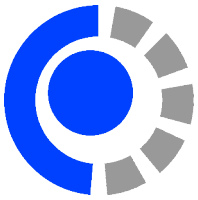
Universidad Nacional del Comahue
Facultad de Informática
Departamento de Programación



Diseño de Algoritmos

Trabajo Práctico 2: asignaciones estables y problemas $\mathcal{NP}$

Ignacio Alejandro Navarro Oliva

2024

Índice

Introducción

En este informe se detallan los ejercicios y su resolución del trabajo práctico 3 de la materia Diseño de Algoritmos, de la Licenciatura en Ciencias de la Computación, carrera dictada en la Universidad del Comahue, Neuquén. Aquellos ejercicios que requieran de código, si bien algunas porciones de este están presentadas en este informe, también se encuentran en un repositorio de GitHub.

1. Asignaciones estables

El problema de asignación de parejas es uno de los cinco problemas representativos propuestos por Jon Kleinberg y Eva Tardós en el libro *Algorithm Design*. Este problema plantea realizar una asignación de N hombres con N mujeres, teniendo en cuenta que cada uno de ellos tiene una lista de preferencias. Gracias al algoritmo diseñado por Gale y Shapley, se puede realizar dicha asignación en tiempo polinomial, y cumpliendo propiedades deseadas como la correctitud, estabilidad y completitud.

El ejercicio plantea agregar una restricción más al problema. Existe ahora una colección de pares P del tipo (h, m) que denotan parejas prohibidas. Dichas parejas no se tienen en cuenta en sus listas de preferencia, y obviamente no deben terminar juntos cuando el programa termina. Entonces, una asignación S es estable si no presenta ninguno de los siguientes tipos de inestabilidad:

1. No existen dos pares (h, m) y (h', m') en S con la propiedad que si $(h', m') \notin S$, h prefiere a m', y m prefiere a h en lugar de h' (inestabilidad básica)
2. Existe un par $(h, m) \in S$, y un hombre h', tal que h' no es parte de ningún par en la asignación, $(h', m) \notin P$, y m prefiere a h' en lugar de h (hombre soltero más deseable y no prohibido)
3. Existe un par $(h, m) \in S$ y una mujer m' tal que m' no es parte de ningún par en la asignación, $(h, m') \notin P$, y h prefiere a m' en lugar de m (mujer soltera más deseable y no prohibida)
4. Existe un hombre h y una mujer m, ninguno de los cuales forman parte de una pareja en el emparejamiento y $(h, m) \notin P$ (hay dos personas solteras sin que nada les impida casarse entre ellas)

Es posible utilizar HashMaps para guardar las preferencias, los hombres y mujeres y las parejas prohibidas, de modo tal que los accesos sean de tiempo constante. Sin embargo, se decidió simplemente emplear arreglos, por su facilidad de escritura y testeo. Los accesos siguen siendo constantes, pero en algunos casos se debe recorrer la estructura entera (como sería el caso de buscar si una es pareja prohibida con un hombre y una mujer determinados).

Las estructuras empleadas son:

- Arreglos bidimensionales para las preferencias de hombres y mujeres. Si una pareja es prohibida, no se consideran en su lista de preferencias y la lista se completa con -1 (se asume como precondición)
- Arreglos unidimensionales con nombres para hombres y mujeres, de modo tal que solamente se empleen sus índices en las demás estructuras (y además, para mejor entendimiento del algoritmo)
- Arreglo bidimensional para las parejas prohibidas. La primera posición indica al hombre y la segunda a la mujer, en cada pareja del arreglo.
- Arreglo unidimensional para indicar la pareja actual de cada mujer
- Arreglo unidimensional para indicar si un hombre tiene pareja o no

Todas tienen como longitud un tamaño fijo N. La única modificación al algoritmo original es la consideración de si el hombre y la mujer actuales forman una pareja prohibida. En ese caso, se debe

ignorar y pasar a la siguiente.

La propiedad de correctitud y estabilidad se sigue manteniendo: el algoritmo sigue proveyendo un conjunto de asignaciónes estable, incluso si existen parejas prohibidas, ya que estas son ignoradas y no se cumpliría ningún caso de inestabilidad del algoritmo original. Lo que sí se ve afectada es la propiedad de completitud: el programa puede que no termine y acabe en un ciclo infinito si es que el conjunto de parejas prohibidas es muy grande, ya que la restricción producida no posibilita una asignación. Se puede adaptar el algoritmo para verificar que no se comparen las mismas parejas repetidamente sin éxito, para al menos indicar que no existe asignación posible.

