidefinidas y si $x^T Ax > 0$ para todo $x \in \mathbb{R}^n$ entonces diremos que A es definida positiva. Cuando A sea positiva semidefinida escribiremos $A \succeq 0$ en tanto que si A es definida positiva escribimos $A \succ 0$. El conjunto de las matrices semidefinidas positivas lo representamos por $S\mathbb{R}^{n\times n}_{+}$ y el de las definidas positivas por $S\mathbb{R}^{n\times n}_{++}$

Las matrices semidefinidas positivas y definidas positivas tienen propiedades muy interesantes las cuales incluyen cierto tipo de descomposiciones que nos serán muy útiles en el diseño de algoritmos.

3.1. Normas matriciales

Recordemos que una norma sobre un espacio vectorial real V es una función $\| \ . \ \|: V \to \mathbb{R}$ tal que se cumplen las siguientes afirmaciones:

(i) para todo $u \in V$, $\| u \| \geq 0$ y $\| u \| = 0$ si y solo si y solo si $u = 0$

(ii) $\| \alpha u \| = |\alpha| \|u\|$ para todo $\alpha \in \mathbb{R}$

(iii) $\| u + w \| \leq \| u \| + \| w \|$ para todo $u, v \in V$

Los espacios vectoriales que consideramos en el presente trabajo son subespacios de $\mathbb{R}^{m\times n}$ y especialmente de $\mathbb{R}^{n\times n}$, sobre estos espacios tenemos las siguientes normas matriciales:

(i) Las normas l_p o normas de Hölder. Para $p \geq 1$ y $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ tenemos que $\| x \| = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$

(ii) Para $p = 1$ tenemos la norma l_1 definida por $\| x \| = \sum_{i=1}^{n} |x_i|$

(iii) Para $p = 2$ Tenemos la norma euclidiana $\| x \| = (\sum_{i=1}^{n} |x_i|^2)^{\frac{1}{2}}$

(iii) Tenemos la norma infinita $\| x \| = \text{máx}\{|x_i| | i = 1, ..., n\}$.

Para nosotros será suficiente la norma euclidiana la cual también se conoce como norma de Frobenius y simbolizaremos como $\| \ . \ \|_F$. Para el producto escalar entre vectores usamos la notación usual $\langle . \rangle$, en particular trabajaremos con un producto escalar entre matrices llamado producto de Frobenius el cual se define de la siguiente manera:

$$A \bullet B = Tr(A^T B) \quad A, B \in \mathbb{R}^{m\times n}$$

3.2. Propiedades Básicas de Matrices

Proposicion 3.1. *Si $P \in \mathbb{R}^{m\times n}$ y $Q \in \mathbb{R}^{n\times m}$ entonces $Tr(PQ) = Tr(QP)$*

Demostración. Tenemos que si $P = [p_{ij}]$ y $Q = [q_{ij}]$ entonces

$$Tr(PQ) = \sum_{i=1}^{n}\sum_{k=1}^{m} p_{ik}q_{ki} = \sum_{k=1}^{m}\sum_{i=1}^{n} q_{ki}p_{ik} = Tr(QP).$$

□

Proposicion 3.2. *Si $U, V \in S\mathbb{R}^{n\times n}$ y P es no singular entonces $U \bullet V = (PUP^T) \bullet (P^{-T}VP^{-1})$ en particular si Q es ortogonal entonces $U \bullet V = (Q^TUQ) \bullet (Q^TVQ)$*

Demostración.

$$\begin{aligned}(PUP^T) \bullet (P^{-T}VP^{-1}) &= Tr((PUP^T)(P^{-T}VP^{-1})) \\ &= Tr(PUVP^{-1}) = Tr(UV) \\ &= U \bullet V\end{aligned}$$

si Q es ortogonal entonces $Q^{-1} = Q^T$ y tenemos el resultado. □

Proposicion 3.3. *Si $A \in S\mathbb{R}^{n\times n}$ entonces sus valores propios son reales y sus espacios propios mutuamente ortogonales.*

Demostración. Consideremos la matriz A como un elemento de $\mathbb{C}^{n\times n}$ y a $\mathbb{C}^n$ con el producto interno hermitiano el cual se define por:

$$\langle X, Y\rangle = \sum_{i=1}^{n} x_i\overline{y}_i$$

Sea λ un valor propio de A asociado al vector propio $X \neq 0$ entonces:

$$\begin{aligned}\lambda\langle X, X\rangle &= \langle X, \lambda X\rangle = \langle X, AX\rangle \\ &= \langle A^TX, X\rangle = \langle \lambda X, X\rangle = \overline{\lambda}\langle X, X\rangle\end{aligned}$$

como $X \neq 0$ tenemos $\lambda = \overline{\lambda}$ de donde $\lambda \in \mathbb{R}$.
Por otro lado sean λ y β valores propios distintos de A asociados respectivamente a los vectores X yZ. Como A es simétrica tenemos que:

$$\begin{aligned}\beta\langle X, Z\rangle &= \langle X, \beta Z\rangle = \langle X, AZ\rangle \\ &= \langle A^TX, Z\rangle = \langle AX, Z\rangle = \langle \lambda X, Z\rangle = \overline{\lambda}\langle X, Z\rangle = \lambda\langle X, Z\rangle\end{aligned}$$

por tanto $(\beta - \lambda)\langle X, Z\rangle = 0$ y como $\beta \neq \lambda$ $\langle X, Z\rangle = 0$ entonces X y Z son ortogonales luego E_λ es ortogonal a E_β □

Theorem 3.4. *$A \in S\mathbb{R}^{n\times n}$ si y solo si A es ortogonalmente diagonalizable, es decir existen matrices reales D diagonal y Q ortogonal tal que $A = Q^TDQ$*

Demostración. Sea $A \in S\mathbb{R}^{n\times n}$ por la proposición 3.3 y por el teorema fundamental del álgebra existe un valor propio λ de A. Veamos que para todo valor propio λ de A se tiene que $ma(\lambda) = mg(\lambda)$.
Sea $\{X_1, X_2, ..., X_d\}$ una base ortonormal para $E_\lambda(A)$ que podemos completar a una base ortonormal de $\mathbb{R}^n$

$$B = \{X_1, X_2, ..., X_d, X_{d+1}, ..., X_n\}$$

Consideremos la trasformación lineal $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, tal que $f(X) = AX$ para $X \in R^n$, la matriz que representa a f en la base canónica de $\mathbb{R}^n$ es A, en tanto que la matriz que representa a f en la base B es

$$M = \left[\begin{array}{cccc|c} \lambda & 0 & \cdots & 0 & \\ 0 & \lambda & \cdots & 0 & M_{12} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & \lambda & \\ \hline 0 & 0 & \dots & 0 & \\ 0 & 0 & \dots & 0 & M_{22} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & 0 & \end{array}\right]$$

Sea Y la correspondiente matriz de cambio de base entonces Y es invertible y tal que $M = Y^T AY$ y por tanto M tiene los mismos valores propios de A con las mismas multiplicidades algebraicas y geométricas.

Como M es simétrica entonces $M_{12} = 0$, por tanto

$$det(A - xI) = det(M - xI) = (\lambda - x)^d det(M_{22} - xI).$$

No es posible que λ sea raíz del polinomio $det(M_{22} - xI)$ pues de ser así la matriz M_{22} tendrá un vector propio $u \neq 0$ asociado a λ y si $u = (y_1, ..., y_{n-d})$ entonces $Z = (0, ..., 0, y_1, ..., y_{n-d})$ es un vector propio de M también asociado a λ.

Pero $dim(E_\lambda(M)) = dim(E_\lambda(A)) = d$ Por tanto las d primeras columnas de M forman una base para $E_\lambda(M)$ pues claramente estas columnas son vectores propios de M asociados a λ, luego $Z \in E_\lambda(M)$ lo cual es una contradicción y por tanto $det(M_{22} - xI)$ no tiene a λ como raíz y por tanto $ma(\lambda) = mg(\lambda) = d$ y así A es diagonalizable.
Si $\lambda_1, \lambda_2, ..., \lambda_k$ son los valores propios distintos de A, por álgebra lineal sabemos que $\mathbb{R}^n = \oplus_{i=1}^k E_{\lambda_i}$, por la proposición 3.3 tenemos que E_{λ_i} es ortogonal a E_{λ_j} para $i \neq j$ por tanto basta aplicar el proceso de ortonormalización de Gramt Smith en cada E_{λ_i} para obtener una base ortonormal B de $\mathbb{R}^n$ formada por vectores propios de A.

La matriz que representa a A en la base B es la matriz diagonal D formada por los valores propios de A donde cada valor propio aparece tantas veces como su multiplicidad geométrica.

Si Q es la matriz cuyas columnas son los vectores de la base B entonces $Q^{-1} = Q^T$ y $A = Q^T DQ$. Para la otra dirección es claro que si existen matrices Q ortogonal y D tales que $A = Q^T DQ$ luego $A \in S\mathbb{R}^{n\times n}$

□

Proposicion 3.5. *$S\mathbb{R}_+^{n\times n}$ es cerrado y convexo*

Demostración. Primero veamos que $S\mathbb{R}_+^{n\times n}$ es cerrado. Para ello sea P un punto límite de $S\mathbb{R}_+^{n\times n}$ entonces existe una sucesión $(A_n) \subset S\mathbb{R}_+^{n\times n}$ tal que $(A_n) \to P$, para todo $n \in \mathbb{N}$ tenemos que $P = (P - A_n) + A_n$, entonces para $x \in \mathbb{R}^n$ fijo:

$$x^T Px = x^T(P - A_n)x + x^T A_n x$$

como $(A_n) \to P$ con la norma de Frobenius entonces

$$\| P - A_n \| = (\sum_{i=1}^{i=n} (p_{ij} - a_{ij}^n)^2)^{\frac{1}{2}} \to 0$$

de donde $(p_{ij} - a_{ij}^n) \to 0$. Como

$$\| x^T(P - A_n)x \|^2 = (p_{ij} - a_{ij}^n)x_i x_j$$

entonces

$$(x^T(P - A_n)x) \to 0$$

Además como (A_n) en $S\mathbb{R}_+^{n\times n}$ entonces $(x^T A_n x)$ es una sucesión de reales positivos por tanto su límite es no negativo y así tenemos que $x^T P x \geq 0$. Como esto se cumple para todo vector $x \in \mathbb{R}^n$ podemos concluir que $P \in S\mathbb{R}_+^{n\times n}$ y por tanto $S\mathbb{R}_+^{n\times n}$ es cerrado.

Ahora probemos que $S\mathbb{R}_+^{n\times n}$ es convexo. Sean $A, B \in S\mathbb{R}_+^{n\times n}$ y $\lambda \in [0,1]$ tenemos que para $x \in \mathbb{R}^n$

$$x^T(\lambda A + (1 - \lambda B)x = \lambda x^T A x + (1 - \lambda)x^T B x \geq 0$$

de donde podemos concluir que

$$\lambda A + (1 - \lambda B) \in S\mathbb{R}_+^{n\times n}$$

para todo $\lambda \in [0,1]$, por tanto $S\mathbb{R}_+^{n\times n}$ es convexo. □

Definition 3.6. *una matriz $A \in \mathbb{R}^{n\times n}$ tiene una descomposición de cholesky si existe L triangular superior tal que $A = LL^T$*

Theorem 3.7. *Sea A una matriz simétrica real de orden n las siguientes afirmaciones son equivalentes:*

(i) $A \succ 0$

(ii) *A tiene factorización de cholesky*

(iii) $\lambda_i > 0$ *para* $i = 1, ..., n$

(iv) $\delta_i > 0$ *para* $i = 1, ..., n$

(v) *Existe una matriz B real de rango n tal que $A = B^T B$*

Demostración. Primero veamos la equivalencia entre (i) y (iv). Supongamos que $A \succ 0$ entonces $f(x) = x^T A x$ es una forma bilineal simétrica y definida positiva sobre cualquier báse de $\mathbb{R}^n$

$$B = \{b_1, b_2, ..., b_n\}$$

entonces f define un producto interno sobre $\mathbb{R}^n$ y por tanto existe una base ortonormal B_o de $\mathbb{R}^n$ respecto a f. La matriz que representa a f en B_o es I_n por tanto tenemos que $C^T A C = I$ donde C es la matriz de cambio de base de B_o a B entonces:

$$A = (C^{-1})^T C^{-1}$$

y por tanto

$$\delta_n = (det(C^{-1}))^2$$

de donde $\delta_n > 0$. Observemos ahora que $M_i(A)$ es la matriz de una forma bilineal sobre $V_i = gen\{b_1, b_2, ..., b_i\}$ para $i = 1, ..., n$ por tanto tenemos que $\delta_i = det(M_i(A)) > 0$. Ahora supongamos que $\delta_i > 0$ para $i = 1, ..., n$.

Construyamos a partir de la base B una base $D = \{d_1, d_2, ..., d_n\}$ tal que

(i) $f(d_i, d_j) = 0$ si $i \neq j$ y

(ii) $f(d_i, d_i) = \alpha_{ii} > 0$ para $i, j = 1, 2, ..., n$ en la base D

Queremos que si $(x_1, x_2, ..., x_n)$ son las coordenadas del vector x entonces $f(x, x) = \alpha_{11}(x_1)^2 + \alpha_{22}(x_2)^2 + ... + \alpha_{nn}(x_1)^2 \geq 0$ lo que mostraría que la función cuadrática es definida positiva.

Para construir la base

$$D = \{d_1, d_2, ..., d_n\}$$

vemos que es suficiente encontrar $d_i = \alpha_{11}b_1 + ... + \alpha_{ii}b_i$ tales que $f(d_i, b_j) = 0$ para $j < i$ y $f(d_i, b_i) = 1$ para todo i así tenemos que si

$$j < i \Rightarrow f(d_i, d_j) = f(d_i, \sum_{i=1}^{j} \alpha_{jk}b_k) = \sum_{i=1}^{j} f(d_i, b_j) = 0$$

también

$$f(d_i, d_j) = f(d_j, b_i) = 0$$

si $j > i$ por simetría y $f(d_i, d_i) = f(d_i, \sum_1^i \alpha_{jk}b_k) = \alpha_{ii}f(d_i, b_i) = \alpha_{ii}$.

