FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico N°2

Nombre: Melina Lazzaro **Padrón:** 105931
Nombre: Nahuel Castro **Padrón:** 106551
Nombre: Ana Gutson **Padrón:** 105853
Nombre: Nicolás Pinto **Padrón:** 105064
Nombre: Iván Litteri **Padrón:** 106223

Fecha de Entrega: 20/10/2021
Grupo: liomessi30

Lineamientos básicos

- El trabajo se realizará en grupos de cinco personas.
- Se debe entregar el informe en formato pdf y código fuente en (.zip) en el aula virtual de la materia.
- El lenguaje de implementación es libre. Recomendamos utilizar C, C++ o Python. Sin embargo si se desea utilizar algún otro, se debe pactar con los docentes.
- Incluir en el informe los requisitos y procedimientos para su compilación y ejecución. La ausencia de esta información no permite probar el trabajo y deberá ser re-entregado con esta información.
- El informe debe presentar carátula con el nombre del grupo, datos de los integrantes y y fecha de entrega. Debe incluir número de hoja en cada página.
- En caso de re-entrega, entregar un apartado con las correcciones mencionadas

Índice general

Capítulo 1

Una campaña publicitaria masiva pero mínima

Una empresa de turismo que vende excursiones desea realizar una campaña publicitaria en diferentes vuelos comerciales con el objetivo de llegar a todos los viajeros que parten del país A y que se dirigen al país B. Estos viajeros utilizan diferentes rutas (algunos vuelos directos, otros armando sus propios recorridos intermedios). Se conoce para una semana determinada todos los vuelos entre los diferentes aeropuertos con sus diferentes capacidades. Además parten del supuesto que durante ese periodo la afluencia entre A y B no se verá disminuida por viajes entre otros destinos.

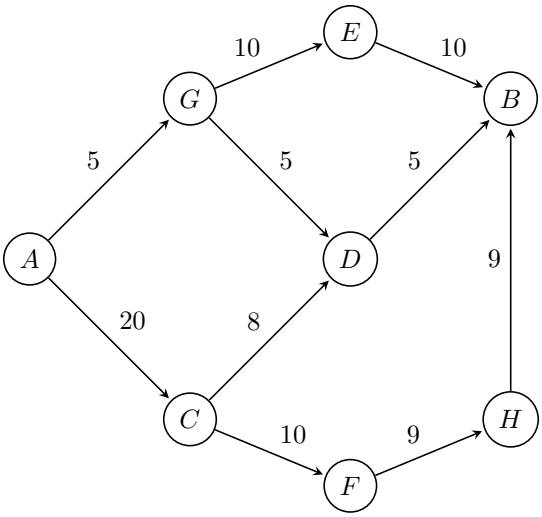
Desean determinar en qué trayectos simples (trayecto de un viaje que inicia desde un aeropuerto y termina en otro) poner publicidad de forma de alcanzar a TODAS las personas que tienen el destino inicial A y el destino final B. Pero además desean que siempre que sea posible seleccionen la combinación que tenga el menor número de vuelos comerciales. Esto es porque pagan tanto por cantidad de vuelos como por pasajeros que cumplan con la condición de ser del país de origen A y con destino final B.

Se pide:

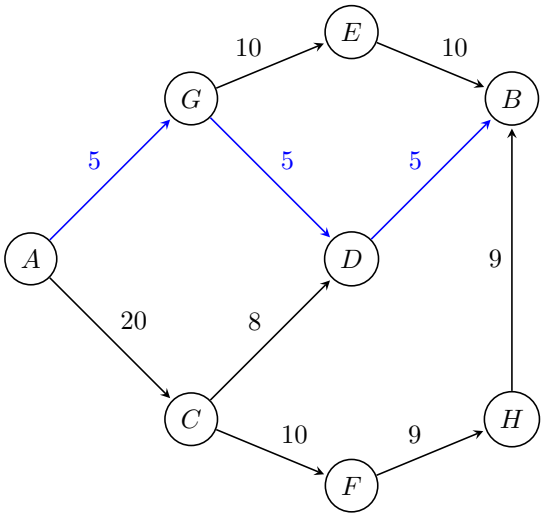
1. Proponer una solución algorítmica que resuelva el problema de forma eficiente. Explicarla paso a paso. Utilice diagramas para representarla.
2. Plantear la solución como si fuese una reducción de problema. ¿Puede afirmar que corresponde a una reducción polinomial? Justificar.
3. ¿Podría asegurar que su solución es óptima?
4. Programe la solución
5. Compare la complejidad temporal y espacial de su solución programada con la teórica. ¿Es la misma o difiere?

