Algoritimo de optimización para el emparejamiento entre dos grupos y su aplicación en una fintech para relacionar un tarjetahabiente y un estudiante

Terry Alexander Cruz Melo
Estudiante: 20151042
Pontifica Universidad Católica del Perú
Lima ,Peru
Email: tcruz@pucp.edu.pe

Lloyd Erwin Castillo Ramos
Estudiante: 20142280
Pontifica Universidad Católica del Perú
Lima ,Peru
Email: castillo.le@pucp.edu.pe

Christian Oswaldo Garcia Vasquez
Estudiante: 20116367
Pontificia Universidad Católica del Peru
Lima ,Peru
Email: a20116367@pucp.pe

***Resumen*—Inversión, el concepto básico es poner a disposición una cantidad limitada de dinero con la finalidad de incrementar las ganancias que se genere de esta. Para tal objetivo, se necesita de un algoritmo que se especialice en la búsqueda de grupos capaces de satisfacer tal meta. El emparejamiento de grupos nos servirá para acomodar y satisfacer las necesidades que se presenten. Sea que una entidad tenga el elemento exacto, un conjunto de valores que cumplan con esa necesidad o que satisfaga un aproximado decente.**

1. Introducción

Dado que, uno de los objetivos para resolver problemas de selección de dos o más grupos con parámetros en común se encuentra en el emparejamiento de un elemento con múltiples elementos de otro grupo , se busca mediante diferentes soluciones, encontrar uno que sea, por su construcción, óptima y solucione de manera eficaz el problema. Es por ello, que se vio en la necesidad de investigar acerca del tema y cómo el matching[**6**] o emparejamiento entre dos grupos puede solucionar problemas de optimización de selección para por ejemplo,el contexto financiero del problema que se va a tocar.

Es así que, dado este problema, lo que se busca mediante el algoritmo de emparejamiento que se va a utilizar es seleccionar un elemento de un grupo con el que se va a trabajar y encontrar posibles coincidencias con varios elementos en otro grupo utilizando en su estructura conceptos básicos como la fuerza bruta pero limitada a ciertos parámetros los cuales definiremos a lo largos de este informe y estructuras de datos como colas y arreglos dinámicos.

Finalmente, el presente artículo está dividido en diferentes secciones. En primer lugar, el siguiente punto a tocar se encuentra la presentación de la formulación del problema. Luego, se dará paso al desarrollo y explicación del algoritmo de emparejamiento a utilizar. A continuación, se analizará la eficiencia del algoritmo. Posteriormente, se describirá la aplicación en la fintech. Y por último, se realizará una conclusión con respecto al tema tratado en el presente artículo.

2. Definición del problema

Sea el primero grupo el cual denotaremos como A , y el segundo B, y seguidamente lo definiremos , como

$$A = \{\, x \mid x \in \mathbb{Q} \wedge x \geq 0 \,\} \tag{1}$$

$$B = \{\, y \mid y \in \mathbb{Q} \wedge y \geq 0 \,\} \tag{2}$$

seguidamente definiremos un épsilon ε , el cual representara la máxima diferencia que podrá haber entre los atributos de estos dos grupos , y α que representara la cantidad máxima de apariciones del grupo B. El problema que queremos afrontar es relacionar un atributo del grupo A con uno o varios del grupo B lo más eficiente posible:

Dado un conjunto de n ítems enumerados del 1 al n , tal que.

$$\textbf{Maximizar} \sum_{y=1}^{n} y_i \text{ sujeto a un x[}\textbf{5}\text{]}$$

este problema resulta un poco tedioso ya que la única manera de relacionarlos mediante un atributo es la fuerza bruta lo que complica la resolución de una manera eficiente, sin embargo es probable que con las variables presentadas previamente un épsilon ε y α se pueda mejorar la eficiencia de cierta forma

3. Emparejamiento

El objetivo de nuestro algoritmo es encontrar el mejor valor o la mejor suma de valores del grupo B para un valor del grupo A, para lograr esto nos valdremos de los parámetros α y ε definidos previamente. El primer paso será almacenar al grupo A para luego emparejarlo con los valores del grupo B, sin embargo este elemento podría no ser el primero en ser evaluado, esto se debe a que algunos elementos del grupo A estarían esperando a que se cumpliera alguna de las tres condiciones que llamaremos prioridades las cuales definiremos posteriormente.