La estrategia algorítmica empleada es la de un algoritmo voraz: toma la mejor asignación posible en el momento, sin tener en cuenta que en el futuro pueda cambiar. Si bien los algoritmos voraces tienen la particularidad de que su decisión actual no cambia en el futuro (lo cual es contradictorio con el algoritmo presentado), esto es debido a que la técnica empleada por el algoritmo es la técnica de búsqueda local con mejora iterativa (no voraz), lo que permite mejorar la posible solución.

2. Problemas $\mathcal{NP}$

2.1. Ejercicio 1

2.1.1. Problema de la Coalición Sospechosa

Una pequeña empresa de tecnología realiza trabajos sensibles para lo cual mantiene un sistema de computadora de alta seguridad con el objeto de garantizar que el mismo no se utilice con algún fin ilícito. Para ello cuenta con software que registra las direcciones IP de los accesos de todos sus usuarios a través del tiempo.

Se asume que cada usuario accede al menos a una dirección IP en algún minuto dado; el software registra en un archivo log lo siguiente: para cada usuario u y cada minuto m, un valor $I(u, m)$ es igual a la dirección IP accedida, si la hubiera. Si no, se setea con el símbolo |, que significa que el usuario u no accedió durante el minuto m.

Los directivos de la empresa han detectado que ayer, el sistema fue utilizado para lanzar un ataque complejo sobre algunos sitios remotos. El ataque fue llevado a cabo a partir de acceder a t direcciones IP distintas en t minutos consecutivos: en el minuto 1, el ataque accedió a la dirección i_1; en el minuto 2, accedió a la dirección i_2; y así para la dirección i_t en el minuto t.

¿Quién pudo haber sido el responsable de llevar adelante el ataque? La empresa chequea los archivos logs y encuentra para su sorpresa que no existe un único usuario u que accedió a cada una de las direcciones IP involucradas en el tiempo t apropiado; en otras palabras, no existe un u tal que $I(u, m) = i_m$ para cada minuto m desde 1 hasta t. La pregunta es: ¿Existió una pequeña coalición de k usuarios que colectivamente podría haber llevado a cabo el ataque?

Se dice que un subconjunto S de usuarios es una coalición sospechosa si, para cada minuto m desde 1 hasta t, existe al menos un usuario $u \in S$ para el cual $I(u, m) = i_m$. (En otras palabras, cada IP fue accedido en el momento apropiado por al menos un usuario de la coalición).

El problema de Coalición Sospechosa pregunta: ¿Dado el conjunto de todos los valores $I(u, m)$,

y un número k, existe una coalición sospechosa de al menos k de tamaño?

2.1.2. Problema en $\mathcal{NP}$ y reducción

El problema de Coalición Sospechosa es un problema $\mathcal{NP}$, ya que dado un conjunto S de usuarios de tamaño k es posible chequear en tiempo polinomial si en cada minuto m de 1 a t, al menos un usuario de S accedió a la dirección IP i_m.

Para encontrar un problema $\mathcal{NP}$-completo para reducirlo al problema de Coalición Sospechosa, es importante resaltar que este se trata de un problema de cubrimiento. Se necesita explicar t accesos sospechosos, y se cuenta con un número k de usuarios para hacerlo. Es natural entonces intentar reducir los problemas Vertex Cover o Set Cover al problema de Coalición Sospechosa. Cabe recalcar que la propiedad de $\mathcal{NP}$-completitud indica que cualquier problema reduce al problema de Coalición Sospechosa, solo que algunos se relacionan mejor y es más fácil encontrar esta reducción.

En el problema de Vertex Cover, se necesita cubrir todos los arcos con solo k nodos. En Coalición Sospechosa, se necesita explicar todos los accesos con solo k usuarios. Este paralelismo entonces nos indica que es posible construir un grafo $G = (V, E)$ y un límite k, donde los nodos de G representan los usuarios y los arcos los accesos.

El grafo G cuenta con m arcos $e_1, \ldots, e_m$ con $e_j = (v_j, w_j)$. Para constuir una instancia de Coalición Sospechosa, se realiza lo siguiente: por cada nodo de G se crea un usuario, y por cada arco $e_t = (v_t, w_t)$ se crea un minuto t (m minutos en total). En el minuto t, los usuarios que participan en el arco e_t acceden a la dirección IP i_t (los demás no acceden a nada). Finalmente, el ataque consiste en accesos a las direcciones IP $i_1, i_2, \ldots, i_m$ en minutos $1, 2, \ldots, m$, respectivamente.

