Sección de Matemáticas
Universidad de La Laguna

Sven Bode Divassón

Problema de asignación y variantes

Assignment problem and variants

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2023

Dirigido por
Carlos González Martín

Carlos González Martín
Departamento de Matemáticas,
Estadística e Investigación
Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

A mi tutor, Carlos González Martín, por su tiempo, dedicación y asesoramiento que han permitido que el trabajo saliese adelante.

A mi familia, que siempre me han apoyado en cualquier circunstancia. Me han ayudado en las dificultades y me han acompañado en las alegrías.

A mi madre, en especial, quien fue mi máximo apoyo y siempre me animó a cursar este grado, y sé que se sentiría muy orgullosa de mi.

Sven Bode Divassón
La Laguna, 10 de julio de 2023

Resumen · Abstract

Resumen

En este trabajo estudiamos el problema de asignación, uno de los exponentes clásicos de los problemas de optimización. Hacemos un recorrido por los conceptos y propiedades que posibilitan el desarrollo de procedimientos para su resolución, en la necesaria cercanía de la Teoría de Grafos y la Programación Lineal. Entre la ingente cantidad de métodos desarrollados para resolver el problema de asignación lineal clásico, merecen nuestra atención tres algoritmos: Húngaro, de Subasta y de Mack. En todos los casos, una vez detallados sus pasos, se acompañan con ejemplos de aplicación. Después, introducimos brevemente algunas variantes del problema de asignación y su formulación matemática, así como sus aplicaciones. Finalmente, aportamos soporte computacional para resolver el problema de asignación lineal con la ayuda de diversos programas informáticos.

Palabras clave: *Problema de asignación – Asignaciones – Grafo bipartito – Politopo de asignación – Programación lineal – Algoritmos – Variantes*

Abstract

In this work we study the assignment problem, one of the classical exponents of optimization problems. We take a tour of the concepts and properties that enable the development of procedures for their resolution, in the necessary proximity to Graph Theory and Linear Programming. Among the huge number of methods developed to solve the linear sum assignment problem, three algorithms deserve our attention: Hungarian, Auction and Mack. In all cases, once the steps are detailed, they are accompanied by application examples. Afterwards, we briefly introduce some variants of the assignment problem and its mathematical formulation, as well as its applications. Finally, we provide computational support to solve the linear sum assignment problem with the help of various computer programs.

Keywords: *Assignment problem – Assignments – Bipartite graph – Assignment polytope – Linear programming – Algorithms – Variants*

Contenido

Introducción

Intuitivamente, podemos describir el problema de asignación como el problema de encontrar parejas, con, exactamente, un elemento de cada uno de dos conjuntos de igual tamaño, cumpliendo criterios como minimizar los costos o maximizar la eficiencia. Este problema puede verse en muchas aplicaciones, como puede ser la programación de horarios, la asignación de recursos a proyectos o la distribución de trabajos en empresas, entre otros, como veremos en los distintos enfoques planteados en este trabajo.

El objetivo del capítulo 1 es introducir los conceptos básicos del problema de asignación. En primer lugar, veremos el planteamiento del problema mediante la matriz de costos asociada y las restricciones correspondientes, acompañado de algunos ejemplos para entender su importancia en la actualidad y un breve resumen sobre los aspectos históricos del problema. Después, en la sección 1.3 vamos a enunciar los principales teoremas y resultados que nos ayudarán a entender la formalización del problema y sentarán las bases de los algoritmos de resolución. Veremos el trasfondo teórico que sustenta al problema de asignación a través de las asignaciones, los grafos bipartitos y el politopo de asignación. En la sección 1.4 presentamos el modelo matemático del problema de asignación lineal y su correspondiente problema dual, ya que trabajaremos con ambos en el siguiente capítulo

En el capítulo 2, introduciremos la clasificación de los algoritmos de resolución para el problema de asignación lineal y, a continuación, investigaremos en detalle tres de estos algoritmos justificando sus metodologías y aplicándolos a sencillos ejemplos. El primero, el famoso método húngaro [7] de H. Kuhn, fue el primer algoritmo polinomial en resolver el problema de asignación lineal. El segundo es el algoritmo de subasta [3], un algoritmo con alta eficiencia en la práctica y aplicable incluso a algunas de las variantes del problema de asignación. El último, es el algoritmo de Mack [9], un algoritmo poco mencionado en la literatura, pero muy fácil e intuitivo de entender y aplicar, adecuado para la enseñanza de pregrado.

En el capítulo 3, veremos algunas de las muchas variantes del problema de asignación, así como su formulación, sus aplicaciones más importantes e indicaremos si tienen o no algoritmos de resolución eficientes y cuáles son.

En el capítulo 4, aprovecharemos algunos programas informáticos para resolver el problema de asignación lineal. Presentamos ayuda computacional para poder resolver ejemplos planteados con anterioridad o ejemplos con una matriz de costos de grandes dimensiones que son muy laboriosos de realizar a “mano”.

1

Problema de asignación

En este capítulo presentaremos el problema de asignación, su formulación matemática y ejemplos. Resumiremos los aspectos históricos y hablaremos de la importancia actual del problema. También, abordaremos el estudio matemático que contiene conceptos y propiedades en los que, posteriormente, se basará la metodología de resolución.

1.1. Planteamiento

El problema de asignación trata de asignar un número de orígenes, por ejemplo trabajadores, a un número de destinos, por ejemplo tareas, donde son necesarias las siguientes condiciones:

1. Se tiene el mismo número de orígenes que de destinos: n.
2. Cualquier origen O_i puede ser asignado a cualquier destino D_j.
3. Cada asignación entre el origen O_i y el destino D_j tiene asociado un costo c_{ij} que puede variar dependiendo del origen y del destino.
4. Se requiere asignar todos los orígenes y todos los destinos.
5. Se debe asignar un único origen a cada destino.
6. Se debe asignar una único destino a cada origen.

Podemos resumir lo descrito previamente introduciendo la tabla:

	D_1	D_2	$\dots$	D_n
O_1	c_{11}	c_{12}	$\dots$	c_{1n}
O_2	c_{21}	c_{22}	$\dots$	c_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
O_n	c_{n1}	c_{n2}	$\dots$	c_{nn}

Con el cumplimiento de estas condiciones, el objetivo es encontrar la asignación que minimice la suma total de los costos.

1.1.1. Ejemplos

Veamos ahora un ejemplo sobre el problema de asignación donde mostramos un sencillo problema práctico de una empresa:
Ejemplo. Una compañía de manufactura desea realizar una jornada de mantenimiento preventivo a sus cuatro máquinas principales. La compañía cuenta con 4 proveedores distintos de servicios de mantenimiento. Entonces debe asignarse un equipo de mantenimiento a cada máquina, teniendo en cuenta que, según el grado de especialización de cada equipo prestador de servicios de mantenimiento, el costo de la tarea varía para cada máquina. Se debe asignar el equipo de mantenimiento correcto a la máquina correspondiente con el objetivo de minimizar el costo total de la jornada. Los costos asociados aparecen en la siguiente tabla:

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	58	58	60	54
Equipo de matenimiento 2	66	70	70	78
Equipo de matenimiento 3	106	104	100	95
Equipo de matenimiento 4	52	54	64	54

Tabla 1.1. Tabla de costes asociada al Ejemplo 1.1.1.

Este es un sencillo ejemplo con un tamaño matricial pequeño, pero el problema de asignación abarca contextos más complejos y de dimensiones más grandes. Por ejemplo, Schwartz [11] planteó un problema de asignación lineal que surgió en operaciones militares. Durante la Guerra Fría entre EE. UU. y la Unión Soviética, EE. UU. tuvo que asignar miles de interceptores ante la posible amenaza de un ataque soviético con miles de misiles, donde cada receptor estaba dedicado a intentar interceptar un misil soviético. Esto generó un problema de asignación lineal de gran tamaño que se resolvió mediante el algoritmo de subasta.
Además, los problemas de asignación aparecen con frecuencia como subproblemas en problemas de optimización combinatoria más complicados, como el problema de asignación cuadrática, el problema del viajante, los problemas de rutas de vehículos o problemas de planificación. Podemos encontrar estas aplicaciones y muchas más en [5].

1.2. Aspectos históricos e importancia del problema de asignación

El problema de asignación es un problema de optimización clásico, que se ha estudiado desde hace mucho tiempo y que ha tenido importantes desarrollos teóricos y prácticos.

Uno de los trabajos pioneros se sitúa en 1784, donde G. Monge estudió el transporte de moléculas. Dichas moléculas se tenían que transportar todas de un área a otra y Monge investigó la manera de que la distancia recorrida fuese lo más pequeña posible. En definitiva, un problema de asignación con enorme costo matricial al que Monge dio un método geométrico, considerando una línea tangente a las dos áreas. Aunque geométricamente intuitivo, el método no es del todo correcto, como señaló Appell en 1928.
Frobenius fue uno de los primeros matemáticos en estudiar el problema de asignación en el siglo XIX, y propuso un método para encontrar una solución óptima en casos particulares. El trabajo de Frobenius (en términos de matrices y determinantes) sentó las bases para el estudio posterior del problema de asignación.
Dénes Kőnig, matemático húngaro, se dio cuenta de que el teorema de Frobenius se puede formular de manera equivalente en términos de grafos bipartitos, al introducir una construcción, ahora bastante estándar, de asociar un grafo bipartito con una matriz A_{ij}. En 1914, Kőnig presentó un teorema que demuestra que todo grafo bipartito regular tiene un emparejamiento perfecto.
En 1931, Kőnig enunció un teorema más general, conocido como el teorema de Kőnig, que caracteriza el tamaño máximo de un emparejamiento en un grafo bipartito. Este teorema fue muy importante en el estudio del problema de asignación, ya que permitió reducir el problema a la búsqueda de un emparejamiento perfecto en un grafo bipartito.
Poco después, Jenő Egerváry (1931) formuló una versión ponderada del teorema de Kőnig. Este caracteriza el peso máximo de un emparejamiento en un grafo bipartito y, por lo tanto, se aplica al problema de asignación.
En 1949, el psicólogo, Thorndike, estudió el problema de clasificación del personal (asignación de puestos de trabajos). Thorndike le trasladó el problema a un matemático, el cual observó que el problema tiene un número finito de permutaciones, es decir, desde su punto de vista solo había que probar todas las posibles permutaciones y elegir la mejor. En cambio, Thorndike observó: "Cuando consideremos solo 10 vacantes y 10 hombres, hay alrededor de 3,5 millones de permutaciones. Probar todas las posibles permutaciones y elegir la mejor puede ser una solución matemática pero no es práctica". En consecuencia, el problema de asignación ha ayudado a comprender que un algoritmo finito no tiene por qué ser práctico y que existe una brecha entre el tiempo de cómputo exponencial y el tiempo de cómputo polinomial.
Un gran avance en la solución del problema de asignación se produjo cuando George Dantzig, en 1951, mostró que el problema de asignación se puede formular como un problema de programación lineal que, automáticamente, tiene una solución entera binaria como consecuencia del teorema de Birkhoff (1946). Por tanto, el problema de asignación también se puede resolver con el método simplex.
En 1955, Harold Kuhn, un matemático estadounidense, presentó un método

combinatorio básico, que usa el esquema primal-dual de la programación lineal, para el problema de asignación, basado en el trabajo de Dénes Kőnig y Jenő Egerváry y, por ello, lo bautizó como el Método Húngaro. Kuhn se contentó con afirmar que el número de iteraciones es finito, pero J. Munkres en 1957, observó que el método se ejecuta en un tiempo fuertemente polinomial $O(n^3)$, donde n es el tamaño de la matriz de costos.
En las décadas siguientes, se estudiaron diversas extensiones y variaciones del problema de asignación. Además, se desarrollaron nuevos algoritmos y enfoques para resolver el problema. El problema de asignación también ha encontrado aplicaciones prácticas en campos como la logística, el transporte y la planificación. Hoy en día, el problema de asignación sigue siendo un área activa de investigación, con nuevas aplicaciones y variantes que se proponen y nuevos algoritmos y técnicas de optimización que se desarrollan. En [10] tenemos una buena referencia para encontrar los antecedentes históricos del problema de asignación.

1.3. Fundamentos Matemáticos

En esta sección vamos a tratar los aspectos teóricos más importantes para el problema de asignación, es decir, vamos a focalizarnos en los principales teoremas y resultados relevantes para entender la formalización del problema y su resolución.
Como se indicó anteriormente, el problema de asignación trata de asignar n elementos (por ejemplo trabajos) a n máquinas (o trabajadores) de "la mejor manera posible". Es decir, por un lado tenemos la asignación como estructura combinatoria y, por otro lado, una función objetivo que impone elegir la "mejor" opción.

1.3.1. Asignaciones

Una asignación se puede definir como una aplicación biyectiva ϕ entre dos conjuntos finitos U y V de n elementos ($|U| = |V| = n$). Entonces representamos una asignación como una permutación ϕ, que podemos escribir como:

$$\phi = \begin{pmatrix} 1 & 2 & \dots & n \\ \phi(1) & \phi(2) & \dots & \phi(n) \end{pmatrix}$$

donde el elemento $1 \in V$ está asignado a $\phi(1) \in W$, el elemento $2 \in V$ está asignado a $\phi(2) \in W$, ..., el elemento $n \in V$ está asignado a $\phi(n) \in W$. Por ejemplo, una permutación de un conjunto de 4 elementos sería:

$$\phi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

El conjunto de todas las posibles asignaciones de un conjunto $\{1, 2, \dots, n\}$ se denota como $\mathcal{S}_n$ y tiene $n!$ elementos, es decir, $n!$ posibles permutaciones diferentes. A cada permutación ϕ del conjunto de elementos $\{1, 2, \dots, n\}$ le corresponde una única matriz $X_\phi = x_{ij}$ donde

$$x_{ij} = \begin{cases} 1, & si\ j = \phi(i) \\ 0, & en\ otro\ caso \end{cases}$$

Por ejemplo, en el caso de la permutación ϕ_1, vista anteriormente, le corresponde la matriz:

$$X_{\phi_1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Las matrices de permutación se caracterizan por el siguiente sistema de ecuaciones:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1, 2, \ldots, n \tag{1.1}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1, 2, \ldots, n \tag{1.2}$$

y la condición:

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, 2, \ldots, n \tag{1.3}$$

El primer conjunto de ecuaciones 1.1 afirma que cada fila de una matriz de permutación suma 1. El segundo conjunto de ecuaciones 1.2 afirma que cada columna de una matriz de permutación suma 1. Además, la condición 1.3 dice que la matriz de permutación está formada únicamente por ceros o unos. Estos tres conjuntos de restricciones se denominan *restricciones de asignación*.

Otra forma de introducir asignaciones es utilizando grafos bipartitos, como veremos a continuación.