Para los vectores d_i tenemos que:

$$d_1 = \alpha_{11}b_1, d_2 = \alpha_{21}b_1 + \alpha_{22}b_2, ..., d_i = \alpha_{i1}b_1 + \alpha_{i2}b_2 + ... + \alpha_{ii}b_i$$

luego para d_1 tenemos que $\alpha_{11} = \frac{1}{f(b_1,b_1)}$ para que se cumplan las condiciones (a) y (b) para d_i debemos tener que:

$$f(\sum_{k=1}^{i} \alpha_{ik}(b_k, b_1)) = 0, f(\sum_{k=1}^{i} \alpha_{ik}(b_k, b_2)) = 0, ..., (\sum_{k=1}^{i} \alpha_{ik}f(b_k, b_i)) = 1$$

equivalentemente: $(\sum_{k=1}^{i} \alpha_{ik}f(b_k, b_1)) = 0, (\sum_{k=1}^{i} \alpha_{ik}f(b_k, b_2)) = 0, ..., (\sum_{k=1}^{i} \alpha_{ik}f(b_k, b_i)) = 1$.

Observemos que este es un sistema no homogéneo en las α_{ij} cuya matriz de coeficientes es $M_i(A)$ con determinante $\delta_i > 0$, por tanto existe solución para las α_{ij} lo que garantiza la existencia de los vectores d_i que cumplan (i) y (ii).

Por la regla de Cramer tenemos que: $\alpha_{ii} = \frac{\delta_i}{\delta_{i-1}} > 0$. Es fácil ver que (i) y (iii) también son equivalentes. Primero supongamos que $A \succ 0$ y sean λ un valor propio de A y $X \neq 0$ un vector propio asociado a λ entonces $\lambda = \frac{x^T Ax}{\|x\|} > 0$.

Ahora supongamos que los valores propios de A son todos positivos entonces $A \succeq 0$ y por el teorema 1.4 $A = Q^T DQ$ donde Q es ortogonal y D es diagonal con los valores propios de A sobre

la diagonal principal. Como los valores propios de A son positivos entonces D tiene raíz cuadrada por tanto $A = Q^T DQ = Q^T(\sqrt{D})^2 Q = ((\sqrt{D})Q)^T(\sqrt{D})Q = B^T B$ donde $B = (\sqrt{D})Q^T$ es invertible.

Entonces para $x \in \mathbb{R}^n$ tenemos que $x^T Ax = x^T B^T Bx = (Bx)^T BX = (\parallel Bx \parallel)^2 \geq 0$ y si $x^T Ax = 0$ entonces $\parallel Bx \parallel^2 = 0$ de donde $\parallel Bx \parallel = 0$ lo que implica $x = 0$ pues B es invertible luego A es definida positiva.

Este mismo razonamiento nos muestra la equivalencia entre (i) y (v). Ahora veamos que (i) implica (ii). Para ello observemos que si U es triangular superior entonces $A = U^T U$ y entonces al igualar la primera componente de $U^T U$ con la primera componente de A obtenemos $u_{11} = a_{11}$ como A es definida positiva $a_{11} > 0$ luego $u_{11} = \sqrt{(a_{11})}$ en general al igualar el producto de la primera fila de U^T por la j-ésima columna de U con la componente $1j$ de A obtenemos $u_{11}u_{1j} = a_{1j}$ de donde $u_{1j} = \frac{a_{1j}}{u_{11}}$ y de esta manera calculamos todos los elementos de la primera fila de U.

Ahora al hacer el producto de la segunda fila de U^T por la segunda columna de U tenemos: $u_{12}^2 + u_{22}^2 = a_{22}$ de donde

$$a_{22} = \sqrt{(a_{22} - u_{12}^2)}$$

de forma similar si multiplicamos la segunda fila de U^T por la columna j de U e igualando el resultado con la componente a_{2j} tenemos:

$$u_{12}u_{1j} + u_{22}u_{2j} = a_{2j}$$

de donde

$$u_{2j} = \frac{(a_{2j} - u_{2j}u_{1j})}{u_{22}}$$

y de manera general tenemos $t = a_{ii} - (\sum_{k=1}^{i-1} u_{ki})$ para $i = 1, ..., n$, $u_{ii} = \sqrt{t}$, $u_{ij} = \frac{a_{ij} - (\sum_{k=1}^{i-1} u_{ki}u_{kj})}{u_{ii}}$.
En el otro sentido es claro que si A tiene una descomposición de Cholesky entonces es definida positiva. □

Theorem 3.8. *Sea $A \in S\mathbb{R}^{n\times n}$ las siguientes afirmaciones son equivalentes:*

(i) A es semidefinida positiva

(ii) Los valores propios de A son no negativos

(iii) Existe una matriz B real de orden n tal que $A = B^T B$

Demostración. Supongamos (i) y sea λ un valor propio de A y $X \leq 0$ un vector propio asociado a λ tenemos que $0 \leq X^T AX = X^T \lambda X = \lambda X^T X = \parallel X \parallel^2$ de donde $\lambda \geq 0$.

Ahora supongamos (ii) por el teorema 1.3 $A = P^T DP$ con $P^{-1} = P^T$ y $D = Diag(\lambda_1, \lambda_2, ..., , \lambda_n)$ donde $\lambda_1, \lambda_2, ..., , \lambda_n$ son los valores propios de A.
Como $\sqrt{\lambda_i}$ existe para$i = 1, ..., n$, tenemos que $D = (\sqrt{D})^2$ donde $\sqrt{D} = Diag(\sqrt{\lambda_1}, \sqrt{\lambda_2}, ..., \sqrt{\lambda_n})$
Así tenemos que

$$A = P^T DP = P^T(\sqrt{D})^2 P = P^T(\sqrt{D})(\sqrt{D})P = (\sqrt{D}P)^T \sqrt{D}P = B^T B$$

con $B = \sqrt{D}P$.

Ahora supongamos que $A = BB^T$ donde B es real, luego para $X \in \mathbb{R}^n$, $X^TAX = X^TB^TBX = (BX)^TBX = \| BX \|^2 \geq 0$ □

Proposicion 3.9. *Si $U \succeq 0$ entonces $u_{kk} \geq 0$ Para $k = 1, ..., n$ y si $u_{kk} = 0$ algún $1 \leq k \leq n$ entonces \$u_{kj} = u_{jk} = 0$ para $j = 1, ..., n$. Similarmente si $U \succ 0$ entonces $u_{kk} > 0$*

Demostración. ya que $u_{kk} = e_k^TUe_k$ y $U \succeq 0$ entonces $u_{kk} \geq 0$, para el caso en que $U \succ 0$ tenemos que $u_{kk} > 0$.
Ahora supongamos que $U \succeq 0$ y $u_{kk} = 0$. Veamos el caso especial en que U es simétrica por el hecho 6 tenemos que $U = B^TB$ de donde $0 = u_{kk} = b_k^Tb_k$ y así $b_k = 0$ luego $u_{kj} = u_{jk} = b_k^Tb_j = 0$ para todo j.

Ahora consideremos el caso general en que U es cualquier matriz semidefinida positiva tal que $u_{kk} = 0$. Sea $A = U + U^T$ $A \succ 0$, simétrica y $a_{kk} = 2ukk = 0$, por el resultado anterior $a_{kj} = a_{jk} = 0$ de donde $u_{kj} + u_{jk} = 0$ o bien $u_{kj} = -u_{jk}$ para todo j, así tenemos que si $B = U^TU$ entonces $b_{kk} = \sum u_{kj}u_{jk} = -\sum u_{kj}^2 = \sum u_{jk}^2$, como $B \succeq 0$ debemos tener que $u_{kj} = u_{jk} = 0$. □

Proposicion 3.10. *Sean $U \in S\mathbb{R}_+^n$ y $P \in M_{n\times m}(\mathbb{R})$ entonces $PUP^T \succeq 0$. Si $P \in GL_n(\mathbb{R})$ tendremos $U \succ 0$ si y solo si $PUP^T \succ 0$*

Demostración. Para todo $x \in \mathbb{R}^n$, tenemos que $x^T(PUP^T)x = (P^Tx)^TU(P^Tx) \geq 0$ ya que $U \succeq 0$.Supongamos que $P \in GL_n(\mathbb{R})$, si $U \succ 0$ entonces $U \succeq 0$ y aplicando el resultado precedente tenemos $PUP^T \succeq 0$.

Ahora sea $x \in \mathbb{R}^n$ tal que $x^T(PUP^T)x = 0$ como $U \succ 0$ entonces $P^Tx = 0$ y ya que $P \in GL_n(\mathbb{R})$ también $P^T \in GL_n(\mathbb{R})$ y asi $PUP^T \succ 0$. Ahora supongamos que $PUP^T \succ 0$, tenemos que $U = P^{-1}(PUP^T)(P^{-1})^T$ y por el resultado anterior tenemos $U \succ 0$ □

Proposicion 3.11. *Si $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21}^T & U_{22} \end{pmatrix} \succeq 0$ entonces $U_{11} \succeq 0$*

Demostración. Supongamos que $U_{11} \in M_k\mathbb{R}$ y sea $x \in \mathbb{R}^k$ entonces para $Z = (x^T, 0)$ Tenemos que $0 \leq ZUZ^T = ZU_{11}Z^T$ de donde concluimos que $U_{11} \succeq 0$.

Veamos que este resultado se cumple para cuando las matrices son positivas definidas, en efecto si $U \succ 0$ entonces por definición $U \succeq 0$ y por el resultado precedente $U_{11} \succeq 0$ veamos que $U \succ 0$ sea $X \in \mathbb{R}^k$ si $x^TU_{11}x = 0$ para $Z = (x^T, 0)$ claramente tendremos $ZUZ^T = 0$ de donde $Z = 0$ y entonces $x = 0$.
Combinando los hecho 8 y 9 tenemos que existe una matriz de permutación que siempre es invertible tal que

$$PUP^T = \begin{pmatrix} U_{22} & U_{12}^T \\ U_{21}^T & U_{11} \end{pmatrix}$$

de donde concluimos que U_{22} es semidefinida positiva o definida positiva según el caso. □

Proposicion 3.12. *Sea $U \in S\mathbb{R}^{n\times n}$ tal que $U = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$ con A, C simétricas y $A \succ 0$ entonces $U \succeq 0 (U \succ 0)$ si y solo si $C - B^TA^{-1}B \succeq 0 (\succ 0)$. La matriz $C - B^TA^{-1}B$ se llama complemento de shur de A*

Demostración. Para probar este hecho usaremos la siguiente factorización de la matriz U.

$$U = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} = \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

. Observemos que

$$P^T = \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix}$$

es invertible. Por el hecho 9

$$\begin{pmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{pmatrix} = (P^{-1})^T U P^{-1}$$

es semidefinida (definida) positiva según U lo sea. También por el hecho 9 tenemos

$$C - B^T A^{-1} B \succeq 0 (\succ 0)$$

según lo que sea U. Para probar el reciproco supongamos que $C - B^T A^{-1} B \succeq 0 (\succ 0)$, como

$$U = P^T \begin{pmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{pmatrix} P$$

siendo P invertible y $A \succ 0$ entonces $\begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \succeq 0 (\succ 0)$ y nuevamente por el hecho 9 tenemos que $U \succeq 0 (\succ 0)$ □

Proposicion 3.13. *Si $U \in S\mathbb{R}^{n \times n}$ entonces $x^T U x = U x x^T$*

Demostración. $U x x^T = Tr(U(xx^T)) = Tr(x^T U x) = x^T U x$ □

Proposicion 3.14. $S\mathbb{R}_+^{n \times n} = (S\mathbb{R}_+^{n \times n})^* = \{V \mid U \bullet V \geq 0, U \in S\mathbb{R}_+^{n \times n}\}$

Demostración. Veamos que

$$S\mathbb{R}_+^{n \times n} \subseteq (S\mathbb{R}_+^{n \times n})^*$$

Sean $U, V \in S\mathbb{R}_+^{n \times n}$, tenemos que

$$V \bullet U = Tr(VU) = Tr(VU^{\frac{1}{2}}U^{\frac{1}{2}}) = Tr(U^{\frac{1}{2}}VU^{\frac{1}{2}}) \geq 0$$

ya que

$$U^{\frac{1}{2}}VU^{\frac{1}{2}} \geq 0$$

asi tenemos que $V \in (S\mathbb{R}_+^{n \times n})^*$. Ahora probemos que

$$(S\mathbb{R}_+^{n \times n})^* \subseteq (S\mathbb{R}_+^{n \times n})$$

para ello supongamos que existe $U \notin S\mathbb{R}_+^{n \times n}$ entonces existe $x \in \mathbb{R}^n$ tal que $x^T U x < 0$ de donde

$$U \bullet (xx^T) = x^T U x < 0$$

y por tanto

$$U \notin (S\mathbb{R}_+^{n \times n})^*$$

□

Proposicion 3.15. *Si $U \succ 0$ entonces $U \bullet V$ para toda $V \neq 0 \succeq 0$ y además $\{V \succeq |U \bullet V \leq \beta\}$ es acotado para cada real positivo β.*

Demostración. Sea λ el menor valor propio de U tenemos que $U \bullet V = (U - \lambda I) \bullet V + \lambda I \bullet V \geq \lambda I \bullet V = \lambda I \bullet D(V) = \lambda \parallel \lambda(V) \parallel_1 \geq \lambda \parallel V \parallel_F$ para $V \succeq 0$ como $V \neq$ entonces $\parallel V \parallel_F > 0$ y así $U \bullet V > 0$ y también tenemos $\lambda \parallel V \parallel_F \leq \beta$ de donde $\parallel V \parallel_F \leq \frac{\beta}{\lambda}$ lo cual demuestra la segunda afirmación del enunciado. □

Proposicion 3.16. *Si $U, V \succeq 0$ entonces $U \bullet V = 0$ si y solo si $UV = 0$*

Demostración. $U = P^T D(U) P$ y $U = Q^T D(V) Q$ entonces $0 = U \bullet V = (P^T D(U) P)(Q^T D(V) Q) = Tr[(P^T D(U) P)(Q^T D(V) Q)] = Tr[(QP)^T D(U) D(V)(QP)] = Tr(UV) = \parallel UV \parallel_F$ de donde $UV = 0$. Ahora supongamos que $UV = 0$ entonces $U \bullet V = Tr(UV) = Tr(0) = 0$ □

Proposicion 3.17. *Si $U, V \in S\mathbb{R}^{n \times n}$ entonces $UV = VU$ si y solo si UV es simétrica si y solo si U y V son simultáneamente diagonalizables.*

Demostración. Supongamos que $UV = VU$ entonces $(UV)^T = V^T U^T = VU = UV$ o sea UV es simétrica. Si $UV \in S\mathbb{R}^{n \times n}$ entonces $UV = (UV)^T = V^T U^T = VU$ Ahora veamos que U y V conmutan si y solo si diagonalizan simultáneamente.

