Local Replacement and Scheduling: Ensemble Computation, Partition into Triangles, Sequencing with Intervals and Multiprocessor Scheduling

Demostraciones de problemas NP-Completos

Rodrigo Arévalo Gaytán Fausto David Hernández Jasso

Universidad Nacional Autónoma de México

2023-06-21

Contenido

Contenido

Partition into Triangles

- Ejemplar genérico: Una gráfica $G = (V, E)$, con $|V| = 3q$ para un entero positivo q.
- Pregunta de decisión: Hay una partición de V en q conjuntos disjuntos $V_1, V_2, ..., V_q$ de tres vértices cada uno tal que para cada $V_i = \{v_{i[1]}, v_{i[2]}, v_{i[3]}\}$, las tres aristas $\{v_{i[1]}, v_{i[2]}\}, \{v_{i[1]}, v_{i[3]}\}, \{v_{i[2]}, v_{i[3]}\}$ pertenezcan a E?

Algoritmo no determinista

- Fase Adivinadora
  - Sabemos que $|V| = 3q$ entonces supongamos que tenemos un dado equilibrado de q lados. Para cada $v \in V$ tiramos nuestro dado, sea n el número obtenido por el dado, si el subconjunto de vértices V_n tiene menos de 3 vértices entonces agregamos el vértice v a V_n, si el subconjunto de vértices V_n ya tiene 3 vértices entonces buscamos el siguiente V_m que tenga menos de 3 vértices y agregamos v a V_m, si llegamos a V_q y no pudimos agregar a v entonces desde V_n buscamos el anterior V_l que tenga menos de 3 vértices y agregamos v a V_l, esto lo hacemos hasta recorrer por completo V.
- Fase Verificadora
  - Si la cardinalidad del conjunto de vértices de la gráfica G no es múltiplo de 3 ningún candidato a solución será válido, entonces devolvemos NO en dicho caso.
  - Si por cada subconjunto de 3 vértices $u, v, w \in V_i$ tenemos que las aristas $(u, v), (u, w), (v, w) \in E$ entonces regresamos SI en otro caso regresamos NO.

Teorema y Demostración

- Teorema: El problema partición en traingulos es NP-Completo.
- Demostración: Transformamos cobertura exacta por conjuntos de 3 a partición en triangulos.
- Para demostrar el problema usaremos la técnica de reemplazo local. Pero primero ¿qué es la cobertura exacta por conjuntos de 3?

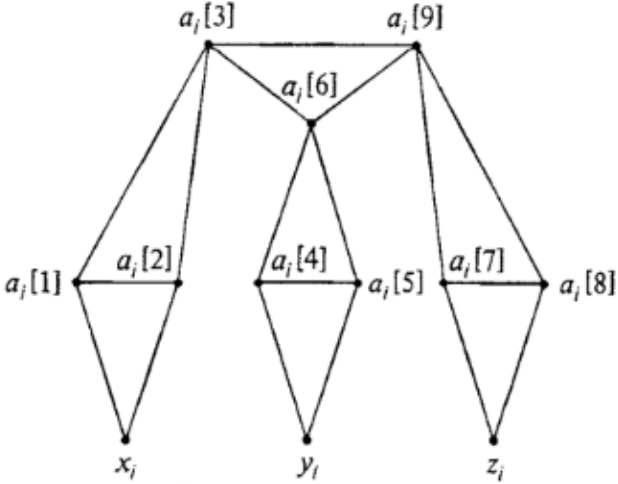
Cobertura exacta por conjuntos de 3 (X3C)

- Ejemplar genérico: Un conjunto X con $|X| = 3q$ y una colección C de subconjuntos de 3 elementos de X.
- Pregunta de decisión: ¿Existe algún subconjunto C' de C en donde cada elemento de X aparezca **exactamente una vez** en C'?
- Ejemplo: Supongamos que tenemos $X = \{1, 2, 3, 4, 5, 6\}$.
  Si tenemos $C = \{\{1, 2, 3\}, \{2, 3, 4\}, \{1, 2, 5\}, \{2, 5, 6\}, \{1, 5, 6\}\}$, entonces podríamos elegir $C' = \{\{2, 3, 4\}, \{1, 5, 6\}\}$ como cobertura exacta porque cada elemento de X aparece **exactamente una vez**.

Transformación

- Sea X el conjunto con $|X| = 3q$ y la colección C de subconjuntos de 3 elementos de X un ejemplar de $X3C$, construiremos una gráfica $G = (V, E)$ con $|V| = 3q'$ tal que la partición deseada exista para G si y solo si C contiene una cobertura exacta.
- Las unidades básicas del ejemplar de $X3C$ son los subconjuntos de 3 elementos de C.
- El reemplazo local, por cada subconjunto $c_i = \{x_i, y_i, z_i\} \in C$, crea la siguiente gráfica:

Transformación



Transformación

Así $G = (V, E)$ se define como

$$V = X \cup \bigcup_{i=1}^{|C|} \{a_i[j] : 1 \leqslant j \leqslant 9\}$$

$$E = \bigcup_{i=1}^{|C|} E_i$$

Notemos que sólo los vértices que se conectan con aristas que pertenecen a más de un E_i son aquellos que se encuentran en el conjunto X.

También podemos ver que $|V| = |X| + 9|C| = 3q + 9|C|$ así que $q' = q + 3|C|$.

Complejidad en tiempo de la transformación

La transformación del ejemplar de X3C al ejemplar de partición en triangulos se hace en tiempo polinomial. Tenemos que por cada subconjunto en C se agregaran 12 vértices y 18 aristas a los respectivos conjuntos, lo que hace que la complejidad tenga $30|C|$ operaciones, lo que deja la complejidad en tiempo como $O(|C|)$.

X3C = YES => PIT = YES

Si $c_1, c_2, ..., c_q$, los subconjuntos de 3 elementos de C, están en alguna cobertura exacta para X, entonces la partición correspondiente $V = V_1 \cup V_2 \cup ... \cup V_q$ de V está dada al tomar

$$\{a_i[1], a_i[2], x_i\}, \{a_i[4], a_i[5], y_i\}, \{a_i[7], a_i[8], z_i\}, \{a_i[3], a_i[6], a_i[9]\}$$

de los vértices conectados por E_i cuando $c_i = \{x_i, y_i, z_i\}$ está en la cobertura exacta.