1.3.2. Grafo bipartito

Recordemos que un *grafo bipartito* $G = (V, W; E)$ es un grafo tal que el conjunto de vértices es una unión disjunta $V \sqcup W$ y todas las aristas son de la forma $\{i, j\} \in E$ donde $i \in V$ y $j \in W$.
Ahora, vamos a definir el concepto de *emparejamiento* y, a posteriori, el concepto de *emparejamiento perfecto*.

Definición 1.3.1 *Sea G=(V,W;E) un grafo bipartito, se denomina* ***emparejamiento*** *al subconjunto M de E de forma que cada vértice de G incide (es vértice inicial o vértice final), a lo sumo, en una arista de M.*

Cuando una arista $e = \{i, j\} \in E$ empareja el vértice i con el vértice j, llamamos a los vértices i y j *emparejados*. Recordemos que la cardinalidad de M, denotada $|M|$, se llama *cardinalidad del emparejamiento*.

Definición 1.3.2 *Un emparejamiento M se denomina* ***emparejamiento perfecto*** *si* $|M| = |V| = |W| = n$.

Es decir, tenemos un emparejamiento perfecto si cada vértice de G incide exactamente con una arista de M. Entonces, cada asignación se puede representar como un emparejamiento perfecto.
Además, un emparejamiento es *maximal* si no se puede ampliar agregando más aristas sin romper la condición de que cada vértice sea incidente en, a lo sumo, una arista. Y un emparejamiento es *máximo* si tiene tantas aristas como sea posible, es decir, tiene cardinalidad máxima. Hay que tener en cuenta que un emparejamiento maximal puede no ser máximo pero todo emparejamiento máximo es maximal. Por tanto, todo emparejamiento perfecto es un emparejamiento máximo y, a su vez, un emparejamiento maximal.
El *problema de determinar un emparejamiento perfecto en un grafo bipartito* tiene un papel fundamental en la teoría del problema de asignación ya que este problema equivale a la determinación de un emparejamiento perfecto $M \subset E$ en un grafo bipartito $G = (V, W; E)$, es decir, si existe un conjunto de n aristas en G tal que cada vértice indice exactamente en una arista.
En 1935 Philip Hall planteó una condición necesaria y suficiente para el existencia de un emparejamiento perfecto en un grafo bipartito. Primero, hay que recordar que para un vértice $i \in V$ definimos la *vecindad de un vértice*, denotada $N(i)$, al conjunto de los vecinos de i, es decir, el conjunto de todos los vértices $j \in W$ que están conectados con i por una arista en E. Además, para cualquier subconjunto V' de V tenemos que:

$$N(V') = \left\{ j \in \ W \mid \{i', j\} \in E, \ \ \forall\, i' \in V' \right\}$$

Hall interpretó un emparejamiento perfecto como un matrimonio entre el conjunto V de damas y el conjunto W de caballeros, donde el conjunto $N(i)$ contiene a los pretendientes de cada dama $i \in V$. El teorema afirma que cada dama puede casarse con uno de sus pretendientes. Además como tenemos que $|V| = |W| = n$, todas las damas y caballeros pueden casarse. Por ello, es conocido como el *Teorema del Matrimonio.*

Teorema 1.3.1 (Teorema del Matrimonio, Hall, 1935) *Sea $G = (V, W; E)$ un grafo bipartito. G tiene una asignación (emparejamiento perfecto) sí, y sólo si, $|V| = |W|$ y todos los subconjuntos V' de V satisfacen:*

$$|V'| \leq \ |N(V')| \quad \textit{(condición de Hall)}$$

Demostración. Es trivial que la condición de Hall es necesaria para la existencia de un emparejamiento tal que todos los vértices de V estén emparejados. Por tanto, nos queda probar la suficiencia de esta condición. Procedemos por inducción sobre n ($|V| = n$). Si $|V| = 1$, la única dama se puede casar con su pretendiente $N(V)$, el único caballero de W.
Supongamos cierto para k, es decir, que el teorema se cumple para todos los grafos bipartitos con $|V| = k$. Veámos que se cumple el teorema para un grafo

bipartito $\overline{G} = (\overline{V}, \overline{W}; \overline{E})$ con $|\overline{V}| = k + 1$ si suponemos cierta la condición de Hall para $\overline{G}$. Consideremos los dos siguientes casos:

- $|V'| < |N(V')|$, $\forall\ V' \in \overline{V}$. En este caso, si emparejamos un vértice $i \in V'$ cualquiera con uno de sus vecinos $j \in N(i)$, entonces al eliminar i, j y todas sus aristas incidentes, nos queda el grafo $G = (V, W; E)$ con $|V| = k$ que cumple la condición de Hall por hipótesis de inducción. Por lo tanto, G tiene un emparejamiento de tamaño $|\overline{V}| - 1$. Si añadimos la arista $\{i, j\}$ tenemos un emparejamiento de tamaño $|\overline{V}|$ en $\overline{G}$.
- $|V'| = |N(V')|$, para un $V' \in \overline{V}$. Por hipótesis de inducción podemos emparejar cada vértice $i \in V'$ con el apropiado vértice $j \in N(V')$. Ahora, eliminamos los conjuntos $V' \in \overline{V}$, $N(V') \in \overline{W}$ y las aristas incidentes de $\overline{E}$ con estos vértices. Entonces, tenemos un nuevo grafo $\tilde{G} = (\tilde{V}, \tilde{W}; \tilde{E})$. Veamos que cumple la condición de Hall. Para ello, vamos a proceder por reducción al absurdo. Supongamos que el nuevo grafo $\tilde{G}$ no cumple la condición de Hall. Entonces, existe un subconjunto U de $\tilde{V}$ cuyos vecinos $N_{\tilde{G}}(U)$ cumplen que $|U| > |N_{\tilde{G}}(U)|$. Usando que $N_{\tilde{G}}(U) = N(U) \setminus N(V')$ tenemos que:

  $$N(V' \cup U)| = |N(V')| + |N(U) \setminus N(V')| < |V'| + |U| = |V' \cup U|$$

  Pero esto significa que la condición de Hall no se mantiene en $\overline{G}$, lo cual es una contradicción. □

Se puede expresar el Teorema de Hall en el lenguaje de las matrices 0 - 1 usando la matriz de adyacencia de un grafo. Recordar que la *matriz de adyacencia* de un grafo bipartito $G = (V, W; E)$ es una matriz $A = (a_{ij})$ de dimensión $|V|$ x $|W|$ (en este problema, $n \times n$) definida por:

$$a_{ij} = \begin{cases} 1, & si\ \{i, j\} \in E \\ 0, & en\ otro\ caso \end{cases}$$

Un emparejamiento perfecto corresponde, en este caso, a una selección de n entradas 1 en la matriz de adyacencia A, una en cada fila y columna. Por tanto, decimos que la matriz A contiene una matriz de permutación. Debido a la condición de Hall vista anteriormente, para $k = 0, 1, \ldots, n - 1$, la matriz A no contiene una $(k + 1) \times (n - k)$ submatriz de 0 elementos, ya que de lo contrario los $k + 1$ vértices correspondientes a las filas de esta submatriz tendrían menos de $k+1$ vecinos. Así pues, tenemos el siguiente *teorema de Frobenius* equivalente al teorema 1.3.1:

Teorema 1.3.2 (Frobenius, 1917) *Sea A una matriz arbitraria de orden $n \times n$ con entradas 0 y 1. La matriz A contiene una matriz de permutación sí y sólo si, para $k = 0, 1, \ldots, n-1$, la matriz A no contiene una $(k+1) \times (n-k)$ submatriz de 0 elementos.*

Como consecuencia tenemos que cada submatriz $(k + 1) \times (n - k)$, de cualquier matriz de permutación $n \times n$, contiene al menos una entrada 1.

Ahora veremos el *teorema de Kőnig*, de gran relevancia como base de algunos algoritmos de resolución para el problema de asignación. Pero primero, definamos la *cobertura de vértices*.

Definición 1.3.3 *Sea G=(V,W;E) un grafo bipartito, se denomina* ***cobertura de vértices*** *al subconjunto $C \in V \sqcup W$ de vértices de forma que cada arista de G incide en al menos un vértice de C.*

Teorema 1.3.3 (Teorema de Kőnig, 1916) *En un grafo bipartito, el mínimo número de vértices en una cobertura de vértices es igual a la máxima cardinalidad de un emparejamiento.*

$$\min_{Cobertura\ del\ vértice\ C} |C| = \max_{Emparejamiento\ M} |M|$$

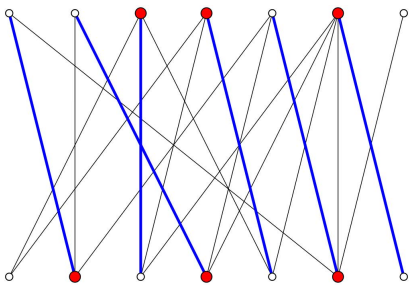

Figura 1.1. Ejemplo de tamaño seis con máximo emparejamiento (azul) y cubrimiento mínimo (rojo).

En algunos algoritmos de resolución para el problema de asignación lineal que veremos en el capítulo 2, haremos uso de la siguiente formulación equivalente del teorema de Kőnig. Dado un grafo bipartito $G = (V, W; E)$ con $|V| = |W| = n$, definimos la *matriz de adyacencia complementaria* B de G como la matriz $B = (b_{ij})$ de dimensión $n \times n$ donde:

$$b_{ij} = \begin{cases} 0, & si\ \{i,j\} \in E \\ 1, & en\ otro\ caso \end{cases}$$

Por tanto, si la matriz de adyacencia A tiene una entrada $a_{ij} = 0$, entonces $b_{ij} = 1$ y viceversa.
Ahora definimos la *cobertura de ceros* como un subconjunto de filas y columnas de la matriz B que contiene todos los elementos 0 de B. Una fila (o columna) que pertenece a la cobertura de ceros se denomina *fila cubierta (o columna cubierta)*. La *cobertura de ceros mínima* es una cobertura de ceros con un número mínimo de filas y columnas cubiertas. Esta cobertura de ceros mínima está directamente relacionada con una cobertura de vértices en el grafo bipartito $G = (V, W; E)$. Si la cobertura de vértices contiene un vértice $i \in V$, entonces le corresponde una columna cubierta de B. Si la cobertura de vértices contiene un vértice

$j \in W$, entonces le corresponde una fila cubierta de B. Por tanto, toda cobertura de vértices lleva a una cobertura de ceros en B y viceversa. En consecuencia, tenemos la siguiente proposición:

Proposición 1.1. *Existe una asignación ϕ tal que $b_{i\phi(i)} = 0,\ \forall i = 1, \ldots, n$, si y solo si la cobertura de ceros mínima tiene n elementos.*

1.3.3. Politopo de asignación

Se denomina a una matriz X *doblemente estocástica*, si es una matriz $n \times n$ en cuyas entradas se cumple:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1, 2, \ldots, n \tag{1.4}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1, 2, \ldots, n \tag{1.5}$$

$$x_{ij} \geq 0, \quad \forall i, j = 1, \ldots, n \tag{1.6}$$

Vimos, en la subsección 1.3.1, el sistema de ecuaciones correspondiente a la matriz de permutación, observando que guarda una estrecha relación con estas condiciones. De hecho, toda matriz de permutación X_ϕ es una matriz doblemente estocástica. En consecuencia, cada asignación ϕ de n elementos está descrita por una matriz doblemente estocástica donde, en particular, $x_{ij} \in \{0, 1\}\ \forall i, j = 1, 2, \ldots, n$. El conjunto formado por todas las matrices doblemente estocásticas se denomina *politopo de asignación* P_A. Birhkoff enunció un teorema importante al respecto:

Teorema 1.3.4 (Teorema de Birhkoff, 1946) *Los vértices del politopo de asignación corresponden únicamente a matrices de permutación*

Por lo tanto, cada matriz doblemente estocástica se puede escribir como una combinación convexa de matrices de permutación, como veremos en la demostración. Para probar el teorema de Birhkoff necesitamos, previamente, enunciar y demostrar el siguiente lema.

Lema 1.2. *Para cualquier $k \in \{0, 1, \ldots, n-1\}$, si una matriz cuadrada R, de dimensión $n \times n$, con entradas no negativas y sumas iguales de filas y columnas contiene una submatriz $(k+1) \times (n-k)$ de 0 elementos, entonces $R = 0$.*

Demostración. Sea la matriz cuadrada R, de dimensión $n \times n$, con entradas no negativas que contiene una submatriz $(k+1) \times (n-k)$ de 0 elementos y las sumas de filas y columnas iguales a α. Podemos reordenar las entradas de R permutando filas y columnas de la siguiente manera:

$$\left(\begin{array}{c|c} R_1 & R_2 \\ \hline 0 & R_3 \end{array}\right)$$

La suma de todos los coeficientes de las primeras $n-k$ columnas corresponde a $R_1+0=(n-k)\alpha$. La suma de todos los coeficientes de las últimas $k+1$ filas corresponde a $0+R_3=(k+1)\alpha$. Además, la suma de todos los elementos en R es $n\alpha$. Por tanto, tenemos que:

$$n\alpha=\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij} \geq \sum_{i=1}^{n}\sum_{j=1}^{n-k} r_{ij} + \sum_{i=n-k}^{n}\sum_{j=1}^{n} r_{ij} = (n-k)\alpha+(k+1)\alpha=(n+1)\alpha$$

Para que se cumpla la igualdad α tiene que ser 0. Entonces $R=0$. □

Demostración (teorema de Birhkoff). Sea X una matriz doblemente estocástica arbitraria. Sabemos por el lema que acabamos de demostrar que la matriz X no contiene ninguna submatriz $(k+1)\times(n-k)$ de 0 elementos, ya que de lo contrario tendríamos que $X=0$ y no podría ser doblemente estocástica. Según el teorema 1.3.2 (Teorema de Frobenius), existe una matriz de permutación $P_1=(p_{ij})$ tal que si $p_{ij}=1$, entonces $x_{ij}>0$. Sea λ_1 la entrada positiva más pequeña de X para la cual $p_{ij}=1$ se cumpla. Entonces $X-\lambda_1 P_1$ es una matriz con entradas no negativas e igual suma de filas y columnas. Repetimos el proceso siempre que no haya una submatriz $(k+1)\times(n-k)$ de 0 elementos. Nos queda que:

$$X=\lambda_1 P_1+\lambda_2 P_2+\ldots+\lambda_r P_r+R$$

donde $\lambda_1,\lambda_2,\ldots,\lambda_r$ son valores positivos y $P_1,P_2;\ldots,P_r$ matrices de permutación. Tenemos que la matriz R no tiene entradas negativas y las sumas de filas y columnas igual a α, tal que $\alpha=1-\lambda_1-\lambda_2-\ldots-\lambda_r$ con $\alpha\geq 0$. Además contiene una submatriz $(k+1)\times(n-k)$ de 0 elementos. Aplicando el lema 1.3.3, tenemos que $R=0$. Por lo tanto, $\lambda_1+\lambda_2+\ldots+\lambda_r=1$ y la matriz X es una combinación convexa de las matrices de permutación $P_1,P_2,\ldots,P_r$. □

El politopo de asignación

$$P_A=\left\{x\in\mathbb{R}^{n^2}:\sum_{j=1}^{n}x_{ij}=1,1\leq i\leq n;\sum_{i=1}^{n}x_{ij}=1,1\leq j\leq n;x\geq 0\right\}$$

tiene como matriz de coeficientes:

$$A=\left(\begin{array}{c|c|c|c} 1\,1\ldots 1 & & & \\ & 1\,1\ldots 1 & & \\ & & \ldots\ldots & \\ & & & 1\,1\ldots 1 \\ \hline 1 & 1 & & 1 \\ \;1 & \;1 & & \;1 \\ \;\;\ddots & \;\;\ddots & \ldots\ldots & \;\;\ddots \\ \;\;\;1 & \;\;\;1 & & \;\;\;1 \end{array}\right)$$

Esta matriz A puede verse como la matriz de incidencia de un grafo bipartito completo. Sea $K_{n,n} = (U, V; E)$ con $|U| = |V| = n$ y $E = \{i, j\}$, $\forall i \in U, j \in V$. Las primeras n filas corresponden a los vértices de U y las últimas n filas corresponden a los vértices de V. Cada columna corresponde con una arista. Veamos una defición para, a posteriori, ver una propiedad que cumple la matriz de coeficientes A.