Supongamos que $UV = VU$, por el teorema 1.4 $U = Q^T D(U) Q$ donde Q es una matriz ortogonal cuyas columnas son los vectores propios de U y forman una base de $\mathbb{R}^n$.

Sea λ un valor propio de U y X un vector propio asociado a λ, observemos que $UVX = VUX = \lambda VX$ lo que significa que $VX \in E_\lambda(U)$. Así tenemos que los espacios $E_\lambda(U)$ son invariantes por la acción de V. Como $\mathbb{R}^n = \bigoplus E_\lambda(U)$ entonces la matriz H que representa a V en la base de $\mathbb{R}^n$ formada por los vectores propios de U esta formada por los bloques $H_1, H_2, ..., H_n$ donde H_i es la matriz que representa la restricción de V en $E_i(\lambda)$.

Como la base B de $\mathbb{R}^n$ formada por las columnas de Q es ortogonal entonces la matriz C de cambio de base de B a la base canónica de $\mathbb{R}^n$ está formada por las columnas de B es decir $C = Q$. Como $H = C^T V C$ entonces H es simétrica y por tanto diagonalizable sobre B que es lo que queríamos probar. □

3.3. Teoría de grafos

Los grafos son una herramienta muy útil para modelar diferentes tipos de problemas y particularmente resultan ser un formato natural para los problemas combinatorios.

Recordemos que un grafo es un par ordenado $G = (V, E)$ donde V es un conjunto finito y $E \subset V \times V - \{(i, i) | i = 1, ..., n\}$, en el presente trabajo siempre tendremos que $V = \{1, ..., n\}$ Cuando queremos hacer énfasis en que V es el conjunto de vertices del grafo G escribimos $V(G)$ en lugar de V pero si no hay lugar a confusión simplemente representamos el conjunto de vertices por V y la misma situación la tenemos para el conjunto E. El orden de G el el número de elementos de V en tanto que el tamaño de G el el número de elementos de E. Si $e = (u, v) \in E$ decimos que u y v son incidentes con e o que e es incidente con u y v en tanto que u y v soa adyacentes

Decimos que el grafo G es vacío si V es el conjunto vacío, discreto si V es no vacío y E es vacío, es completo si $E = V \times V - \{i, i) | i = 1, ..., n\}$ en este caso se dice que G es un $n - clique$. Dado un grafo G decimos que el grafo H es subgrafo de G y escribimos $H \leq G$ se $V(H) \subset V(G)$ y $E(H) \subset E(G)$. Si $F \subset V(G)$ entonces el grafo inducido por F es $(V(F), E(F))$ donde $V(F) = F$ y $E(F) = \{(u, v) \in E | u, v \in F\}$. Decimos que $S \subset V$, S no vacío es un conjunto independiente si el grafo inducido por S es discreto. Decimos que S cubre a G si todo vértice de G es adyacente con algún vértice en S.

El complemento de un grafo G es $\overline{G} = (V, \overline{E})$ donde $\overline{E}$ es el complemento de E. Una función $f : V(G) \to \{1, ..., k\}$ tal que $f(i) \neq f(j)$ si $i \neq j$ se llama un k-coloreamiento de G

4. Optimización Semidefinida

Un problema de optimización semidefinida consiste en maximizar o minimizar una función lineal en las variables de una matriz simétrica sujeta a un conjunto finito de restricciones lineales de igualdad y a la restricción de que la matriz de variables sea semidefinida positiva. La forma usual de representar un problema de optimización semidefinida es:

$$\begin{aligned} \min_{X \in S\mathbb{R}^{n\times n}} \; & C \bullet X \qquad (P) \\ s.a \quad & A_i \bullet X = b_i, \; i = 1, \ldots, m \\ & X \succeq 0 \end{aligned}$$

Donde $A_i \in SR^{n\times n}$, $b \in R^m$, $C \in SR^{n\times n}$ son los parametros del problema y $X \in SR^{n\times n}$ es la variable. Cuando escribimos un problema de optimización semidefinida de esta manera decimos que está en la forma primal estándar. Su forma dual estándar es:

$$\begin{aligned} \max_{(y,S)\in\mathbb{R}^m \times S\mathbb{R}^{n\times n}} \; & b^T y \qquad (D) \\ s.a \quad & \sum_{i=1}^{m} y_i A_i + S = C \\ & S \succeq 0 \end{aligned}$$

Inicialmente no hay garantía de que los problemas (P) y (D) alcancen su valor óptimo en un punto de su dominio de hecho no podemos asegurar que estos problemas sean acotados, sin embargo en el presente trabajo en cada una de las situaciones en que plantearemos problemas de optimización semidefinida tendremos condiciones suficientes para que hayan soluciones en los correspondientes dominios.

Proposicion 4.1. *Los problemas (P) y (D) son duales lagrangianos uno del otro.*

Demostración. El dual lagrangiano de (P) es

$$\max_{y\in\mathbb{R}^m} h(y)$$

Donde

$$h(y) = \min_{X\in S\mathbb{R}^{n\times n}} C \bullet X - \sum_{i=1}^{m} y_i (A_i \bullet X - b_i) \text{ sujeto a: } X \succeq 0$$

de donde

$$h(y) = \sum_{i=1}^{m} y_i b_i + \min_{X \in S\mathbb{R}^{n\times n}} (C - \sum_{i=1}^{m} y_i A_i) \bullet X \text{ sujeto a: } X \succeq 0$$

$$= \sum_{i=1}^{m} y_i b_i + \min_{X \in S\mathbb{R}^{n\times n}} S \bullet X \text{ sujeto a: } X \succeq 0$$

En este punto podemos considerar la familia de matrices $X_\epsilon = \epsilon I$ claramente $\epsilon I \succeq 0$ para todo $\epsilon > 0$ y tenemos que

$$\min_{X \in S\mathbb{R}^{n\times n}} S \bullet X = \lim_{\epsilon \to 0} S \bullet X_\epsilon = 0$$

Ahora, si para $y \in \mathbb{R}^m$,

$$C - \sum_{i=1}^{m} y_i A_i$$

no es semidefinida positiva, entonces por el hecho 2.14 existe

$$U \in S\mathbb{R}^{n\times n}_+$$

tal que

$$C - \sum_{i=1}^{m} y_i A_i \bullet U < 0$$

de donde

$$\min_{X \in SR^{n\times n}} (C - \sum_{i=1}^{m} y_i A_i) \bullet X$$
$$X \succeq 0$$

$= -\infty$ entonces $h(y) = -\infty$ de donde podemos afirmar que h no toma su valor máximo en y, en conclusión tenemos que h toma su valor máximo en

$$F = \{y \in \mathbb{R}^m | (C - \sum_{i=1}^{m} y_i A_i) \succeq 0\}$$

además para $y \in F$ tenemos que para $X_\epsilon = diag(\epsilon)$ donde $\epsilon > 0$.

$$(C - \sum_{i=1}^{m} y_i A_i) \bullet X_\epsilon = \epsilon Tr(C - \sum_{i=1}^{m} y_i A_i) \to 0$$

cuando $\epsilon \to 0$ de esta manera tenemos que $h(y) = b^T y$ y por tanto tenemos que

$$h(y) = \max_{(y,S) \in R^m \times SR^{n\times n}} b^T y \text{ sujeto a: } \sum_{i=1}^{m} y_i A_i + S = C, S \succeq 0 \qquad (D)$$

Ahora veamos que (P) es el dual de (D).

Tenemos que el dual lagrangiano de (P)

$$\min_{X \in S\mathbb{R}^{n\times n}} h(X)$$

Donde

$$h(X) = \max_{y\in\mathbb{R}^m, S\succeq 0} b^T y - \sum_{i=1}^{m} y_i(A_i + S - C) \bullet X = C \bullet X - \min_{X\in S\mathbb{R}^{n\times n}} \min h(X)$$

Donde

$$h(X) = \min_{y\in\mathbb{R}^m} (\sum_{i=1}^{m} y_i A_i \bullet X - b^T y) - \min_{S\succeq 0} S \bullet X = C \bullet X$$

$$= \sum_{i=1}^{m} y_i A_i \bullet X - b^T y - \min_{S\succeq 0} S \bullet X = C \bullet X$$

$$= \min_{y\in\mathbb{R}^m} \sum_{i=1}^{m} (A_i \bullet X - b_i) y_i - \min_{S\succeq 0} S \bullet X$$

Si para algún $i = 1, ..., m$, $A_i \bullet X - b_i \neq 0$, cómo $y \in \mathbb{R}^m$ entonces $h(X) = \infty$ entonces $h(X)$ alcanza su valor mínimo sobre los $X \in S\mathbb{R}^{n\times n}$ tal que $A_i \bullet X - b_i = 0$ para $i = 1, ..., m$ por el hecho 2.13 $h(X) = \infty$ a menos que $X \succeq 0$ por tanto concluimos que

$$\min_{X\in S\mathbb{R}^{n\times n}} h(X) =$$

$$\begin{aligned} \min_{X\in SR^{n\times n}} \; & C \bullet X \quad (P) \\ & A_i \bullet X = b_i, ß = 1, \ldots, m \\ & X \succeq 0 \end{aligned}$$

□

Theorem 4.2. *Dualidad Débil Si X es una solución factible de (P) y (y, S) es una solución factible de D entonces $C \bullet X - b^T y = X \bullet S \geq 0$*

Demostración. Tenemos que:

$$C \bullet X - b^T y = (\sum_{i=1}^{m} y_i A_i + S) \bullet X - b^T y$$

$$= \sum_{i=1}^{m} (A_i \bullet X) y_i + S \bullet X - b^T y =$$

$$\sum_{i=1}^{m} b_i y_i + S \bullet X - b^T y = b^T y + S \bullet X - b^T y = S \bullet X$$

.

Ahora tengamos en cuenta que

$$S \bullet X = Tr(SX) = Tr(SX^{\frac{1}{2}} X^{\frac{1}{2}}) = Tr(X^{\frac{1}{2}} S X^{\frac{1}{2}})$$

En este punto aprovechamos que $X \succeq 0$ y por tanto tiene una raíz cuadrada, así como las propiedades de la traza. Por otro lado la matriz $X^{\frac{1}{2}}SX^{\frac{1}{2}} \succeq 0$ ya que para todo vector $u \in \mathbb{R}^n$ tenemos que $U^T X^{\frac{1}{2}} S X^{\frac{1}{2}} U = (UX^{\frac{1}{2}})^T S X^{\frac{1}{2}} U \geq 0$ ya que $S \succeq 0$.

Como $SX \succeq 0$ entonces todos sus valores propios son positivos y cómo $S \bullet X = Tr(SX)$ es la suma de los valores propios de SX podemos concluir que $S \bullet X \geq 0$ □

Queremos saber cómo se relacionan los problemas (P) y (D) ya hemos visto que cada uno de ellos es el dual lagrangiano del otro y ahora el teorema de dualidad débil nos dice que las soluciones factibles de D están acotadas superiormente por las soluciones factibles de (P). Es importante ver como estan relacionados estos dos problemas esta realción está expresadajgwudwdt67 en el teorema de dualidad fuerte pero antes de enunciar este teorema veamos algunas definiciones:

Definition 4.3. *Considere los siguientes conjuntos:*

(i) $F(P) = \{X \in S\mathbb{R}^{n\times n} | A_i \bullet X = b_i, i = 1, ..., m, X \succeq 0\}$

(ii) $F^0(P) = \{X \in F(P) | X \succ 0\}$

(iii) $F(D) = \{(y, S) \in \mathbb{R}^m \times S\mathbb{R}^{n\times n} | \sum_{i=1}^m y_i A_i + S = C, S \succeq 0\}$

(iii) $F^0(D) = \{(y, S) \in F(D) | S \succ 0\}$

Theorem 4.4. *Dualidad Fuerte Si $F(P)$ y $F^0(D)$ son no vacíos. entonces (P) tiene un conjunto compacto no vacío de soluciones y los valores óptimos de (P) y (D) coinciden.*

Demostración. Ver [3] □

5. Complejidad computacional.

Un algoritmo es un conjunto de reglas para realizar una Tarea. A principios del siglo XX hubo varios intentos para formalizar la noción de algoritmo. La alternativa principal y generalmente aceptada es el concepto de máquina de Turing (propuesta en 1926 por el matemático inglés Alan Turing) que describimos a continuación.

Definición 5.1. Una máquina de Turing determinista consiste de una cinta infinita dividida en casillas y una cabeza que hace operaciones de lectura y escritura. Fijamos un alfabeto Γ finito de los símbolos que la máquina sabe leer y escribir y también un conjunto finito F de estados internos de la máquina. En cada unidad de tiempo la máquina lee el símbolo $s \in \Gamma$ escrito en la cinta y hace las siguientes operaciones:

1. Escribe un símbolo distinto en el cuadrado que está leyendo.
2. Cambia su estado interno.
3. Mueve la cabeza lectora una casilla a la izquierda o a la derecha en la cinta o se queda estacionaria.

Las reglas de transición que describen completamente el funcionamiento de la máquina son una función δ que, para cada estado f y cada símbolo $s \in \Gamma$ nos dice

$$\delta(f, s) = (f', s', k)$$

Lo cual quiere decir: Si la máquina lee el símbolo s en el estado interno f entonces escribirá el símbolo s', pasará al estado interno f' y moverá la cabeza lectora a la izquierda, a la derecha o se queda quieta según $k \in \{I, D, Q\}$. Una máquina de Turing se dice no determinista si δ no es una función, es decir si para input (f, s) permitimos que la máquina pase a uno de un conjunto *de varios estados posibles*

$$\delta(f, s) = \{(f'_1, s'_1, k_1), \ldots, (f'_m, s'_m, k_m)\}$$

Así, con un mismo input una máquina de Turing no determinista tiene varias evoluciones futuras.

Asumimos además que todas nuestras máquinas de Turing, deterministas o no, tienen dos estados especiales $A, R \in F$ que se llaman estado de aceptación y de rechazo en los que la máquina de Turing se detiene.

Definición 5.2. Dado un alfabeto finito Γ el conjunto de todas las cadenas sobre Γ se denota por Γ^*

Observación 5.1. Lo importante de una máquina de Turing es que para especificarla completamente necesitamos sólo una cantidad finita de información: el alfabeto Γ, el conjunto de estados internos F y la función δ (o correspondencia δ para el caso no determinista) cuyo dominio es el conjunto finito $\Gamma \times F$ luego también puede especificarse usando una cantidad finita de información.