Y tomando

$$\{a_i[1], a_i[2], a_i[3]\}, \{a_i[4], a_i[5], a_i[6]\}, \{a_i[7], a_i[8], a_i[9]\}$$

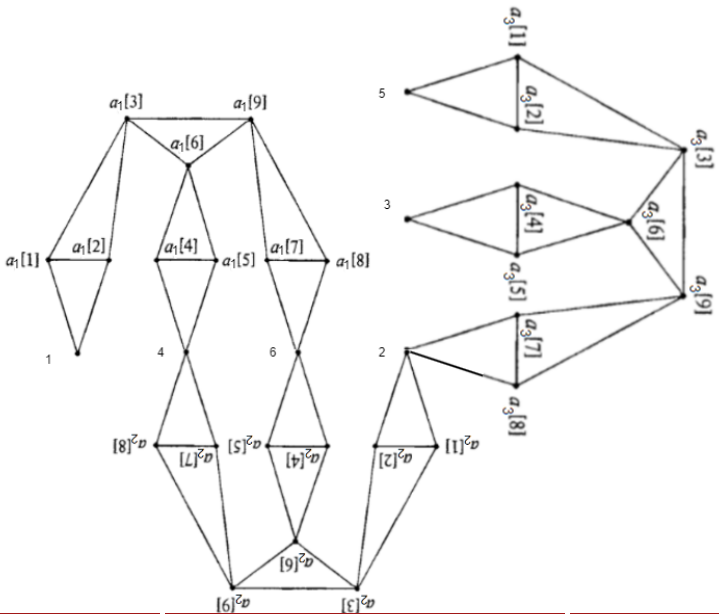
de los vértices conectados por E_i cuando $c_i = \{x_i, y_i, z_i\}$ no está en la cobertura exacta. Esto asegura que cada elemento de X este incluido en exactamente un subconjunto de 3 vértices en la partición

X3C = YES <= PIT = YES

Si $V_1 \cup V_2 \cup ... \cup V_q$ es una partición en triangulos de G, la cobertura exacta está dada al elegir los $c_i \in C$ tal que $\{a_i[3], a_i[6], a_i[9]\} = V_j$ para alguna j, $1 \leqslant j \leqslant q'$

Ejemplo de transformación

Supongamos que tenemos un ejemplar de X3C con un conjunto $X = \{1, 2, 3, 4, 5, 6\}$ y una colección $C = \{\{1, 4, 6\}, \{2, 4, 6\}, \{2, 3, 5\}\}$, entonces nuestro ejemplar de partición en triángulos nos quedaría como


$a_1[1]$
$a_1[2]$
$a_1[3]$
$a_1[4]$
$a_1[5]$
$a_1[6]$
$a_1[7]$
$a_1[8]$
$a_1[9]$
1
4
6
$a_2[1]$
$a_2[2]$
$a_2[3]$
$a_2[4]$
$a_2[5]$
$a_2[6]$
$a_2[7]$
$a_2[8]$
$a_2[9]$
2
3
5
$a_3[1]$
$a_3[2]$
$a_3[3]$
$a_3[4]$
$a_3[5]$
$a_3[6]$
$a_3[7]$
$a_3[8]$
$a_3[9]$

Contenido

Forma canónica o estándar

- **Ejemplar genérico:** Una colección C de subconjuntos de un conjunto finito A y un entero positivo J.
- **Pregunta de decisión:** ¿Existe una secuencia

$$\langle z_1 = x_1 \cup y_1, z_2 = x_2 \cup y_2, \ldots, z_j = x_j \cup y_j \rangle$$

con j operaciones de unión, tal que $j \leq J$, donde cada x_i y y_i es $\{a\}$ para algún $a \in A$ o es z_k para alguna $k < i$, además x_i y y_i son disjuntos para todo $1 \leq i \leq j$, para todo subconjunto $c \in C$ existe alguna z_i con $1 \leq i \leq j$, tal que z_i es igual a c?

Entendiendo el enunciado

- A es un conjunto finito.
- C es una colección formado por subconjuntos de A, es decir, para todo c en la colección $c \in \mathcal{P}(A)$
- Cada x_i y y_i son conjuntos, como los describimos anteriormente sí x_i y y_i no es de la forma $\{a\}$ para algún $a \in A$, entonces son un elemento z_k *"anterior"* de la secuencia.
- Se busca que la secuencia cumpla que para todo elemento $c \in C$ siempre debe de existir un elemento de la secuencia que sea idéntico a él, es decir, sí $c \in C$ entonces existe z_k tal que:
  - z_k forma parte de la secuencia.
  - $z_k = c$.

Ejemplo

- Sea $A = \{a, b, c, d, e\}$, $C = \{\{a, b\}, \{b, c, d\}, \{c, e\}\}$ y $J = 4$.
- Una secuencia que cumple lo que nos pide el problema es:

$$\langle z_1 = \{a, b\}, z_2 = \{b, c\}, z_3 = \{b, c, d\}, z_4 = \{c, e\}\rangle$$

- En la secuencia anterior, notemos que se cumple $j \leq J$.
- En la secuencia anterior x_i y y_i para $1 \leq i \leq j$ con $j = 4$ están dados por:
  - $x_1 = \{a\}$, $y_1 = \{b\}$ ya que $z_1 = x_1 \cup y_1 = \{a\} \cup \{b\} = \{a, b\}$
  - $x_2 = \{b\}$, $y_2 = \{c\}$ ya que $z_2 = x_2 \cup y_2 = \{b\} \cup \{c\} = \{b, c\}$
  - $x_3 = z_2$, $y_3 = \{d\}$ ya que $z_3 = x_3 \cup y_3 = z_2 \cup y_3 = \{b, c\} \cup \{d\} = \{b, c, d\}$.
  - $x_4 = \{c\}$, $y_4 = \{e\}$ ya que $z_4 = x_4 \cup y_4 = \{c\} \cup \{e\} = \{c, e\}$

  Consecuentemente se cumple que para todo x_i y y_i para $1 \leq i \leq 4$ es de la forma $\{a\}$ para algún $a \in A$ o es z_k para alguna $k < i$.

Ejemplo

Notemos que $x_i \cap y_i = \varnothing$ ya que:

- $x_1 \cap y_1 = \{a\} \cap \{b\} = \varnothing$
- $x_2 \cap y_2 = \{b\} \cap \{c\} = \varnothing$
- $x_3 \cap y_3 = z_2 \cap y_3 = \{b, c\} \cap \{d\} = \varnothing$.
- $x_4 \cap y_4 = \{c\} \cap \{e\} = \varnothing$

Por lo tanto se cumple que para todo x_i y y_i son disjuntos.

Ejemplo

Nombraremos a los elementos del conjunto C

$$c_1 = \{a, b\}$$
$$c_2 = \{b, c, d\}$$
$$c_3 = \{c, e\}$$

Así notemos que

$$c_1 = z_1$$
$$c_2 = z_3$$
$$c_3 = z_4$$

Por lo tanto se cumple que para todo $c \in C$ existe z_k en la secuencia tal que $c = z_k$, así la secuencia cumple con lo solicitado por el problema.