Definición 1.3.4 *Una matriz entera se dice totalmente unimodular si cada submatriz cuadrada tiene determinante 0, 1 ó -1*

Proposición 1.3. *La matriz de coeficientes de un problema de asignación es totalmente unimodular*

Demostración. Sea A la matriz de coeficientes correspondiente a las restricciones de asignación. Vamos a proceder por inducción sobre n.
Si $n = 1$, tenemos que cualquier submatriz 1×1 de A tiene determinante 1. Supongamos que todas las matrices de tamaño $k \times k$ o menor cumplen esta proposición. Veamos que es cierto para cualquier submatriz D de tamaño $(k + 1)\times(k+1)$ de la matriz A. En el primer caso, si la matriz D contiene una columna llena de 0, entonces D es singular (det $D = 0$). En el segundo caso, si cada columna de la matriz D tiene dos entradas 1 diferentes implica que una entrada proviene de la parte superior de A (las primeras n filas) y la segunda entrada proviene de las últimas n filas. Por tanto, la suma de las filas de la parte superior de la matriz A son iguales a la suma de las restantes filas teniendo así que el det $D = 0$, pues los vectores fila son linealmente dependientes. Entonces, veamos el último caso en el que asumimos que una columna contiene solo una entrada 1. Sea esa entrada a_{ij} y sea D_{ij} la matriz de tamaño $k\times k$ formada al eliminar la fila i y la columna j de la matriz D. Entonces, $\det(D)=\pm a_{ij}$ $\det(D_{ij})=\pm\det(D_{ij})$, donde $\det(D_{ij})$ es igual a 0 o ± 1 por hipótesis de inducción. En conclusión, $\det(D) \in \{0, \pm 1\}$. □

1.4. Problema de asignación lineal

Formalmente, sean n orígenes que deben ser asignados a n destinos. Tenemos una matriz de costo $C = c_{ij}$ de dimensión $n \times n$ donde cada c_{ij} es el coste de asignar cada Origen $O_i,\ i = 1, \ldots, n$ a cada Destino $D_j,\ j = 1, \ldots, n$. Pretendemos seleccionar n elementos de C tal que haya exactamente un elemento en cada fila y un elemento en cada columna de manera que la suma de los costos sea un mínimo.
Alternativamente, podemos definir el problema de asignación lineal mediante teoría de grafos: problema de emparejamiento perfecto ponderado (*weighted perfect matching problems*). Dado un grafo bipartito $G = (V, W; E)$ con un vértice de V para cada fila de la matriz de costos y un vértice de W para cada columna,

donde c_{ij} son los costos para todas las aristas $\{i,j\} \in E$ $(i,j \in \{1,2,\ldots,n\})$, encuentre un emparejamiento perfecto con el mínimo coste:

$$min\left\{ \sum_{\{i,j\}\in M} c_{ij} : M \; es \; un \; emparejamiento \; perfecto \right\}$$

Se trabajará tanto con el grafo bipartito G como con la matriz de costes C, pretendiendo dar una visión más completa del problema de asignación. Sin perdida de generalidad, suponemos que los costos c_{ij} son no negativos, ya que la mayoría de algoritmos de preprocesamiento producen una matriz de costos reducidos no negativa. Si no lo fueran, bastaría hacer un sencillo cambio que consiste en restar a cada coeficiente el mínimo de los coeficientes de la matriz de coste, es decir, $\min_{i,j} c_{ij}$, y el problema resultante sería equivalente. De esta forma, podemos también manejar el problema de asignación cuando se trata de **maximizar** resolviendo la transformada con costos $\tilde{c}_{ij} = -c_{ij}$. También asumimos, en general, que los valores en C son finitos $(< \infty)$.

1.4.1. Modelo Matemático

Definimos la matriz $X = (x_{ij})$ formada por las variables binarias:

$$x_{ij} = \begin{cases} 1, & si \; el \; origen \; i \; se \; asigna \; al \; destino \; j \\ 0, & en \; otro \; caso \end{cases}$$

Definimos el modelo lineal del problema de asignación:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1.7}$$

Sujeto a las restricciones:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1,2,\ldots,n \tag{1.8}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1,2,\ldots,n \tag{1.9}$$

$$x_{ij} \in \{0,1\}, \quad \forall i,j = 1,\ldots,n \tag{1.10}$$

Nótese que la matriz X resultante es una matriz de permutación, como vimos en la sección 1.3.1. Además, la matriz A, correspondiente a las restricciones de asignación, es la matriz de incidencia del grafo bipartito $G = (V,W;E)$ que definimos previamente, y que también vimos en la sección 1.3.3. Esta matriz A tiene una fila i por cada vértice $i \in V$, una fila $n+j$ por cada vértice $j \in W$ y una

columna por cada arista $\{i, j\} \in E$. Las entradas de esta matriz son 0 y 1. En la sección 1.3.3, demostramos la propiedad de que esta matriz A es *totalmente unimodular*.
Entonces, un problema lineal entero con una matriz de restricciones totalmente unimodular tiene **solución óptima entera**. Así pues, podemos relajar las condiciones del problema de asignación, es decir, reemplazamos $x_{ij} \in \{0, 1\}$ por

$$x_{ij} \geq 0, \ \ (i, j = 1, 2, \ldots, n) \tag{1.11}$$

1.4.2. Problema dual

Acabamos de ver que podemos relajar las condiciones $x_{ij} \in \{0, 1\}$ por $x_{ij} \geq 0$. Por tanto, podemos formular el correspondiente problema dual del problema 1.7.

$$max \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \tag{1.12}$$

sujeto a:

$$u_i + v_j \ \leq c_{ij} \qquad i, j = 1, \ldots, n \tag{1.13}$$

$$u_i, v_j \in \mathbb{R} \qquad i, j = 1, \ldots, n \tag{1.14}$$

donde u_i y v_j $(i, j = 1, \ldots, n)$ son las variables duales asociadas a las restricciones del problema primal.
En consecuencia, podemos poner la restricción del problema dual 1.13 como:

$$c_{ij} - u_i - v_j \geq 0 \quad (i, j = 1, \ldots, n) \tag{1.15}$$

donde $c_{ij} - u_i - v_j = \bar{c}_{ij}$ son los *costos reducidos*.
Recordemos una propiedad muy importante de dualidad, el *teorema fuerte de dualidad*, que relaciona la solución óptima del problema dual con la solución óptima del problema primal.

Teorema 1.4.1 (Teorema fuerte de dualidad) *Si el problema primal (dual) tiene solución óptima, entonces el problema dual (primal) tiene solución óptima y los valores óptimos de ambos problemas coinciden.*

En consecuencia, ambos problemas (primal y dual), al ser evaluados en sus respectivas soluciones óptimas, proveen idéntico valor óptimo. Es más, resulta suficiente resolver uno de ellos y luego utilizar la propiedad de *condición de holgura complementaria* para encontrar la solución óptima (y valor óptimo) de su problema equivalente.

Teorema 1.4.2 (Condición de holguras complementarias) *Sean $\overline{x}$ solución factible del problema primal e $\overline{y}$ solución factible del problema dual. $\overline{x}$ es solución óptima del problema primal e $\overline{y}$ es solución óptima del problema dual, sí y sólo si,*

$$\overline{x}(c - A^t\overline{y})^t = 0$$

Aplicándolo al problema de asignación con las variables duales u_i y v_j $(i, j = 1, \ldots, n)$, se tiene que cumplir:

$$\overline{x}_{ij}\overline{c}_{ij} = 0 \qquad \forall\ i, j = 1, \ldots, n \tag{1.16}$$

De hecho, veamos la función objetivo transformada para cada solución factible primal $\overline{x}$:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - u_i - v_j)\overline{x}_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\overline{x}_{ij} \text{ - } \sum_{i=1}^{n} u_i \sum_{j=1}^{n}\overline{x}_{ij} \text{ - } \sum_{j=1}^{n} v_j \sum_{i=1}^{n}\overline{x}_{ij}$$

Recordemos las restricciones del problema primal:

$$\sum_{i=1}^{n}\overline{x}_{ij} = 1 \ \ \forall\ i = 1, 2, \ldots, n$$
$$\sum_{j=1}^{n}\overline{x}_{ij} = 1 \ \ \forall\ j = 1, 2, \ldots, n$$

Al añadir estas restricciones a la función objetivo nos queda:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\overline{x}_{ij} \text{ - } \sum_{i=1}^{n} u_i \text{ - } \sum_{j=1}^{n} v_j$$

Observamos aquí que, para cualquier solución factible $\overline{x}$, la diferencia es una **constante** $\sum_{i=1}^{n} u_i$ - $\sum_{j=1}^{n} v_j$.

La transformación de la matriz de costos C a $\overline{C}$ es un caso especial de *transformaciones admisibles*. Una transformación T de la matriz de costos $C = (c_{ij})$ a $\overline{C} = (\overline{c}_{ij})$ se denomina admisible si para cada permutación ϕ de n elementos se cumple la siguiente igualdad:

$$\sum_{i=1}^{n} c_{i\phi(i)} = \sum_{i=1}^{n}\overline{c}_{i\phi(i)} + z(T)$$

donde $z(T)$ es una constante que depende solo de la transformación. Una transformación admisible no cambia el orden de las permutaciones (soluciones factibles del problema de asignación lineal) con respecto al valor de la función objetivo. Por tanto, los problemas lineales con matrices de costo C y $\overline{C}$ son equivalentes.

Tendremos que tenerlo en cuenta para los algoritmos de resolución del problema de asignación lineal, pues trataremos tanto con el problema primal como con el problema dual.

2

Métodos de Resolución

En este capítulo vamos introducir la clasificación de los métodos de resolución para el problema de asignación lineal. Posteriormente, explicaremos en detalle tres algoritmos de resolución acompañados de un ejemplo donde se apliquen: algoritmo húngaro, algoritmo de subasta y algoritmo de Mack.

2.1. Introducción a los métodos de resolución

El problema de asignación lineal se puede resolver de manera eficiente. Existe una cantidad asombrosa de algoritmos para resolver el problema y el diseño de los métodos de resolución ha sido objeto de investigación durante muchos años. Podemos clasificar estos métodos en tres grupos.

El primero está constituido por algoritmos de tipo *primal-dual*. Entre ellos el famoso **algoritmo húngaro**, presentado por H.Kuhn en la década de 1950 y que explicaremos más adelante en detalle. En 1976, Lawler hizo una implementación para el algoritmo húngaro que mejoró su complejidad en $O(n^3)$. Dentro de este grupo podemos considerar también el **algoritmo de subasta**, propuesto en 1981 por Bertsekas. Este algoritmo tiene una complejidad de tiempo pseudopolinomial pero una alta eficiencia promedio en la práctica. Posteriormente, Bertsekas y Eckstein obtuvieron un algoritmo de subasta en tiempo polinomial, combinando la subasta y una técnica particular de escalado (conocida como ϵ-relajación). El método primal-dual realiza los siguientes pasos:

1. Encontrar una solución dual factible.
2. Resolver un problema primal restringido, es decir, encontrar una solución primal (parcial) que cumpla la condición de holguras complementarias con la solución dual.
3. Terminar en caso de que la solución del problema primal restringido resuelva el problema de asignación. De lo contrario, resolver un problema dual restrin-

gido para ajustar la solución dual. Esto lleva a un problema primal menos restringido y se regresa al segundo punto.

El segundo grupo son los métodos basados en *caminos más cortos*. Son algoritmos en los que existe viabilidad dual y se debe alcanzar la viabilidad primal. Esto se logra, considerando el problema de asignación como un problema de flujo de costo mínimo, que se puede resolver encontrando caminos más cortos que incrementan el flujo o caminos incrementales más cortos (*shortest augmenting paths*) en un grafo auxiliar. Dentro de este grupo, destacamos dos algoritmos, ambos de complejidad $O(n^3)$: el algoritmo de Tomizawa (1971) y el algoritmo de Hung & Rom (1980). El algoritmo de Tomizawa se inicializa con una solución primal parcial y una solución dual factible. La asignación parcial se incrementa a una solución completa mediante pasos en el problema primal. En cada uno de los pasos se determina un camino incremental más corto utilizando el método de Dijkstra (1959). En cada paso del algoritmo de Hung & Rom se determina un árbol de caminos mínimos, que conduce a encontrar más caminos incrementales en cada iteración.
Destacamos en este grupo el **método Bradford de Mack** (1969) al ser un algoritmo sencillo de aplicar y fácil de entender y, por ello, profundizaremos más adelante en sus pasos. Este método tiene similitudes con el método de Hung & Rom y el método húngaro en su formulación original con complejidad $O(n^4)$. Una adaptación para mejorar su complejidad temporal a $O(n^3)$ nos presenta un algoritmo parecido al de Tomizawa. Los pasos de los algoritmos de caminos incrementales más cortos son:

1. Inicialización.
2. Terminación, si las filas están asignadas.
3. Incremento. Construir la red auxiliar y determinar, desde una fila i sin asignar hasta una columna j sin asignar, un camino alterno de mínimo costo reducido total, y utilizar para incrementar la solución.
4. Ajustar la solución dual para restaurar la condición de holguras complementarias. Ir al paso 2.

