




Teoría de Algoritmos I 75.29

Trabajo Práctico N° 2

Grupo Rosita Forever

Apellido y Nombre	Padrón	Correo electrónico
Jamilis, Netanel David	99093	njamilis@fi.uba.ar
Del Torto, Agustín	98867	adeltorto@fi.uba.ar
Daverede, Agustín	98540	agusdaverede@yahoo.com.ar
Betz, Joaquín	104348	

Github: 

<https://github.com/netaneldj/7529-TDA/TP2>

Indice

1	Introducción	2
1.1	Objetivos	2
1.2	Resumen	2
2	El productor agropecuario	3
2.1	Solución	3
2.2	Subproblema	3
2.3	Relación de recurrencia	3
2.4	Pseudocódigo	4
2.5	Complejidad	5
2.6	Programa	5
3	La explotación minera	6
3.1	Solución	6
3.2	Pseudocódigo	7
3.3	Complejidad	8
3.4	Ejemplo	8
4	Conclusión	12
4.1	El productor agropecuario	12
4.2	La explotación minera	12
5	Bibliografía y Referencias	13

1 Introducción

1.1 Objetivos

Los objetivos del presente trabajo son los siguientes:

- Resolver un problema aplicando Programación Dinámica
- Aplicar los conceptos aprendidos en clase en un problema de Redes Neuronales
- Preparar un informe técnico

En las siguientes secciones se verá plasmado el cumplimiento de los objetivos.

1.2 Resumen

El presente trabajo se centrará en la presentación y consecuente resolución de los problemas planteados.

En primer lugar, se tratará el problema del productor agropecuario, el cual se resolverá aplicando la técnica de programación dinámica. Se hará un exhaustivo análisis de la solución propuesta.

Luego, se repetirá el mismo análisis para el problema de la explotación minera, donde se aplicarán redes neuronales.

En último lugar, se analizarán los resultados obtenidos, donde se tendrán en cuenta los conocimientos adquiridos en el desarrollo del trabajo y la bibliografía consultada para la realización del presente informe.

2 El productor agropecuario

Un productor agropecuario quiere maximizar su producción para su campo. Puede elegir entre n productos. Puede elegir 1 por trimestre.

Por otro lado, existe un estudio que indica que tipo de cultivos no deben ser consecutivos para evitar que se agote el suelo. En primer lugar no se puede repetir cultivos en trimestres consecutivos, pero además hay una restricción entre ciertos productos contiguos. Son recomendaciones de tipo: Luego del producto A no se puede cultivar el producto B.

2.1 Solución

Nuestra solución consiste en ir construyendo dinámicamente la lista de cultivos óptimos trimestre a trimestre. Partimos de los posibles cultivos del primer trimestre y en cada paso incorporamos para cada cultivo el óptimo de los nuevos cultivos.

Al llegar al ultimo trimestre solamente debemos elegir la lista de cultivos de mayor ganancia.

2.2 Subproblema

El subproblema en nuestro planteo es la elección de los cultivos óptimos para cada lista de cultivos en cada trimestre.

2.3 Relación de recurrencia

$$\begin{cases} \text{Si } i \text{ es un cultivo del primer trimestre:} & \text{Ganancia_acumulada}(i) = \text{Ganancia}(i) \\ \text{Si } i \text{ pertenece a un trimestre } > 1: & \text{Ganancia_acumulada}(i) = \text{Ganancia}(i) + \max(\text{ganancia}(\text{anteriores_permitidos}(i))) \end{cases}$$

El arreglo *anteriores_permitidos* contiene la ganancia acumulada de los cultivos del anterior trimestre que el plantarlos antes no imposibilite plantar el cultivo i .

2.4 Pseudocodigo

def productor():

 cultivosXtrimestre = separar_cultivos_por_trimestres()

 restriccionesXcultivo = armar_diccionario_restricciones()

 plantaciones = crear_diccionario()

 inicializar_plantaciones_primer_trimestre()

 for trimestre in cultivosXtrimestre:

 cultivos_del_trimestre = cultivosXtrimestre[trimestre]

 for cultivo in cultivos_del_trimestre:

 ganancia_max_acumulada = calcular_maximo_entre_cultivos_anteriores_posibles()

 cultivos_optimos = lista_cultivo_optimo_trimestre_anterior + cultivo

 agregar_ganancia_y_cultivos_optimos_a_plantaciones()

 cultivos_por_trimestre=obtener_cultivos_optimos_cultivo_mayor_ganancia_ultimo_trimestre()