Definición 5.3. El input de una máquina de Turing M es lo que está escrito en la cinta cuando la máquina empieza su ejecución. Decimos que una máquina de Turing (determinista o no) acepta el input w si existe alguna ejecución de M con input w que lleva a la máquina al estado de aceptación A. De manera semejante decimos que M rechaza el input w si existe alguna ejecución de w con input w que lleva a la máquina al estado de rechazo R.

Definición 5.4. El Lenguaje aceptado por una máquina de Turing M esta formado por el conjunto de cadenas de Γ^* cuyo computo termina en el estado A de M. El lenguaje aceptado por una maquina de Turing M lo denotaremos por $L(M)$. Tenemos que:

$$L(M) = \{w \in \Gamma^* | \text{el computo de } M \text{ sobre } w \text{ termina en el estado } A \text{ de } M\}$$

Definición 5.5. Dada una máquina de Turing M y $w \in \Gamma^*$, $T(w)$ representa el numero de aplicaciones de la función δ para que la cadena w sea aceptada o rechazada.

Definición 5.6. Una máquina de Turing M es de tiempo polinomial si existe un polinomio $p : N \longrightarrow N$ tal que $T(w) \leq p(|w|)$, donde $|w|$ representa el número de simbolos de Γ que hay en la cadena w.

Definición 5.7. Un problema de decisión es una función de un conjunto I de instancias a un conjunto de dos valores $V = \{0, 1\}$.
Ejemplos:

1. Sea I es el conjunto de grafos finitos y $f : I \to V$ es la función que asigna 1 al grafo g si g es 3-coloreable y 0 de lo contrario.

2. Sea I el conjunto de fórmulas del calculo proposicional en letras $\{p_i : i \in \mathbb{N}\}$ y $h : I \to V$ es la función que asigna 1 a la fórmula ϕ ssi existe una valuación (asignación de valores de verdad a las letras proposicionales) que hace que ϕ sea verdadera.

Definición 5.8. Un problema de decisión $h : I \to V$ está en P si existe una máquina de Turing determinista que acepta las instancias positivas de I (aquellas en las que h vale 1) y rechaza las instancias negativas de I (en as que h vale 0) en un numero de pasos polinomial en la longitud del input (el número de casillas no vacías). Un problema de decisión está en NP si existe una máquina de Turing **no determinista** que acepta las instancias positivas de I (aquellas en las que h vale 1) y rechaza las instancias negativas de I (en as que h vale 0) en un numero de pasos polinomial en la longitud (el número de casillas no vacías) del input.

Observación 5.2. Es posible verificar que un problema de decisión está en NP si y solo si para toda instancia positiva y candidato a solución es posible verificar en tiempo polinomial determinista que el candidato efectivamente satisface la propiedad.
Ejemplo: 2-sat está en NP porque dada una fórmula y una asignación de valores de verdad para las letras proposicionales es posible verificar, en tiempo polinomial determinista si **esa** asignación satisface o no la fórmula.

Como toda máquina de Turing determinista es un tipo especial de máquina de Turing no determinista es claro que $P \subseteq NP$. La pregunta de si $P = NP$ es uno de los problemas abiertos más interesantes de las matemáticas.

6. Algunos problemas de la clase NP

Dado un problema de decisión h podemos usar la observación 5.2 para determinar si $h \in NP$. A continuación presentamos una lista de problemas de la clase NP.

2-sat: Una instancia de este problema consiste en un conjunto de clausulas $\{C_1, ..., C_m\}$ y un conjunto de variables booleanas $\{x_1, ..., x_n\}$ donde $C_k = x_i \vee x_j$ para $1 \leq k \leq m$ y $1 \leq i \leq j \leq n$ hay que determinar si existe una asignación de valores de verdad de las variables $x_1, ..., x_n$ que hagan verdadera cada clausula $C_1, ..., C_m$.

n-sat: Una instancia de este problema consiste en un conjunto de clausulas $\{C_1, ..., C_m\}$ y un conjunto de variables booleanas $\{x_1, ..., x_r\}$ donde cada C_k es una disyunción de máximo n de las variables $x_1, ..., x_r$.
Se tarta de determinar si existe una asignación de valores de verdad de las variables $x_1, ..., x_r$ que hagan verdadera cada clausula $C_1, ..., C_m$.

Clique: Una instancia de este problema consiste en un grafo G y un entero positivo k.
Se trata de encontrar un conjunto de k vértices mutuamente adyacentes.

Set packing: Una instancia de set packing consisite en una familia S_j de conjuntos y un entero positivo k .
Se trata de determinar si S_j contiene k conjuntos mutuamente disyuntos.

Vertex cover: Una instancia de problema consiste en un grafo $G = (V, A)$ y un entero positivo r.
Se trata de encontrar $N \subset V$ tal que $|N| \leq r$ y cada arco de G es adyacente con un vértice en N.

Set Covering: Una instancia de Set covering consiste en una familia de conjuntos finitos $\{S_j\}$ y un entero positivo r.
Se trata de determinar si existe una familia $\{T_h\} \subset \{S_j\}$ tal que $|\{T_h\}| \leq r$ y $\cup T_h \subset \cup S_j$.

Undirected Hamilton Circuit: Una instancia de este problema consiste en un grafo G.
Se tarta de determinar si G contiene un ciclo que contiene cada vértice del grafo G exactamente una vez.

Chromatic Number: Una instancia de este problema consiste en un grafo G y un entero positivo k.
Se trata de determinar si G es k-coloreable es decir si existe una función $\phi : V \to Z_k$ tal que si u y v son adyacentes entonces $\phi(u) \neq \phi(v)$

Exact Cover: Una instancia de Exact cover consiste en una familia de subconjuntos $\{S_j\}$ de un conjunto $\{x_1, ..., x_t\}$.
Se trata de hallar una subfamilia disyunta dos a dos $\{T_h\} \subset \{S_j\}$ tal que $\cup T_h = \cup S_j$

Knapsack: Una instancia de Knapsack consiste en un vector $(a_1, a_2, ..., a_r, b) \in Z^{r+1}$.
Se trata de determinar si la ecuación $\sum_{i=1}^{r} x_i = b$ tiene una solución en $\{0, 1\}^r$

Partition: Una instancia de Partition consiste en un vector $(c_1, c_2, ..., c_s) \in Z^s$.
Se trata de determinar si existe $I \subset \{1, 2, ..., s\}$ tal que $\sum_{h \in I} c_h = \sum_{h \notin I} c_h$

Number-Cut: una instancia de Number- cut consiste en un grafo G una función $w : E(G) \to \mathbb{Q}^+ \cup \{0\}$ y un entero positivo k.
Se tarta de determinar si existe un subconjunto $S \subset V$ tal que $\sum w(u, v) \geq k$ donde $u \in S$ y $v \notin S$

Observación 6.1. Para probar que un problema de desición esta en NP se puede construir una maquina de Turing no determinista de tiempo polinomial que decida cada instancia del problema o podemos usar la observación 5.2 que es equivalente. En el siguiente teorema usamos esta segunda opción para probar que 3-sat esta en NP.

Proposicion 6.1. 3-$Sat \in NP$

Demostración. Sea (C, X) una instancia de 3-sat con $X = \{x_1, x_2, ..., x_n\}$ y $C = \{c_1, ..., c_m\}$ donde cada c_i es una disyunción de máximo tres literales y sea t una asignación de valores de verdad de las variables $x_1, x_2, ..., x_n$. Tenemos que $c_i = y_{i_1} \vee y_{i_2} \vee y_{i_3}$ donde $y_{i_k} \in \{x_{i_k}, \overline{x_{i_k}}\}$ para $k = 1, 2, 3$, como $|t| = n$ podemos verificar en un máximo de n pasos si el literal y_{i_k} es falso o verdadero bajo la asignación t, por tanto podemos determinar en un máximo de $3n$ pasos el valor de verdad de la clausula c_i para $i = 1, ..., m$ y así saber en un máximo de $3nm$ pasos si la instancia (C, X) de 3-sat se satisface bajo la asignación t.
De esta manera probamos que 3-Sat $\in NP$. De forma analoga podemos probar que los demas problemas de la lista son de la clase NP. □

Observación 6.2. En su famoso artículo Reducibility Among Combinatorial Problems, Richard Karp muestra que aún que la clase NP abarca un gran número de problemas combinatorios de diversosos dominios como son por ejemplo la teoría de conjuntos, la teoria de números y la teoría de grafos, entre muchos de estos problemas se aprecia una especie de semejanza en el sentido de que un problema se puede transformar en otro de manera que el primer problema es solo una versíon del segundo. Entre otras cosas esta noción de reducción o transformación es muy útil para encontrar tratamientos generales a los problemas tipicos de la clase NP.

7. Reducciones entre problemas combinatorios

7.1. Reducciones

Definición 7.1. Reducción Polinomial

Sea Σ un alfabeto finito y $L \subset \Sigma^*$ y $M \subset \Sigma^*$ decimos que L es reducible a M y escribimos $L \propto M$ si existe una función $f : \Sigma^* \to \Sigma^*$ computable en tiempo polinomial por una máquina de Turing determinista tal que para cada $x \in \Sigma^*$ tenemos que $x \in L$ si y solo si $f(x) \in M$.
Si $L \propto M$ y $M \propto L$ entonces decimos que los lenguajes L y M son equivalentes y escribimos $L \sim M$.

Observación 7.1. El concepto de reducibilidad polinomial es una herramienta muy útil para determinar la clase de complejidad de un problema dado, en efecto, supongamos que queremos conocer la clase de complejidad de un problema A y de alguna manera logramos probar que $A \propto B$ entonces el problema A pertenece a la clase de complejidad del problema B luego si conocemos la clase de complejidad de B también conocemos la clase de A.

Ya mencionamos que aun no se sabe si $P = NP$ lo que significa que no se sabe cuantas clases de complejidad existen y por tanto resulta muy útil para la investigación el criterio de reducibilidad que permite encontar problemas de una misma clase de complejidad.

Un hecho que resulta inmediato útil es que si $L \propto M$ y $M \propto N$ entonces $L \propto N$

Definición 7.2. Problemas NP-Completos

Decimos que $L \in NP$ es NP-completo si $M \propto L$ para cada $M \in NP$. A primera vista resultan extraños los problemas NP-Completos, sin embargo Stephen Cook demostro en 1971 la existencia de problemas NP-Completos lo que constituye un avance importante para las ciencias de la computación.

La siguiente sección se basa en los trabajos de Stephen Artur Cook y Richard Manning Karp, el primero demostro la existencia de problemas NP-completos en un artículo titulado The complexity of theorem proving procedures una demostaración del este teorema conocido como teorema de Cook la encontramos en [3], donde especificamente se demuestra que 3-sat es NP-completo.

Un aspecto llamativo del teorema de Cook es que abre la puerta para encontrar por medio de reducciones polinomicas otros problemas NP-Completos esto es precisamente lo que hace Richard Karp en su artículo Reducibility Among Combinatirial Problems donde presenta una lista de problemas NP-completos junto con las correspondientes formulas de reducción. Las reducciones transforman a 3-Sat (Que es NP-Completo por el teorema de Cook) en otros problemas combinatorios que en consecuencia resultan NP-Completos, algunas de estas formulas son naturales pero otras son realmente ingeniosas. Nuestro trabajo en esta sección fue demostar las reducciones propuestas por Karp.

Las reducciones presentadas por Karp en su artículo también pueden ser útiles en el sentido de permitirnos usar un algoritmo de aproximación de un problema A para aproximar otro problema B cuando el problema B se pueda reducir al problema A. Esto es lo que intentamos en la sección 4 cuando intentamos usar un algoritmo diseñado para Max cut por Goemas y Williamson para resolver otros problemas combinatorios.

Proposicion 7.1. *Max 3-Sat* $\propto$ *Chromatic Number.*

Sea (C, X) una instancia de 3-Sat donde $C = \{c_1, \ldots, c_m\}$ es el conjunto de clausulas y $X = \{x_1, x_2, \ldots, x_k\}$ es el conjunto de variables. Sin perdida de generalidad suponemos $k \geq 4$. A partir de (C, X) construimos la instancia $(G, k+1)$ de Chromatic Number donde $V = \{x_1, \ldots, x_k\} \cup \{\bar{x}_1, \ldots, \bar{x}_k\} \cup \{v_1, \ldots, v_k\} \cup \{c_1, \ldots, c_m\}$ y $A = \{(x_i, \bar{x}_i) |\ 1 \leq i \leq k\} \cup \{(v_i, v_j) |\ 1 \leq i, j \leq k, i \neq j\} \cup \{(x_i, v_j) |\ 1 \leq i, j \leq k, i \neq j\} \cup \{(v_i, \bar{x}_j) |\ 1 \leq i, j \leq k, i \neq j\} \cup \{(x_i, c_j) |\ x_i \notin c_j\} \cup \{(\bar{x}_i, c_j) |\ \bar{x}_i \in c_j\}$.

Veamos que (C, X) se satisface si y solo si $(G, k+1)$ se satisface.
Si (C, X) se satisface entonces existe una asignación t de X que hace que cada clausula en C sea verdadera. Definamos $f : V \to \mathbb{Z}_{k+1}$ de la siguiente manera: para cada $1 \leq i \leq k$ si $u_i \in \{x_1, \ldots, x_k\} \cup \{\bar{x}_1, \ldots, \bar{x}_k\}$ y u_i es verdadera bajo la asiganción t entonces $f(u_i) = i$, en caso que u_i sea falsa entonces $f(u_i) = k+1$ y $f(v_i) = i$ para $i = 1, 2, \ldots, k$.
Por otro lado, para cada $c_i \in C$ elegimos $u_j \in C_i$ tal que u_j es verdadera bajo t y hacemos $f(c_i) = f(u_j)$. Es claro que f es una función, probemos que es un coloreamiento de G. Sean $u, v \in V$ tales que u y v son adyacentes. Consideremos cada una de las siguientes posibilidades:

(i) $u = x_i$ y $v = \bar{x}_i$. Como x_i y $\bar{x}_i$ tienen valores de verdad contrarios entonces $f(u) = i$ y $f(v) = k+1$ en caso de ser u verdadera y v falsa, o bien, $f(u) = k+1$ y $f(v) = i$ en caso contrario, ya que $1 \leq i \leq k$ en cualquiera de los dos casos tenemos $f(u) \neq f(v)$.

