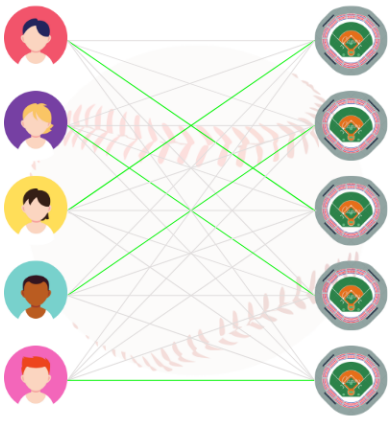
Diseño y Análisis de Algoritmos

Proyecto # 1: La Pelota



Laura Victoria Riera Pérez
Marié del Valle Reyes

Cuarto año. Ciencias de la Computación.
Facultad de Matemática y Computación, Universidad de La Habana, Cuba

4 de abril de 2023

I. Repositorio del proyecto

https://github.com/computer-science-crows/algorithms-design-and-analysis

II. Definición inicial del problema

Para un campeonato de pelota, el manager debe elegir de un conjunto de n personas, a su equipo de p jugadores, y a k espectadores especiales para que suban la moral del equipo. De cada persona i, el manager conoce el valor que aporta a la moral del equipo a_i y el valor que aporta siendo situado en la posición j, $s_{i,j}$. Determine una alineación entre jugadores en el campo y espectadores de forma que el equipo tenga la mayor cantidad de valor acumulado posible.

III. Definición en términos matemático - computacionales

1. Preliminares

Definición 1. *Sea $G = (V, E)$ un grafo. Se dice que G es bipartito si $V(G)$ es la unión de dos conjuntos independientes disjuntos. Si entre todo par de nodos de diferentes particiones existe una arista, se dice que es un grafo bipartito completo.*

Definición 2. *Un emparejamiento M es un conjunto de aristas en un grafo que son independientes, o sea, que no comparten vértices.*

Definición 3. *Dado un emparejamiento M se dice que:*

- *Si la arista $e = (v, w) \in M$, se dice que v y w son saturados por M.*
- *Un conjunto de vértices es saturado por M cuando M satura a todos los vértices del conjunto.*
- *Se dice que M es perfecto si satura a $V(G)$.*
- *M es máximo cuando no existe M_1 tal que $|M_1| > |M|$.*

Definición 4. *Sea G un grafo y M un emparejamiento del mismo. Un camino simple en G es M-alternativo si sus aristas alternan entre pertenecer y no pertenecer a M.*

Definición 5. *Sea G un grafo y M un emparejamiento del mismo. Un camino simple en G es M-aumentativo si es M-alternativo y sus extremos no son saturados por M.*

Teorema 1. *Sea G un grafo y M un emparejamiento del mismo. M es maximal si y solo si no existen caminos M-aumentativos.*

II. Problema de asignación

Dado un grafo bipartito completo ponderado $G = (V, E)$, donde $V = L \cup R$. Se asume que los vértices de los conjuntos L y R contienen n vértice cada uno, por tanto el grafo contiene n^2 aristas. Para $l \in L$ y $r \in R$, se denota el peso de la arista (l, r) como $w(l, r)$, lo cual representa ganancia de emparejar el vértice l con el vértice r.

El objetivo es encontrar el emparejamiento perfecto M^* cuyas aristas tengan el peso máximo total de todos los emparejamientos perfectos posibles.

Sea $w(M) = \sum_{(l,r)\in M} w(l, r)$ el peso total de las aristas en el emparejamiento M, se quiere encontrar el emparejamiento perfecto M^* tal que,

$$w(M*) = \max\{w(M) : M \text{ es un emparejamiento perfecto}\}$$

.

A encontrar un emparejamiento perfecto de peso máximo se le llama **problema de asignación**. Una solución del problema de asignación es un emparejamiento perfecto que maximice el costo total.