Algoritmo no determinista

- Fase adivinadora:
  - Sea S una secuencia vacía
  - Pondremos todos los elementos de la colección C en un frasco.
  - Tenemos un dado equilibrado con $|C|$ caras.
  - Tiramos el dado y obtendremos un número j tal que $j \in \{1, 2, \ldots, |C|\}$.
  - Sacaremos j conjuntos de la colección C de la siguiente manera:
    - Al sacar el i-ésimo conjunto c del frasco lo agregamos a la secuencia S y nos referiremos a éste como z_i.
    - Para x_i tomamos un subconjunto de c tal que sea de cardinalidad 1.
    - Para y_i tomamos el subconjunto restante de c.
- Fase verificadora:
  - Si $|C| > J$ entonces no habrá solución para el ejemplar, en otro caso verificamos que la secuencia cumpla las restricciones, si S no cumple alguna, entonces S no es una solución para el ejemplar, si S cumple todas las restricciones entonces sí una solución para el ejemplar.

Complejidad del algoritmo no determinista

- Fase adivinadora:
  - Agregar elementos a la secuencia nos toma $O(|C|)$ ya que en el peor de los casos $j = |C|$.
  - Definir a cada x_i y y_i con $1 \leq i \leq j$ nos toma $O(|C|)$.
- Fase verificadora:
  - Comprobar que $j \leq J$ nos toma $O(1)$.
  - Comprobar que cada x_i y y_i cumple con la forma solicitada nos toma tiempo $O(|C| + |A|)$ ya que existen j de cada uno de ellos entonces debemos comprobar $2j$ conjuntos que cumplan con lo solicitado, recordemos que éstos su vez pueden tener dos formas:
    - $\{a\}$ con $a \in A$.
    - z_k con $k < i$.

    así la complejidad de éste paso es $O(|C| \cdot (|C| + |A|))$.
  - Comprobar que para cada $c \in C$ existe un elemento de la secuencia z_i tal que $z_i = c$ está dado por $O(j \cdot |C|)$.
  - Comprobar que cada $x_i \cap y_i = \varnothing$ nos toma j operaciones ya que hay j x_i y j y_i por lo tanto nos toma ésta comprobación tiempo $O(|C|)$.

Vertex cover (Cobertura de vértices)

Definición:
Una cobertura de vértices es un subconjunto V' de vértices de la gráfica $G = (V, E)$ tal que para cada arista $(u, v) \in E$, $u \in V'$ ó $v \in V'$.

Ejemplo

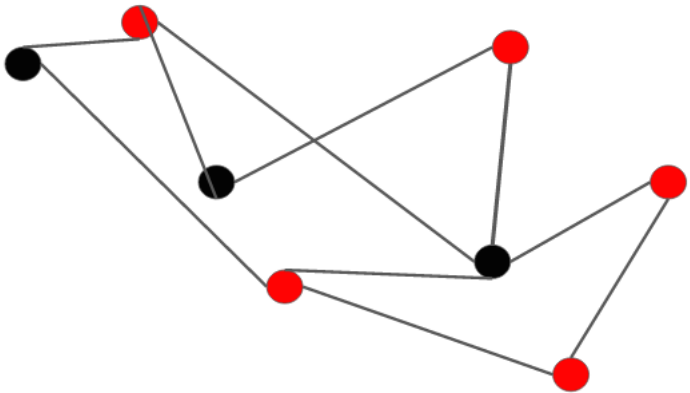


Figura 1: Gráfica G

Ejemplo

En la gráfica G, el conjunto de vértices de color rojo, forma una cobertura de vértices en G mientras que el conjunto de vértices de color negro no.

Problema de la cobertura de vértices

Dada una gráfica G y un entero positivo M, ¿G tiene una cobertura de vértices de tamaño a lo más M?
El problema de la **cobertura de vértices** es **NP**-Completo

Teorema

El problema **Ensemble Computation** es **NP**-Completo

Demostración

Transformaremos **Vertex Cover** a **Ensemble Computation**.
Sea $G = (V, E)$ una gráfica tal que:

- V es el conjunto de vértices de la gráfica.
- E es el conjunto de aristas de la gráfica.

y sea K un entero positivo tal que $K \leq |V|$, lo anterior es un ejemplar genérico de **Vertex Cover**.

Demostración

Sea a_0 un elemento nuevo tal que $a_0 \notin V$. Entonces para cada arista $\{u, v\} \in E$ formamos el conjunto $\{a_0, u, v\}$, el conjunto que contiene a todos éstos conjuntos descritos anteriormente, lo nombraremos como conjunto C. Es decir, C está definido de la siguiente manera:

$$C = \{\{a_0, u, v\} \mid \{u, v\} \in E\}$$

Definimos al conjunto A como $A = V \cup \{a_0\}$
Definimos al entero J como $J = K + |E|$, notemos que $J \in \mathbb{Z}^+$ ya que $K \in \mathbb{Z}^+$ y $|E| \in \mathbb{Z}^+$

Demostración

Ahora lo que tenemos es un ejemplar genérico de **Ensemble Computation**. Éste ejemplar se construyó en tiempo polinomial ya que:

- Formar el conjunto A nos tomó $|V| + 1$ operaciones así la complejidad en tiempo de la construcción está dada por $O(|V|)$.
- Formar el conjunto C nos tómo $|E|$ operaciones así la complejidad en tiempo en la construcción está dada por $O(|E|)$, a su vez $|E|$ está acotado por $O(|V|^2)$ ya que el mayor número de aristas que puede tener una gráfica está dado por $\frac{|V|(|V|-1)}{2}$ *(Gráfica completa)*.
- Formar el número J se hace en tiempo constante $O(1)$ ya que siempre se efectúan dos operaciones.

Demostración

Afirmación:

G tiene una cobertura de vértices de a los más K sí y sólamente sí existe la secuencia deseada de $j \leq J$ operaciones para el conjunto C

Demostración

Ida

Supongamos que V' es una cobertura de vértices para la gráfica G y $|V'| \leq K$. Notemos que podemos agregar vértices a V' y continuará siendo una cobertura de vértices. Sin pérdida de generalidad asumiremos que $|V'| = K$. Nombramos los elementos de V' como $v_1, v_2, \ldots, v_K$ y etiquetamos a las aristas en E como $e_1, e_2, \ldots, e_m$ donde $m = |E|$.

Demostración

Ya que V' es una cobertura de vértices entonces cada arista e_j contiene al menos un elemento de V'. Consecuentemente podemos escribir a e_j como

$$e_j = \{u_j, v_{r[j]}\}$$

donde $r[j]$ es un entero tal que $1 \leq r[j] \leq K$. La siguiente secuencia:

$$\langle z_1 = \{a_0\} \cup \{v_1\}, z_2 = \{a_0\} \cup \{v_2\}, \ldots, z_K = \{a_0\} \cup \{v_K\},$$
$$z_{K+1} = \{u_1\} \cup z_{r[1]}, z_{K+2} = \{u_2\} \cup z_{r[2]}, \ldots, z_J = \{u_m\} \cup z_{r[m]}\rangle$$

cumple con los propiedades solicitadas por **Ensemble Computation**.