(ii) $u = v_i$ y $v = v_j$ con $1 \leq i, j \leq k$, $i \neq j$, por la definición de f tenemos $f(u) = i \neq j = f(v)$.

(iii) $u = v_i$ y $v = x_j$ con $i \neq j$. En este caso hay dos posibilidades: si x_j resulta verdadera, entonces $f(x_j) = j \neq i = f(v_i)$, si x_j es falsa, $f(x_j) = k+1 \neq i = f(v_i)$ de donde $f(u) \neq f(v)$.

(iv) $u = v_i$ y $v = \bar{x}_j$. Con un razonamiento similar al del caso anterior concluimos que $f(u) \neq f(v)$

(v) $u = x_i$ y $v = c_j$. Ya que c_j es verdadera existe $u_j \in c_j$ tal que u_j es verdadera y por tanto $f(c_j) = f(u_j) = j \leq k$. Si x_i es falsa entonces $f(x_i) = k+1 \neq j = f(c_j)$. Si x_i es verdadera y $x_i \in c_j$ entonces u y v no son adyacentes y no hay nada que probar. La otra posibilidad es que x_i sea verdadera y $x_i \notin c_j$, entonces $x_i \neq u_j$ y así $f(v) = f(c_j) = f(u_j) = j \neq i = f(u)$.

(vi) $u = \bar{x}_i$ y $v = c_j$. Por un razonamiento similar al del caso anterior concluimos que $f(u) \neq f(v)$. Asi concluimos que $f : V \to \mathbb{Z}_{k+1}$ es un $(k+1)$-coloreamiento de G, por tanto $(G, k+1)$ se satisface.

Ahora supongamos que $(G, k+1)$ se satisface entonces existe una función $f : V \to \mathbb{Z}_{k+1}$ tal que si u y v son adyacentes entonces $f(u) \neq f(v)$. A partir de esto mostremos que cada clausula $c_j \in C$ contiene por lo menos una variable x_i de X.
Razonemos por el absurdo y supongamos que existe $c_j = \bar{x}_{1j} \vee \bar{x}_{2j} \vee \bar{x}_{3j}$, donde $x_{1j}, x_{2j}, x_{3j} \in X$.

Por la construcción del grafo G c_j es adyacente a toda variable $x_i \in X$, por tanto $f(c_j) \neq i$ para $i = 1, \ldots, k$. Como c_j es adyacente a $\bar{x}_{1j}$ y $f(\bar{x}_{1j}) = k+1$ entonces $f(c_j) \neq k+1$ y así $f(c_j) \notin \mathbb{Z}_{k+1}$ lo cual contradice el hecho de que f es función, luego cada clausula $c_j \in C$ contiene alguna variable $x_i \in X$ por tanto asignando valor verdadero a cada variable de X entoces (C, X) se satisface.

Ahora veamos que esta reducción se realiza en tiempo polinomial. Para construir V realizamor $3k + m$ pasos, que es un polinomio, y A lo construimos en menos de $4k^2 + 2km$ pasos, que es un polinomio, luego la construcción se realizó en tiempo polinomial.

Proposicion 7.2. *Chromatic Number* $\propto$ *Exact Cover.*

Dada una instancia (G, k) de Chromatic Number donde $G = (V, A)$ construimos una instancia de exact cover donde

$$\mathcal{U} = V \cup A \cup \{(u, e, f)|\ u \in e \text{ y } 1 \leq f \leq k\}$$

Para cada $(u, f) \in V \times \{1, 2, \ldots, k\}$ definimos

$$S_{(u,f)} = \{u\} \cup \{(u, e, f)|\ u \in e\}$$

.

Para cada tripleta $(e, f_1, f_2) \in A \times \{1, 2, \ldots, k\} \times \{1, 2, \ldots, k\}$ con $f_1 \neq f_2$ definimos

$$S_{(e,f_1,f_2)} = \{e\} \cup \{(u, e, f)|\ f \neq f_1\} \cup \{(v, e, g)|\ g \neq f_2\}$$

donde $e = (u, v)$. Es fácil probar que el conjunto $\mathcal{U}$ junto con la familia de conjuntos $\{S_{(u,f)}\} \cup \{S_{(e,f_1,f_2)}\}$.

Constituyen una instancia de exact cover. Por ejemplo es claro que para cada $(u, f) \in V \times \{1, ..., k\}$ y para cada $(e, f_1, f_2) \in V \times \{1, ..., k\} \times \{1, ..., k\}$, $S_{(u,f)} \subset \mathcal{U}$ y $S_{(e,f_1,f_2)} \subset \mathcal{U}$ respectivamente. Es decir $\{S_{(u,f)}\} \cup \{S_{(e,f_1,f_2)}\}$ es una familia de subconjuntos de $\mathcal{U}$.

Por otro lado dado $x \in \mathcal{U}$ entonces $x \in V$ o $x \in A$ o $x \in \{(u, e, f)|u \in e\ ,\ 1 \leq f \leq k\}$.

Si $x \in V$ entonces $x \in S_{(x,f)}$, si $x \in A$ entonces $x \in S_{(x,f_1,f_2)}$, si $x = (u, e, f)$ entonces $x \in S_{(u,f)}$ así podemos concluir que la familia $\{S_{(u,f)}\} \cup \{S_{(e,f_1,f_2)}\}$ cubre a $\mathcal{U}$.

Ahora veamos que la instancia (G, k) se satisface si y solo si la instancia $(\mathcal{U}, \{S_{(u,f)}\} \cup \{S_{(e,f_1,f_2)}\})$ se satisface.

Primero supongamos que (G, k) se satisface, entonces existe una función $\phi : V \to \mathbb{Z}_k$ tal que si u y v son adyacentes entonces $\phi(u) \neq \phi(v)$. Para cada $u \in V$ sea $\phi_u = \phi(u)$. Definimos la familia de conjuntos

$$\mathcal{F} = \{S_{(u,\phi_u)}|u \in V\} \cup \{S_{(e,\phi_u,\phi_v)}|\ e = (u, v)\}$$

.

Es inmediato que $\mathcal{F}$ es una subfamilia de $\{S_{(u,f)}\} \cup \{S_{(e,f_1,f_2)}\}$. Veamos que $\mathcal{F}$ es disyunta dos a dos. Sean $A, B \in \mathcal{F}$, $A \neq B$. Analicemos cada caso posible:

(i) $A = S_{(u,\phi_u)}$ y $B = S_{(v,\phi_v)}$. Razonemos por el absurdo y supongamos que $A \cap B \neq \varnothing$ luego existe $x \in A \cap B$ y así $x = (u, e, \phi_u)$ y $x = (v, e', \phi_v)$ lo cual implica que $u = v$ lo que implica $A = B$ lo que es una contradicción, por tanto $A \cap B = \varnothing$.

(ii) Si $A = S_{(u,\phi_u)}$ y $B = S_{(v,\phi_v,\phi_w)}$ razonemos por el absurdo y supongamos que $A \cap B \neq \varnothing$ entonces existe $x \in A \cap B$ como $x \in A$ entonces $x = u$ o $x = (u, e^{'}, \phi_u)$ y como $x \in B$ entonces $x = e$ o $x = (v, e, \phi_v)$ con $f \neq f_v$ o bien con $f \neq f_w$. Si $x = u$ entonces $x \notin B$ lo cual es una contradiccón. Si $x = (u, e^{'}, \phi_u) \in A$ y $x = (v, e, f) \in B$ entonces $u = v$ y $e^{'} = e$ y $\phi_u = f$ de donde $(u, e, \phi_u) \in B$ lo cual es una contradicción.
Si $x = (w, d, f)$ con un razonamiento igual al anterior llegamos a una contradiccón.

(iii) Si $A = S_{(e,\phi_u,\phi_v)}$ y $B = S_{(e_1,\phi_{u1},\phi_{v1})}$ nuevamente supongamos $A \cap B \neq \varnothing$ y sea $x \in A \cap B$ como $x \in A$ entonces $x = e$ o $x = (u, e, f)$ con $f \neq \phi_u$ o bien $x = (v, e, f)$ con $f \neq \phi_v$ pero como también $x \in B$ debemos tener $x = e_1$ o $x = (u_1, e_1, f)$ con $f \neq \phi_{u1}$ o $x = (v_1, e_1, f)$ con $f \neq \phi_{v1}$

Si suponemos que $x = e$ entonces solo es posible que $x = e_1$ de donde $e = e_1$ entonces $(u, v) = (u_1, v_1)$ lo que implica $A = B$ lo cual es una contradicción.
Si $x = (u, e, f) \in A$ y $x = (u_1, e_1, f) \in B$ entonces $e = e_1$. De forma similar se llega a una contradiccón con cada una de las restantes posibilidades. Por tanto tenemos que $\mathcal{F}$ es disyunta dos a dos.

Ahora veamos que $\mathcal{F}$ cubre a $\mathcal{U}$. Sea $x \in \mathcal{U}$ entonces $x \in V \cup A \cup \{(u, e, f) | u \in e, 1 \leq f \leq k\}$

Si $x \in V$ entonces $x \in S_{(x,\phi_x)} \in \mathcal{F}$. Si $x \in A$ entonces $x = (u, v)$ luego $x \in S_{(x,\phi_u,\phi_v)} \in \mathcal{F}$. Si $x = (u, e, f)$ con $e = (u, v)$ y $1 \leq f \leq k$. Aqui hay dos posiblidades si $f = \phi_u$ entonces $x \in S_{(u,\phi_u)} \in \mathcal{F}$ pero si $f \neq \phi_u$ entonces $x \in S_{(e,\phi_u,\phi_v)} \in \mathcal{F}$ de esta manera $(\mathcal{U}, \{S_{(u,f)}\}_{(e,f_1,f_2)})$ se satisface.
Esto significa que existe una subfamilia $\mathcal{F}$ de $\{S_{(u,f)}\}_{(e,f_1,f_2)})$ que cubre a $\mathcal{U}$ y es disyunta dos a dos. Como $V \subset V$ entonces para cada $u \in V$ existe $E \in \mathcal{F}$ tal que $u \in E$ ya que $u \notin S_{(e,f_1,f_2)}$ para cada $e \in A$ y cada $f_1, f_2 \in \{1, ...k\}$ entonces debemos tener que $E = S_{(u,f)}$ para algún $1 \leq f \leq k$. Como $\mathcal{F}$ es disyunta dos a dos realmente este f es único y lo denotamos con f_u. En resumen tenemos que para cada $u \in V$ existe un único $1 \leq f_u \leq k$ tal que $S_{(u,f)} \in \mathcal{F}$

Por otro lado $A \subset \mathcal{U}$ entonces para cada $e \in A$ existe por lo menos un $F \in \mathcal{F}$ tal que $e \in F$. Como $e \notin S_{(u,f)}$, para cada $u \in V$ y para todo $1 \leq f \leq q$ entonces $F = S_{(e,f_1,f_2)}$. Nuevamente como $\mathcal{F}$ es una familia disyunta dos a dos realmente existe un único elemento de $\{S_{(e,f_1,f_2)}\}$ que contiene a e y lo denotamos con S_{e,f_1e,f_2e}. Realmente podemos probar que $\mathcal{F} = \{S_{(u,f_u)} | u \in V\} \cup \{S_{(e,f_1e,f_2e)} | e \in A\}$ y para ello es suficiente probar que $\mathcal{F} = \{S_{(u,f_u)} | u \in V\} \cup \{S_{(e,f_1e,f_2e)} | e \in A\}$ cubre a $\mathcal{U}$.

Sea $x \in \mathcal{U}$, si $x \in V$ entonces $x = S_{(x,f_x)}$. Si $x \in A$ entonces $x \in S_{(x,f_1x,f_2x)}$, si $x = (u, e, f)$ donde $u \in e$ y $1 \leq f \leq k$ entonces $x \in S_{(u,f)}$ luego $\{S_{(u,f_u)} | u \in V\} \cup \{S_{(e,f_1e,f_2e)} | e \in A\}$ cubre a $\mathcal{U}$ y asi $\mathcal{F} = \{S_{(u,f_u)} | u \in V\} \cup \{S_{(e,f_1e,f_2e)} | e \in A\}$.

Para mostrar que la instancia (G, k) se satisface definamos $\phi : V \to \mathbb{Z}_k$ donde $\phi_u = f_u$. Por la unicidad de f_u ϕ es función y si u y v son adyacentes no es posible que $\phi(u) = \phi(v)$ ya que entonces $f_u = f_v$. Por unicidad el único elemento de $\mathcal{F}$ que puede contener elementos de $S_{(e,\phi_v,\phi_v)}$ es $S_{(e,f_1e,f_2e)}$ por tanto $S_{(e,\phi_u,\phi_v)} \subset S_{(e,f_1e,f_2e)}$

Como estos conjuntos son finitos y ademas tienen el mismo número de elementos entonces $S_{(e,\phi_u,\phi_v)=S_{(e,f_1e,f_2e)}}$ y así $\phi_u = f_1e$, $\phi_v = f_2e$ de donde $f_1e = f_2e$ como esto es una contradicción por reducción al absurdo (G, k) se satisface.

Veamos que esta reducción se realiza en tiempo polinomial. Para construir $\mathcal{U}$ se realizan $|V|+|E|+f|V||E|$ pasos, para construir cada $S_{(u,f)}$ se realizan $f|V||E|$ pasos igual

que para construir $S_{(e,f_1,f_2)}$ y asi esta reducción se lleva a cabo en tiempo polinomial. con $f \neq f_u$ entonces $x \neq x$ y si $x = (v, e, f)$ entonces $x \neq x$. Luego por reducción al absurdo podemos concluir que $A \cap B \neq \varnothing$, Así tenemos que $F_{(u,v)}$ es una familia de conjuntos disyunta dos a dos.

Ahora veamos que $\mathcal{U} = \cup F_{(u,v)}$, sea $x \in \mathcal{U}$, si $x = u$, para cualquier $u \in V$ tenemos $u \in S_{(u,f_u)}$ entonces $x \in \cup F_{(u,v)}$, si $x = e$ entonces $x \in S_{(e,f_u,f_v)}$ donde $e = (u, v)$ y por tanto $x \in \cup F_{(u,v)}$.
Si $x = (u, e, f)$ para algún $1 \leq f \leq k$, si $f \neq f_u$ entonces $x \in S_{(e,f_u,f_v)}$ donde $e = (u, v)$ y así $x \in \cup F_{(u,v)}$ si $f = f_u$ entonces $x \in S_{(u,f_u)}$ y entonces $x \in \cup F_{(u,v)}$.