1.1. Solución algorítimica que resuelve el problema de forma eficiente. Paso a paso

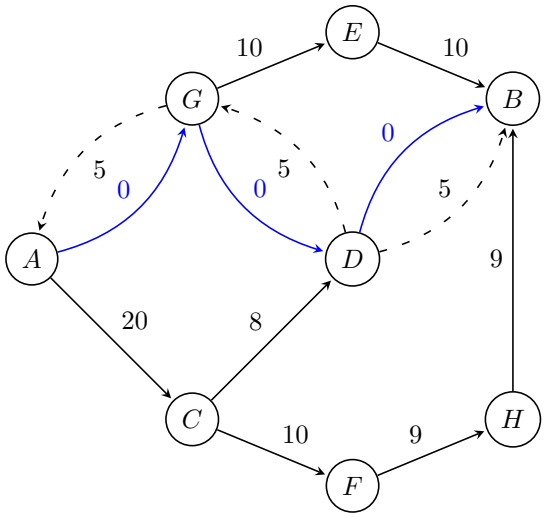
Modelamos el problema con un grafo donde los vuelos son aristas cuyos pesos son sus capacidades máximas y los aeropuertos son vértices. Luego, resolvemos mediante el algoritmo de Edmonds-Karps de la siguiente manera:



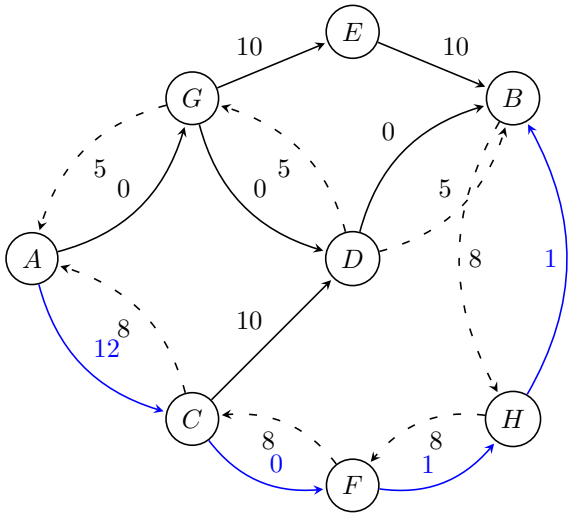
1. Se utiliza el algoritmo de BFS (Breadth First Search o Búsqueda en Anchura) para encontrar el camino mínimo de A a B, solo considerando las aristas que no estén saturadas.

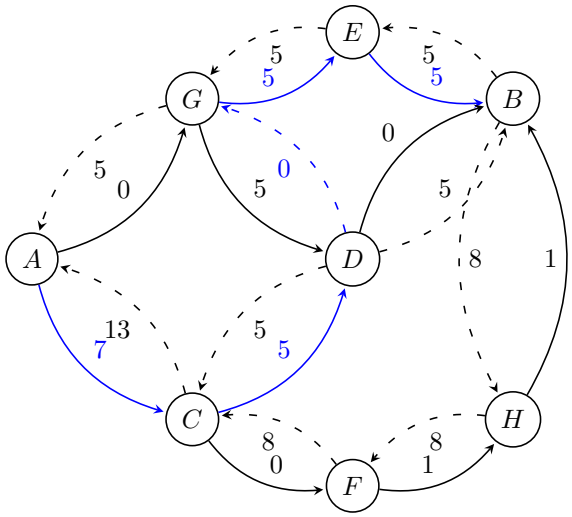


2. Se calcula la menor capacidad en el camino, a la cual llamamos cuello de botella, que le resta a la capacidad de la arista original y se le suma a la capacidad de la arista residual (la arista en el sentido contrario, de capacidad máxima 0).



3. Se repiten los pasos 1-2 hasta que no queden caminos de A a B con capacidad disponible.





Finalmente, tenemos el grafo residual y el flujo máximo que es la suma de todos los cuellos de botella.