 return cultivos_por_trimestre

2.5 Complejidad

def productor():

 cultivosXtrimestre = separar_cultivos_por_trimestres() $O(K)$

 restriccionesXcultivo = armar_diccionario_restricciones() $O(L)$

 plantaciones = crear_diccionario() $O(1)$

 inicializar_plantaciones_primer_trimestre() $O(M)$

 for trimestre in cultivosXtrimestre:

 cultivos_del_trimestre = cultivosXtrimestre[trimestre] $O(1)$

 for cultivo in cultivos_del_trimestre:

 ganancia_max_acumulada = calcular_maximo_entre_cultivos_anteriores_posibles()
 cultivos_optimos = lista_cultivo_optimo_trimestre_anterior + cultivo

 agregar_ganancia_y_cultivos_optimos_a_plantaciones() $O(1)$

 cultivos_por_trimestre = obtener_cultivos_optimos_cultivo_mayor_ganancia_ultimo_trimestre() $O(M)$

 return cultivos_por_trimestre

$O(M*M*N)$

$O(M*M*N)$

K: es la cantidad de líneas del archivo que contiene la información de los cultivos.

L: es la cantidad de líneas del archivo que contiene la información de las restricciones.

M: es la cantidad de cultivos máximos que hay por trimestre.

N: es la cantidad de trimestres a evaluar.

K está acotada por $N*M$ es decir que todos los trimestres contengan exactamente M cultivos. Asimismo L está acotada por $M*(M-1)$ siendo que todos los cultivos tengan una restricción con los otros.

De esta forma no afectan a la complejidad de nuestro algoritmo.

2.6 Programa

Para poder ejecutar el archivo main.py se debe abrir la consola en el mismo directorio que el código fuente e ingresar:

```
$ python3 main.py cultivos.txt restricciones.txt
```

Donde *cultivos.txt* es el archivo de cultivos y *restricciones.txt* es el archivo con las restricciones para la rotación de cultivos. Los formatos de ambos archivos fueron detallados en el enunciado.

3 La explotación minera

Una compañía minera requiere que le ayudemos a analizar su nueva explotación. Ha realizado el estudio de suelos de diferentes vetas y porciones del subsuelo. Con estos datos ha construido una regionalización del mismo. Cada región cuenta con un costo de procesamiento y una ganancia por extracción de materiales preciosos. En algunos casos el costo supera al beneficio. Al ser un procesamiento en profundidad, ciertas regiones requieren previamente procesar otras para acceder a ellas.

La compañía nos solicita que le ayudemos a maximizar su ganancia, determinando cuales son las regiones que tiene que trabajar.

3.1 Solución

Para modelar el problema, representaremos cada región como nodos. Cada nodo estará asociado tanto con las regiones que deberían ser explotadas previamente como con las que habilitaría la explotación de la región que este representa. Así, obtendremos un esquema idéntico al problema de "selección de proyectos" visto en clase.

A cada arista de esta red se le asigna un peso equivalente a la **cota de ganancia** $C + 1$ (suma de ganancias netas positivas más uno) y sentido hacia los nodos pre-requeridos.

Agregaremos entonces, para terminar de adaptar el modelo, un par de nodos sumidero y fuente, de cada lado del grafo. La fuente antecede a los que tengan ganancia neta positiva, y el sumidero sucede a las regiones con ganancia neta negativa. Cada arista de estos dos extremos que agregamos tendrá un peso o capacidad equivalente al módulo de la ganancia neta de su región asociada. Definimos **ganancia neta** como la diferencia entre la ganancia que ofrece una región y su costo.

A partir de este modelo, como se demostró en clase, el problema se reduce a obtener el corte mínimo del grafo obtenido. Para esto se implementa el **algoritmo de Ford-Fulkerson**, ya demostrado óptimo. Una vez obtenido este corte mínimo, podremos obtener el valor de la **ganancia máxima posible** ($G_{max} = C - CorteMinimo$) y las **regiones a explotar para conseguirla** (representadas por los nodos que quedan del lado de la fuente)

3.2 Pseudocódigo

Lllamar G al grafo que contiene a cada región y a cada región derivada
Lllamar $v_{\{i\}}$ a cada nodo que represente una región del grafo del problema
Lllamar $Gn_{\{i\}}$ a la ganancia neta de la región i
Lllamar $e_{\{i\}}$ a cada arista del grafo del problema, con peso $Gn_{\{i\}}$

Lllamar C a la cota de ganancia neta (suma de ganancias netas positivas)
Lllamar s al nodo fuente y t al nodo sumidero
Lllamar G_{\max} a la ganancia máxima

Construir G , utilizando las regiones y los nodos s y t .
Los nodos con $Gn \geq 0$ deben suceder a s y los $Gn < 0$ anteceder a t .