3.1. Definición de TAD

Para nuestro algoritmo nos valdremos principalmente de dos estructuras de datos: double ended queue el cual abreviaremos como $deque$ y arreglos dinámicos, usaremos el deque para almacenar al grupo A (da) , y al grupo de espera de A (dp), y usaremos arreglos dinámicos para almacenar los elementos del grupo B (db) y un grupo a eliminar del mismo grupo (eb).

3.1.1. Dequeue. Utilizaremos el dequeue para poder utilizar operaciones tales como poder encolar tanto por delante como por detrás elementos a dicha cola que pueden ser accedidos de manera directa en tiempo constante. Púes de esta manera, podremos procesar en la cola elementos del grupo A y mediante las condiciones explicadas posteriormente si no se cumple ninguna de ellas, este elemento se añadirá a una cola de espera para poder ser procesado un nuevo elemento de la cola principal.

Algorithm 1: Incersión del grupo de espera al grupo A

```
if db ≠ ∅ ∧ (da ≠ ∅ ∨ dp ≠ ∅) then
    if dp ≠ ∅ then
        while dp ≠ ∅ do
            poner adelante al ultimo de dp en da
            sacar por atras un elemento de dp
        end
    end
end
```

3.1.2. Arreglo dinámicos. Para este caso, el uso de arreglos dinámicos se encuentra ligado a almacenar en ellos, para un caso las cantidades las cuales van a ser evaluadas y elegidas que cumplan con las condiciones establecidas. Y por otro lado, almacenar las cantidades seleccionadas para luego su próxima eliminación cuando se encuentre en el caso donde se encuentren sumas máximas no exactas al valor pedido a encontrar del grupo B en el grupo A.

3.2. Prioridades

3.2.1. Primera prioridad. El objetivo de la primera prioridad es buscar la cantidad idéntica de manera lineal, es decir buscaremos la primera coincidencia del elemento de un elemento a en el conjunto de valores del arreglo db .Previo a esto tendremos que haber extraído el elemento a del deque da.

Algorithm 2: Primera prioridad

```
encontrado=0
for y = 0 hasta |B| do
    if y-ésimo elemento de B es igual a la cantidad
     idéntica X then
        encontrado=1
        imprimir(.Encontrado")
        asignar al y-ésimo de elemento B el último
         elemento de B
        eliminar(último elemento de B)
        break
    end
end
```

3.2.2. Segunda prioridad. El segundo objetivo es también buscar la cantidad idéntica de un valor a', sin embargo al no haberse cumplido con la prioridad uno sabremos que no será factible encontrar un elemento que coincida con este por lo tanto conseguiremos este objetivo sumando cierta de cantidad alfa de elementos del dequeue da que sea igual al elemento a'. para esto usaremos fuerza bruta buscando todas las posibilidades entre B que cumplan con esta condición. Si se encuentra un valor para el elemento a' , se añaden los elementos del vb al arreglo eb.

Algorithm 3: Segunda prioridad

```
cantM = α
for x = 0 hasta 2^|B| do
    sum = 0
    cantidad = 0
    for y = 0 hasta |B| do
        if y-ésimo bit es el conjunto en x then
            sum = sum + B[y]
        end
    end
    if
     sum = x∧cantidad ≤ α∧cantidad < cantM
     then
        prender bandera
        seleccionar el conjunto i-ésimo
        cantM = cantidad
    end
end
```

3.2.3. Tercera prioridad. Al no haberse cumplido ninguna de las prioridades anteriores mencionadas, el objetivo de será buscar la mejor opción para a' , tal que:

$$abs(\textbf{Maximizar} \sum_{y=1}^{n} y_i \text{ - } x) \leq \epsilon \wedge n \leq \alpha$$

Para este caso también utilizaremos fuerza bruta buscando todas las posibilidades entre B que cumplan con esta condición. Igualmente que en la prioridad dos, si se encuentra

un valor para el elemento x , se añaden los elementos del db al arreglo eb.

Algorithm 4: Tercera prioridad

```
epsMin = ϵ
for x = 0 hasta 2^|B| do
    sum = 0
    cantidad = 0
    for y = 0 hasta |B| do
        if y-ésimo bit es el conjunto en x then
            sum = sum + B[y]
        end
    end
    if abs(sum − x) ≤ ϵ ∧ abs(sum − x) ≤
     eps ∧ cantidad < cantM then
        prender bandera
        seleccionar el conjunto i-ésimo
        epsMin = abs(sum-x)
    end
end
```

4. Eliminación de elementos

Luego de haberse cumplido la prioridad , pasaremos a almacenar cada elemento seleccionado en el conjunto de elementos a eliminar, para luego recorrer el conjunto a eliminar y buscar en el conjunto B el elemento y eliminarlo.