Demostración

Regreso

Supongamos que S es la secuencia deseada es decir, definimos a S como

$$S = \langle z_1 = x_1 \cup y_1, \ldots, z_j = x_j \cup y_j \rangle$$

con $j \leq J$. Supondremos que S es la secuencia que contiene el menor número de operaciones de la forma $z_i = \{u\} \cup \{v\}$ para $u, v \in V$. Notemos que S no puede contener operaciones de ésta forma.

Demostración

Supongamos que $z_i = \{u\} \cup \{v\}$ con $u, v \in V$.
Notemos que $\{u, v\} \notin C$ ya que por construcción los elementos de C son de la forma $\{a_0, u, v\}$ con a_0 un nuevo elemento, y $\{u, v\} \in E$, de nuevo por construcción debe ocurrir que $\{u, v\} \in E$.
Como S cumple con las condiciones dadas entonces debe de existir una $z_j = \{a_0, u, v\}$ ya que $\{a_0, u, v\} \in C$, como $z_i = \{u, v\}$ entonces $z_j = \{a_0\} \cup z_i$ y por ende $i < j$, es decir z_j aparece "después" en S.
Por construcción tenemos $\{u, v\}$ es subconjunto de un solo elemento de C. Así z_i no puede ser usado en ningún elemento de la secuencia más que en z_j.

Demostración

Notemos lo siguiente

$$z_i = \{u\} \cup \{v\} \qquad \text{y} \qquad z_j = \{a_0, u, v\} = \{a_0\} \cup z_i$$

Podemos redefinirlo como:

$$z_i = \{a_0\} \cup \{u\} \qquad \text{y} \qquad z_j = \{a_0, u, v\} = \{v\} \cup z_i$$

Demostración

Observemos que seguimos cumpliendo las restricciones impuestas por el problema, lo anterior implica que hemos encontrado una secuencia S' con menor número de operaciones de tipo $\{u\} \cup \{v\}$ lo cual es una contradicción. Por lo tanto las operaciones en S solamente tienen dos formas:

- $z_i = \{a_0\} \cup \{u\}$ para $u \in V$
- $\{a_0, u, v\} = \{v\} \cup z_i$ para $\{u, v\} \in E$

Demostración

De nuevo, por construcción tenemos $|C| = |E|$, como todo $c \in C$ debe de ser igual a un z_k de S entonces S contiene exactamente $|E|$ operaciones del tipo $\{a_0, u, v\}$. Observamos que

$$j \leq J$$
$$j - |E| \leq J - |E|$$

Definimos a K como $J-|E|$, y tenemos a lo más K de operaciones de la forma $\{a_0\} \cup \{u\}$. Ahora definiremos al conjunto V' que describe las operaciones anteriores, es decir

$$V' = \{\{a_0\} \cup \{u\} \mid u \in V\}$$

Consecuentemente $|V'| \leq K$ como cada $\{a_0, u\}$ es un subconjunto de algún elemento de C, entonces V' es una cobertura de vértices de G.

Ejemplificación de la transformación

Consideremos la gráfica $G = (V, E)$ definida como sigue:

$$\begin{aligned} V =& \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\} \\ E =& \{\{v_1, v_2\}, \{v_1, v_5\}, \{v_2, v_5\}, \{v_2, v_6\}, \{v_2, v_3\}, \\ & \{v_2, v_7\}, \{v_3, v_4\}, \{v_4, v_7\}, \{v_4, v_8\}, \{v_5, v_6\}\} \end{aligned}$$

Observamos que:

$$\begin{aligned} |V| &= 8 \\ |E| &= 10 \end{aligned}$$

Una cobertura de vértices para G es el conjunto V' definido como sigue:

$$V' = \{v_2, v_4, v_5\}$$

Nuestro entero positivo es $K = 3$.

Ejemplificación de la transformación

Sea a_0 un nuevo elemento, claramente se cumple que $a_0 \notin V$. Formamos el conjunto C de la siguiente forma:

$$\begin{aligned} C = \{&\{a_0, v_1, v_2\}, \{a_0, v_1, v_5\}, \\ &\{a_0, v_2, v_5\}, \{a_0, v_2, v_6\}, \\ &\{a_0, v_2, v_3\}, \{a_0, v_2, v_7\}, \\ &\{a_0, v_3, v_4\}, \{a_0, v_4, v_7\}, \\ &\{a_0, v_4, v_8\}, \{a_0, v_5, v_6\}\} \end{aligned}$$

Definimos a A como

$$A = V \cup \{a_0\} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, a_0\}$$

Notemos que todo $c \in C$ cumple con $c \in \mathcal{P}(A)$
Definimos a J como

$$J = K + |E| = 3 + 10 = 13$$

Así tenemos un ejemplar genérico de **Ensemble Computation.**

Técnica para la demostración del problema

Reemplazo Local

Aplicación en la vida real

- Encontrar circuitos con el menor número de compuertas lógicas.
- Encontrar una manera eficiente de realizar determinadas multiplicaciones entre números, matrices, etc.

Contenido

Sequencing within intervals

- Ejemplar genérico: Un conjunto finito T de tareas y por cada $t \in T$, un entero *tiempo de liberación* $r(t) \geqslant 0$, un *tiempo limite* $d(t) \in \mathbb{Z}^+$ y una longitud $l(t) \in \mathbb{Z}^+$.
- Pregunta de decisión: ¿Existe un horario factible para T, una función $\sigma : T \to \mathbb{Z}^+$ tal que para cada $t \in T$, $\sigma(t) \geqslant r(t), \sigma(t) + l(t) \leqslant d(t)$ y si $t' \in T - \{t\}$ entonces $\sigma(t') + l(t') \leqslant \sigma(t)$ o $\sigma(t') \geqslant \sigma(t) + l(t)$?
  (la tarea t es ejecutada desde el tiempo $\sigma(t)$ al tiempo $\sigma(t) + l(t)$, no se puede ejecutar hasta el tiempo $r(t)$, debe ser completada para el tiempo $d(t)$ y su ejecución no puede traslaparse con la ejecución de otra tarea t').

Algoritmo no determinista

- Fase Adivinadora
  - Tiramos un dado equilibrado de $|T|$ lados y del número n que salga tomamos la tarea t_n del conjunto T, la renombramos como t'_1 hacemos su horario factible igual a $\sigma(t'_1) = 0$ volvemos a tirar el dado y del número m que salga tomamos la tarea t_m del conjunto T, y la renombramos como t'_2, hacemos el $\sigma(t'_2) = \sigma(t'_1) + l(t'_1)$, ..., volvemos a tirar el dado y del número j que salga tomamos la tarea t_j del conjunto T, y la renombramos como t'_i, tal que t'_{i-1} sea la tarea anterior a la que le asignamos su horario factible, hacemos el $\sigma(t'_i) = \sigma(t'_{i-1}) + l(t'_{i-1})$, y hacemos esto hasta asignarle un horario factible a la tarea $t_{|T|}$.
- Fase Verificadora
  - Por cada tarea verificamos que $\sigma(t) \geqslant r(t), \sigma(t) + l(t) \leqslant d(t)$ y si $t' \in T - \{t\}$ entonces $\sigma(t') + l(t') \leqslant \sigma(t)$ o $\sigma(t') \geqslant \sigma(t) + l(t)$, si se cumple para todas las tareas devolvemos SI, en otro caso regresamos NO.