En nuestro problema, el conjunto V sería los nodos representando los jugadores y las posiciones, donde L correspondería a los jugadores y R a las posiciones. El conjunto E lo conformarían las aristas que van de cada nodo del conjunto L a cada nodo del conjunto R. El peso de cada arista sería $s[i][j] + a[i]$.

IV. Línea de pensamiento

Como primera solución al problema de asignación fue implementado un *backtrack*. Esta es una solución correcta, ya que prueba todas las combinaciones y se queda con la que más valor aporte, pero muy ineficiente, $O(n!)$. En una computadora de 32GB de RAM, intel core i7-11na generación, se puede resolver para una cantidad máxima de personas y/o posiciones de n = 11.

Se intentó resolver mediante un algoritmo *greedy* pero la forma de garantizar el óptimo en la solución pensada, era en el caso peor tan mala como backtrack.

El próximo pensamiento fue resolverlo con *programación dinámica*, pero luego de graficar el backtrack y aplicar memorización para varios casos se observó que ninguno presentaba subproblemas solapados. Además se cree que este problema no tiene subestructura óptima.

Este problema puede ser modelado como un problema de optimización lineal, y por tanto se le dió solución con el algoritmo *Simplex*, el cual es exponencial.

Investigando el estado del arte y siguiendo la idea de modelarlo mediante grafos se decidió implementar para su solución el algoritmo Húngaro [1], el cual corre en tiempo polinomial.

V. Algoritmo Húngaro

El método Húngaro es un algoritmo de optimización-combinatoria que resuelve el problema de asignación en tiempo polinomial.

En vez de trabajar con un grafo bipartito completo G, el algoritmo Húngaro trabaja con un subgrafo de G llamado **subgrafo de igualdad**. El subgrafo de igualdad depende de asignar un atributo h a cada vértice. El atributo h se llama **etiqueta** del vértice. Se dice que h es un **etiquetado de vértice factible** de G si $l.h + r.h \geq w(l,r)$ para todo $l \in L$ y $r \in R$. Un etiquetado de vértice factible siempre existe, como el **etiquetado de vértice por defecto** dado por

$$l.h = \max\{w(l,r) : r \in R\} \qquad \text{para todo } l \in R, \tag{1}$$

$$r.h = 0 \qquad \text{para todo } r \in R \tag{2}$$

Dado un etiquetado de vértice factible h, el **subgrafo de igualdad** $G_h = (V, E_h)$ de G consiste de los mismos vértice de G y el subconjunto de aristas $E_h = \{(l,r) \in E : l.h + r.h = w(l,r)\}$.

El subgrafo de igualdad puede cambiar en el tiempo y tiene la propiedad que cualquier emparejamiento perfecto en el subgrafo de igualdad es también una solución óptima del problema de asignación como se demuestra en el siguiente teorema.

Teorema 2. *[1] Sea $G = (V, E)$, donde $V = L \cup R$, un grafo bipartito completo donde cada arista $(l,r) \in E$ tiene peso $w(l,r)$. Sea h un etiquetado de vértice factible de G y G_h el subgrafo de igualdad de G. Si G_h contiene un emparejamiento perfecto M^*, entonce M^* es una solución óptima del problema de asignación G.*

Demostración. Si G_h tiene un emparejamiento perfecto M^*, entonces debido a que G_h y G tienen el mismo conjunto de vértices, M^* es también un emparejamiento perfecto en G. Debido a que cada arista de M^* pertenece a G_h y cada vértice tiene exactamente una arista incidente del emparejamiento perfecto, entonces se tiene

$$w(M*) = \sum_{(l,r)\in M*} w(l,r) \tag{3}$$

$$= \sum_{(l,r)\in M^*} (l.h + r.h) \qquad \text{(porque todas las aristas de } M^* \text{ pertenecen a } G_h) \tag{4}$$

$$= \sum_{l\in L} l.h + \sum_{r\in R} r.h \qquad \text{(porque } M^* \text{ es un emparejamiento perfecto)} \tag{5}$$

$$\tag{6}$$

Sea M un emparejamiento perfecto cualquiera de G, se tiene

$$w(M) = \sum_{(l,r)\in M} w(l,r) \tag{7}$$

$$\leq \sum_{(l,r)\in M} (l.h + r.h) \qquad \text{(porque } h \text{ es un etiquetado de vértice factible)} \tag{8}$$

$$= \sum_{l\in L} l.h + \sum_{r\in R} r.h \qquad \text{(porque } M \text{ es un emparejamiento perfecto)} \tag{9}$$