Algorithm 5: Eliminación

```
if encontrado = cierto then
    for x = 0 hasta |B| do
        if x-esimo bit esta el conjunto seleccionado
         then
            agregar por atras B[x] a eb
        end
    end
    for z = 0 hasta |eb| do
        eliminar el elemento eb[z] de B
    end
end
```

5. Análisis de eficiencia

En este apartado nos centraremos en la complejidad[**2**] espacial, es decir, en función a la cantidad de datos de entrada. El algoritmo sigue un total de tres formas de abordar el emparejamiento, por tanto, es de importancia explicar cada una de sus complejidades, además de que cada forma es excluyente una de otra, es decir solo una de las opciones se puede dar y pasa a evaluar al siguiente elemento a evaluar:

Primera prioridad: Bucle for, tamaño cantidad de estudiantes $\leq$ N . El algoritmo recorre una cantidad fija de datos en búsqueda de un monto exacto, un recorrido lineal. Una vez encontrada la coincidencia, pasa a evaluar otro elemento de la cola de personas tarjetahabiente.

Complejidad: $O(N)$

Segunda prioridad: Doble bucle for : 1er bucle, tamaño (2 a la cantidad de estudiantes = 2^N

2do bucle, tamaño (cantidad de estudiantes $\leq$ N) En esta prioridad, el algoritmo se centra en buscar y sumar una cantidad de montos que presentan los alumnos y si esta cantidad es exacta al monto exacto que se pide, termina aquella iteración y empieza un nuevo elemento de la cola de personas tarjetahabiente. Realiza un recorrido lineal y a la vez pasa a sumar un monto en una sumatorio, resultando en una complejidad exponencial.

Complejidad: $O(2^N \cdot N)$

Tercera prioridad: Doble bucle for : 1er bucle, tamaño (2 a la cantidad de estudiantes =2^N

2do bucle, tamaño (cantidad de estudiantes $\leq$ N) A diferencia del anterior prioridad, este se centra en buscar y sumar el monto más aproximado posible al monto meta. Compara la diferencia entre esta sudatoria y el monto meta y lo almacena para comparar posibles resultados. Habiendo recorrido todos los estudiantes y teniendo el monto aproximado se procede a presentarlo como opción. Complejidad exponencial.

Complejidad: $O(2^N \cdot N)$

En resumen, el algoritmo ,en general, sigue un complejo computacional de orden $O(2^N \cdot N)$ debido al uso de conceptos como bitmasking y fuerza bruta. Esto significaría que hace un uso grande de recursos, sin embargo; en cada prioridad, el algoritmo libera memoria por cada iteración que pida calcular el emparejamiento. Por lo tanto el consumo de recursos puede verse aliviado en cada iteración que desarrolle.

Finalmente, la eficiencia del algoritmo depende netamente de la cantidad de estudiantes y de habientes disponibles para el algoritmo.

6. Aplicación en la fintech

Para el caso del la aplicación de la fintech(Financial Technology),se busca automatizar,para este problema, la selección de una mejor opción de selección de un elemento de un grupo con uno o mas elementos de otro.Para este caso en específico, se denota como grupo A a aquellas personas que buscan encontrar ofertas por parte del grupo B, que definido anteriormente, los primeros estarán en una cola esperando a ser atendidos, en caso no se pueda satisfacer a dicha persona de ese grupo, esta entra en espera para atender a un siguiente elemento del grupo.Los segundos,son procesados y eliminados si cumplen con las condiciones antes dadas para ser parte de la solución de selección del grupo de personas que solicitan un crédito con un monto a satisfacer.

6.1. Financial Technology (*fintech*)

Anteriormente, se definió la aplicación del algoritmo en la fintech, sin embargo para poder entender dicha aplicación se definirá a continuación el concepto de dicho término y de la StartUp WePayU.

En primer lugar, fintech o tecnología financiera es un término que refiere, según la web del banco BBVA como:"las empresas que prestan servicios financieros a través de la tecnología"[**1**].Por lo que, podemos inferir que esta industria aplica en sí, tecnologías actuales a rubros financieros[**3**] para poder resolver de manera más eficiente por ejemplo pagos o transacciones de dinero por bancos u otro tipo de transacciones mediante aplicativos o servicios web.