El tercer grupo está constituido por algoritmos basados en la *programación lineal* (versiones especializadas del método simplex). Los mejores resultados que se han publicado son de Barr, Glover & Klingman (1977). Una dificultad con estos métodos es la degeneración, donde casi la mitad de las variables básicas son iguales a cero. Otro incoveniente es la compleja implementación en comparación con los otros enfoques. Experimentos computaciones demuestran que este tipo de algoritmos son superados por los mejores algoritmos primal-dual y dual. Por ello, no vamos a entrar en detalle con ningún método de esta categoria. Aun así, a este grupo pertenece el algoritmo presentado por Balinski en 1985 con complejidad $O(n^3)$. Desde su primera publicación, se han propuesto algunas variantes pero, hasta el momento, no se han presentado resultados computacionales.

2.2. El Método Húngaro

En 1955, Harold W.Kuhn publicó dos famosos artículos que presentaban el método húngaro. Este resultado histórico permitió, por primera vez, encontrar una solución óptima al problema de asignación que ningún ordenador en la tierra podía resolver en ese momento. Kuhn le dio este nombre porque el algoritmo se basó, en gran medida, en los trabajos anteriores de dos matemáticos húngaros: Dénes Kőnig y Jenő Egerváry. James Munkres revisó el algoritmo en 1957 y observó que es fuertemente polinomial.
En el caso de problemas de pequeña dimesión, el método húngaro es un recurso didáctico que nos muestra como resolver el problema de asignación "a mano", sin ayuda de un ordenador. También existen implementaciones informáticas, basadas en el propio método húngaro, para resolver problemas de mayor dimensión.

2.2.1. Pasos del algoritmo húngaro

El método húngaro es un algoritmo primal-dual. Comienza con una solución factible dual, forzando las condiciones de holguras complementarias, intenta encontrar una solución factible primal. En las diferentes iteraciones, maneja soluciones primales "parcialmente" factibles hasta encontrar una que sea factible primal. En este punto, la solución primal también es óptima, según la teoría de la dualidad. Veamos los pasos del algoritmo de Kuhn-Munkres en donde se transforma la matriz de costes C hasta conseguir la asignación óptima:

1. Obtener ceros por filas. Restamos el elemento mínimo de cada fila a todos los elementos pertenecientes a dicha fila.

   $$u_i = \min_j \{c_{ij}\}$$

   Los nuevos elementos de la tabla son $c'_{ij} = c_{ij}$ - u_i, $\forall i, j = 1, \ldots, n$, que mantienen la factibilidad dual al ser costos no negativos.

2. Obtener ceros por columnas. Restamos el elemento mínimo de cada columna a todos los elementos pertenecientes a dicha columna.

   $$v_j = \min_i \{c'_{ij}\}$$

   Los nuevos elementos de la tabla son $c''_{ij} = c'_{ij}$ - $v_j = c_{ij}$ - u_i - v_j, $\forall i, j = 1, \ldots, n$, que mantienen la factibilidad dual al ser costos no negativos.

3. Trazar, en la matriz resultante, el mínimo número de líneas horizontales y verticales de tal manera que se cubran todos los ceros. Sea N este mínimo número de líneas, tenemos dos posibilidades:

a) Si $N = n$, entonces podemos realizar una asignación óptima con la matriz resultante.
Pasamos al 5.
b) Si $N < n$, entonces no podemos realizar una asignación óptima.
Pasamos al 4.

4. Crear nuevos ceros. Seleccionamos el elemento mínimo de la submatriz formada por las columnas y filas no cubiertas por las N líneas. Restamos este elemento mínimo a todos los elementos no cubiertos que forman la submatriz y sumamos el mismo elemento a los elementos que se encuentran en la intersección entre las líneas verticales y horizontales.
Sea I el conjunto de índices de filas correspondiente a las filas no cubiertas y sea J el conjunto de índices correspondiente a las columnas no cubiertas. El elemento mínimo de los costos reducidos descubiertos es:

$$\delta = min\{\overline{c}_{ij} : i \in I, j \in J\}$$

Entonces nos queda la siguiente transformación:

$$\overline{c}_{ij} := \begin{cases} c_{ij} - \delta, & si\ i \in I, j \in J \\ c_{ij} + \delta, & si\ i \notin I, j \notin J \\ c_{ij}, & en\ otro\ caso \end{cases}$$

Obtenemos así, una nueva matriz modificada. Volvemos al 3.

5. Construcción de solución óptima. Elegimos la fila o columna con menor número de ceros. Asignamos un cero y lo encuadramos. Tachamos todos los ceros de la misma fila y columna de tal manera que no se puedan considerar para una asignación posterior. Repetimos la asignación en filas y columnas continuando por aquella que tenga el mínimo número de ceros sin tachar. Al final, se obtiene un único cero asignado (encuadrado) en cada columna y en cada fila de la matriz. La asignación correspondiente a estos ceros encuadrados será la asignación óptima. Termina el algoritmo.

2.2.2. Justificación de los pasos del algoritmo húngaro mediante grafos bipartitos

Partimos de la matriz de costos $C = (c_{ij})$ de dimensión $n \times n$. En la sección 1.4.2 vimos como la transformación T de la matriz de costos C a la matriz de costos reducidos $\overline{C} = (\overline{c}_{ij}) = (c_{ij} - u_i - v_j)$, no supone un cambio en la solución óptima del problema lineal. Por ello, ahora, podemos hacer las siguientes transformaciones en la matriz de costos C: restar en las filas y/o columnas una constante. Restamos el elemento mínimo de cada fila, u_i, en la

fila correspondiente y el elemento mínimo de cada columna, v_j, en la columna correspondiente.

$$u_i = \min_j\{c_{ij}\}$$
$$v_j = \min_i\{c_{ij} - u_i\}$$

Obtenemos, así, la matriz de costos reducidos $\overline{C}$ donde los elementos son de la forma: $\overline{c}_{ij} = c_{ij}$ - u_i - v_j, $\forall i, j = 1, \ldots, n$.
El algoritmo húngaro comienza con una solución dual factible u y v, que satisface la restricción del problema dual $c_{ij} - u_i - v_j \geq 0$ $(i, j = 1, \ldots, n)$, y una solución primal parcial (en el que caso de que la cardinalidad de la asignación sea menor a n filas) satisfaciendo la condición de holguras complementarias $x_{ij}(c_{ij} - u_i - v_j) = 0$, $\forall\ i, j = 1, \ldots, n$.
Una vez obtenida la solución parcial, el algoritmo húngaro resuelve en cada iteración un problema primal restringido independiente de los costos, operando en el grafo bipartito parcial $\overline{G}$. Sea el grafo bipartito $\overline{G} = (V, W; \overline{E})$, donde $V = W = \{1, 2, \ldots, n\}$ y $\overline{E} = \{\{i, j\} : \overline{c}_{ij} = c_{ij}$ - u_i - $v_j = 0\}$, es decir, solo contiene las aristas de E con coste reducido igual a 0. Sea M el emparejamiento correspondiente a la asignación parcial obtenida. El objetivo es aumentar la cardinalidad del emparejamiento actual ($|M|$) mediante *árboles alternos*. Si el intento tiene éxito, se obtiene una nueva solución primal en la que se mejora la asignación anterior. De los contrario, se actualiza la solución dual para que se obtengan nuevas aristas con costos reducidos igual a 0. Para describir el proceso necesitamos algunas definiciones básicas:

Definición 2.2.1 *Una arista de i a j en un grafo bipartito G se dice que está emparejada, si pertenece a M. En caso contrario, se denomina arista libre.*

Definición 2.2.2 *Un vértice de un grafo bipartito G se dice que está emparejado, si pertenece a M. En caso contrario, se denomina vértice libre.*

Definición 2.2.3 *Un camino alterno en un grafo bipartito G, respecto al emparejamiento M, es un camino donde las aristas están emparejadas y libres de manera alternada en M.*

Definición 2.2.4 *Un árbol alterno en un grafo bipartito G, respecto al emparejamiento M, es un árbol que tiene una raíz en un vértice $r \in V$ y todos los caminos que parten desde r son alternos.*

Definición 2.2.5 *Un camino incremental P en un grafo bipartito G es un camino alterno que une dos vértices libres $v \in V$ y $w \in W$, es decir, sus aristas final e inicial no están asignadas.*

Comenzamos con un emparejamiento parcial M. El algoritmo húngaro consiste en buscar un camino incremental mediante caminos alternos. El algoritmo

construye un árbol y busca un posible camino incremental P a partir de un vértice $r \in V$ no emparejado. Si se encuentra tal camino P, tenemos un emparejamiento con mayor cardinalidad que el emparejamiento anterior. Este se obtiene intercambiando aristas emparejadas y no emparejadas a lo largo de P, es decir, estableciendo $x_{ij} = 1$ para las aristas no emparejadas {i,j} de P (aristas impares $|P|/2+1$) y $x_{ij} = 0$ para las aristas emparejadas {i,j} de P (aristas pares $|P|/2$). En este caso, aumentamos el número de variables primales x_{ij} que son iguales a 1 y, de esta manera, mejorar la solución primal sin tener que cambiar la solución dual. En definitiva, buscamos un camino alterno que comience en un vértice no asignado $r \in V$ haciendo crecer progresivamente un árbol alterno con raíz en r, resultando un nuevo emparejamiento M con mayor cardinalidad que el emparejamiento anterior.
El objetivo final es encontrar un emparejamiento perfecto (cada vértice de G incide exactamente con una arista de M) en el grafo bipartito parcial $\overline{G}$. En consecuencia del Teorema 1.3.3 (Teorema de Kőnig), la máxima cardinalidad del emparejamiento M corresponde con el mínimo número de vértices en una cobertura de vértices de un grafo bipartito. También vimos como toda cobertura de vértices lleva a una cobertura de ceros. Entonces, definimos I como el conjunto de índices de filas correspondiente a las filas no cubiertas y sea J el conjunto de índices correspondiente a las columnas no cubiertas. Debido a la proposición 1.1, la constante $z(T)$ de la transformación de la matriz de costos es igual a 0 ($|I| + |J| = n$), sí y sólo si existe una permutación ϕ tal que $c_{i\phi(i)} = 0$, para $1 \geq i \geq n$. En este último caso tenemos una solución factible primal y, por tanto, la **solución óptima**. Si el mínimo número de filas y columnas cubiertas es menor que n ($|I|+|J| \neq n$) tenemos que seguir buscando caminos incrementales. Si no podemos encontrar más caminos incrementales, actualizamos las variables duales para conseguir más costos reducidos iguales a 0:

$$u_i := \begin{cases} u_i - \delta, & si\ i \in I \\ u_i, & en\ otro\ caso \end{cases}$$

$$v_j := \begin{cases} v_j, & si\ j \in J \\ v_j + \delta, & en\ otro\ caso \end{cases}$$

donde $\delta = min\{\overline{c}_{ij} : i \in I, j \in J\}$, es decir, está definida como el mínimo de los costos reducidos descubiertos.
Entonces nos queda la siguiente transformación:

$$\overline{c}_{ij} := \begin{cases} c_{ij} - \delta, & si\ i \in I, j \in J \\ c_{ij} + \delta, & si\ i \notin I, j \notin J \\ c_{ij}, & en\ otro\ caso \end{cases}$$

Esta transformación preserva la no negatividad de la matriz de costos reducidos, es decir, la factibilidad del problema dual y agrega una nueva arista al grafo bipartito parcial $\overline{G}$.
Ahora, volvemos a intentar encontrar un camino incremental que mejore la asig-

nación parcial actual. Como máximo hay n caminos incrementales porque la cardinalidad de un emparejamiento no puede ser mayor que n y, ya vimos, como el proceso acaba cuando $|I| + |J| = n$, es decir, cuando la suma de las filas y columnas no cubiertas sea n, teniendo así un emparejamiento perfecto.

Complejidad computacional del Método Húngaro

El algoritmo húngaro, presentado por Kuhn, en su formulación original resuelve el problema en una complejidad de $O(n^4)$.
Cuando usamos *árboles alternos*, tenemos como máximo n caminos incrementales y las actualizaciones de variables duales entre dos incrementos consecutivos se realizan en $O(n^2)$. Por tanto, en total, tenemos un algoritmo de complejidad $O(n^3)$.

2.2.3. Resolución de un ejemplo mediante el algoritmo húngaro

Recordemos el ejemplo 1.1.1 y su tabla de costos asociada 1.1. Vamos a emplear el método húngaro para minimizar el costo total de este problema.

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	58	58	60	54
Equipo de matenimiento 2	66	70	70	78
Equipo de matenimiento 3	106	104	100	95
Equipo de matenimiento 4	52	54	64	54

- **Paso 1.** Restamos el elemento mínimo de cada fila a todos los elementos pertenecientes a dicha fila. Así pues, restamos 54, 66, 95, 52 para las filas primera, segunda, tercera y cuarta, respectivamente. Ahora tenemos la tabla:

$$u_1 = min\{58, 58, 60, 54\} = 54$$

$$u_2 = min\{66, 70, 70, 78\} = 66$$

$$u_3 = min\{106, 104, 100, 95\} = 95$$

$$u_4 = min\{52, 54, 64, 54\} = 52$$

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	4	4	6	0
Equipo de matenimiento 2	0	4	4	12
Equipo de matenimiento 3	11	9	5	0
Equipo de matenimiento 4	0	2	12	2

- **Paso 2.** Restamos el elemento mínimo de cada columna a todos los elementos pertenecientes a dicha columna.

$$v_1 = min\{4, 0, 11, 0\} = 0$$

$$v_2 = min\{4, 4, 9, 2\} = 2$$

$$v_3 = min\{6, 4, 5, 12\} = 4$$

$$v_4 = min\{0, 12, 0, 2\} = 0$$

Así pues, restamos 0, 2, 4, 0 para las columnas primera, segunda, tercera y cuarta, respectivamente. Ahora tenemos la nueva tabla:

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	4	2	2	0
Equipo de matenimiento 2	0	2	0	12
Equipo de matenimiento 3	11	7	1	0
Equipo de matenimiento 4	0	0	8	2

- **Paso 3.** Realizamos el mínimo número de líneas horizontales y verticales de tal manera que se cubran todos los ceros.

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	4	2	2	0
Equipo de matenimiento 2	0	2	0	12
Equipo de matenimiento 3	11	7	1	0
Equipo de matenimiento 4	0	0	8	2

En total hemos hecho tres líneas (dos horizontales y una vertical) y la matriz tiene 4 filas (columnas). Por tanto, como $3 < 4$, tenemos que ir al **Paso 4.**

- **Paso 4.** Seleccionamos el elemento mínimo de la submatriz formada por las columnas y filas no cubiertas por las tres líneas.

$$\delta = min\{4, 2, 2, 11, 7, 1\} = 1$$

Restamos 1 a todos los elementos de las filas y columnas no cubiertas y sumamos 1 a los elementos que se encuentren en la intersección entre las líneas verticales y horizontales (filas y columnas cubiertas), es decir, a los elementos 12 y 2.
Volvemos al **Paso 3.**

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	3	1	1	0
Equipo de matenimiento 2	0	2	0	13
Equipo de matenimiento 3	10	6	0	0
Equipo de matenimiento 4	0	0	8	3

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	3	1	1	0
Equipo de matenimiento 2	0	2	0	13
Equipo de matenimiento 3	10	6	0	0
Equipo de matenimiento 4	0	0	8	3

- **Paso 3.** Realizamos el mínimo número de líneas horizontales y verticales de tal manera que se cubran todos los ceros.