Ahora veamos que esta construcción se realiza en tiempo polinomial. Para construir $\mathcal{U}$ se realizan $|V| + |E| + f|V||E|$ pasos, para construir $S_{(u,f_u)}$ se realizan máximo $f|V||E|$ pasos, igual que para construir $S_{(e,f_u,f_v)}$ luego la reducción se lleva a cabo en tiempo polinomial.

Proposicion 7.3. *Exact cover* $\propto$ *Knapsack.*

Sea $(\mathcal{F}, S)$ una instancia de exact cover donde $\mathcal{F} = \{F_1, ..., F_k\}$ es una familia finita de conjuntos finitos y $S = F_1 \cup F_2 \cup ... \cup F_k$.
Recordemos que exact cover consiste en determinar si existe una subfamilia disyunta de $\mathcal{F}$ que cubra a S, a partir de $(\mathcal{F}, S)$ construyamos la siguiente instancia de knapsack:

$(a_1, ..., a_k, b)$ donde $a_i = |F_i|$ y $b = |S|$ para $i = 1, ..., k$.
Recordemos que Knapsack consiste en determinar si existe o no una sucesión $(x_1, ..., x_k) \subset \{0, 1\}^k$ tal que $\sum_{j=1}^{k} a_j x_j = b$.
Demostremos que $(\mathcal{F}, S)$ se satisface si y solo si $(a_1, ..., a_k, b)$ se satisface.
Supongamos que $(\mathcal{F}, S)$ se satisface, entonces existe $\{F_{i_1}, ..., F_{i_r}\} \subset \mathcal{F}$ tal que $F_{i_r} \cap F_{i_p} = \phi$ si $r \neq p$ y $S = F_{i_1} \cup F_{i_2} \cup ... \cup F_{i_r}$.
Luego $b = |S| = \sum_{j=1}^{r} |F_{i_j}| = \sum_{i=1}^{r} a_{i_j}$ entonces si hacemos $x_t = 1$ para $t \in \{i_1, ..., i_r\}$ y $x_t = 0$ si $t \notin \{i_1, ..., i_r\}$ para cada $1 \leq t \leq k$ entonces $b = \sum_{i=1}^{r} a_i x_i$ y así knapsack se satisface.
Ahora supongamos que la instancia $(a_1, ..., a_k, b)$ se satisface, entonces existe $\{i_1, ..., i_t\}$ tal que $x_i = 1$ si $i \in \{i_1, ..., i_t\}$ y $x_i = 0$ si $i \notin \{i_1, ..., i_t\}$. Ahora tenemos que $|S| = b = \sum_{i=1}^{k} a_i x_i = \sum_{j=1}^{t} a_{i_j} x_{i_j} = \sum_{j=1}^{t} a_{i_j} = \sum_{j=1}^{t} |F_{i_j}|$.
De aqui tenemos que $\{F_{i_j} | j = 1, ..., t\}$ es disjunto dos a dos y $S = F_{i_1} \cup ... \cup F_{i_t}$ por tanto exact cover se satisface.
Es fácil verificar que que esta reducción se realiza en tiempo polinomial.

Proposicion 7.4. *Knapsack* $\propto$ *Partition.*

Dada la instancia $(a_1, a_2, ..., a_r, b)$ de Knapsack construimos instancia $(c_1, c_2, ..., c_r, c_{r+1}, c_{r+2})$ de partition donde $c_i = a_i$ para $i = 1, ..., r$, $c_{r+1} = b + 1$ y $c_{r+2} = 1 - b + \sum_{i=1}^{r} c_i$.
Primero Supongamos que la instancia $(a_1, a_2, ..., a_r, b)$ de Knapsack se satisface entonces existe $x = (x_1, ..., x_r) \in \{0, 1\}^r$ tal que $\sum_{i=1}^{r} a_i x_i = b$. Sin perdida de generalidad asumamos que las variables $x_1, ..., x_k$ toman valor 1 para $0 \leq k \leq r$ y que las variables $x_{k+1}, ..., x_r$ toman valor 0, luego $\sum_{i=1}^{k} a_i = b$ y entonces $\sum_{i=1}^{r} a_i - b = \sum_{i=k+1}^{r} a_i$.

Para ver como $(c_1, c_2, ..., c_r, c_{r+1}, c_{r+2})$ se satisface tomemos $S = \{a_1, ..., a_k, \sum_{i=1}^{r} a_i - b + 1\}$ entonces $\overline{S} = \{a_{k+1}, ..., a_r, b+1\}$ y así tenemos que $\sum S = \sum_{i=1}^{k} a_i - b + 1 + \sum_{i=1}^{r} a_i = 1 + \sum_{i=1}^{r} a_i$ y $\sum \overline{S} = \sum_{i=k+1}^{r} a_i + b + 1 = \sum_{i=k+1}^{r} a_i + \sum_{i=1}^{k} a_i + 1 = 1 + \sum_{i=1}^{r} a_i$ y de esta manera $\sum S = \sum \overline{S}$

lo que significa que $(c_1, c_2, ..., c_r, c_{r+1}, c_{r+2})$ se satisface.

Ahora supongamos que $(c_1, c_2, ..., c_r, c_{r+1}, c_{r+2})$ se satisface es decir existe $S \subset \{c_1, c_2, ..., c_r, c_{r+1}, c_{r+2}\}$ tal que $\sum S = \sum \overline{S}$.
Observemos que no es posible que $c_{r+1}, c_{r+2} \in S$ ya que entonces tendríamos $\sum S \geq c_{r+1}+c_{r+2} = b+1+\sum_{i=1}^{r} a_i - b+1 = 1+\sum_{i=1}^{r} a_i > \sum \overline{S}$. Por tanto podemos asumir sin perdida de generalidad que $c_{r+2} \in S$ y $c_{r+1} \in \overline{S}$ de esta manera $S = \{a_k, ..., a_{k_t}, c_{r+2}\}$. Donde $a_{k1}, ..., a_{k_t} \in \{a_1, ..., a_r\}$, $c_{r+2} = 1 - b + \sum_{i=1}^{r} a_i$ y $\overline{S} = T \cup \{b+1\}$ donde $T = \{a_{k1}, ..., a_{k_t}\}^c$. Como $\sum S = \sum \overline{S}$ entonces $\sum_{i=1}^{t} a_{k_j} + \sum_{i=1}^{r} a_i - b + 1 = 1 + b + \sum T$ entonces $\sum_{i=1}^{t} a_{k_j} + \sum_{i=1}^{r} a_i - \sum T = 2b$ luego $2\sum_{i=1}^{t} a_{k_j} = 2b$ de donde $\sum_{i=1}^{t} a_{k_j} = b$ entonces elegimos $x = (x_1, ..., x_r)$ donde $x_{k_j} = 1$ para $j = 1, ..., t$ y $x_i = 0$ para las demás variables entonces x satisface la instancia $(a_1, ..., a_r, b)$ de Knapsack. Observemos que para construir $(c_1, c_2, ..., c_r, c_{r+1}, c_{r+2})$ a partir de $(a_1, ..., a_r, b)$ realizamos $3r + 3$ pasos que es tiempo polinomial.

Proposicion 7.5. *Partition $\propto$ Number Cut.*

Sea $(c_1, \ldots, c_s)$ una instancia de Partition. A partir de esta instancia definimos la instancia (G, w, W) de Number Cut donde $V = \{1, 2, \ldots, s\}$, $A = \{(i, j) |\ i, j \in V,\ i \neq j\}$, $w(i, j) = c_i c_j$ y $W = \frac{1}{4}(\sum_{i=1}^{s} c_i)^2$. Es claro que esta instancia se realiza en tiempo polinomial.

Supongamos que $(c_1, c_2, \ldots, c_s)$ se satisface, entonces existe $T \subseteq \{c_1, c_2, \ldots, c_s\}$ tal que $\sum T = \sum \bar{T}$. Tomemos $S \subseteq V$ tal que $S = \{i \in V |\ c_i \in T\}$ entonces $\bar{S} = \{i \in V |\ c_i \in \bar{T}\}$. Observemos que $w(S, \bar{S}) = \sum_{i \in S,\ j \in \bar{S}} w(i, j) = \sum_{i \in S,\ j \in \bar{S}} c_i c_j = \sum_{i \in S} c_i \sum_{j \in \bar{S}} c_j = (\sum T)(\sum \bar{T}) = (\sum T)^2$ ya que $\sum T = \sum \bar{T}$.

Por otro lado $\sum_{i=1}^{s} c_i = \sum T + \sum \bar{T} = 2\sum T$ de donde $\sum T = \frac{1}{2}\sum_{i=1}^{s} c_i$. Luego $w(S, \bar{S}) = (\sum T)^2 = \frac{1}{4}(\sum_{i=1}^{s} c_i)^2 = W$. Luego (G, w, W) se satisface.
Ahora supongamos que la instancia (G, w, W) de Max Cut se satisface con $W = \frac{1}{4}(\sum_{i=1}^{s} c_i)^2$ donde $V = \{1, 2, \ldots, s\}$ y $A = \{(i, j) |\ i, j \in S,\ i \neq j\}$.

Entonces existe $S \subseteq V$ tal que $w(S, \bar{S}) \geq \frac{1}{4}(\sum_{i=1}^{s} c_i)^2$. Como $w(S, \bar{S}) = (\sum_{i \in S} c_i)(\sum_{i \in \bar{S}} c_i)$ entonces $(\sum_{i \in S} c_i)(\sum_{i \in \bar{S}} c_i) \geq \frac{1}{4}(\sum_{i=1}^{s} c_i)^2$, entonces $4(\sum_{i \in S} c_i)(\sum_{i \in \bar{S}} c_i) \geq (\sum_{i \in S} c_i + \sum_{i \in \bar{S}} c_i)^2$. Si $A = \sum_{i \in S} c_i$ y $B = \sum_{i \in \bar{S}} c_i$ entonces $4AB \geq (A+B)^2 = A^2 + 2AB + B^2$ entonces $(A-B)^2 \leq 0$ de donde $A = B$, es decir, $\sum_{i \in S} c_i = \sum_{i \in \bar{S}} c_i$, entonces $(c_1, \ldots, c_s)$ se satisface.

Como ya mencionamos Stephen Cook probo en 1971 que el problema de decisión 3-sat es NP-Completo. Presentamos a continuación este teorema sin demostración.

Theorem 7.6. 3*-sat es NP-completo.*

Demostración. Ver [2] □

De las proposiciones 7.1, 7.2, 7.3, 7.4, 7.5 de la definición 7.1 de la observación 7.1 y del teorema 7.6 tenemos el siguiente teorema:

Theorem 7.7. *Los problemas de decisión Chromatic Number, Exact Cover, Knapsack, Partition y Number-Cut son NP-Completos.*

8. Algunos Problemas De Optimización Discreta

Un problema de optimización discreta consiste en hallar el vaor máximo o mínimo de una función f sobre un dominio finito. Por ejemplo el problema de hallar el menor entero k tal que un grafo G dado sea k-coloreable es un problema de optimización discreta. Para cada uno de los problemas NP de la sección anterior damos a continuación su respectiva versión de optimización.

1. Max-2sat: Dadas las clausulas $C = \{C_1, ..., C_m\}$ sobre de variables booleanas $\{x_1, ..., x_n\}$ donde $C_k = x_i \vee x_j$ para $1 \leq k \leq m$ y $1 \leq i \leq j \leq n$ y una función de pesos $w : C \to \mathbb{Q}^+$.
   Se trata de hallar la asignación de valores de verdad de las variables $\{x_1, ..., x_n\}$ que maximiza la suma de los pesos de las clausulas verdaderas

2. Max-nsat: Es el mismo problema que Max-2sat salvo que cada clausula puede contener hasta n literales

3. Max-Clique:Dado un grafo G se trata de encontrar el mayor entero positivo k tal que G contiene un k-Clique

4. Max-set packing: Dada una familia S_j se tarta de hallar el mayor entero positivo k tal que S_j tiene k conjuntos mutuamente disyuntos

5. min-Vertex cover: Dado un grafo $G = (V, A)$ se tarta de hallar el menor entero positivo r tal que existe$N \subset V$ con $|N| = r$ y cada arco de G es adyacente con un vertice en N

6. min-set covering: Dada una familia de conjuntos finitos $\{S_j\}$ se trata de hallar el menor entero positivo r tal que existe una familia $\{T_h\} \subset \{S_j\}$ tal que $|\{T_h\}| = r$ y $\cup T_h \subset \cup S_j$

7. min-undirected Hamilton Circuit: dado un grafo G se tarta de hallar el ciclo de longitud mínima que contiene cada vértice del G exactamente una vez

8. min-chromatic number: Dado un grafo G se trata de hallar el menor entero positivo k tal que G sea k-coloreable

9. min-exact cover: Dada una familia de subconjuntos $\{S_j\}$ de un conjunto $\{x_1, ..., x_t\}$ se trata de hallar una subfamilia disyunta dos a dos con menor orden $\{T_h\} \subset \{S_j\}$ tal que $\cup T_h = \cup S_j$

10. Max-knapsack: Dado un vector $(a_1, a_2, ..., a_r) \in Z^n$ se trata de hallar los valores de las variables x_i en $\{0, 1\}$ que maximizan la expresión $\sum x_i$

11. min-partition: Dado un vector $(c_1, c_2, ..., c_s) \in Z^s$ se trata de determinar el $I \subset \{1, 2, ..., s\}$ tal que $\sum_{h \in I} c_h - \sum_{h \notin I} c_h$ sea mínima

12. Max-cut: $G = (V, A)$ y una función de pesos $w : A \to \mathbb{Q}^+$ se trata de hallar un subconjunto $S \subset V$ tal que $\sum_{\{u,v\} \in A, u \in S, v \notin S} w(u, v)$ sea máximo.

Observación 8.1. Consideremos cualquiera de los problemas de la lista anterior por ejemplo Max-cut, por un momento supongamos que tenemos un algoritmo que resuelve Max-cut en tiempo polinomial entonces dada cualquier instancia (G, w, k) del problema de decisión Number-cut podemos aplicar nuestro algoritmo para hallar el corte máximo de (G, w) y inmediatamente saber si la instancia (G, w, k) se satisface o no. Así tendríamos un algoritmo que decide el problema NP-Completo Number-cut en tiempo polinomial y en consecuencia llegaríamos a que $P = NP$. Este razonamiento se aplica cualquiera de los problemas de optimización de la lista.

Definición 8.1. Un problema Q sea de decisión o no se es NP-Hard, si la existencia de un algoritmo de tiempo polinomial para su solución implica que $P = NP$.