4. Utilizando el grafo residual obtenemos el corte mínimo de la siguiente manera:

   a) Dividimos el grafo en 2 grupos, el grupo A será el que contenga todos los vértices accesibles desde el vértice fuente. El grupo B tendrá al resto de vértices.

   b) Observando el grafo original, diremos que todas las aristas que conecten un vértice del grupo A con un vértice del grupo B, están en el corte mínimo.

5. Finalmente, las publicidades serán colocadas en todas las aristas que estén en el corte mínimo.

1.2. Solución algorítima como si fuese una reducción del problema

Acá va el 1.2

1.3. Optimalidad de la solución

La solución propuesta por Edmonds-Karp es óptima ya que el algoritmo termina cuando no quedan caminos de A a B sin saturar, el flujo calculado es el máximo.

1.4. Solución Programada (clickeame)

1.5. Comparación temporal y espacial de la solución programada

La complejidad teórica del algoritmo de Edmons-Karp es $O(VE^2)$ ya que utiliza BFS para encontrar los caminos. La complejidad temporal del corte mínimo es $O(V^2)$ ya que en el peor de los casos todos los vértices están conectados ente sí y las aristas que conectan con el destino son las que generan el cuello de botella. La complejidad total de ambas operaciones es $O(VE^2)$.

Con respecto a su complejidad espacial, se utiliza un grafo residual de V.2E y BFS utiliza una cola de a lo sumo V elementos, por lo que la complejidad espacial de Edmons-Karp es $O(VE)$. La complejidad espacial del corte mínimo es $O(V)$ ya que en el peor de los casos hay V vértices en el grupo A. Por lo tanto, la complejidad temporal total de la operación es $O(VE)$

Las complejidades temporal y espacial de la solución programada coinciden con las teóricas.

Capítulo 2

Equipos de socorro

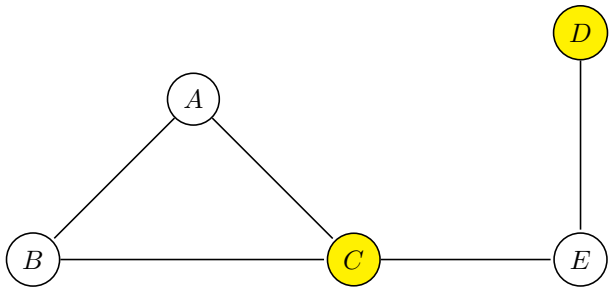
El sistema ferroviario de un país cubre un gran conjunto de su territorio. El mismo permite realizar diferentes viajes con transbordos entre distintos ramales y subramales que pasan por sus principales ciudades. Dentro de su proceso de mejoramiento del servicio buscan que ante una emergencia en una estación se pueda llegar de forma veloz y eficiente. Consideran que eso se lograría si el equipo de socorro se encuentra en esa misma estación o en el peor de los casos en una estación vecina (que tenga una trayecto directo que no requiere pasar por otras estaciones). Como los recursos son escasos desean establecer la menor cantidad de equipos posibles (un máximo de k equipos pueden solventar). Se solicita nuestra colaboración para dar con una respuesta a este problema.

Se pide:
1. Utilizar el problema conocido como "set dominante" para demostrar que corresponde a un problema NP-Completo.
2. Asimismo demostrar que el problema set dominante corresponde a un problema NP-Completo.
3. Con lo que demostró responda: ¿Es posible resolver de forma eficiente (de forma "tratable") el problema planteado?

HINT: podría ser una buena idea utilizar 3SAT o VERTEX COVER.

2.1.

En este problema podemos considerar las ciudades como vertices de un grafo y las aristas como los caminos que los conectan.
El problema presentado se puede resolver como el problema del set dominante, en este problema se busca un subset de vertices 'T' tal que todos los vertices que no estan en T sean adyacentes a al menos a un vertice que pertenezca a este subset.
Lo que debemos hacer es encontrar el numero minimo de vertices con los que podemos formar T, puesto que este problema se trata de elegir la cantidad minima de equipos de supervivencia para cubrir todas las ciudades y sus adyacentes, con lo que estariamos cumpliendo con el requerimiento pedido



En este caso vemos como las ciudades coloreadas son las que deberian tener el equipo de socorro, puesto que se cubren a si mismas y a las adyacentes, llegando asi a todas las ciudades
Con esto explicado sabemos que si el problema del set dominante es NP-completo por consiguiente nuestro problema tambien lo sera

2.2.