  En total hemos hecho cuatro líneas (una horizontal y tres verticales) y la matriz tiene 4 filas (columnas). Por tanto, como $4 = 4$, tenemos que ir al **Paso 5.**

- **Paso 5.** Asignamos ceros. Elegimos la fila o columna con menor número de ceros. Asignamos un cero y lo encuadramos. Tachamos todos los ceros de la misma fila y columna de tal manera que no se puedan considerar para una asignación posterior. Repetimos el proceso hasta tener todos los ceros encuadrados.

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**
Equipo de matenimiento 1	3	1	1	$\boxed{0}$
Equipo de matenimiento 2	$\boxed{0}$	2	$\not{0}$	13
Equipo de matenimiento 3	10	6	$\boxed{0}$	$\not{0}$
Equipo de matenimiento 4	$\not{0}$	$\boxed{0}$	8	3

  Interpretando la última tabla tenemos que: el equipo de mantenimiento 1 se encargará de la máquina 4. El equipo de mantenimiento 2 se encargará de la máquina 1. El equipo de mantenimiento 3 se encargará de la máquina 3. El equipo de mantenimiento 4 se encargará de la máquina 2.
  Con esta asignación el coste total (óptimo) es: $54 + 66 + 100 + 54 = 274$.

2.3. Algoritmos de subasta

El algoritmo de subasta original fue introducido en 1981 por Dimitri P. Bertsekas para una versión de maximización del problema de asignación lineal. Según distintos artículos de la literatura, se puede considerar como un algoritmo

primal-dual o simplemente dual. Años más tarde, Bertsekas y Eckstein combinaron el algoritmo con una técnica denominada ϵ-*scaling* que veremos en detalle en esta sección.
El algoritmo de subasta mantiene una solución primal x (no factible) y una solución dual factible (u, v) que satisface la condición de holguras complementarias. El nombre de *subasta* proviene de la siguiente interpretación: queremos maximizar la función objetivo del problema de asignación lineal bajo las restricciones de asignación. Imaginemos que cada columna j representa un objeto que está a la venta en una subasta y cada fila i es un cliente. Para el objeto j, c_{ij} es el costo correspondiente para el cliente i. La variable dual v_j es el precio actual del objeto j, por lo que la diferencia $c_{ij} - v_j$ es el margen de beneficio del cliente i en relación al objeto j.
Sin embargo, con el objetivo de mantener la uniformidad en nuestro planteamiento, describiremos el algoritmo de subasta para el caso de minimización del problema de asignación lineal.

2.3.1. Algoritmo de subasta ingenuo

El algoritmo de subasta ingenuo tiene un fallo grave, como se verá más adelante. No obstante, este fallo ayudará a motivar un algoritmo más sofisticado y correcto.
Recordemos que el problema dual viene dado por:

$$max \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$$

$$s.a. \qquad u_i + v_j \ \leq c_{ij} \qquad i, j = 1, \ldots, n$$

Además, podemos observar que, para un vector v, el vector factible u que proporciona el valor máximo de la función objetivo viene dado por:

$$u_i = min\{c_{ij} - v_j : j = 1, 2, \ldots, n\} \ para \ i = 1, 2, \ldots, n \tag{2.1}$$

Como consecuencia directa, podemos expresar el problema dual de manera equivalente como:

$$max \ q(v) = \sum_{i=1}^{n} \min_{j} \{c_{ij} - v_j\} + \sum_{j=1}^{n} v_j \tag{2.2}$$

El algoritmo de subasta ingenuo se basa en iteraciones, donde cada una está compuesta de dos fases:

1. **Fase de oferta.** Sea NV el conjunto de filas no asignadas y consideremos la solución primal (no factible) $x_{i\phi(i)} = 1$ para $i \in V \backslash NV$ ($x_{ij} = 0$ en caso contrario).

Para un vector v, denotamos a la columna en la que se produce el valor mínimo de $c_{ij} - v_j$ como:

$$\phi(i) = arg\ min\{c_{ij} - v_j \ : \ j = 1, 2, \ldots, n\} \ \ (i = 1, 2, \ldots, n) \tag{2.3}$$

Entonces el par $(x, (u, v))$, con u dado por 2.1, satisface las condiciones de holguras complementarias. Por tanto, la solución primal x será factible (y por lo tanto óptima) solo si $NV = \emptyset$ y $\phi(i) \neq \phi(k)$ para $i \neq k$.
Una vez tengamos calculado u_i para una fila i, denotamos al segundo valor mínimo $c_{ij} - v_j$ para la fila i como:

$$u_i^s = min\{c_{ij} - v_j \ : \ j = 1, 2, \ldots, n, \quad j \neq \phi(i)\} \tag{2.4}$$

2. **Fase de asignación.** Empezando con la asignación vacía ($NV = V$), en cada iteración se selecciona una fila no asignada $i \in NV$. Pueden ocurrir dos casos:
Caso 1. ($u_i < u_i^s$) o ($u_i = u_i^s$ y $\phi(i) \neq \phi(k)\ \forall k \in V \backslash NV$).
Se asigna la fila i a la columna $\phi(i)$ y quitamos i del conjunto NV. Si la columna $\phi(i)$ tenía asignada otra fila, dicha fila ingresa en el conjunto NV. Ahora, las variables duales se actualizan $v_{\phi(i)} = v_{\phi(i)} - (u_i^s - u_i)$ y $u_i = u_i^s$ preservando la factibilidad dual. Si, después de todas las iteraciones, conseguimos que $NV = \emptyset$, el algoritmo termina eficientemente.
En la interpretación del caso 1, el nuevo cliente i (sin ningún objeto asignado) elige el objeto j, otorgándole el máximo margen de beneficio, y puja por su precio (fase de oferta) por la diferencia entre este margen y el segundo margen máximo de beneficio, es decir, por la mayor cantidad por la que j todavía le dará el máximo margen de beneficio. Luego, el artículo j se asigna al cliente i (fase de asignación) en lugar de otro cliente (si lo hay) que había pujado anteriormente por j.
Caso 2. ($u_i = u_i^s$ y $\phi(i) = \phi(k)$ para algún $k \in V \backslash NV$).
Aquí reside el gran inconveniente del algoritmo de subasta ingenuo. Se genera un ciclo infinito cuando más de un objeto ofrece el valor máximo, lo cual genera que el precio de los objetos no cambie y el algoritmo se atasque.
Para solventarlo, Bertsekas propone un proceso similar al método húngaro donde se quiere encontrar un camino incremental. Si es así, se encuentra un camino incremental que tiene costo reducido cero desde i hasta una columna j no asignada y, luego, se obtiene una asignación más intercambiando aristas asignadas y no asignadas a lo largo del camino (en consecuencia, se actualizan las variables duales). Si no se encuentra el camino incremental, se actualizan las variables duales determinado el costo reducido mínimo δ y se asigna la fila i a la columna $\phi(i)$, teniendo que entrar la fila k en NV.
De hecho, Bertsekas también consideró un algoritmo subasta-húngaro que combina los dos métodos, donde comienza aplicando el algoritmo de subasta

y, cuando las iteraciones no conducen a un incremento de objetos asignados a los clientes, cambia al algortimo húngaro.
Por último, Bertsekas y Eckstein, para solventar los inconvenientes generados por el caso 2, formularon un algoritmo de subasta de una versión relajada de las condiciones de holguras complementarias, el cual tiene una complejidad polinomial y, por ello, vamos a verlo en detalle a continuación.

2.3.2. Algoritmo de subasta polinomial

Bertsekas y Eckstein introducen un mecanismo de perturbación para poder salir del ciclo infinito generado en el caso 2 del algoritmo de subasta ingenuo. El algoritmo opera usando un concepto denominado *ϵ-scaling*. En definitiva, el algoritmo se aplica en repetidas ocasiones, comenzando con un valor de ϵ suficientemente grande y reduciendo iterativamente este valor hasta que se logre el valor crítico que garantice la optimalidad de la solución encontrada.

Un par $(x,(u,v))$ de una solución primal y dual satisface la *ϵ-holguras complementarias* si, dado un valor positivo ϵ, tenemos que:

$$x_{ij}(c_{ij} - u_i - v_j) \leq \epsilon \quad (i,j = 1,2,\ldots,n) \tag{2.5}$$

En otras palabras, asignamos la fila i a la columna j si el costo reducido correspondiente no excede de cero más de ϵ.

Consideremos un par primal-dual factible $(x,(u,v))$ que satisface 2.5 y tenemos que ϕ almacena la solución primal. Si sumamos todos los pares (i,j) obtenemos:

$$\sum_{i=1}^{n} c_{i\phi(i)} - \left(\sum_{i=1}^{n} u_i + \sum_{i=1}^{n} v_i\right) \leq n\epsilon \tag{2.6}$$

Es decir, la diferencia entre el valor de la solución primal y el de la solución dual está acotado por $n\epsilon$. Por tanto, podemos enunciar la siguiente proposición.

Proposición 2.1. *Si los costos c_{ij} son enteros y un par primal-dual factible satisface 2.5 para $\epsilon < 1/n$, entonces la solución es óptima para el problema de asignación lineal.*

La idea es la siguiente: se define el problema ϵ-relajado como el problema de encontrar una solución que satisfaga 2.5. Entonces, se modifica el algoritmo de subasta para, inicialmente, resolver el problema ϵ-relajado para un valor de ϵ grande. Luego, se disminuye el valor de ϵ y se vuelve a optimizar la solución, iterando hasta que $\epsilon < 1/n$.
Bertsekas y Eckstein implementan este enfoque **multiplicando todos los costos por n+1** y resolviendo una secuencia de problemas ϵ-relajados con valores

de ϵ enteros que disminuyen desde un valor inicial grande hasta uno. Si un par primal-dual $(x,(u,v))$ es óptimo para este problema relajado, entonces el par $(x,(\overline{u},\overline{v}))$, con $\overline{u}_i = u_i/(n+1)\ \forall i$ y $\overline{v}_j = v_j/(n+1)\ \forall j$, satisface $(1/(n+1))$-holguras complementarias para el problema original. Por tanto, se aplica la proposición 2.1.
Cada subproblema se resuelve mediante un algoritmo similar al algoritmo de subasta ingenuo, que alterna una fase de oferta y una fase de asginación. En este caso, la fase oferta se modifica de modo que esta siempre incremente el precio de los objetos. Por tanto, el algoritmo se puede describir en los siguientes pasos:

1. **Inicialización.** Comenzamos con un ϵ suficientemente grande. Consideremos la asignación inicial vacía $\phi = \emptyset$ y un vector de precio v que satisfacen la condición ϵ-holguras complementarias.

2. **Fase de oferta.** Se selecciona una fila no asignada i y la oferta se determina como:
$$v_{\phi(i)} - (u_i^s - u_i) - \epsilon \tag{2.7}$$
donde $u_i, \phi(i), u_i^s$ se obtienen al igual que en 2.1, 2.3 y 2.4, respectivamente.

3. **Fase de asignación.** La fila i es asignada a la columna $\phi(i)$ y la correspondiente variable dual se actualiza estableciendo $v_{\phi(i)} = v_{\phi(i)} - (u_i^s - u_i) - \epsilon$. Además, si la columna $\phi(i)$ ya estaba asignada a una fila, dicha fila queda sin asignar.
Si todos los clientes i han sido asignados, es decir, $NV = \emptyset$, entonces pasamos al paso 4. De lo contrario, volvemos al paso 2.

4. **Comprobación de optimalidad.** Si $\epsilon > 1$, entonces se disminuye el valor de ϵ y se vuelve a realizar otra iteración para resolver otro subproblema.
Si $\epsilon = 1$, entonces hemos llegado a la solución óptima y termina el algoritmo.

Complejidad computacional del algoritmo de subasta

El algoritmo de subasta, propuesto por Bertsekas, es de complejidad pseudopolinomial y asciende a $O(n^3 + n^2 R)$ donde $R = máx\{c_{ij} : 1 \leq i, j \leq n\}$. En el caso del algoritmo combinado subasta-húngaro, se puede demostrar que es de complejidad $O(n^3)$.
Para el algoritmo de subasta con *ϵ-scaling*, propuesto por Bertsekas y Eckstein, tenemos que la disminución de ϵ es tal que, como máximo, se resuelven $O(log(nR))$ problemas ϵ-relajados. Entonces, la complejidad total es $O(nm\, log(nR))$, siendo m el número de aristas en un grafo.

2.3.3. Resolución de un ejemplo mediante el algoritmo de subasta

Vamos a ver un sencillo ejemplo donde poder aplicar el algoritmo de subasta de Bertsekas y Eckstein y, así, entender mejor su funcionamiento.
Ejemplo. Una empresa de transportes tiene tres modelos diferentes de camiones. Dependiendo de la pericia y el estilo de pilotaje del conductor, el camión consume más o menos combustible. En la actualidad, la empresa cuenta con tres conductores. Los costos (en euros/hora) por el uso adicional de combustible se muestran en la siguiente tabla adjunta. El objetivo es minimizar el costo total y que cada conductor sea adjudicado a un solo camión.

	Camión 1	**Camión 2**	**Camión 3**
Conductor 1	7	7	8
Conductor 2	2	8	5
Conductor 3	3	6	7

Multiplicamos todos los costos por $n + 1 = 4$ para obtener la matriz escalada.

	Camión 1	**Camión 2**	**Camión 3**
Conductor 1	28	28	32
Conductor 2	8	32	20
Conductor 3	12	24	28

- **1. Inicialización.** Comenzamos con un ϵ suficientemente grande ($\epsilon = 4$). Consideremos la asignación inicial vacía ($\phi = \emptyset$) y un vector de precio $v = (0, 0, 0)$ que satisfacen la condición ϵ-holguras complementarias.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 1$ y se calcula la oferta mediante 2.7.
  $i = 1$, $(c_{ij} - v_j) = (28, 28, 32) - (0, 0, 0) = (28, 28, 32)$

- **3. Fase de asignación.** El conductor $i = 1$ se asignada a la columna $\phi(1) = 1$ y se actualiza la correspondiente variable dual.
  $i = 1$, $\phi = (1, -, -)$ y $v = (-4, 0, 0)$.
  No se han asignado todos los conductores ($NV \neq \emptyset$), entonces volvemos al paso 2.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 2$ y se calcula la oferta mediante 2.7.