Teorema y Demostración

- Teorema: El problema secuenciación dentro de intervalos es NP-Completo.
- Demostración: Transformamos partición a secuenciación dentro de intervalos.
- Para demostrar el problema usaremos la técnica de reemplazo local. Pero primero ¿qué es el problema de partición?

Partición

- Ejemplar genérico: Un multiconjunto S de enteros positivos.
- Pregunta de decisión: Existen 2 subconjuntos S_1 y S_2 tal que la suma de los enteros de S_1 sea igual a la suma de los enteros de S_2?
- Ejemplo: Supongamos que tenemos $S = \{3, 1, 1, 2, 2, 1\}$.
  Sea $S_1 = \{1, 1, 1, 2\}$ y $S_2 = \{2, 3\}$ vemos que ambos subconjuntos suman 5 y devuelven YES a la pregunta de decisión. Podemos ver que otra partición la tenemos con los subconjuntos $S_1 = \{3, 1, 1\}$ y $S_2 = \{2, 2, 1\}$ vemos que ambos subconjuntos suman 5.
- Dato curioso: Este problema es llamado el problema difícil más fácil.

Transformación

- Sea el conjunto finito A y un tamaño $s(a)$ para cada $a \in A$ el ejemplar genérico del problema partición y $B = \Sigma_{a \in A}\ s(a)$.
- Las unidades básicas del ejemplar de partición son los elementos individuales $a \in A$.
- El reemplazo local para cada $a \in A$ es una sola tarea t_a con $r(t_a) = 0$, $d(t_a) = B+1$ y $l(t_a) = s(a)$. El ejecutor es una sola tarea $\bar{t}$ con $r(\bar{t}) = \lceil B/2 \rceil$, $d(\bar{t}) = \lceil (B+1)/2 \rceil$ y $l(\bar{t}) = 1$.

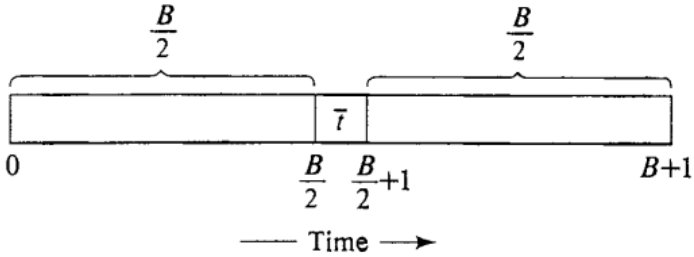
Complejidad en tiempo de la transformación

Claramente este ejemplo puede ser construido en tiempo polinomial desde el ejemplar de partición.

- Tomar cada elemento $a \in A$ y asignarle un valor $r(t_a) = 0$, $d(t_a) = B + 1$ y $l(t_a) = s(a)$ tiene una complejidad $O(3|A|)$, junto con la generación de la tarea $\bar{t}$ tenemos que la complejidad en tiempo para esta transformación es de tiempo $O(3(|A| + 1))$.

Restricciones por $\bar{t}$

- $\bar{t}$ nos asegura que el horario factible no puede ser construido cuando B es un número impar, porque tendríamos que $r(\bar{t}) = d(\bar{t})$. Supongamos que $B = 3$ entonces $r(\bar{t}) = \lceil 3/2 \rceil, d(\bar{t}) = \lceil (3+1)/2 \rceil => r(\bar{t}) = \lceil 1{,}5 \rceil, d(\bar{t}) = \lceil 4/2 \rceil => r(\bar{t}) = 2, d(\bar{t}) = 2$ y $\bar{t}$ no puede ser agendada. Así que asumimos que B es par.
- La segunda restricción la tenemos, ya que B es par, $r(\bar{t}) = B/2$ y $d(\bar{t}) = r(\bar{t})+1$, así que algún horario factible debe tener $\sigma(\bar{t}) = B/2$. Esto divide el tiempo disponible para agendar las tareas restantes en dos bloques separados, cada uno con longitud de $B/2$ como vemos en la siguiente imagen:



Así el problema de agendación se convierte es un problema de elegir subconjuntos, los que son agendados antes de $\bar{t}$ y los que son agendados después de $\bar{t}$.

Partición = YES => SWI = YES

- Como la cantidad total de tiempo disponible en los dos bloques es igual al total de la longitud B de las tareas restantes, tenemos que llenar cada bloque de forma exacta.
- Esto puede hacerse si y solo si hay un subconjunto $A' \subseteq A$ tal que

$$\sum_{a \in A'} s(a) = B/2 = \sum_{a \in A - A'} s(a)$$

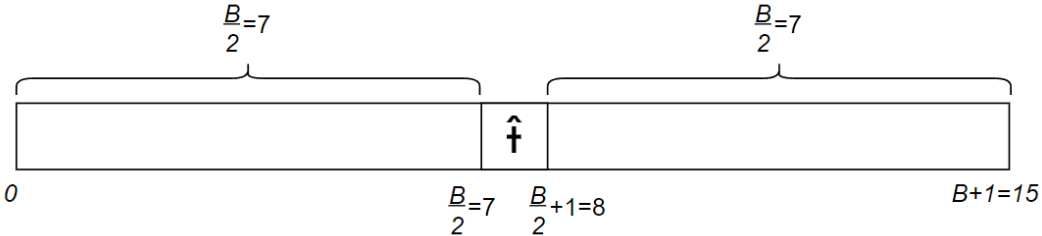
- Así el subconjunto A' deseado existe para el ejemplar de partición si y solo si existe un horario factible para el ejemplar correspondiente de secuenciación dentro de intervalos.

Ejemplo de transformación

Sea $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ con $s(a_1) = 1, s(a_2) = 1, s(a_3) = 2, s(a_4) = 3, s(a_5) = 3, s(a_6) = 4$, tenemos que $B = \sum_{1<i<6} s(a_i) = 14$.