Entonces se tiene

$$w(M) \leq \sum_{l\in L} l.h + \sum_{r\in R} r.h = w(M^*), \tag{10}$$

por tanto M^* es un emparejamiento perfecto de máximo costo en G. ■

Entonces, el objetivo del algoritmo es encontrar un emparejamiento perfecto en un subgrafo de igualdad.

1. Explicación del algoritmo

El algoritmo Húngaro empieza con un etiquetado de vértice factible h que es el de defecto 1 y cualquier emparejamiento M en el subgrafo de igualdad G_h. En la resolución de este problema se utilizó un algoritmo de emparejamiento maximal greedy. Luego, el algoritmo repetidamente encuentra un **camino M-aumentativo** P en G_h utilizando una variante de Búsqueda Primero a lo Ancho (*en inglés*, **BFS**).

El algoritmo BFS empieza la búsqueda desde todos los vértices no saturados de L, los cuales al inicio se insertan en la cola Q. La condición de parada es que se descubra algún vértice no saturado de R, ya que un camino M-aumentativo es aquel que empieza en un vértice no saturado de L y termina en un vértice no saturado de R, tomando aristas no saturadas de L a R y aristas saturadas de R a L. El resultado del algoritmo es un bosque primero a lo ancho $F = (V_f, E_f)$, donde cada vértice no saturado de L es raíz de algún árbol de F.

Si la búsqueda de un camino M-aumentativo falla, se debe actualizar el etiquetado de vértice factible para adicionar a G_h al menos una arista nueva.

Una vez encontrado un camino M-aumentativo, actualiza el emparejamiento para que este sea la diferencia simétrica de M y P, incrementando así el tamaño del emparejamiento. Mientras haya algún subgrafo de igualdad que contenga un camino M- aumentativo, el tamaño del emparejamiento puede incrementar, hasta que un emparejamiento perfecto se logre.

Existen casos en los que la cola Q se vacía sin que se halla llegado a encontrar un vértice no saturado de R que conforme un camino M-aumentativo. Cuando esto ocurre, el algoritmo Húngaro actualiza el etiquetado de vértices factible h de acuerdo al siguiente lemma.

Lema 1. *[1] Sea h un etiquetado de vértice factible en el grafo bipartito completo G con el grafo de igualdad G_h, y se M un emparejamiento para G_h y F el bosque construído a partir de una Búsqueda Primero a lo Ancho (en inglés, BFS) sobre el subgrafo de igualdad G_h. Entonces, la etiqueta h',*

$$v.h' = \begin{cases} v.h - \delta & si\ v \in F_l, \\ v.h + \delta & si\ v \in F_r, \\ v.h & e.o.c \end{cases} \tag{11}$$

donde

$$\delta = min\{l.h + r.l - w(l,r) : l \in F_l, r \in F_r\} \tag{12}$$

con $F_l = L \cap V_f$ y $F_r = R \cap V_f$ son vértices del bosque F que pertenecen a L y a R, respectivamente, es una etiqueta de vértice factible para G con las siguientes propiedades:

1. *Si (u,v) es una arista de bosque F para G_h, entonces (u,v) $\in E_{h'}$.*
2. *Si (l,r) pertenece al emparejamiento M para G_h, entonces (l,r) $\in E_{h'}$.*
3. *Existen vértices $l \in F_l$ y $r \in R - F_r$ tales que (l,r) $\notin E_h$, pero (l,r) $\in E_{h'}$.*