$i = 2$, $(c_{ij} - v_j) = (8, 32, 20) - (-4, 0, 0) = (12, 32, 20)$

- **3. Fase de asignación.** El conductor $i = 2$ es asignado a la columna $\phi(2) = 1$ y se actualiza la correspondiente variable dual.
  $i = 2$, $\phi = (-, 1, -)$ y $v = (-16, 0, 0)$.
  Además, la columna 1 ya estaba asignada a la fila 1, entonces la fila 1 queda sin asignar.
  No se han asignado todos los conductores ($NV \neq \emptyset$), entonces volvemos al paso 2.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 3$ y se calcula la oferta mediante 2.7.
  $i = 3$, $(c_{ij} - v_j) = (12, 24, 28) - (-16, 0, 0) = (28, 24, 28)$

- **3. Fase de asignación.** El conductor $i = 3$ se asignada a la columna $\phi(3) = 2$ y se actualiza la correspondiente variable dual.
  $i = 3$, $\phi = (-, 1, 2)$ y $v = (-16, -8, 0)$.
  No se han asignado todos los conductores ($NV \neq \emptyset$), entonces volvemos al paso 2.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 1$ y se calcula la oferta mediante 2.7.
  $i = 1$, $(c_{ij} - v_j) = (28, 28, 32) - (16, -8, 0) = (44, 36, 32)$

- **3. Fase de asignación.** El conductor $i = 1$ se asignada a la columna $\phi(1) = 3$ y se actualiza la correspondiente variable dual.
  $i = 1$, $\phi = (3, 1, 2)$ y $v = (-16, -8, -8)$.
  Todos los conductores i han sido asignados ($NV = \emptyset$), entonces pasamos al paso 4.

- **4. Comprobación de optimalidad.** Como $\epsilon = 4 > 1$, entonces se disminuye el valor de ϵ y se vuelve a realizar otra iteración para resolver otro subproblema.

Ahora, procedemos a disminuir ϵ y a resolver otro subproblema ϵ-relajado mediante el mismo algoritmo iterativo.

- **1. Inicialización.** Comenzamos con $\epsilon = 1$ (valor final). Consideremos la asignación inicial vacía ($\phi = \emptyset$) y el vector de precios del subproblema anterior $v = (-16, -8, -8)$ que satisfacen la condición ϵ-holguras complementarias.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 1$ y se calcula la oferta mediante 2.7.
  $i = 1$, $(c_{ij} - v_j) = (28, 28, 32) - (-16, -8, -8) = (44, 36, 40)$

- **3. Fase de asignación.** El conductor $i = 1$ se asignada a la columna $\phi(1) = 2$ y se actualiza la correspondiente variable dual.
  $i = 1$, $\phi = (2, -, -)$ y $v = (-16, -13, -8)$.
  No se han asignado todos los conductores ($NV \neq \emptyset$), entonces volvemos al paso 2.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 2$ y se calcula la oferta mediante 2.7.
  $i = 2$, $(c_{ij} - v_j) = (8, 32, 20) - (-16, -13, -8) = (24, 45, 28)$

- **3. Fase de asignación.** El conductor $i = 2$ es asignado a la columna $\phi(2) = 1$ y se actualiza la correspondiente variable dual.
  $i = 2$, $\phi = (2, 1, -)$ y $v = (-21, -13, -8)$.
  No se han asignado todos los conductores ($NV \neq \emptyset$), entonces volvemos al paso 2.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 3$ y se calcula la oferta mediante 2.7.
  $i = 3$, $(c_{ij} - v_j) = (12, 24, 28) - (-21, -13, -8) = (33, 37, 36)$

- **3. Fase de asignación.** El conductor $i = 3$ se asignada a la columna $\phi(3) = 1$ y se actualiza la correspondiente variable dual.
  $i = 3$, $\phi = (2, -, 1)$ y $v = (-25, -13, -8)$.
  Además, la columna 1 ya estaba asignada a la fila 2, entonces la fila 2 queda sin asignar.
  No se han asignado todos los conductores ($NV \neq \emptyset$), entonces volvemos al paso 2.

- **2. Fase de oferta.** Se selecciona el conductor no asignado $i = 2$ y se calcula la oferta mediante 2.7.
  $i = 2$, $(c_{ij} - v_j) = (8, 32, 20) - (-25, -13, -8) = (33, 45, 28)$

- **3. Fase de asignación.** El conductor $i = 2$ es asignado a la columna $\phi(2) = 3$ y se actualiza la correspondiente variable dual.
  $i = 2$, $\phi = (2, 3, 1)$ y $v = (-25, -13, -14)$.
  Todos los conductores i han sido asignados ($NV = \emptyset$), entonces pasamos al paso 4.

- **4. Comprobación de optimalidad.** Como $\epsilon = 1$, entonces hemos llegado a la solución óptima y termina el algoritmo.

Interpretando los resultados tenemos que: el conductor 1 va a conducir el camión 2, el conductor 2 va a conducir el camión 3 y el conductor 3 va a conducir el camión 1.

2.4. El método Bradford de Mack

El método de asignación lineal propuesto por Mack en 1969 es fácil de entender y fácil de aplicar. Este método tiene algunas similitudes con el algoritmo de Tomizawa (1971) y con el método húngaro. El algoritmo de Mack se basa en dos observaciones triviales para el problema de asignación lineal:

1. La matriz de costos se puede reducir sin influir en la solución óptima.
2. Se encuentra una solución óptima si, para una cierta matriz de costos reducidos, los mínimos de cada fila ocurren en columnas diferentes.

Siempre que 2 no se cumpla, el algoritmo de Mack ajusta los costos reducidos de tal manera que los mínimos de las filas se distribuyen en más columnas. Veamos los pasos del algoritmo de Mack.

2.4.1. Pasos del algoritmo de Mack

Antes de comenzar, vamos a explicar la siguiente notación que usaremos durante los pasos del algoritmo de Mack: llamaremos la base de la fila i, denotada como $base_i$, al mínimo costo reducido actual de cada columna para una cierta fila i.

1. Inicialización: Determinar las bases en la matriz de costos. Marcamos cada base con un asterisco encima.

2. Terminación: si cada columna tiene una base el algoritmo termina.
   En caso contrario, pasar al 3.

3. Seleccionar la columna j que contenga más de una base.
   Sea $COL = \{j\}$ el conjunto con el índice de la columna seleccionada y sea $FIL = \{i \mid base_i \in COL\}$ el conjunto con los índices de las filas correspondientes a las bases de la columna seleccionada.

4. Para cada $i \in FIL$ establecemos $m_i = min\{\overline{c}_{ik} - \overline{c}_{i,base_i} \mid k \notin COL\}$.
   Determinar $\delta = min\{m_i \mid i \in FIL\}$.
   Sea rr la fila y kk la columna donde se obtuvo delta.

5. Ajuste de la solución dual: incrementar δ a todos los costos reducidos de todas las columnas pertenecientes a COL.
   Si la columna kk no contiene una base, pasar al 6.
   De lo contrario, pasar al 7.

6. Incremento: se ha encontrado un camino que consta de, alternativamente, bases y bases alternativas, comenzando en la columna kk:
   Alterar el conjunto actual de bases a lo largo del camino alterno.
   Ir al 2.

7. La columna kk es base para alguna(s) fila(s):
   Marcar la columna kk como base alternativa para la fila rr.
   Ahora, $COL = COL \cup \{kk\}$ y $FIL = FIL \cup \{i \mid base_i = kk\}$.
   Ir al 4.

2.4.2. Mejora en la complejidad computacional del algoritmo de Mack

En este apartado se ofrece una alternativa para mejorar la eficiencia del 4 que requiere $O(n^2)$ operaciones y, así, mejorar la complejidad del método.

- **Paso 4.** Para $k \in COL$ tenemos $mm_k = min\{\overline{c}_{ik} - \overline{c}_{i,base_i} \mid i \in FIL\}$.
  Determinar $\delta = min\{mm_k \mid k \in COL$.
  Sea rr la fila y kk la columna donde se obtuvo delta.

Sea $\delta - sum$ la suma de todos los valores δ por los cuales las columnas en COL se incrementaron en aplicaciones anteriores del paso 5 durante la iteración actual, observamos que:

$$mm_k + \delta - sum = d_k \quad (k = 1, \ldots, n)$$

Por tanto, actualizar los mínimos de las filas equivale a determinar el camino más corto desde la columna original j (del paso 2) hasta cualquier columna desocupada en una red auxiliar. Podemos usar el método de Dijkstra siempre y cuando consideremos los costos reducidos no negativos. Entonces, en esta formulación, los mínimos de las columnas sobre las filas pertenecientes a FIL se pueden actualizar en $O(n)$ operaciones. Esta alternativa del paso 4 produce un método que tiene ciertas similitudes con el algoritmo de Tomizawa (1971).

Complejidad computacional del Método Bradford de Mack

El algoritmo de Mack, en su formulación original, resuelve el problema de asignación lineal en una complejidad $O(n^4)$. Esto se debe a que el 4 requiere

$O(n^2)$ operaciones para actualizar los mínimos de las filas sobre las columnas que no están en COL.
En la mejora que se propone para el algoritmo de Mack, los mínimos de las columnas sobre las filas en FIL se pueden actualizar en $O(n)$ operaciones, mejorando la complejidad computacional del algoritmo de Mack a $O(n^3)$.

2.4.3. Resolución de un ejemplo mediante el algoritmo de Mack

Vamos a plantear un breve y sencillo problema de asignación para resolverlo usando el algoritmo de Mack.
Ejemplo. Una empresa compra cuatro tipos de impresoras distintas, de inyección de tinta, térmica, láser y led. Las impresoras se deben asignar a los departamentos de recursos humanos, facturación, marketing y dirección. Debido a la frecuencia de uso en cada departamento y al tipo de impresora hay un costo aproximado en la asignación que se adjunta en la siguiente tabla. El objetivo es minimizar el costo total.

	R. Humanos	**Facturación**	**Marketing**	**Dirección**
Inyección	3	7	6	6
Térmica	1	6	8	8
Láser	3	0	8	1
Led	0	7	9	9

- **Paso 1.** Inicialización: Determinamos las bases en la matriz de costos. Marcamos las bases con un asterisco.

	R. Humanos	**Facturación**	**Marketing**	**Dirección**
Inyección	3^*	7	6	6
Térmica	1^*	6	8	8
Láser	3	0^*	8	1
Led	0^*	7	9	9

- **Paso 2.** No tenemos una base en cada columna, por lo tanto, el algoritmo no termina.

- **Paso 3.** Seleccionamos la primera columna que contiene más de una base, en concreto tiene tres.
  Sea $COL = \{j\} = \{1\}$ y sea $FIL = \{i = \{1, 2, 4\} \mid base_i \in COL\}$.

- **Paso 4.** Para cada $i = \{1, 2, 4\} \in FIL$ establecemos $m_i = min\{\overline{c}_{ik} - \overline{c}_{i,base_i} \mid k \notin COL\}$.
  $m_1 = min\{\overline{c}_{1k} - \overline{c}_{1,base_1} \mid k \notin COL\} = min\{7-3,\ 6-3,\ 6-3\} = 3$
  $m_2 = min\{\overline{c}_{2k} - \overline{c}_{2,base_2} \mid k \notin COL\} = min\{6-1,\ 8-1,\ 8-1\} = 5$
  $m_4 = min\{\overline{c}_{4k} - \overline{c}_{4,base_4} \mid k \notin COL\} = min\{7-0,\ 9-0,\ 9-0\} = 7$
  Determinar $\delta = min\{m_i \mid i \in FIL\} = min\{3, 5, 7\} = 3$.
  δ se obtiene en la fila 1 y columna 3.

- **Paso 5.** Ajustamos la solución dual: incrementamos $\delta = 3$ a todos los costos reducidos de la primera columna perteneciente a COL.

	R. Humanos	**Facturación**	**Marketing**	**Dirección**
Inyección	6^*	7	6	6
Térmica	4^*	6	8	8
Láser	6	0^*	8	1
Led	3^*	7	9	9

  La columna 3 no contiene una base, pasamos al **paso 6**.

- **Paso 6.** Al incrementar las entradas de la columna 1 creamos una posición alternativa para una de las bases en esta columna, en concreto, se encuentra una posición alternativa para $base_1$ de la columna 1 a la columna 3, usando un camino incremental de longitud dos.

	R. Humanos	**Facturación**	**Marketing**	**Dirección**
Inyección	6	7	6^*	6
Térmica	4^*	6	8	8
Láser	6	0^*	8	1
Led	3^*	7	9	9

  Vamos al **paso 2**.

- **Paso 2.** No tenemos una base en cada columna, por lo tanto, el algoritmo no termina.

- **Paso 3.** Seleccionamos la primera columna que contiene más de una base, en concreto tiene dos.
  Sea $COL = \{j\} = \{1\}$ y sea $FIL = \{i = \{2, 4\} \mid base_i \in COL\}$.

- **Paso 4.** Para cada $i = \{2, 4\} \in FIL$ establecemos $m_i = min\{\overline{c}_{ik} - \overline{c}_{i,base_i} \mid k \notin COL\}$.
  $m_2 = min\{\overline{c}_{2k} - \overline{c}_{2,base_2} \mid k \notin COL\} = min\{6-4,\ 8-4,\ 8-4\} = 2$
  $m_4 = min\{\overline{c}_{4k} - \overline{c}_{4,base_4} \mid k \notin COL\} = min\{7-3,\ 9-3,\ 9-3\} = 4$
  Determinar $\delta = min\{m_i \mid i \in FIL\} = min\{2, 4\} = 2$.
  δ se obtiene en la fila 2 y columna 2.

- **Paso 5.** Ajustamos la solución dual: incrementamos $\delta = 2$ a todos los costos reducidos de la primera columna perteneciente a COL.

	R. Humanos	**Facturación**	**Marketing**	**Dirección**
Inyección	8	7	6^*	6
Térmica	6^*	6	8	8
Láser	8	0^*	8	1
Led	5^*	7	9	9

  La columna 2 ya contiene una base, pasamos al **paso 7**.

- **Paso 7.** La columna 2 es base para la fila 3:
  Marcamos la columna 2 como base alternativa para la fila 2.
  Ahora, $COL = COL \cup \{2\}$ y $FIL = FIL \cup \{i = \{3\} \mid base_3 = 2\}$.
  Vamos al **paso 4**.

- **Paso 4**. Para cada $i = \{2, 3, 4\} \in FIL$ establecemos $m_i = min\{\overline{c}_{ik} - \overline{c}_{i,base_i} \mid k \notin COL\}$.
  $m_2 = min\{\overline{c}_{2k} - \overline{c}_{2,base_2} \mid k \notin COL\} = min\{8-6,\ 8-6\} = 2$
  $m_3 = min\{\overline{c}_{3k} - \overline{c}_{3,base_3} \mid k \notin COL\} = min\{8-0,\ 1-0\} = 1$
  $m_4 = min\{\overline{c}_{4k} - \overline{c}_{4,base_4} \mid k \notin COL\} = min\{9-5,\ 9-5\} = 4$
  Determinar $\delta = min\{m_i \mid i \in FIL\} = min\{2, 1, 4\} = 1$.
  δ se obtiene en la fila 3 y columna 4.