Haciendo un razonamiento analogo al de la observación 8.1 tenemos el siguiente teorema:

Theorem 8.1. *Todos los problemas de optimización en la lista presentada en esta sección son NP-Hard.*

En la siguiente sección presentamos el α-Algoritmo de aproximación aleatoria para Max -cut desarrollado por Goemans y Williamson (ver [7]) este algoritmo logra aproximar Max-cut por lo menos en un 87,8 % de la solución óptima del problema en tiempo polinomial. El principal proposito del presente trabajo es usar las reducciones de la sección 7 para hallar relaciones algebraicas (preferiblemente cuadráticas) entre la función objetivo de un problema de la lista distinto de Max-cut y la función objetivo de Max-cut a fin de usar el algoritmo G-W para resolver el nuevo problema.

9. Un Algoritmo De Aproximacion Aleatorio Para Max Cut Y Max 2-Sat

En el artículo [7] Michel Goemans y David Williamson construyeron un algoritmo de aproximación aleatorio para el problema del corte máximo en un grafo con pesos dado. Este algoritmo corre en tiempo polinomial y obtiene un resultado que es $\geq$ 87,8 % del valor del corte máximo. Esta basado en ideas de optimización semidefinida que discutiremos a continuación.

Recordemos que un grafo con capacidades es una tripleta $G = (V, E, w)$, donde V es el conjunto de vertices del grafo que en este trabajo sera finito, E es el conjunto de aristas o lados de G. Asumimos que G es no dirigido y no tiene aristas multiples ni loops por tanto w es una función que a cada elemento de $V \times V$ le asigna un racional no negativo de manera que $w(i, i) = 0$ $w(i, j) = 0$ si $(i, j) \notin E$ y $w(i, j) = w(j, i)$ para todo $i, j \in V$.

Por sencillez escribiremos w_{ij} en lugar de $w(i, j)$. Para $S \subset V$, $\overline{S}$ denota el complemento de S llamamos al par $(S, \overline{S})$ un corte de G. Denotamos con $cut(G)$ el conjunto de todos los cortes de G, para $(S, \overline{S}) \in cut(G)$, definimos su peso como:

$$W(S, \overline{S}) = \frac{1}{2} \sum_{\substack{i \in S \\ j \in \overline{S}}} w_{ij}$$

El problema de max cut consisite en maximizar $W(S, \overline{S})$ con $S \subset V$. Como ya vimos Max cut es NP-Hard por tanto a menos que $P = NP$ no existe un algoritmo que resuelva este problema en tiempo polinomial.

En general dado que no se ha encontrado un algoritmo que resuelva un problema NP-Hard en tiempo polinomial algunos autores han diseñado algoritmos que en tiempo polinomial dan soluciones aproximadas a estos problemas. Esta estrategia se ha intentado bastante con el problema de Max cut por ejemplo se han publicado trabajos como los de [vitanányi 1981], [Poljak and

Turzík] o [Hofmeister and lefmann 1995].

Pero el aporte mas importante en el diseño de algoritmos de aproximacion para problemas NP-Hard lo hicieron Michael Goemans y David P Williamson en su famoso artículo Improved Approximaton Algorithms For Maximum Cut And Satisfiability Problems using Semidefinite Programming donde usando programación semidefinida construyeron un algoritmo que produce en tiempo polinomial una aproximación de al menos 0.87856 de la solución óptima de Max cut donde hasta entonces solo se habían logrado construir algoritmos que lograban el 0.5 de la solucion optima del problema.

Uno de nuestros objetivos en el presente trabajo es aplicar el algoritmo de Goemans y Williamson para hallar buenas aproximaciones de otros problemas de optimización de la clase NP-Hard.

9.1. El problema Cuadrático Entero Para Max Cut

Veamos como el problema de max cut es equivalente a un problema cuadrático sobre los enteros para ello asociemos a cada $S \subset V$ la función $X_S : V \longrightarrow \{1, -1\}$ donde

$$X_S(i) = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \in \overline{S} \end{cases}$$

Sea $\chi_{(S,\overline{S})}$ la función característica asociada al corte $(S, \overline{S})$ es decir

$$\chi_{(S,\overline{S})}(i,j) = \begin{cases} 1 & \text{if } (i,j) \in (S, \overline{S}) \\ 0 & \text{if } (i,j) \notin (S, \overline{S}) \end{cases}$$

tenemos que

$$\sum_{\substack{i \in S \\ j \in \overline{S}}} w(i,j) = \frac{1}{2} \sum_{(i,j)} w_{ij} \chi_{(S,\overline{S})}(i,j)$$

observemos que

$$\chi_{(S,\overline{S})}(i,j) = (1 - X_S(i)X_S(j))/2$$

y por tanto

$$W(S, \overline{S}) = \frac{1}{2} \sum_{i<j} w_{ij}(1 - X_S(i)X_S(j))$$

Además observemos que $(S, \overline{S}) \in cut(G)$ esta univocamente determinado por la función $X_S = (X_S(1), ..., X_S(n))$ luego podememos definir

$$W(X_S(1), ..., X_S(n)) = W(S, \overline{S})$$

es decir para $S \in V$, el peso del corte $(S, \overline{S})$ es

$$W(X_S(1), ..., X_S(n)) = \frac{1}{2} \sum_{i<j} w_{ij}(1 - X_S(i)X_S(j))$$

como

$$X_S(i) = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \in \overline{S} \end{cases}$$

Para $i = 1, ...n$.

Así podemos definir W sobre el conjunto

$$\{(x_1, ..., x_n) : x_i \in \{1, -1\}, i = 1, ..., n\}$$

como

$$W(x_1, ..., x_n) = \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j), x_i \in \{1, -1\}$$

donde $x_i \in \{1, -1\}$. Entonces el problema de max cut es equivalente al siguiente problema cuadrático sobre los enteros

$$\max_{x_i \in \{1,-1\}} \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j), \quad 1 \leq i \leq n \qquad (Q)$$

9.2. El Problema Geométrico Para Max Cut

El problema geométrico para Max cut es:

$$\max_{v_i, v_j \in S^m} \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i.v_j), \quad 1 \leq i \leq n \qquad (P)$$

El cual se obtiene relajando el dominio del problema cuadrático a toda la esfera m-dimensional. Lo valioso del programa (P) es que este puede resolverse usando programacion semidefinida lo cual permitio a Goemans y Williamson desarrollar el siguiente algoritmo de aproximación aleatoria para max cut el cual llamaremos algoritmo G-W y tiene los siguientes pasos:

a) Resolver (P) como un problema de programación semidefinida obteniendo un conjunto de n vectores $v_1, ...v_n$ sobre la esfera m-dimensional los cuales permiten realizar el grafo G sobre S^{m-1}

b) generar un vector r uniformemente distribuido sobre la esfera

c) $S = \{i | v_i.r \geq 0\}$

G-W es un algoritmo de redondeo aleatorio, que garantiza que el valor esperado del corte W es al menos 0.87856 veces el valalor del Max cut. Para obtener un vector aleatorio r uniformemente distribuido sobre la esfera unitaria tomamos numeros $x_1, x_2, ..., x_n$ de la distribución normal con media cero y desviación estandar 1 entonces si $d = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$ tenemos que

$$\frac{1}{d}(x_1, x_2, ..., x_n)$$

es un vector aleatorio sobre S^{n-1}, si $r = (x_1, x_2, ..., x_n)$ donde $x_1, x_2, ..., x_n$ se eligen de la distribución normal con media cero y desviación estandar 1 entonces la función de densidad de probabilidad para r es $f(y_1, y_2, ..., y_n) = \prod \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y_i^2} = \frac{1}{(2\pi)^{\frac{n}{2}}} e^{-\frac{1}{2}(y_1^2 + y_2^2 + ... + y_n^2)}$ lo que nos muestra que r esta uniformemente distribuido sobre la esfera unidad con probabilidad $\frac{1}{(2\pi)^{\frac{n}{2}}} e^{-\frac{1}{2}}$ sobre cada punto de la esfera unitaria.

9.3. Resultados Fundamentales

Veamos que P es equivalente a un problema de optimización semidefinidad, es claro que P es equivalente a

$$\min_{v_i, v_j \in S^m} \sum_{i<j} w_{ij} v_i . v_j, \quad 1 \leq i \leq n \qquad (P_1)$$

Theorem 9.1. (P_1) es equivalente al problema de optimización semidefinida

$$min W \bullet Y \qquad (P_2)$$

Donde $Y \in S\mathbb{R}^{n \times n}$, $Y = [y_{ii}] \succeq 0$, $y_{ii} = 1$ para $i = 1, ..., n$ y $W = [w_{ij}]$

Demostración. Para cada solución factible $u = (v_1, ..., v_m)$ de (P_1) definimos la matriz $Y_u = [y_{ij}]_m$ donde $y_{ij} = v_i . v_j$. Observemos que la matriz Y_u esta formada por los productos punto de los vectores $v_1, ..., v_m$ por tanto resulta ser una suma finita de matrices de la forma uu^T con $u \in \mathbb{R}^m$ que claramente es simétrica y semidefinida positiva.

Por tanto Y_u es simétrica y definida positiva. Por otro lado $y_{ii} = v_i . v_i = 1$ entonces Y_u es una solución factibe de (P_2) y también podemos observar que el valor de (P_1) en u coincide con el valor de (P_2) en Y_u. Ahora supongamos que $Y = [y_{ij}]_m$ es una solución factible de P_2 como Y es simétrica y semidefinida positiva.

Por el teorema 1.8 existe una matriz B real tal que $Y = B^T B$ digamos que las columnas de B son $v_1, ..., v_m$ como $v_i^T v_i = v_i . v_i = y_{ii} = 1$ entonces $v_i \in S^m$ para $i = 1, ..., m$ luego $u_Y = (v_1, ..., v_m)$ es una solución factible para (P_1) ademas $W \bullet Y = \sum_{i<j} w_{ij} v_i . v_j$ es decir que el valor de Y en (P_2) coincide con el de u_Y en (P_1). Así probamos que (P_1) y (P_1) so equivalentes.

□

Observación 9.1. La importancia de este teorema no es nada proporcional a lo sencillo de su demostración, el resutado permite vincular la relajacion (P) del algoritmo G-W con un problema de optimización semidefinida el cual puede resolverse en tiempo polinomial en forma tan exacto como se quiera. Este es quiza el aspecto que hace mas valioso el artículo de Goemans y Williamson [7] ya que para cualquier problema que se pueda relajar de forma similar al problema (P) se le puede aplicar el algoritmo G-W para obtener en tiempo polinomial una solución aproximada.

G-W es un algoritmo de aproximación y por tanto es natural preguntarse que proporción nos da G-W de la solución exacta de Max cut. Mostraremos que si Z_P es la solución optima del programa P y W es el valor del corte dado por GM entonces $\alpha Z_P \leq E(W)$ donde

$$\alpha = \min_{0 < \theta \leq \pi} \left(\frac{2}{\pi} \frac{\theta}{1 - \cos(\theta)} \right) > 0{,}87856$$

.

Si Z_{MC} es la solución óptima para Max cut entonces tenemos que: $\alpha Z_{MC} \leq \alpha Z_P \leq E(W) \leq Z_{MC} \leq Z_P$ o bien $\alpha \leq \frac{E(W)}{Z_{MC}} \leq 1$ es decir el valor esperado del corte aleatorio dado por G-W es al menos 087856 veces el valor óptimo de Max cut. Las siguientes proposiciones son útiles para llegar a este resultado.

Proposicion 9.1. $\alpha = \min_{0<\theta\leq\pi} \left(\frac{2}{\pi} \frac{\theta}{1-\cos\theta} \right) > 0.87856$

Proposicion 9.2. *Sean Sea $v_1, v_2, ..., v_n$ vectores en S^{m-1} y r un vector aleatorio uniformemente distribuido sobre S^{m-1} entonces*

$$Pr[sgn(v_i.r) \neq sgn(v_j.r)] = \frac{arccos(v_i.v_j)}{\pi}$$

donde

$$sgn(v_i) = \begin{cases} 1 & if\ v_i.r \geq 0 \\ -1 & if\ v_i.r < 0 \end{cases}$$

Demostración. Observemos que $[sgn(v_i.r) \neq sgn(v_j.r)] \Leftrightarrow [(v_i.r \geq 0 \wedge v_j.r < 0) \vee (v_i.r < 0 \wedge v_j.r \geq 0)]$ y $(v_i.r \geq 0 \wedge v_j.r < 0)$ si y solo si r esta en el sector circular n-dimensional determinado por $\theta_{ij} = arccos(v_i.v_j)$.

Es un ejercicio bastante interesante de cálculo vectorial se probar que el area superficial de la esfera n-dimensional es

$$S = \frac{2\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})}$$

y la del sector circular n-dimensional determinado por θ_{ij} es

$$S_{\theta_{ij}} = \frac{\pi^{\frac{n-2}{2}} r^{n-1}\theta}{\Gamma(\frac{n}{2})}$$

donde Γ representa la funcion gamma como r es un vector aleatorio uniformemente distribuido sobre la esfera n-dimensional entonces de acuerdo a las leyes de la probabilidad

$$p(v_i.r \geq 0 \wedge v_j.r < 0) = \frac{S_{\theta_{ij}}}{S} = \frac{\frac{\pi^{\frac{n-2}{2}} r^{n-1}\theta}{\Gamma(\frac{n}{2})}}{\frac{2\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})}} = \frac{\theta_{ij}}{2\pi}$$

de manera análoga

$$p(v_i.r < 0 \wedge v_j.r \geq 0) = \frac{\theta_{ij}}{2\pi}$$

y por tanto

$$p[sgn(v_i.r) \neq sgn(v_j.r)] = \frac{\theta_{ij}}{\pi}$$

□

Comentario: El primer paso del algoritmo G-W se basa en un embebimiento o realización del garfo G en la esfera m-dimensional, esto se logra sustituyendo el vértice x_i por un vector de S^{m-1} el cual denotamos por v_i y trazamos sobre S^{m-1} el arco A_{ij} que une a los vectores v_i y v_j si y solo si $x_i x_j \in E(G)$.
D

e esta manera tenemos que el plano aleatorio P_r determinado por el vector r determina una partición aleatoria en $cut(G)$ esto convierte a W en una variable aleatoria sobre $cut(G)$ usaremos el resultado anterior para calcular el valor esperado de W en el siguiente teorema veremos que $E(W) \geq \frac{1}{2}\alpha \sum_{i<j} w_{ij}(1 - v_i.v_j)$ independientemente de los vectores que tomemos en S^{m-1} para ralizar el grafo G.