Para demostrar que un problema es NP-Completo basta con elegir un problema $Y \in NP - Completo$ y reducirlo polinomialmente a nuestro problema. Para ello podemos elegir el problema de vertex cover que es parte de los problemas NP Completos.

Cada instancia del problema de vertex cover consiste de un grafo $G = (V, E)$ y un numero k que sera el numero con el que intentaremos crear un subset de vértices que cubra todo el grafo. Para reducir este problema a uno de set dominante podemos modificar el grafo que tenemos de forma tal que el set dominante nos devuelva si se puede resolver el problema con el mismo numero k para el set dominante y para el vertex cover. La forma de crear el grafo seria para cada arista E que una a los vértices u,v del grafo G, crear un nuevo vértice auxiliar uv y unirlo a estos dos vértices u y v formando así triángulos

El nuevo grafo G' puede crearse en tiempo polinomial agregando las aristas correspondientes a los nuevos vértices en tiempo O(V+E). La prueba de que esto funciona viene dado por las siguientes afirmaciones:

Si algún problema es un NP, entonces, dado un 'certificado', que es una solución al problema y una instancia del problema (en este caso, un grafo G y un entero positivo k), se podrá verificar, comprobando si la solución dada es correcta o no, el certificado en tiempo polinomial.

El certificado es una secuencia de vértices que forman un Set Dominante en el grafo. Podemos validar esta solución comprobando que:
i. todos los vértices pertenecen a los vértices del grafo,
ii. todos los vértices que no forman parte de esta secuencia son adyacentes a algunos de los vértices de este conjunto.

Esto se puede hacer en tiempo polinomial, es decir $O(V + E)$ usando la siguiente estrategia:

```
flag = verdadero
para cada vertice v en V:
    si v no pertenece al Set Dominante:
        verificar el conjunto de aristas correspondientes a v
        si v no es adyacente:
            para cada uno de los vertices en el Set Dominante:
                flag = falso
si (flag)
```

la solucion es correcta
sino
la solucion es incorrecta

Ahora, sin embargo, para probar que un Set Dominante es un NP-Hard, habrá que reducir algún problema NP-Hard que conozcamos a este problema. Haremos una reducción desde el problema de Cobertura de Vértices al del problema del Set Dominante.

Cada paso del problema de Cobertura de Vértices consta de un grafo G, con V vértices y E aristas, y un entero K que consta del subconjunto de vértices, ya que la entrada se puede convertir en un nuevo problema del Set Dominante. Este nuevo problema tendrá un grafo G', con V' vértices y E' aristas= (V', E').

Este nuevo grafo G' será construido de la siguiente manera:

E':
Para cada arista del grafo G, cuyos vértices que conecta son a y b:
Agregar un nuevo vértice v
Unirlo con a
Unirlo con b

Se genera, entonces, por cada arista del grafo G, un nuevo vértice v con dos aristas.

V': Agregar todos los vértices V del grafo original G.

Este nuevo grafo G' se puede obtener en tiempo polinomial: el agregar nuevas aristas por cada nuevo vértice conlleva una complejidad temporal $O(V + E)$. Esta reducción puede probarse mediante las siguientes dos afirmaciones:

- Supongamos que el grafo G tiene una cubierta de vértices CV de tamaño k. Cada arista en G tiene uno de los vértices que pertenecen a CV. Por lo tanto, para cada arista e, que consta de vértices {a, b}, al menos a o b es parte de CV. Entonces, si a está contenido en CV, por ende el vértice adyacente es b, también está cubierto por alguno de los elementos en CV. Es decir,

  a,b vértices unidos por e arista
  $a \in$ CV $\quad \vee \quad b \in$ CV
  b adyacente a $a \quad \Rightarrow \quad b$ cubierto por $w \in$ CV

  Ahora, para todos los vértices ab recientemente agregados por cada arista, el vértice es adyacente tanto a a como a b, uno de los cuales es al menos parte de CV, como se demostró anteriormente. Por lo tanto, los vértices adicionales para todas las aristas también están cubiertos por CV.

  Además, el conjunto de vértices que forman la cobertura de vértices de tamaño k conforman el set dominante en el grafo G'. Por lo tanto, si G tiene una cobertura de vértices, G' tiene un set dominante del mismo tamaño. Es decir,

  $$\text{tamaño CV}(G) = \text{tamaño SD}(G')$$

- Suponemos que el grafo G' tiene un set dominante de tamaño k. Pueden surgir dos posibilidades:
  i. el vértice en el SD es un vértice original,

ii. el vértice en el SD pertenece al vértice ab recién agregado por cada arista a, b.