- **Paso 5.** Ajustamos la solución dual: incrementamos $\delta = 1$ a todos los costos reducidos de la primera y la segunda columna pertenecientes a COL.

	R. Humanos	**Facturación**	**Marketing**	**Dirección**
Inyección	9	8	6^*	6
Térmica	7^*	7	8	8
Láser	9	1^*	8	1
Led	6^*	8	9	9

La columna 4 no contiene una base, pasamos al **paso 6**.

- **Paso 6.** Existe un camino, alternando entre bases y bases alternativas, a lo largo del cual el conjunto actual de bases se puede distribuir en una columna más.

	R. Humanos	**Facturación**	**Marketing**	**Dirección**
Inyección	9	8	6^*	6
Térmica	7	7^*	8	8
Láser	9	1	8	1^*
Led	6^*	8	9	9

Vamos al **paso 2**.

- **Paso 2.** Terminación: cada columna tiene una base, por tanto, el algoritmo termina.
  Interpretando la última tabla tenemos que: la impresora de inyección va destinada al departamento de marketing. La impresora térmica va destinada al departamento de facturación. La impresora láser va destinada al departamento de dirección. La impresora led va destinada al departamento de recursos humanos.
  Con esta asignación el coste total (óptimo) es: $6 + 6 + 1 + 0 = 13$.

3

Variantes del problema de Asignación

En este tercer capítulo, empezaremos describiendo variantes aplicadas al problema de asignación lineal. Luego, veremos el planteamiento y formulación matemática de algunas variantes más complejas, como puede ser el problema de asignación de cuello de botella lineal, el problema de asignación cuadrática o el problema de asignación de índices múltiples.

3.1. Variantes del problema de asignación lineal

En esta sección veremos pequeñas modificaciones del problema de asignación lineal y como afrontarlas para su resolución.

3.1.1. Problema de asignación lineal con asignaciones inaceptables

En esta variante del problema de asignación lineal a algún o algunos orígenes o destinos no se les puede asignar un costo. Trasladado a la práctica, podemos interpretarlo como que el trabajador i no está capacitado para realizar el trabajo j.

Para solucionar este inconveniente, se asigna un costo arbitrariamente grande que representaremos mediante la letra M. M es un número tan grande que si se le resta un número finito cualquiera, sigue quedando un valor mayor que los demás. De esta manera, evitamos que resulte asignado. Ahora podemos resolver este problema con cualquiera de los métodos de resolución expuestos en el capítulo 2. Un posible ejemplo de la tabla de costos se muestra a continuación:

	D_1	D_2	D_3	D_4
O_1	c_{11}	c_{12}	c_{13}	c_{14}
O_2	M	c_{22}	c_{23}	c_{24}
O_3	c_{31}	c_{32}	c_{33}	c_{34}
O_4	c_{41}	c_{42}	M	c_{44}

3.1.2. Problema de asignación lineal asimétrico

En esta variante, la cardinalidad del conjunto de orígenes es menor que la cardinalidad del conjunto de destinos, es decir, tenemos una matriz de costos rectangular tal que $n \leq m$. Entonces, tenemos que asignar n filas a n columnas diferentes de modo que la suma de los costos sea mínima, dejando $m-n$ columnas sin asignar.

Para solucionar este problema, añadimos $m-n$ filas ficticias con costos igual a 0 (en la práctica estas asignaciones no se realizan) de tal manera que tengamos un problema de asignación lineal con matriz de costos cuadrada. Ahora podemos resolver este problema con cualquiera de los métodos de resolución expuestos en el capítulo 2. Un posible ejemplo de la tabla de costos se muestra a continuación:

	D_1	D_2	D_3	D_4	D_5
O_1	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}
O_2	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}
O_3	c_{31}	c_{32}	c_{33}	c_{34}	c_{35}
O_4	0	0	0	0	0
O_5	0	0	0	0	0

3.1.3. Problema de semiasignación lineal

En esta variante, tenemos una matriz de costos $n \times m$, donde $n \geq m$, y un vector de demanda positivo b de m elementos. El problema de semiasignación pide que se seleccione un elemento por fila para que el número de elementos en cada columna j sea b_j y la suma de los elementos seleccionados sea un mínimo. En términos de formulación matemática tenemos que:

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \tag{3.1}$$

sujeto a:

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i = 1, 2, \ldots, n \tag{3.2}$$

$$\sum_{i=1}^{n} x_{ij} = b_j, \quad \forall j = 1, 2, \ldots, m \tag{3.3}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \ldots, n \tag{3.4}$$

Hay que tener en cuenta que el problema tiene una solución solo si $\sum_{j=1}^{m} b_j = n$, para todo j. Barr, Glover y Klingman dieron una adaptación de su algoritmo de base alterna al problema de semiasignación.

3.2. Problema de asignación de cuello de botella lineal

Los problemas de asignación de cuello de botella lineales fueron introducidos por Fulkerson, Glicksberg y Gross. Este tipo de problemas ocurren en casos como el siguiente donde se pretende minimizar un tiempo: supongamos que se van a asignar n trabajos a n máquinas (o trabajadores) de la mejor manera posible. El coeficiente de costos c_{ij} es el tiempo que necesita la máquina j en completar el trabajo i. El objetivo es minimizar el tiempo total hasta que se completen los trabajos. Si asumimos que las máquinas trabajan en paralelo y queremos asignar los trabajos a las máquinas de manera que el último tiempo en finalizar sea lo más temprano posible, obtenemos el problema de asignación de cuello de botella lineal:

$$\min_{\phi \in S_n} \max_{1 \leq i \leq n} c_{i\phi(i)} \tag{3.5}$$

Además, como ya vimos, si describimos las permutaciones mediante las correspondientes matrices de permutación $X = (x_{ij})$, podemos modelar el problema de cuello de botella lineal como:

$$min \max_{1 \leq i \leq n} c_{ij}\ x_{ij} \tag{3.6}$$

Sujeto a las restricciones de asignación 1.8, 1.9, 1.10.
Para resolver los problemas de asignación de cuello de botella lineales tenemos los algoritmos de umbral, un método dual y un método de camino incremental más corto que se pueden consultar en [5].

3.3. Problema de asignación cuadrática

Los problemas de asignación cuadrática pertenecen a los problemas de optimización combinatoria más difíciles. Debido a sus múltiples aplicaciones en el mundo real, muchos autores han investigado esta clase de problemas, entre ellos, destacamos a E. Çela y su libro [6].
El problema de asignación cuadrático fue presentado por Koopmans y Beckmann en 1957 como un modelo matemático para la localización de actividades económicas indivisibles. Como ejemplo, queremos asignar n instalaciones a n ubicaciones con un costo proporcional al flujo entre las instalaciones multiplicado por las distancias entre las ubicaciones más los costos de ubicar las instalaciones en sus respectivas ubicaciones. El objetivo es asignar cada instalación a una ubicación tal que se minimice el costo total. Podemos modelar este problema mediante tres matrices $n \times n$:
$A = a_{ik}$, donde a_{ik} es el flujo de la instalación i a la instalación k;
$B = b_{jl}$, donde b_{jl} es la distancia desde la ubicación j a la ubicación l;
$C = (c_{ij})$, donde c_{ij} es el costo de colocar la intalación i en la ubicación j;

Entonces, podemos escribir el problema de asignación cuadrática de Koopmans-Beckmann como:

$$\min_{\phi \in S_n} \left(\sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik}\ b_{\phi(i)\phi(k)} + \sum_{i=1}^{n} c_{i\phi(i)} \right) \tag{3.7}$$

donde cada producto individual $a_{ik}\ b_{\phi(i)\phi(k)}$ es el costo de transporte causado por la asignación de la instalación i a la ubicacion $\phi(i)$ y la intalación k a la ubicación $\phi(k)$. Por lo tanto, cada término $c_{i\phi(i)} + \sum_{k=1}^{n} a_{ik}\ b_{\phi(i)\phi(k)}$ es el coste total dado, para la instalación i, por el costo de instalarla en la ubicación $\phi(i)$, más los costos de transporte a todas las instalaciones k, si están instaladas en las ubicaciones $\phi(1), \phi(2), \ldots, \phi(n)$.
Cuando expresamos las permutaciones por matrices de permutación $X = (x_{ij})$, podemos modelar el problema de asignación cuadrático de Koopmans-Beckmann como un programa cuadrático entero de la forma:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} a_{ik}\ b_{jl}\ x_{ij}\ x_{kl} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\ x_{ij} \tag{3.8}$$

sujeto a a las restricciones de asignación 1.8, 1.9, 1.10.
Además de la ubicación de las instalaciones, los problemas de asignación cuadrática aparecen en una variedad de aplicaciones, como la fabricación de ordenadores, la porgramación, las comunicaciones de procesos, el balanceo de turbinas, etc. Uno de los casos especiales más destacados del problema de asignación cuadrático es el problema del viajante.
En cuanto a su resolución, hay una gran diferencia entre el problema de asignación lineal y el cuadrático. Mientras que para los problemas de asignación lineal existen métodos de resolución eficientes en tiempo polinomial (permitiendo resolver problemas para valores grandes de n), los problemas de asignación cuadrática son problemas NP-hard. Esto signfica que la solución óptima solo se puede encontrar mediante enumeración (implícita) de todas las posibilidades a menos que $P = NP$. Las instancias de referencia más grandes de problemas de asignación cuadrática para los que se ha determinado una solución óptima tienen un tamaño de alrededor de $n \approx 30$ en tiempos de ejecución elevados. Por esta razón, debido a la relevancia teórica y práctica del problema, se han propuesto muchas heurísticas constructivas y decenas de métodos de búsqueda locales que suministran una solución subóptima más o menos buena, es decir, se intenta encontrar soluciones de calidad en tiempos de cáculo cortos.

3.4. Problema de asignación de índices múltiples

Los problemas de asignación de índices múltiples fueron introducidos por Pierskalla en 1968 como una extensión del problema de asignación lineal. Durante mucho tiempo solo se han considerado problemas de asignación de 3 índices,

mientras que en los últimos años se han investigado problemas con más de 3 índices, principalmente en el contexto de problemas de asociación de datos y seguimiento de objetivos múltiples. En el caso de problemas de asignación de 3 índices vamos a mencionar dos modelos: el problema de asignación axial de 3 índices y el problema de asignación plano de 3 índices.

3.4.1. Problema de asignación axial de 3 índices

En un problema de horarios se requiere asignar n cursos a n franjas horarias y a n salas. Sea c_{ijk} el costo de asignar el curso i al horario j en la sala k, queremos encontrar una asignación ϕ de los cursos a las franjas horarias y una asignación ψ de los cursos a las salas tal que el costo total sea mínimo. Este ejemplo pertenece al llamado problema de asignación axial de 3 índices:

$$\min_{\phi,\psi \in S_n} \sum_{i=1}^{n} c_{i\phi(i)\psi(i)} \tag{3.9}$$

Dado que las dos permutaciones que describen una solución factible pueden elegirse arbitrariamente, el problema de asignación axial tiene $(n!)^2$ soluciones factibles. Podemos escribir este problema como un programa lineal entero:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk}\ x_{ijk} \tag{3.10}$$

sujeto a:

$$\sum_{j=1}^{n} \sum_{k=1}^{n} x_{ijk} = 1 \quad (i = 1, 2, \ldots, n) \tag{3.11}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{n} x_{ijk} = 1 \quad (j = 1, 2, \ldots, n) \tag{3.12}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijk} = 1 \quad (k = 1, 2, \ldots, n) \tag{3.13}$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j, k = 1, 2, \ldots, n) \tag{3.14}$$

Las aplicaciones de los problemas de asignación axial de 3 índices surgen en una gran cantidad de situaciones, por ejemplo, el ensamblaje de placas de un circuito impreso, la inversión de capital en diferentes ubicaciones durante un horizonte de tiempo o en un tren de laminación, como mencionaron Qi y Sun, donde los lingotes deben programarse a través de pozos de remojo (donde se estabiliza la temperatura) para minimizar el tiempo de inactividad del tren de laminación.

En contraste con el prolema de asignación lineal, Karp mostró que el problema de asignación axial de 3 índices es NP-hard. Crama y Spieksma incluso demostraron que ningún algoritmo de tiempo polinomial puede lograr una relación de rendimiento constante a menos que $P = NP$.

3.4.2. Problema de asignación plano de 3 índices

El problema de asignación plano de 3 índices se puede formular de la siguiente manera: decimos que n permutaciones $\phi_1, \phi_2, \ldots, \phi_n$ son mutuamente distintas si $\phi_r(i) \neq \phi_s(i)$ para cualquier $i = 1, 2, \ldots, n$ y $r \neq s$. Dados n^3 coeficientes de costos c_{ijk} $(i, j, k = 1, 2, \ldots, n)$, el problema es encontrar n permutaciones mutuamente distintas tales que

$$\sum_{k=1}^{n} \sum_{i=1}^{n} c_{i\phi(i)k} \tag{3.15}$$

sea un mínimo. Entonces, el corrrespondiente programa lineal entero es

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk}\ x_{ijk} \tag{3.16}$$

sujeto a:

$$\sum_{k=1}^{n} x_{ijk} = 1 \quad (i, j = 1, 2, \ldots, n) \tag{3.17}$$

$$\sum_{i=1}^{n} x_{ijk} = 1 \quad (j, k = 1, 2, \ldots, n) \tag{3.18}$$

$$\sum_{j=1}^{n} x_{ijk} = 1 \quad (i, k = 1, 2, \ldots, n) \tag{3.19}$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j, k = 1, 2, \ldots, n) \tag{3.20}$$

Los problemas de asignación planos de 3 índices tienen un papel importante en los problemas de horarios. Por ejemplo, consideremos un instituto en el que n profesores deben impartir clases a n grupos en n franjas horarias diferentes. Asumimos que cada profesor tiene que enseñar a cada grupo una sola vez. Una solución factible a este problema es una asignación de los profesores a los grupos por cada horario fijo y una asignación de las franjas horarias a los profesores por cada grupo fijo. En resumen, una solución factible se puede representar como un problema de asignación plano de 3 índices.
Frieze mostró que el problema es NP-hard. Este tipo de problema no se ha investigado tan a fondo como el problema de asignación axial de 3 índices, ya que es mucho más difícil de resolver.

4

Soporte computacional

En este capítulo presentaremos soporte computacional para la resolución del problema de asignación lineal. Explicaremos un código desarrollado en Python utilizando una librería de Google OR-Tools y un programa realizado en R.

4.1. Resolución del problema de asignación lineal

En esta sección vamos a utilizar Google Colaboratory, un entorno virtual gratuito que permite programar y ejecutar en Python y R desde nuestro navegador. También ofrece la posibilidad de acceder a multitud de cuadernos públicos compartidos por la comunidad para aprender y obtener ideas de otros usuarios.