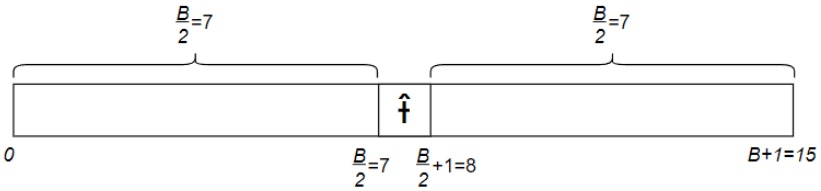
- Asignamos los valores $r(t_a) = 0, d(t_a) = B + 1, l(t_a) = s(a)$ para cada $a \in A$:
  - $r(t_{a_1}) = 0, \quad d(t_{a_1}) = B + 1 = 15, \quad l(t_{a_1}) = s(a_1) = 1$
  - $r(t_{a_2}) = 0, \quad d(t_{a_2}) = B + 1 = 15, \quad l(t_{a_2}) = s(a_2) = 1$
  - $r(t_{a_3}) = 0, \quad d(t_{a_3}) = B + 1 = 15, \quad l(t_{a_3}) = s(a_3) = 2$
  - $r(t_{a_4}) = 0, \quad d(t_{a_4}) = B + 1 = 15, \quad l(t_{a_4}) = s(a_4) = 3$
  - $r(t_{a_5}) = 0, \quad d(t_{a_5}) = B + 1 = 15, \quad l(t_{a_5}) = s(a_5) = 3$
  - $r(t_{a_6}) = 0, \quad d(t_{a_6}) = B + 1 = 15, \quad l(t_{a_6}) = s(a_6) = 4$
- Agregamos la tarea $\bar{t}$ con $r(\bar{t}) = \lceil B/2 \rceil = \lceil 14/2 \rceil = 7$, $d(\bar{t}) = \lceil (B+1)/2 \rceil = \lceil (14+1)/2 \rceil = \lceil 15/2 \rceil = \lceil 7{,}5 \rceil = 8$ y $l(\bar{t}) = 1$.

Ejemplo de transformación

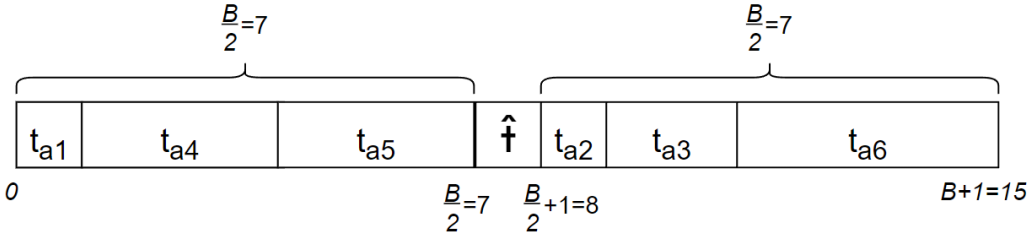


Ejemplo de transformación

Elegimos el subconjunto $A' = \{a_2, a_3, a_6\}$ y tenemos que $A - A' = \{a_1, a_4, a_5\}$



Ejemplo de transformación



Aplicaciones en la vida real

- Agendar las tareas que debe realizar una maquina.
- Crear horarios bonitos para la escuela.

Contenido

Forma canónica o estándar

- **Ejemplar genérico:** Dado un conjunto de trabajos $\mathcal{J} = \{J_1, \ldots, J_n\}$, una gráfica acíclica dirigida $P = (J, A)$ *A es un orden parcial de precedencia*, un entero m *(el número de procesadores)* y un entero T *(el lapso para cumplir los trabajos de $\mathcal{J}$)*.
- **Pregunta de decisión:** ¿Existe una función S de $\mathcal{J}$ a $\{1, 2, \ldots, T\}$, tal que S cumple con lo siguiente:
  - Para todo $j \leq T$ entonces $|\{J_i \mid S(J_i) = j\}| \leq m$
  - Sí $(J_i, J_j) \in A$ entonces $S(J_i) < S(J_j)$?

Entendiendo el enunciado

- Todos los trabajos tardan una unidad de tiempo en ejecutarse.
- Los trabajos son la tareas que debe realizar el procesador, como lo puede ser la ejecución de un programa.
- El **orden parcial** quiere decir que hay trabajos los cuales se pueden ejecutar sí y solamente sí los trabajos que los **preceden** ya han sido ejecutados.

Entendiendo el enunciado

- La restricción de *"Para todo* $j \leq T$ *entonces* $|\{J_i \mid S(J_i) = j\}| \leq m$ *"*, nos quiere decir que el número de tareas ejecutados en el tiempo j son a lo más m, ya que solamente tenemos m procesadores para realizar las tareas.
- La restricción de *"Sí* $(J_i, J_j) \in A$ *entonces* $S(J_i) < S(J_j)$ *"* indica la precedencia de las tareas es decir tenemos que ejecutar primero el trabajo J_i que el trabajo J_j ya que J_i tiene mayor precedencia o su equivalente: J_j no se puede ejecutar hasta que ya se haya ejecutado J_i.

Algoritmo no determinista

- Fase adivinadora:
  - Para la creación de la función S (recordemos que $S : \mathcal{J} \longrightarrow \{1, \ldots, T\}$), damos la siguiente regla de correspondencia:
    - Por cada $J \in \mathcal{J}$, tiramos un dado equilibrado de T caras, y el resultado que nos salga sera el lapso para cumplir el trabajo.
- Fase verificadora:
  - Checamos que cada restricción del problema se cumplan, en caso de que no se cumpla alguna entonces no es una solución. En otro caso sí es una solución para el ejemplar genérico.

Complejidad del algoritmo no determinista

- Fase adivinadora:
  - Tiramos $|\mathcal{J}|$ veces el dado equilibrados de T caras, así la complejidad en tiempo de éste paso está dado por $O(|\mathcal{J}|)$
  - Asignarle a cada $J \in \mathcal{J}$ el lapso para cumplir el trabajo tiene complejidad en tiempo de $O(|\mathcal{J}|)$.
- Fase verificadora:
  - Definimos al conjunto C_j como

$$\{J_i \mid S(J_i) = j\}$$

    es decir, C_j es el conjunto de trabajos que están en $\mathcal{J}$ tal que esos trabajos fueron ejecutados en el tiempo j.
  - Comprobar que para un tiempo j tal que $j \leq T$ se cumpla que $|C_j| \leq m$ nos toma tiempo $O(|C_j|)$.
  - Así para todo tiempo j tal que $j \leq T$ se cumpla que $|C_j| \leq m$ nos toma tiempo $O\left(T \cdot \sum_{j=1}^{T} |C_j|\right)$.
  - Comprobar que para todo $(J_i, J_j) \in A$ se cumpla que $S(J_i) < S(J_j)$ tiene complejidad en tiempo de $O(|A|)$.