En el caso ii., dado que cada nuevo vértice está conectado a los dos vértices de la arista, a y b, por lo tanto, puede ser reemplazado por a o b. Como estos tres vértices forman un triángulo, entonces, incluso sustituyendo la vista con a o b, podemos continuar abarcando todos los vértices que se extendieron antes de reemplazar. Esto conducirá a la eliminación de todos los vértices recién agregados mientras abarca todas las aristas del grafo G'. Los vértices recién agregados están dominados por el SD modificado y cubren todas las aristas en G con al menos a o b para cada arista ab. Por lo tanto, si G' tiene un set dominante de tamaño k, G tendrá una cubierta de vértice con el tamaño máximo k.

En resumen, podemos decir que el grafo G' contiene un set dominante si el grafo G contiene una cobertura de vértices. Por lo tanto, cualquier instancia del problema del set dominante puede reducirse a una instancia del problema de cobertura de vértices. Entonces, el set dominante también es NP-Hard. Dado que la cobertura de vértices se encuentra en las clases NP y NP-Hard, el set dominante de un grafo es NP-Completo.

2.3.

Con lo demostrado actualmente no podemos afirmar que el problema se puede resolver de una forma eficiente, tomando eficiente como una resolucion polinomica, puesto que si bien demostramos que el problema es NP-Completo, no podemos afirmar que sea un problema del tipo P. La igualdad entre los problemas tipo P y NP es uno de los problemas que se siguen intentando demostrar, y su respuesta hasta el dia de hoy sigue siendo una incognita, asi que solamente sabiendo que el problema es NP-Completo no podemos determinar que sea tipo P

Capítulo 3

Un poco de teoría

1. Defina y explique qué es una reducción polinomial y para qué se utiliza.
2. Explique detalladamente la importancia teórica de los problemas NP-Completos.
3. Tenemos un problema A, un problema B y una caja negra NA y NB que resuelven el problema A y B respectivamente. Sabiendo que B es P

a. Qué podemos decir de A si utilizamos NA para resolver el problema B (asumimos que la reducción realizada para adaptar el problema B al problema A es polinomial)

b. Qué podemos decir de A si utilizamos NB para resolver el problema A (asumimos que la reducción realizada para adaptar el problema A al problema B es polinomial)

c. ¿Qué pasa con los puntos anteriores si no conocemos la complejidad de B, pero sabemos que A es NP-C?

3.1.

Una reduccion polinomial es un procedimiento que permite pasar una instancia de un problema A a otro problema B en una complejidad polinomial, y se usa la "facilidad"del problema B para probar la "facilidad"de A.

Un problema A es reducible a un problema B si existe una función polinomial $f : \Sigma^* \to \Sigma^*$ donde para cada string w, $w \in A \leftrightarrow f(w) \in B$. Podemos decir que f es una reducción polinomial de A a B.

Dada una instancia α de un problema A, se usa un algoritmo de reducción polinomial para transformarlo a una instancia β de un problema B, resolvemos β mediante un algoritmo de decisión de complejidad polinomial y esta solución la transformaremos en la solución α.

Se concluye que el segundo es al menos tan dificil (misma complejidad temporal para resolverlo) que el primero. Es por ello que las reducciones se pueden utilizar para probar que un problema es NPHARD reduciendo un problema que previamente se conocıa como NPC a una instancia del problema a demostrar.

Figura 3.1: Ejemplo gráfico de la reducción y resolución de Y

3.2.

Los problemas NP-Completos (NP-C) son aquellos que están incluidos en las clases NP y NP-Hard. Es decir, los problemas NP-Completos se pueden verificar en tiempo polinomial y cualquier problema NP se puede reducir a este problema en tiempo polinomial.

Los lenguajes NP-Completos son importantes porque se piensa que todos los lenguajes NP-Completos tienen una dificultad similar, en ese proceso, resolver uno implica que otros también se resuelven.

Si se demuestra que algún problema NP-Completo está en P, entonces se demuestra que todos los NP están en P, teniendo en cuenta que cualquier problema NP se puede reducir a un problema NP-Completo y usando esta reducción y el algoritmo para el problema NP-Completo dado para resolver cualquier problema en NP. Por lo tanto, todos tendrán soluciones de tiempo polinomiales.