Demostración. Primero se demuestra que h' es un etiquetado de vértice factible para G. Debido a que h es un etiquetado de vértice factible, se tiene $l.h + r.h \geq w(l,r)$ para todo $l \in L$ y $r \in R$. Para que h' no sea un etiquetado de vértice factible, entonces se necesitaría que $l.h' + r.h' < l.h + r.h$ para algún $l \in L$ y $r \in R$. La única forma en que esto pudiera ocurrir sería para algún $l \in F_l$ y $r \in R - F_r$. En esta instancia, la cantidad de decrecimiento es igual a δ, entonces $l.h' + r.h' = l.h - \delta + r.h$. Por ecuación 12, se tiene que $l.h - \delta + r.h \geq w(l,r)$ para cualquier $l \in F_l$ y $r \in R - F_r$, por tanto $l.h' + r.h' \geq w(l,r)$. Para cualquier otra arista, se tiene $l.h' + r.h' \geq l.h + r.h \geq w(l,r)$. Por tanto, h' es un etiquetado de vértice factible.

Ahora se mostrará la veracidad de las propiedades:

1. Si $l \in F_l$ y $r \in F_r$, entonces se tiene $l.h' + r.h' = l.h + r, h$ debido a que δ se adiciona a la etiqueta de l y se substrae de la etiqueta de r. entonces, si una arista pertenece a F para el grafo G_h, también pertenece a $G_{h'}$.

2. Se afirma que para el momentp en que el algoritmo Húngaro computa el nuevo etiquetado de vértice factible h', para toda arista $(l,r) \in M$, se tiene que $l \in F_l$ si y solo sí $r \in F_r$.

   Para demostrar por qué, se considera el vértice saturado r y la arista $(l,r) \in M$.

   Primero se supone que $r \in F_r$, entonces la búsqueda encuentra r y lo pone en la cola. Cuando r se remueve de la cola, l es descubierto, entonces $l \in F_l$.

   Luego se supone que $r \notin F_r$, por tanto r no se ha descubierto. Se demostrará que $l \notin F_l$. La única arista en G_h que entra l es (r,l), y dado que r no se ha descubierto, la búsqueda no ha tomado esta arista; si $l \in F_l$, no es por la arista (r,l). La única otra forma que un vértice en L puede estar en F_l es si es raíz de la búsqueda, pero solo vértices no saturados de L son raíces y l está saturado. Por tanto, $l \notin F_l$ y la afirmación se cumple.

   Se conoce que para $l \in F_l$ y $r \in F_r$ se cumple $l.h' + r.h' = l.h + r.h$. En caso contrario, cuando $l \in L - F_l$ y $r \in R - F_r$, se tiene que $l.h' = l.h$ y $r.h' = r.h$, entonces $l.h' + r.h' = l.h + r.h$. Por tanto, si la arista (l,r) está en el emparejamiento M para el grafo G_h, entonces $(l,r) \in E_{h'}$.

3. Sea (l,r) una arista que no pertenece a E_h, tal que $l \in F_l$, $r \in R - F_r$ y $\delta = l.h + r.h - w(l,r)$. Entonces, por definición de δ, existe al menos una de esas arista. Luego, se tiene

$$\begin{aligned} l.h' + r.h' &= l.h - \delta + r.h \\ &= l.h - (l.h + r.h - w(l,r)) + r.h \\ &= w(l,r) \end{aligned}$$

   y por tanto $(l,r) \in E_{h'}$.

■

II. Correctitud

III. Complejidad Temporal

La complejidad temporal del algoritmo Húngaro implementado es $O(n^4)$, donde $|V| = 2n$ y $|E| = n^2$ en el grafo original G.

VI. Generador de casos de prueba

Se generaron 3000 casos de prueba utilizando backtrack los cuales pueden ser encontrados en *json/test_cases.json*.

VII. Tester

Las soluciones implementadas fueron testeadas para todos los casos de prueba generados y pueden encontrarse en *json/tests/simplex$_s$olution.json* y *json/tests/hungarian$_s$olution.json*

Referencias

[1] Cormen, Thomas H. y otros. *Introduction to Algorithms*. The MIT Press. 4ta Edición. Cambridge, Massachusetts. 2022.