Clique

Definición:
Un **clique** es un subconjunto V' de vértices de la gráfica $G = (V, E)$ tal que sí $u, v \in V'$ entonces $(u, v) \in E$

Ejemplo

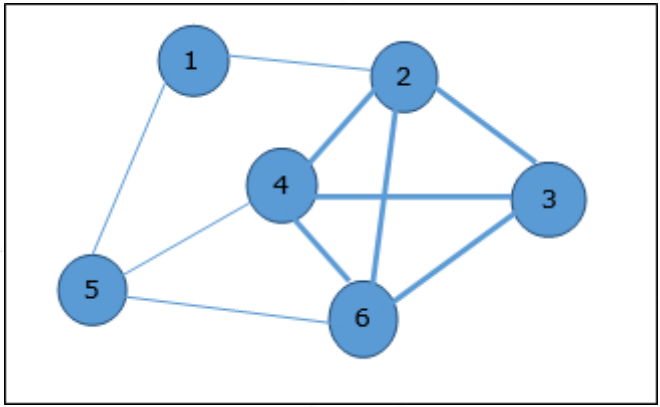


Figura 2: Gráfica G

Ejemplo

El conjunto de vértices de G definido como $\{2, 4, 6, 3\}$ es un **clique** en G mientras que el conjunto de vértices de G definido como $\{1, 2, 4\}$ ya que $\{1, 4\} \notin E$

Problema del Clique

Dada una gráfica G y un entero K, ¿G contiene un clique de tamaño al menos K?
El problema del **clique** es **NP**-Completo

Teorema

El problema **Multiprocessor Scheduling** es **NP**-Completo.

Demostración

Transformaremos **Clique** a **Multiprocessor Scheduling**.
Sea $G = (V, E)$ una gráfica tal que:

- V es el conjunto de vértices de la gráfica.
- E es el conjunto de aristras de la gráfica.

y sea $k \in \mathbb{Z}^+$ tal que $k \leq |V|$, lo anterior es un ejemplar genérico de **Clique**.
Construiremos un conjunto de trabajos $\mathcal{J}$, una gráfica acíclica $P = (\mathcal{J}, A)$ con orden parcial de precedencia A y enteros m y T de tal manera que existe la función S de $\mathcal{J}$ a T sí y sólo sí G tiene un **clique**.

Demostración

Supondremos que G no tiene vértices aislados, ésto no hace que perdamos generalidad ya que los vértices aislados *(los que no tienen aristas incidiendo en ellos)* podemos eliminarnos de la gráfica y éstos no afectan el tamaño del **clique** ya que por definición ninguno de éstos vértices aislados forma parte de él.
Definimos al conjunto de trabajos $\mathcal{J}$ de la siguiente manera:

$$\mathcal{J} = V \cup E \cup B \cup C \cup D.$$

donde B, C y D no son $\varnothing$ y además son disjuntos entre sí. Definimos a los anteriores conjuntos como sigue:

$$B = \{b_1, \ldots, b_{|B|}\}$$
$$C = \{c_1, \ldots, c_{|C|}\}$$
$$D = \{d_1, \ldots, d_{|D|}\}$$

Demostración

$|B|$, $|C|$ y $|D|$ satisfacen las siguientes igualdades:

$$m = k + |B| = \binom{k}{2} + |V| - k + |C| = |E| - \binom{k}{2} + |D|$$

$$min\,(|B|, |C|, |D|) = 1$$

A T se le asigna el valor de 3.
Para construir la gráfica $P = (\mathcal{J}, A)$, iniciamos añadiendo a A todas las duplas posibles de la forma b_i, c_j y c_j, d_k. Después agregamos las duplas (v, e) para todo $v \in V$ y $e \in E$ tal que e incide en v. Ahora tenemos un ejemplar genérico de **Multiprocessor Scheduling**.

Demostración

El ejemplar se construyó en tiempo polinomial ya que:

- Construir el conjunto $\mathcal{J}$ está dado por $O(|V| + |E| + |B| + |C| + |D|)$.
- Construir los números m y T se hace en tiempo constante $O(1)$ ya que siempre se efectúan el mismo número de operaciones sin importar la cardinalidad de B, C, D, E y V en el caso de m.
- Construir el conjunto A está dado por $O(|B| \cdot |C| \cdot |D| + |V| \cdot |E|)$.

Demostración

Afirmación:
Existe la función S para $\mathcal{J}$ bajo el orden parcial de precedencia A, con m y T sí y solamente sí G tiene un **clique** *de tamaño k.*

Demostración

Ida

- Supongamos que existe la función S del conjunto $\mathcal{J}$ al conjunto $\{1, \ldots, T\}$ tal que cumple:
  - Para todo $j \leq T$ entonces $|\{J_i \mid S(J_i) = j\}| \leq m$
  - Sí $(J_i, J_j) \in A$ entonces $S(J_i) < S(J_j)$
- Como $T = 3$ y por construcción de A tenemos que todos los trabajos que están en B deben de ser ejecutados antes que aquellos trabajos están en C y éstos a su vez se ejecutan primero que los trabajos que están en D.

Demostración

Necesariamente tenemos que:

$$S(b) = 1 \ \forall b \in B$$
$$S(c) = 2 \ \forall c \in C$$
$$S(d) = 3 \ \forall d \in D$$

Notemos que:

$$|\mathcal{J}| = T \cdot m$$

Ya que:

$$|\mathcal{J}| = |V| + |E| + |B| + |C| + |D|$$

porque V, E, B, C y D son todos disjuntos entre sí.

Demostración

Por las restricciones que pusimos al contruir a m tenemos lo siguiente:

$$|B| = m - k$$
$$|V| + |C| = m + k - \binom{k}{2}$$
$$|D| + |E| = m + \binom{k}{2}$$

Demostración

Así

$$\begin{aligned}|\mathcal{J}| &= m - k + m + k - \binom{k}{2} + m + \binom{k}{2} \\ &= m + m + m - k + k - \binom{k}{2} + \binom{k}{2} \\ &= m + m + m = 3 \cdot m\end{aligned}$$

como $T = 3$ entonces

$$|\mathcal{J}| = T \cdot m$$

Demostración

- Consecuentemente los m procesadores siempre están ejecutando algún trabajo durante los 3 períodos de ejecución.
- Notemos que el último período además de los trabajos que están en el conjunto D por la restricción de que $m = |E| - \binom{k}{2} + |D|$ tenemos que se ejecutan $|E| - \binom{k}{2}$ trabajos que corresponden a las aristas de G.

Demostración

- Esto es así ya que sí el trabajo correspondiente a un vértice se ejecuta en el último período de ejecución, por construcción el trabajo de las aristas que inciden en ese vértice no se podrían ejecutar hasta después del tercer período, es por eso que supusimos que en G no hay vértices aislados, ya que sí los hubiera, entonces S no existiría con $T = 3$ como hemos supuesto.
- Los $\binom{k}{2}$ trabajos restantes corresponden a trabajos de las aristas que se tienen que ejecutar en el segundo período.
- Los trabajos correspondiente a los vértices se ejecutan en el primer y segundo período.