Si algún problema NP-Completo no está en P, entonces esto significa que este lenguaje está en NP pero no en P, por lo tanto, esto certifica que P y NP no son iguales.

Por lo tanto, P frente a NP se puede resolver si se demuestra que algún problema de NP-Completo está o no, en P.

3.3.

Analicemos, para comenzar, los datos que tenemos:

- Una caja negra NA que resuelve A,
- una caja negra NB que resuelve B,
- un problema A,
- un problema B que es P.

Con este último dato, sabemos entonces también que B es NP.

Si se quisiera hallar una relación entre A y B, y sus resoluciones, deberíamos conocer, al menos, si alguno es reducible en tiempo polinómico al otro, es decir:

$$A \leq_p B \ \vee \ B \leq_p A$$

Mientras tanto, no hay nada más que podamos decir.

3.3.a.

Ahora se cuenta con más información, tal que si se recopila toda la información, se tiene:

- Una caja negra NA que resuelve A,
- **Una caja negra NA que resuelve B**,
- una caja negra NB que resuelve B,
- un problema A,
- un problema B que es P, y por ende NP,
- **una relación entre los problemas A y B tal que $B \leq_p A$.**

Este último dato lo que quiere decir es que B es reducible en tiempo polinomial a A, o lo que es lo mismo, que A es al menos tan difícil como B.
Es decir, la complejidad de resolver el problema A es mayor o igual a la complejidad de resolver el problema B.
Se concluye, entonces, que B está incluído en el mismo grupo que A. No se puede asegurar nada más sobre A, ya que tranquilamente podría ser un problema por fuera del conjunto NP.

3.3.b.

Se nos plantea otra situación similar a la anterior, los datos ahora son:

- Una caja negra NA que resuelve A,
- **Una caja negra NB que resuelve A**,
- una caja negra NB que resuelve B,
- un problema A,
- un problema B que es P, y por ende NP,
- **una relación entre los problemas A y B tal que $A \leq_p B$.**

Se sabe entonces que A puede ser resuelto con la misma caja negra NB que B. Además $A \leq_p B$, que como se dijo anteriormente, quiere decir que B es al menos tan difícil como A.

Si B es al menos tan difícil como A, A es resuelto con NB, y a su vez, NB también resuelve B, recordando que B es P, resulta que:

$$A \text{ es P}$$

Y por ende, también es NP.

3.3.c.

La situación a analizar es:

- Una caja negra NA que resuelve A,
- una caja negra NB que resuelve B,
- un problema A, que es NP-C,
- un problema B.

Es decir, en esta situación, se desconoce la complejidad de B.

A su vez, tampoco se puede decir si el problema A tiene una solución que pertenece al conjunto P, pero al conocer que es NP-C si podemos decir que es NP y NP-HARD al mismo tiempo.

Esto último es debido a que todavía no se ha demostrado que P = NP. Llegado el caso que se demuestre, entonces, ahí sí se podría decir que A es P, y por ende también NP.