6.2. WePayU

Como se mencionó al comienzo de este informe aplicaremos este algoritmo para un caso de la vida real, el cual es la empresa WePayU[**4**], específicamente trabajaremos con el problema el cual presentan actualmente. Antes de explicar el problema, explicaremos de manera breve qué es lo que hace WePayU, e iremos específicamente a lo que nos interesa. WePayU conecta a personas que necesitan dinero en efectivo con estudiantes universitarios que necesitan pagar su boleta o pensión académica, la propuesta de valor que ellos ofrecen es brindarles un descuento adicional a las pensiones y así el estudiante termina pagando menos de lo que pagaría si lo haría de una manera convencional, este descuento lo terminaría pagando el retirador, ya que este mismo esta interesado en retirar dinero en efectivo de su tarjeta, pagando un descuento mucho menor al que paga en el banco.

Figura 1. Esquema de como funciona WePayU

6.3. Aplicación del algoritmo

Ya teniendo un contexto del problema al cual queremos aplicar el algoritmo y como también ya habiendo dado a conocer el algoritmo, ya podríamos relacionar ambos.

En este contexto podremos definir

$$A = \text{Retiradores o tarjetahabientes}$$

$$B = \text{Estudiantes}$$

$\varepsilon =$ máxima diferencia entre un tarjeta habiente objetivo y la suma de los montos de la boleta de los estudiantes

$\alpha =$ máxima cantidad de boletas o estudiantes que podrá pagar un tarjetahabiente o retirador.

Ya definiendo estas variables podríamos aplicar el algoritmo al problema que presenta actualmente la empresa el cual es poder relacionar y/o emparejar de la mejor manera a los estudiantes con los retiradores, apesar que no me hemos tenido el tiempo para mostrar resultados estadísticos de mejora comparando nuestro algoritmo con la manera de como están resolviendo el problema actualmente. Es imposible negar que el emparejamiento de manera computacional es mucho mas eficiente y eficaz con respecto a hacerlo de manera manual, esto se nota cuando se incrementa la cantidad de elementos tanto en el conjunto B como en el A, en este caso el número de estudiantes y el número de retiradores.

7. Conclusiones

Finalizando con el presente artículo,el algoritmo propuesto de emparejamiento entre dos grupos busca asociar un elemento con uno o mas de otro grupo satisfaciendo los parámetros distintivos dados durante el desarrollo del problema.Además, durante la construcción del algoritmo, se utilizaron conceptos de fuerza bruta para encontrar la solución mas óptima dado un tarjetahabiente y un grupo de estudiantes pero se utilizó estructuras adicionales para la resolución del problema así como manejo de bitmasking para la selección de posibles soluciones y el almacenamiento de los grupos en arreglos dinámicos y una double ended queue.

Por otro lado, con respecto a la aplicación en la fintech, podemos pensar que los parámetros utilizados como el épsilon, para la diferencia máxima entre los montos de un tarjetahabiente y uno o mas estudiantes,el alfa,la cantidad máxima de estudiantes para un tarjetahabiente,pueden no ser suficientes para poder reducir el universo de posibilidades y dar una óptima respuesta.Por lo tanto, como futura optimización del mismo algoritmo se puede proponer el uso de otro discriminante como por ejemplo un número máximo de operaciones hecha por un tarjetahabiente entre otros parámetros.

Agradecimientos

Mediante el presente trabajo, agradecemos al Dr. y profesor Ivan Sipirán por enseñarnos a aplicar y entender el uso de estrategias algorítmicas para solucionar problemas aplicados en situaciones reales.

Referencias

[1] "Qué es el 'fintech' y cómo las 'startups' quieren innovar en los servicios financieros", BBVA NOTICIAS, 2020. [Online]. Available: https://www.bbva.com/es/que-es-el-fintech/. [Accessed: 14- Jul- 2020].

[2] Webdiis.unizar.es. 2020. [online] Available at: <http://webdiis.unizar.es/asignaturas/AB/material/varios/eficiencia.pdf>[Accessed 16 July 2020].

[3] Economipedia. 2020. Inversión | Economipedia. [online] Available at: <https://economipedia.com/definiciones/inversion.html>[Accessed 16 July 2020].

[4] Wepayu.pe. 2020. Wepayu. [online] Available at: <https://www.wepayu.pe/nosotros>[Accessed 16 July 2020].

[5] "Knapsack problem", En.wikipedia.org, 2020. [Online]. Available: <https://en.wikipedia.org/wiki/Knapsack_problem#:~:text=The%20knapsack%20problem%20is%20a,is%20as%20large%20as%20possible.[Accessed:16-Jul-2020].>

[6] Anany V. Levitin. 2002. Introduction to the Design and Analysis of Algorithms. Addison-Wesley Longman Publishing Co., Inc., USA.