Demostración

- Como en el primer período se ejecutan todos los trabajos correspondientes a B, también se deben de ejecutar k trabajos que corresponden V, ésto se debe a la restricción $m = k + |B|$. En el segundo período se ejecutan $|V| - k$ trabajos que corresponden a los vértices de G.
- Por lo tanto hemos obtenido lo siguiente:
  - En el primer período se ejecutan k trabajos correspondientes a los vértices.
  - En el segundo período se ejecutan $\binom{k}{2}$ trabajos correspondientes a las aristas.

Demostración

- La única manera en que pueda ocurrir lo anterior es que las aristas *(cuyos trabajos correspondientes son ejecutados en el* **segundo período***)* sean de la forma (u, v) con u y v vértices cuyos trabajos correspondientes fueron ejecutados en el **primer período**.
- Así para cualquier par de vértices v' y u' de los k vértices existe una arista (v', u'), que por definición es un **clique**.
- Por lo tanto en G existe un **clique** de tamaño k.

Demostración

Regreso

Si G tiene un **clique** V' de tamaño k, entonces diseñamos una función S de la siguiente manera:

$$S(b) = 1 \ \forall b \in B$$
$$S(c) = 2 \ \forall c \in C$$
$$S(d) = 3 \ \forall d \in D$$
$$S(v) = 1 \text{ sí y sólo sí } v \in V'$$
$$S(v) = 2 \text{ sí y sólo sí } v \notin V'$$
$$S([v,u]) = 2 \text{ sí y sólo sí } v, u \in V'$$
$$S([v,u]) = 3 \text{ sí y sólo sí } v, u \notin V'$$

Así S por construcción cumple con los siguiente:

- Para todo $j \leq T$ entonces $|\{J_i \mid S(J_i) = j\}| \leq m$
- Sí $(J_i, J_j) \in A$ entonces $S(J_i) < S(J_j)$

Ejemplificación de la transformación

Consideremos la gráfica $G = (V, E)$ definida como sigue:

$$V = \{v_1, v_2, v_3, v_4\}$$
$$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_3, v_4\}\}$$

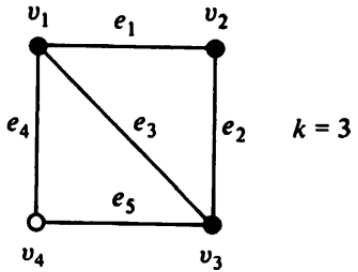


Figura 3: Gráfica G

Ejemplificación de la transformación

Observemos que:

$$|V| = 4$$
$$|E| = 5$$

Redefiniremos a cada arista de E de la siguiente manera:

$$e_1 = \{v_1, v_2\}$$
$$e_2 = \{v_2, v_3\}$$
$$e_3 = \{v_1, v_3\}$$
$$e_4 = \{v_1, v_4\}$$
$$e_5 = \{v_3, v_4\}$$

Ejemplificación de la transformación

Así

$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

Un **clique** de la gráfica G es el conjunto V' definado como sigue:

$$V' = \{v_1, v_2, v_3\}$$

$|V'| = 3$. Definimos a k como 3.

Ejemplificación de la transformación

Definimos el conjunto B como sigue:

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

Definimos el conjunto C como sigue:

$$C = \{c_1\}$$

Definimos el conjunto D como sigue:

$$D = \{d_1, d_2, d_3\}$$

Notemos que los conjuntos B, C, D son disjuntos.
De lo anterior obtenemos que:

$$|B| = 2$$
$$|C| = 1$$
$$|D| = 3$$

Ejemplificación de la transformación

Definimos a m como sigue:

$$m = k + |B|$$
$$m = \binom{k}{2} + |V| - k + |C|$$
$$m = |E| - \binom{k}{2} + |D|$$

Sustituyendo

$$m = 3 + 2 = 5$$
$$m = 3 + 4 - 3 + 1 = 5$$
$$m = 5 - 3 + 3 = 5$$

Notemos que también las cardinalidades de los conjuntos B, C, D cumplen con:

$$\text{mín}\,(|B|, |C|, |D|) = 1$$

Ejemplificación de la transformación

Calcularemos $B \times C$

$$B \times C = \{(b_1, c_1), (b_2, c_2)\}$$

Calcularemos $C \times D$

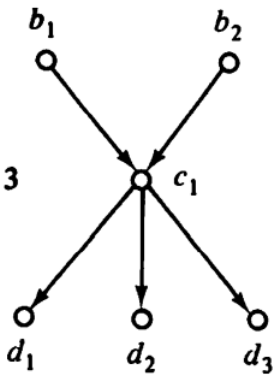
$$C \times D = \{(c_1, d_1), (c_1, d_2), (c_1, d_3)\}$$

Ejemplificación de la transformación

Orden parcial de los trabajos que están en los conjuntos B, C y D

$T = 3, m = 5$

$|B| = 2, |C| = 1, |D| = 3$



Ejemplificación de la transformación

Sea A' el conjunto definido como sigue:

$$A' = \{(v, e) \mid v \in V, e \in E \text{ tal que } e \text{ incide en } v\}$$
$$A' = \{(v_1, e_1), (v_1, e_3), (v_1, e_4), (v_2, e_1), (v_2, e_2), (v_3, e_3), (v_3, e_5), (v_4, e_4), (v_4, e_4)\}$$

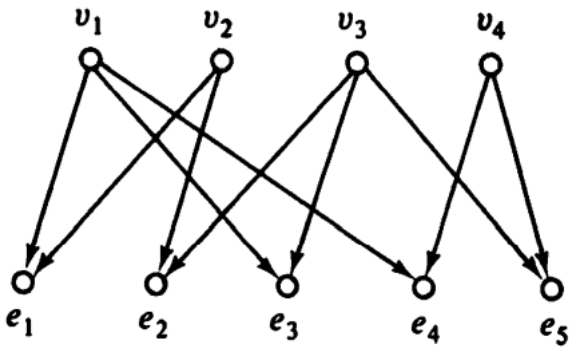
Definimos a A como

$$A = B \times C \cup C \times D \cup A'$$

A lo tomaremos como un orden parcial de precedencia.

Ejemplificación de la transformación

Orden parcial de los trabajos que están en los conjuntos V y E



Ejemplificación de la transformación

- Por lo tanto hemos convertido un ejemplar particular de **Clique** en un ejemplar particular de **Multiprocessor Scheduling**.

Técnica para la demostración del problema

- Reemplazo Local

Aplicación en la vida real

- Encontrar una manera eficiente por parte del **Sistema Operativo** de ejecutar tareas en los múltiples procesadores de la computadora.

Referencias I

[1] Garey. M.R and Johnson, D. S Computer and intractability. A guide to the Theory of NP-Completness *W.H. Freeman and Company. New York (1979), 66-72.*

[2] Papadimitriou, Ch Combinatorial Optimization: Algorithms and Complexity *Dover Publications, INC. Mineola, New York (1982), 363-366.*