Asignar a cada $e_{\{i\}}$ que no conecte con s o t un peso equivalente a $C + 1$
Asignar a las restantes un peso equivalente a $|Gn_{\{i\}}|$.

Proceso FordFulkerson (G, C):

```
    corteMinimo = 0
    Gr = copia de G
    Mientras existe un camino de aumento p en Gr
        bn(p) ::= capacidad mínima del camino p
        Para cada arista  $u \rightarrow v$  en p
            disminuir la capacidad de  $u \rightarrow v$  en bn(p)
            incrementar la capacidad de  $v \rightarrow u$  en bn(p)
        fin para
        corteMinimo += bn(p)
    fin mientras
```

En Gr, cada nodo alcanzable desde s es una región
a explotar (conjunto del corte mínimo de G).
 $G_{\max} = C - \text{corteMinimo}$

Sobre los caminos de aumento:

Buscar un camino de aumento equivale a buscar un camino $s \rightarrow t$
(a través de aristas con capacidad no nula) usando DFS.

3.3 Complejidad

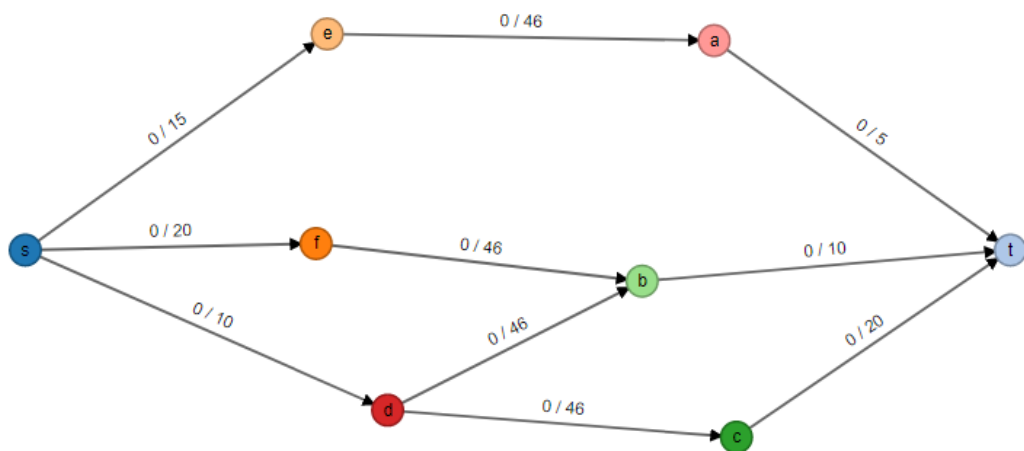
La complejidad temporal del algoritmo está dada por dos factores. La primera es la construcción del grafo que empleará el algoritmo de Ford-Fulkerson para resolver el problema de maximizar la ganancia. Sea E la cantidad de aristas y sea V la cantidad de vertices. Luego, el coste de la construcción del grafo de $\mathcal{O}(E * V)$, siendo esta también su complejidad temporal. Por otro lado, el algoritmo de Ford-Fulkerson, tiene una complejidad de $\mathcal{O}(V * q)$, donde q es la cantidad de pasadas necesarias para hallar el corte mínimo y V la cantidad de aristas. Se estima que el algoritmo tiene una probabilidad de requerir n iteraciones de 1 en 4^n .

3.4 Ejemplo

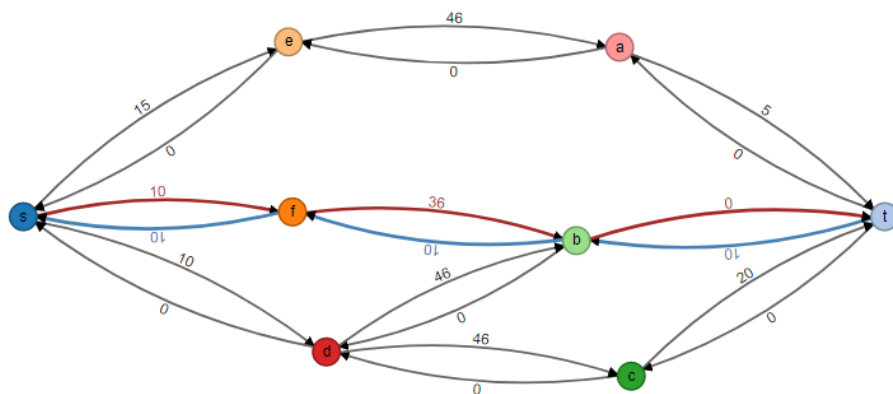
Supongamos que tenemos el siguiente caso:

Región	Costo	Ganancia	Ganancia Neta	Prerequisitos
a	10	5	-5	-
b	15	5	-10	-
c	25	5	-20	-
d	5	15	10	b, c
e	5	20	15	a
f	10	30	20	b

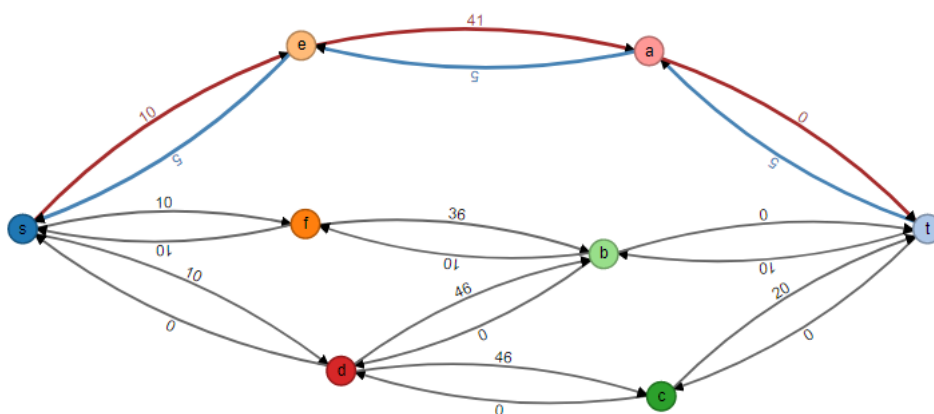
Las unidades son arbitrarias. Pueden representar miles o millones de pesos, por ejemplo. La cota de ganancia es 45 (10 + 15 + 20 por d, e y f). El grafo G entonces será:



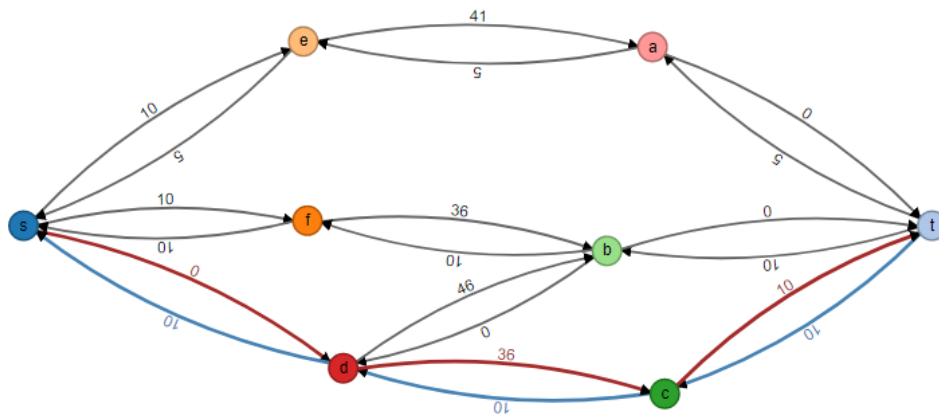
Buscamos un primer camino de aumento. Este bien puede ser s-f-b-t. El cuello de botella en este camino es 10. Actualizamos las aristas de ese camino, disminuyendo la capacidad en 10 para el sentido original (color rojo) e incrementándola en 10 para el sentido inverso (color azul). Como en principio no tendremos aristas en los dos sentidos, las agregamos para todo el grafo.