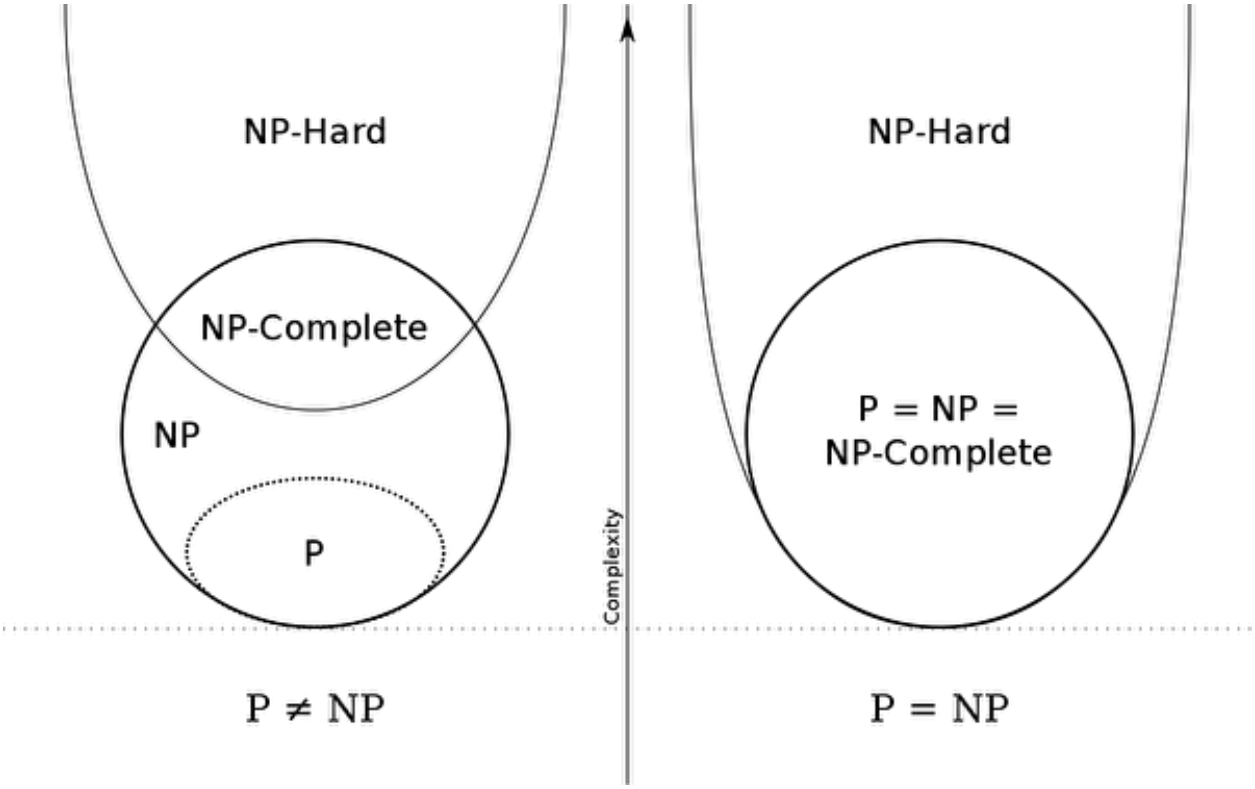


Figura 3.2: Relaciones

Se procede, entonces, a ver cada caso planteado en el enunciado.

a. Se tiene:

- Una caja negra NA que resuelve A,
- **Una caja negra NA que resuelve B**,
- una caja negra NB que resuelve B,
- un problema A, que es NP-C,
- un problema B,
- **una relación entre los problemas A y B tal que $B \leq_p A$.**

Nuevamente, gracias al último dato, se demuestra que B tiene una complejidad menor o igual a la de A. Por ende, B es a lo sumo NP-Hard.

b. Se tiene:

- Una caja negra NA que resuelve A,
- **Una caja negra NB que resuelve A,**
- una caja negra NB que resuelve B,
- un problema A, que es NP-C,
- un problema B,
- **una relación entre los problemas A y B tal que $A \leq_p B$.**

Se sabe que el problema B es al menos tan dificil como A, o mejor dicho, B tiene una complejidad mayor o igual a la complejidad de A.

Se concluye, finalmente, que A es al menos NP-Hard. Esto se puede observar con claridad en el diagrama anterior.

Bibliografía

[1] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms (tercera edición), MIT Press (2009).

[2] J. Kleinberg, E. Tardos, Algorithm Design, Addison Wesley (2006).

[3] Nguyen, M., Hà, M., Nguyen, D. and Tran, T., 2020. Solving the k-dominating set problem on very large-scale networks. Computational Social Networks, 7(1).

[4] GeeksforGeeks. 2021. GeeksforGeeks — A computer science portal for geeks. [online] Available at: <https://www.geeksforgeeks.org/> [Accessed 9 November 2021].

[5] GeeksforGeeks. 2021. GeeksforGeeks — A computer science portal for geeks. [online] Available at: <https://www.geeksforgeeks.org/> [Accessed 9 November 2021].

[6] Stack Overflow. 2021. Stack Overflow - Where Developers Learn, Share, Build Careers. [online] Available at: <https://stackoverflow.com/>.

[7] Stackexchange.com. 2021. Questions - Stack Exchange. [online] Available at: <https://stackexchange.com>.

[8] Lcm.csa.iisc.ernet.in. 2021. 10.1 Importance of NP-Completeness. [online] Available at: <http://lcm.csa.iisc.ernet.in/dsa/node217.html>.