Theorem 9.3. *Si $v_1, v_2, ..., v_n$ son los vertices de una realización del grafo G en S^{m-1} entonces*

$$E(W) \geq \frac{1}{2}\alpha \sum_{i<j} w_{ij}(1 - v_i.v_j)$$

Demostración. Tenemos que

$$E(W) = \frac{1}{2\pi} \sum_{i<j} w_{ij} E(1 - x_i.x_j)$$

=

$$\frac{1}{2\pi} \sum_{i<j} w_{ij} p[(1 - x_i.x_j) \neq 0] = \frac{1}{2\pi} \sum_{i<j} w_{ij}[sgn(v_i.r) \neq sgn(v_j.r)]$$

=

$$\frac{1}{\pi} \sum_{i<j} w_{ij} arccos(v_i.v_j)$$

En la ultima igualdad usamos la proposición anterior.

Así tenemos que

$$E(W) = \frac{1}{\pi} \sum_{i<j} w_{ij} arccos(v_i.v_j) = \sum_{i<j} \frac{2}{\pi} \frac{arccos(v_i.v_j)}{1 - v_i.v_j} \frac{1}{2} w_{ij}(1 - v_i.v_j)$$

=

$$\sum_{i<j} \frac{2}{\pi} \frac{cos\theta_{ij}}{1 - cos\theta_{ij}} \frac{1}{2} w_{ij}(1 - v_i.v_j) \geq \frac{1}{2}\alpha \sum_{i<j} w_{ij}(1 - v_i.v_j)$$

como queriamos □

Comentario: Veremos mas adelante como el problema (P) es equivalente a un problema de programación semidefinida el cual se puede resolver en tiempo polinomial con un error $\epsilon > 0$ así obtendremos vectores $v_1, v_2, ..., v_n \in S^{m-1}$ tales que $Z_P = \underset{v_i \in S^m}{max} \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i.v_j)$, $1 \leq i \leq n$

Asi tenemos que $\alpha Z_{MC} \leq \alpha Z_P \leq E(W) \leq Z_{MC}$ lo que significa que el valor esperado para W es al menos α veces la solución óptima para max cut. En la siguiente proposición mostramos que si el valor máximo de la relajación $\frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i v_j)$ esta por encima de 0.84458 del peso total $W_{Tot} = \sum_{(i,j)} w_{ij}$ entonces el valor de α en el teorema anterior se puede mejorar.

Proposicion 9.4. *Sean $h(t) = arccos(1 - 2t)$, γ el valor mínimo de la función $\frac{h(t)}{t}$ en $(0, 1]$ y $A = \frac{1}{W_{tot}} \sum_{i<j} w_{ij} \frac{(1 - v_i.v_j)}{2}$ si $A \leq \gamma$ entonces $E(W) \geq \frac{h(A)}{A} \sum_{i<j} w_{ij} \frac{(1 - v_i.v_j)}{2}$*

Demostración. Sean $X_e = \frac{(1 - v_i.v_j)}{2}$ y $\lambda_e = \frac{w_{ij}}{W_{Tot}}$ entonces tenemos que $A = \sum_e \lambda_e X_e$ y $E(W) = W_{Tot} \sum_e \lambda_e h(X_e)$. Es fácil comprobar que $\text{mín}_{0<t\leq 1} \frac{(arccos(1-2t))}{t} = \alpha$ y que $\frac{h(\gamma)}{\gamma} = \alpha$ de aquí

tendremos que $h(t) \leq \alpha t$ para todo $t \in (0,1]$.

Definamos la función $g(t)$ tal que $g(t) = \alpha t$ si $0 \leq t \leq \gamma$ y $g(t) = h(t)$ si $\gamma < t \leq 1$, teniendo en cuenta que $g(t)$ es convexa observemos que $E(W) = W_{Tot} \sum_e \lambda_e h(X_e) \geq W_{Tot} \sum_e \lambda_e g(X_e) \geq W_{Tot} g(\sum_e \lambda_e X_e) = W_{Tot} g(A) = W_{Tot} h(A) = \frac{h(A)}{A} \sum_{i<j} w_{ij} \frac{(1 - v_i v_j)}{2}$. Observemos que como $A \leq \gamma$ entonces $\frac{h(A)}{A} \geq \alpha$ □

9.3.1. Un α- Algoritmo Aleatorio De Aproximación Para MAX2SAT

Consideremos el conjunto de n variables boleanas $x_1, x_2, ..., x_n$ definimos una clausula como una disyunción de literales distintos donde un literal es una de las variables del conjunto o su negación.

La longitud de una clausula es el numero de literales que la componen. Una instancia de MAX2SAT consiste en una colección ω de clausula boleanas de longitud 2 cada una de las cuales tiene asociado un peso no negativo. Digamos que si $\Omega = C_1, C_2, ..., C_n$ entonces $\omega(C_i) = w_i$ donde $\omega(C_i)$ representa el peso de la clausula C_i MAX2SAT consiste en hallar los valores de verdad que hay que asignar a las incognitas $x_1, x_2, ..., x_n$ de manera que la subcolección de clausulas de ω que resultan verdaderas sea máximo.

Para formular MAX2SAT como un problema de optimización cuadrática Definamos Γ como el conjunto de todas las proposiciones boleanas que se pueden construir a partir de Ω y $\nu : \Gamma \to 1, 0$ donde $\nu(Z) = 1$ si Z resulta verdadera y $\nu(Z) = 0$ si Z resulta falsa y sea $y_0 \in 1, -1$ tal que a cada variable x_i asignemosle una variable y_i de manera que $y_i = y_0$ si x_i resulta verdadera y $y_i = -y_0$ si x_i resulta falsa.De esta manera el MAX2SAT consisite en

$$\underset{C_i \in \Omega}{max} w_i \nu(C_i), \qquad (SAT)$$

Observemos que

$$\nu(x_i) = \frac{1 + y_0 y_i}{2}$$

,

$$\nu(\overline{x}_i) = \frac{1 - y_0 y_i}{2}$$

donde $\overline{x}_i$ representa la negación de x_i para $i = 1, ..., n$. También tenemos que para $P, Q \in \Gamma$, $\nu(P \wedge Q) = \nu(P)\nu(Q)$, por tanto tenemos que para $C_{ij} = x_i \vee x_j$, $\nu(C_{ij}) = \nu(x_i \vee x_j) = 1 - \nu(x_i \wedge x_j) = 1 - \nu(x_i)\nu(x_j) = 1 - \frac{1+y_0 y_i}{2} \frac{1+y_0 y_j}{2} = \frac{1+y_0 y_i}{4} + \frac{1+y_0 y_j}{4} + \frac{1-y_i y_j}{4}$ si $C_{ij} = x_i \vee \overline{(x_j)}$ entonces realizando un procedimiento similar al anterior llegamos a que $C_{ij} = \frac{1+y_0 y_i}{4} + \frac{1-y_0 y_j}{4} + \frac{1+y_i y_j}{4}$ de esta manera tenemos que (SAT) tiene la forma

$$\sum_{i=1}^{i=n} a_{ij}(1 - x_i x_j) + b_{ij}(1 + x_i x_j)$$

y por tanto G-W es un α - algoritmos de aproximación aleatoria para Max-2Sat

10. Aplicaciones

En esta sección nos proponemos usar el algoritmo G-W para aproximar en tiempo polinomial problemas de optimización de la clase NP-Hard. La idea es la siguiente: Dado un problema P de la lista de problemas de la sección 8 vamos a usar la reducción del correspondiente problema de decisión de P a Number-cut con el proposito de encontrar en relación polinomial (Preferiblemente cuadrática) entre la función objetivo de P y la función objetivo de Max-cut entonces tendremos que G-W sera útil para resolver el problema P

Recordemos que min-partition es el problema de dado un vector $(c_1, c_2, ..., c_s) \in Z^s$ se trata de determinar el $S \subset \{1, 2, ..., s\}$ tal que $|\sum_{h\in S} c_h - \sum_{h\notin S} c_h|$ sea mínima.
Usando la identidad $(A-B)^2 = (A+B)^2 - 4AB$ Tenemos que $|\sum_{h\in S} c_h - \sum_{h\notin S} c_h|^2 = (c_1 + c_2 + ... + c_s)^2 - 4(\sum_{h\in S} c_h)(\sum_{h\notin S} c_h)$ es decir $|\sum_{h\in S} c_h - \sum_{h\notin S} c_h|^2 = (c_1 + c_2 + ... + c_s)^2 - 4(\sum_{i\in S, j\notin S} c_i c_j)$ En esta ultima expresión tenemos que $\sum_{i\in S, j\notin S} c_i c_j$ es la función objetivo de max-cut para el grafo $G = (V, E)$ donde $V = \{1, ..., s\}$ y $E = \{(i, j) | 1 \leq i < j \leq s\}$ y el peso de la arista (i, j) es $w(i, j) = c_i c_j$ por tanto aplicando G-W para aproximar el máximo de $\sum_{i\in S, j\notin S} c_i c_j$ obtenemos en tiempo polinomial mínimo de $|\sum_{h\in S} c_h - \sum_{h\notin S} c_h|$ con una aproximación de por lo menos 87,8 % en decir G-W es un α-Algoritmo de aproximación aleatoria para min-partition.

Observación 10.1. Resulta natural preguntarse si dado un problema de la lista de la sección 8 siempre es posible hallar una representación cuadrática sobre el conjunto $D = \{(y_1, ..., y_n) | y_i = 1, -1\}$ para su función objetivo. Veremos en el siguiente teorema que por lo menos para Max-3 sat esto no es posible.

11. Conclusiones

Recordemos Max cut y el algoritmo G-W que diseñaron Goemans y Willamson para aproximar este problema: Dados (G, w) donde $w : E \to \mathbb{Q}^+ \cup \{0\}$ se trata de hallar $S \subset V$ tal que $w(S, \overline{S})$ sea maximo.

Sabemos que aunque Max cut tiene el conjunto finito $D = \{(S, \overline{S}) | S \subset V\}$ como dominio, no se conoce un algoritmo que resuelva este problema en tiempo polinomial es mas en Max cut es NP completo lo que significa que si para Max cut existe un algoritmo polinomial entonces tendriamos $P = NP$ lo que hace muy plausible que no haya algoritmos polinomiales para Max cut ni para ningun problema de la clase NP. Es quizas por esta razon que hay un gran interes en los algoritmos que aproximan problemas NP-completos.

El trabajo mas destacado es que realizaron Goemans and Williamson en su articulo Iproved Approximatin Algorthms For Max Cut And Satisfiability Problems Using Semidefinite Programmin, esto por varia razones, una porque lograron cotas de aproximación que mejoraban significativamente todo lo que se habia logrado con anterioridad.

El uso de la programación semidefinida fue muy innovador y abrio las puertas para resolver otros problemas con esta tecnica de optimización, ademas su técnica de redondeo aleatorio es muy original y sorprendente pero lo mas importante el trabajo de Goemans y Williamson proporciona un modelo matemático que se puede implementar para resolver otros problemas de la clase NP.

Recordemos el algoritmo G-W para Max cut Dados (G, w) donde $w : E \to \mathbb{Q}^+ \cup \{0\}$ se trata de hallar $S \subset V$ tal que $w(S, \overline{S})$ sea máximo, como ya vimos este problema equivale a resolver el problema cuadrático entero

$$\max_{x_i \in \{1,-1\}} \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j), \quad 1 \leq i \leq n \qquad (Q)$$

el cual se relaja a

$$\max_{v_i \in S^m} \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i.v_j), \quad 1 \leq i \leq n \qquad (P)$$

y el algoritmos G-W es

a) Resolver (P) como un problema de programación semidefinida obteniendo un conjunto de n vectores $v_1, ...v_n$ sobre la esfera m-dimensional los cuales permiten realizar el grafo G sobre S^{m-1}

b) generar un vector r uniformemente distribuido sobre la esfera

c) $S = \{i | v_i.r \leq 0\}$

Notemos que G-W funciona para Max cut porque Max cut se pudo formular como un problema de optimización de una función cuadrática sobre un conjunto de vectores cuyas entradas estan en $\{1, -1\}$. Si $D = \{(x_1, ..., x_n) \in \mathbb{R}^n | x_i \in \{1, -1\}\}$ entonces la función objetivo de (Q) es $f(x) = \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j)$ con $x \in D$ o bien $f(x) = x^T C x$ donde $c_{ij} = -\frac{w_{ij}}{4}$ si i /j y $c_{ii} = \sum_i \frac{w_{ij}}{4}$.

Observemos entonces que dado un problema de optimización M si podemos expresar la función objetivo f_M de M en la forma $f_M = x^T C x$ para alguna matriz cuadrada C y el dominio de f_M es D entonces podemos usar G-W para resolver M. Esto fue por ejemplo lo que hicimos con el problema de listas en la sección de aplicaciones y lo que hicieron Goemans y Williamson con Max 2 Sat en el articulo de estos autores que ya mencionamos.

Referencias

[1] Rodrigo De Castro, *Teoria De La Computacion,* Universidad Nacional De Colombia, Bogota 2004.

[2] Hubertus Th. Jongen, Klaus Meer, Eberhard Triesch, *Optimization Theory,* Kluwer Academic Publisers, Boston 2004.

[3] A. M. J. Todd, *Semidefinite Optimization,* Cambridge University Press, New York 2001.

[4] Richard M. Karp, *Reducibility Among Combinatorial Problems,* University Of California At Berkeley, 1992.

[5] Howard Karloff, Uri Zwick, *A $\frac{7}{8}$-Approximation Algorithm For Max 3 Sat?*

[6] Luca Trevisan, Madhu Sudan, Gregory B. Sordan, David P. Williamson *Gadgets, Approximation And Linear Programming*, Octuber 28, 1998

[7] M. Goemans, D. Williamson, *Improved Approximation Algorithms For Maximum Cut And Satisfiability Problems Using Semidefinite programming* Journal Of The ACM 42:1115-1145, 1995.

[8] Uriel Feige, *Randomized Rounding For Semidefinite Programs - Variations On The Max Cut Example,* Weizman Institute, Rehovot 76100, Israel

[9] Etienne De Klerk, Cristian Dobre *A Comparison Of Lower Bouns For The Symmetric Circulant Traveling Salesman Problem,* September 9, 2009.

[10] Vijay V. Vazirani, *Approximation Algorithms,* College Of Computing, Georgia Institute Of Tecnology 2001.