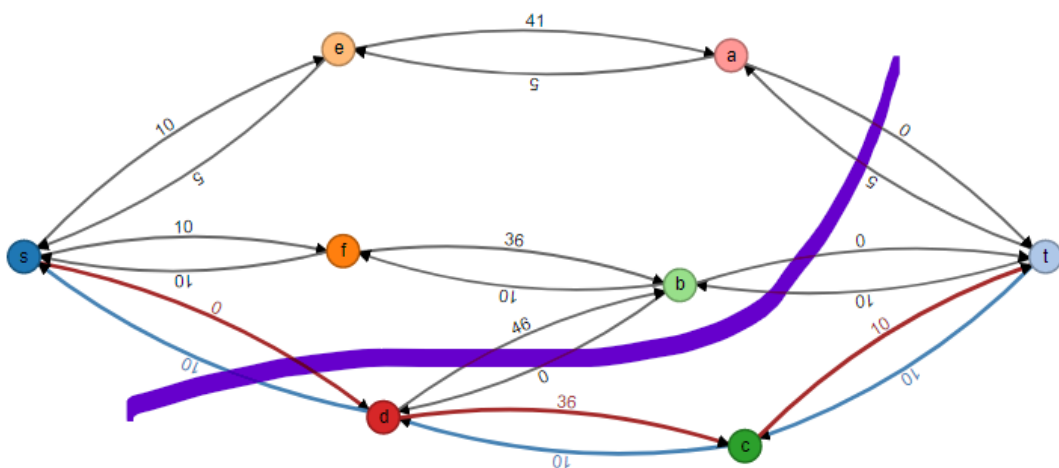
El siguiente camino que tomamos es s-e-a-t. El valor mínimo acá es 5. Actualizamos las aristas.



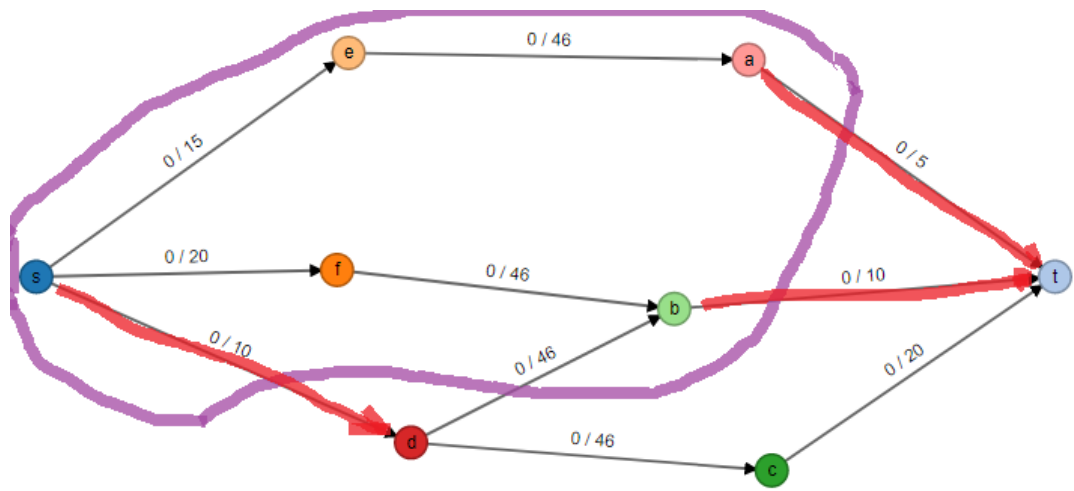
Ahora tomamos el camino s-d-c-t, y su menor capacidad es 10.



Notar que la arista $s \rightarrow d$ quedó sin capacidad, por lo que no es más accesible desde s . Tampoco tenemos otro camino posible desde s hacia t , por lo tanto el algoritmo finalizó. Observando los nodos accesibles desde s , indicamos el corte mínimo con un trazo violeta.



Las regiones a explotar son las que se encuentran del lado s del corte: **a**, **b**, **f** y **e**. Trazando el mismo corte en el grafo original, observamos que las capacidades de las aristas que van desde el lado s del corte hacia el lado t suman 25. Para conocer qué ganancia producirá esta explotación, queda restarle este valor a C ; **la ganancia máxima será de 20**.



4 Conclusión

Habiendo culminado el trabajo podemos efectuar las siguientes conclusiones:

4.1 El productor agropecuario

- El problema se simplifica al ir construyendo varias soluciones en paralelo de forma dinámica.
- En cada trimestre se debe incorporar el cultivo óptimo de acuerdo al cultivo anterior.
- En cada trimestre cada una de nuestras soluciones produce M soluciones, una por cada posible cultivo. Estas son evaluadas y nos quedamos con la permitida de mayor ganancia.

4.2 La explotación minera

- El problema se puede modelar como una red de flujos.
- Se puede solucionar mediante el teorema de corte mínimo-máximo flujo.
- La solución es exclusiva para números enteros, por lo que hay que adaptar los montos de costo y ganancia de cada región de manera acorde.

5 Bibliografía y Referencias

- J. Kleinberg, E. Tardos, Algorithm Design
- T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms
- Clase de Redes de flujo, V. Podberezski.
- Max-flow min-cut theorem, Wikipedia