A partir de la construcción de la reducción, es fácil ver que los problemas son muy similares, que la construcción se realiza en tiempo polinomial (se recorre toda el grafo), y que también el problema de la Coalición Sospechosa es $\mathcal{NP}$-completo. Entonces, si el grafo G para el problema Vertex Cover tiene una solución C con tamaño k, también es solución para el problema de Coalición Sospechosa. Esto sería que, considerando un conjunto S de usuarios, y que para cada t de 1 a m, hay al menos un elemento en C que participa en el arco e_t, el usuario correspondiente en S accedió a la dirección IP i_t, y el nodo correspondiente en C es un extremo en el arco e_t, entonces S es una Coalición Sospechosa.

2.2. Ejercicio 2

2.2.1. Problema de Planificación de Conferencia

Se le ha encomendado organizar un seminario para estudiantes ingresantes que se reunirá una vez por semana durante el próximo semestre. El plan es tener la primera porción del semestre consistente de una secuencia de l conferencias invitadas por oradores externos, y la segunda porción del semestre dedicada a la realización de una secuencia de p proyectos prácticos que los estudiantes desarrollarán.

Hay n opciones para los oradores en general, y en la semana número i (para $i = 1, 2, \ldots, l$) un subconjunto de L_i de estos oradores está disponible para dar la conferencia. Por otro lado, cada proyecto requiere que los estudiantes hayan visto cierto material de base para que puedan completar el proyecto exitosamente. En particular, para cada proyecto j (para $j = 1, 2, \ldots, p$), existe un subconjunto P_j con oradores relevantes por lo que el estudiante necesita haber ido al menos una

vez, a la conferencia de uno de esos oradores en el conjunto P_j para completar el proyecto. Dados estos conjuntos, ¿se puede seleccionar exactamente un orador para cada una de las l primeras semanas del seminario, de manera de elegir oradores que estén disponibles en la semana designada, y para que, para cada proyecto j, los estudiantes hayan visto al menos uno de los oradores en el conjunto de P_j relevante? Se denomina este enunciado como el Problema de la Planificación de Conferencia. Considerar la siguiente instancia de ejemplo. Se supone que $l = 2$, y $p = 3$. Y que existen $n = 4$ oradores denotados A, B, C, D. La disponibilidad de los oradores está dada por los conjuntos $L_1 = \{A, B, C\}$ y $L_2 = \{A, D\}$. Los oradores relevantes para cada proyecto están dados por los conjuntos $P_1 = \{B, C\}$, $P_2 = \{A, B, D\}$ y $P_3 = \{C, D\}$. Para esta instancia la respuesta es SI, ya que se puede elegir al orador B en la primera semana y al orador D en la segunda; de esta manera, para cada uno de los tres proyectos, los estudiantes habrá visto a los oradores relevantes.

2.2.2. Problema en $\mathcal{NP}$ y reducción

El problema está en $\mathcal{NP}$, ya que dada una secuencia de oradores, se puede chequear en tiempo polinomial que todos los oradores están disponibles en las semanas que se programaron, y que por cada proyecto al menos uno de los oradores relevantes fue programado para esa conferencia.

Es posible construir una reducción desde 3-SAT a Planificación de Conferencia. La utilidad de realizar una reducción desde 3-SAT radica en que el problema de la Planificación de Conferencia tiene dos fases: la primera consiste en elegir un orador para cada semana, y la segunda en verificar que el orador elegido sea relevante para cada proyecto. En 3-SAT, primero se asignan las variables, luego se observa cada clausula y la asignación anterior las satisfagan.

Más precisamente, dada una instancia de 3-SAT con clausulas $C_1, \ldots, C_k$ sobre variables $x_1, \ldots, x_n$, se puede construir una instancia del problema de Planificación de Conferencia como se indica a continuación. Por cada variable x_i se crean dos oradores z_i y z'_i que corresponderán a x_i y su negación. Entonces se cuenta con n semanas de conferencias, de las cuales la semana i solo tendrá como oradores disponibles z_i y z'_i. Luego existe una secuencia de k proyectos, en la que cada proyecto j tiene P_j oradores relevantes, que contiene tres oradores correspondientes a los términos de la clausula C_j. Si existe una asignación v que satisface todas las clausulas para la instancia de 3-SAT, entonces en la semana i se eligió al orador entre z_i y z'_i que corresponde con x_i en v. En ese caso, se selecciona al menos un orador de de cada conjunto de oradores relevantes P_j. De manera similar, si se encuentra una manera de elegir oradores tal que haya uno relevante para cada proyecto, entonces se pueden iniciar las variables x_i como 1 si z_i fue elegido, y 0 si z'_i fue elegido. De esa manera al menos una de las tres variables de cada clausula C_j es iniciada de tal manera que la satisface, y entonces surge una asignación satisfacible.

Como la reducción dada es correcta y es de tiempo polinomial es posible afirmar que el problema de Planificación de Conferencia es $\mathcal{NP}$-completo.