4.1.1. Código en Python

Explicaremos el código creado en Python utilizando una librería de Google OR-Tools para resolver el ejemplo 1.1.1. Vamos a aprovechar este conjunto de librerías que ofrecen una amplia gama de herramientas para resolver diversos tipos de problemas de optimización, incluyendo el problema de asignación. Para ello, utilizaremos la resolución de MIP (Mixed Integer Programming) con SCIP (Solving Constraint Integer Programs) que resuelve problemas de programación matemática mixta entera. Podemos encontrar el código y ejecutarlo en el siguiente enlace a un cuaderno de Google Colaboratory `https://colab.research.google.com/drive/1Yj1kREVJKy9daAivgCMYavdu26_odzNK?usp=sharing`. Veamos los pasos de su construcción:

1. Instalamos la librería de Google OR - Tools en nuestro cuaderno de Google Colaboratory. Importamos la clase *pywraplp* del módulo *linear_solver* para modelos de programación lineal. Luego, declaramos la resolución de problemas MIP que abordará nuestro modelo.

```
!pip install ortools

from ortools.linear_solver import pywraplp
solver = pywraplp.Solver.CreateSolver('SCIP')

```

2. Añadimos los datos del problema. Escribimos la matriz de costos asociada a la tabla del ejemplo, y utilizamos la función *len* que cuenta el número de elementos de la matriz por filas y columnas para tener el número de equipos de mantenimiento y el número de máquinas, respectivamente. Creamos las variables binarias x_{ij} y las restricciones del problema de asignación, donde en primer lugar tenemos que cada equipo de mantenimiento tiene que ser asignado a una sola máquina y, en segundo lugar, tenemos que cada máquina tiene que ser asignado a un solo equipo de mantenimiento. Creamos la función objetivo con el criterio de optimización: **minimizar**. Por último, invocamos al agente de resolución.

```
costos = [
    [58, 58, 60, 54],
    [66, 70, 70, 78],
    [106, 104, 100, 95],
    [52, 54, 64, 54],
]
num_equipos = len(costos)
num_maquinas = len(costos[0])

x = {}
for i in range(num_equipos):
    for j in range(num_maquinas):
        x[i, j] = solver.IntVar(0, 1, '')

for i in range(num_equipos):
    solver.Add(solver.Sum([x[i, j] for j in range(num_maquinas)]) == 1)
for j in range(num_maquinas):
    solver.Add(solver.Sum([x[i, j] for i in range(num_equipos)]) == 1)

fun_objectivo = []
for i in range(num_equipos):
    for j in range(num_maquinas):
        fun_objectivo.append(costos[i][j] * x[i, j])
solver.Minimize(solver.Sum(fun_objectivo))

sol = solver.Solve()

```

3. Configuramos los parámetros de salida para imprimir la solución.

```
33  if sol == pywraplp.Solver.OPTIMAL or sol == pywraplp.Solver.FEASIBLE:
34      print('Costo óptimo total = ', solver.Objective().Value(), '\n')
35      for i in range(num_equipos):
36          for j in range(num_maquinas):
37              if x[i, j].solution_value() > 0.5:
38                  print('Equipo de mantenimiento %d asignado a la máquina %d. Costo = %d' %
39                        (i, j, costos[i][j]))
```

Al ejecutar el código se muestra la solución óptima. Este sencillo ejemplo se ha resuelto en menos de 1 segundo. Podemos observar que hemos llegado a la misma solución que en el capítulo 2, donde se usó el método hungaro 2.2.3.

```
Costo óptimo total =  274.0

Equipo de mantenimiento 0 asignado a la máquina 3. Costo = 54
Equipo de mantenimiento 1 asignado a la máquina 0. Costo = 66
Equipo de mantenimiento 2 asignado a la máquina 2. Costo = 100
Equipo de mantenimiento 3 asignado a la máquina 1. Costo = 54
```

4.1.2. Programa en R

En el cuaderno de Google Colaboratory, podemos cambiar el "entorno de ejecución" para trabajar en R. De la misma manera, se podría ejecutar, el programa que vamos a describir, utilizando la herramienta *Rstudio*.
Vamos a presentar un programa desarrollado en R donde se resuelve un problema de asignación lineal con una matriz de costos de dimensión 10×10. Debido a su gran dimensión, llevaría mucho tiempo resolver este ejemplo manualmente. Primero, cabe destacar que R alberga la librería "*lpSolve*", que se utiliza para resolver problemas de programación lineal. Esta se puede aplicar en el problema de asignación mediante la función "*lp.assign*", pero, como ya vimos en el capítulo 2, hay métodos de resolución más eficientes. Por ello, vamos a implementar en nuestro programa de R la librería "*clue*" para usar la función "*solve_LSAP*" (Linear Sum Assignment Problem), la cual resuelve el problema de asignación lineal utilizando el método húngaro. A continuación se muestra este programa desarrollado que conlleva alrededor de 7 segundos ejecutarlo. Además, los dos programas de R que resuelven este ejemplo utilizando las librerías mencionadas, lo podemos encontrar en el siguiente enlace a un cuaderno de Google Colaboratory https://colab.research.google.com/drive/1OIek_PTKnMcqGtKI8p4kOXlQG6boAoIN?usp=sharing.

```r
install.packages('clue')
library('clue')
# Matriz de costos
matrizcostos <- matrix(c(
  10, 5, 8, 7, 9, 6, 3, 4, 1, 2,
  2, 7, 6, 4, 1, 3, 5, 9, 8, 10,
  6, 3, 2, 1, 7, 5, 9, 10, 4, 8,
  8, 6, 9, 3, 5, 4, 7, 2, 13, 1,
  1, 4, 7, 9, 2, 8, 10, 6, 5, 3,
  5, 8, 3, 2, 12, 2, 4, 7, 6, 9,
  9, 10, 1, 6, 11, 7, 2, 5, 3, 4,
  4, 1, 5, 10, 3, 2, 6, 8, 9, 7,
  7, 9, 4, 8, 6, 10, 1, 3, 2, 5,
  3, 2, 10, 5, 4, 9, 8, 2, 7, 6
), nrow = 10, ncol = 10, byrow = TRUE)
# Resolver el problema de asignación lineal
assignment <- solve_LSAP(matrizcostos)
# Mostrar la asignación óptima
print(assignment)
# Mostrar el costo total óptimo
sum(matrizcostos[cbind(seq_along(assignment), assignment)])
```

Las líneas de comando que devuelve el programa son:

```
Optimal assignment:
1 => 9, 2 => 5, 3 => 4, 4 => 10, 5 => 1, 6 => 6, 7 => 3, 8 => 2,
9 => 7, 10 => 8
12
```

Interpretando los resultados, podemos observar que el programa asigna el trabajador 1 con la tarea 9, el trabajador 2 con la tarea 5, el trabajador 3 con la tarea 4, el trabajador 4 con la tarea 10, el trabajador 5 con la tarea 1, el trabajador 6 con la tarea 6, el trabajador 7 con la tarea 3, el trabajador 8 con la tarea 2, el trabajador 9 con la tarea 7 y el trabajador 10 con la tarea 8. El mínimo coste total es 12.

La importancia de esta sección es el dinamismo y la flexibilidad de su resolución. Podemos cambiar los valores de la tabla de costos asociados o su dimensión. También podemos hacer pequeñas modificaciones para maximizar el problema o ajustar pequeños cambios en las restricciones. Todo esto sin tener que reescribir todo el proceso de nuevo. Además, es destacable la rapidez computacional con la que se resuelve el problema.

Conclusiones

El objetivo principal de este trabajo ha sido hacer un estudio del problema asignación y sus métodos de resolución. Se han explicado los fundamentos conceptuales establecidos desde la década de 1920 y se han presentado en detalle los desarrollos teóricos, algorítmicos y prácticos del problema de asignación lineal. Se ha hecho mucho hincapié en tres técnicas algorítmicas de resolución del problema de asignación lineal (algoritmo húngaro, algoritmo de subasta y algoritmo de Mack), introduciendo sus características esenciales a través de una explicación intuitiva e ilustrando su ejecución mediante una serie de ejemplos numéricos completamente desarrollados. Además, se ha complementado el trabajo con soporte computacional, apoyándonos en distintos programas informáticos para resolver, eficientemente, problemas de asignación lineal de gran tamaño sin tener que realizar el proceso a "mano".

Se introducen las variantes del problema de asignación, mediante sus formulaciones matemáticas y aplicaciones. Por ello, podría ser de interés, investigar en detalle las variantes del problema de asignación para futuros proyectos, ya que tiene importantes aplicaciones en la actualidad y una gran repercusión en el área de optimización combinatoria. Por ejemplo, el problema de asignación cuadrática aún carece de métodos efectivos de solución exacta. Después de décadas de investigaciones, las instancias de tamaño 30 requieren años de CPU para resolverse de manera óptima.

Finalmente, tras contribuir a la compresión y resolución del problema de asignación, se puede concluir que, por sencillo que parezca, el problema de asignación tiene múltiples aplicaciones y numerosas variantes que forman un área increíblemente grande en la optimización.

Bibliografía

[1] Balinski, M.L. y Russakoff, A. On the assignment polytope. *SIAM Review*, 1974, vol. 16, n° 4, pp. 516-525.

[2] Bertsekas, D.P. The auction algorithm: A distributed relaxation method for the assignment problem. En Meyer, R.R. y Zenios, S.A. (editores) *Parallel Optimization on Novel Computer Architectures*, vol. 14, Baltzer, Basel, 1988, pp. 105-123.

[3] Bertsekas, D.P. y Eckstein, J. Dual coordinate step methods for linear network flow problems. *Mathematical Programming* 42, 1988, pp. 203-243.

[4] Burkard, R.E. y Çela, E. Linear assignment problems and extensions. En Du, D.Z. y Pardalos, P.M. (editores) *Handbook of Combinatorial Optimization*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999, pp. 75-149.

[5] Burkard, R.E., Dell'Amico, M. y Martello, S. *Assignment Problems*. SIAM, Philadelphia, 2009.

[6] Çela, E. *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[7] *Hungarian algorithm* [en línea]. Wikipedia. Disponible en: https://en.wikipedia.org/wiki/Hungarian_algorithm.

[8] Jonker R. y Volgenant. A.T. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38, 1987, pp. 325-340.

[9] Jonker R. y Volgenant. A.T. Teaching linear assignment by Mack's algorithm. En Lenstra, J.K. Tijms, H. y Volgenant, A.T. (editores) *Twenty-Five Years of Operations Research in the Netherlands: Papers Dedicated to Gijs de Leve*, vol. 70 of CWI Tract, Centre for Mathematics and Computer Science, Amsterdam, 1989, pp. 54-60.

[10] Schrijver, A. On the history of combinatorial optimization (till 1960). En Aardal, K. Nemhauser, G.L. y Weismantel, R. (editores) *Discrete Optimization*, vol.12 of *Handbooks in Operations Research and Management Science*, Elsevier, Amsterdam, 2005, pp. 1-68.

[11] Schwartz, B.L. A computational analysis of the auction algorithm. *European Journal of Operations Research* vol 74, 1994, pp. 161-169.

Assignment problem and variants

Sven Bode Divassón
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu0101110172@ull.edu.es

Abstract

In this work we study the assignment problem. We take a tour of the concepts and properties that enable the development of procedures for their resolution, in the necessary proximity to Graph Theory and Linear Programming. Among all the methods developed to solve the Linear sum assignment problem, we highlight three algorithms: Hungarian, Auction and Mack. We detail its steps and are accompanied by application examples.

1. Introduction

Let n origins be assigned to n destinations. We have a cost matrix $C = c_{ij}$ of dimension $n \times n$ where each c_{ij} is the cost of assigning each Origin $O_i,\ i = 1, \ldots, n$ to each Destination $D_j,\ j = 1, \ldots, n$.

	D_1	D_2	$\ldots$	D_n
O_1	c_{11}	c_{12}	$\ldots$	c_{1n}
O_2	c_{21}	c_{22}	$\ldots$	c_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
O_n	c_{n1}	c_{n2}	$\ldots$	c_{nn}

We intend to select n items from C such that there is exactly one item in each row and one item in each column such that the sum of the costs is a minimum.

We define the matrix $X = (x_{ij})$ formed by the binary variables:

$$x_{ij} = \begin{cases} 1, & if\ origin\ i\ is\ assigned\ to\ destination\ j \\ 0, & otherwise \end{cases}$$

We define the linear model of the assignment problem:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1, 2, \ldots, n$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \ldots, n$$

The matrix corresponding to the constrains is totally unimodular. Then, the integer linear problem has **integer optimal solution**. Thus, we can relax the conditions of the assignment problem, i.e., $x_{ij} \geq 0\ (i, j = 1, 2, \ldots, n)$. Therefore, we can formulate the corresponding dual problem.

$$max \sum_{i=1}^{n} u_i \sum_{j=1}^{n} v_j$$

s.t.

$$u_i + v_j \ \leq c_{ij} \quad i, j = 1, \ldots, n$$

where $u_i, v_j \in \mathbb{R}\ (i, j = 1, \ldots, n)$ are the dual variables associated with the constraints of the primal problem.
By the *strong duality theorem*, both problems have an optimal solution and provide the same optimal value. Applying *complementary slackness* to the assignment problem, must be fulfilled: $\overline{x}_{ij}\overline{c}_{ij} = 0\ \forall\, i, j = 1, \ldots, n$, where $\overline{c}_{ij} = c_{ij} - u_i - v_j$ are the *reduced costs*.

2. Algorithms

The **Hungarian method** is a primal-dual algorithm. Starting with a dual feasible solution, forcing complementary slack conditions, it tries to find a primal feasible solution. In the different iterations, it handles"partiall" primal feasible solutions until it finds one that is primal feasible. At this point, the primal solution is also optimal, according to the duality theory.
The algorithm transforms the cost matrix, always keeping the non-negative costs and, thus, the dual feasibility, until obtaining a sufficient number of zeros to be able to assign a zero in each column and each row of the matrix, and have an optimal solution.

Bertsekas and Eckstein's auction algorithm multiply all costs by $n+1$ and solve a sequence of ϵ-relaxed problems with integer ϵ values decreasing from a large initial value to one. Each ϵ-relaxed problem is defined as the problem of finding a solution that satisfies the ϵ-complementary slackness ($x_{ij}(c_{ij} - u_i - v_j) \leq \epsilon$) and is solved by an iterative algorithm, alternating an bidding phase and an assignment phase. The bidding phase always increments the dual variable v_j and the assignment phase assigns the customers to the items, updating the dual variables and thus preserving the dual feasibility. The algorithm ends when all are assigned.
If a primal-dual pair $(x, (u, v))$ is optimal for this relaxed problem, then the pair $(x, (\overline{u}, \overline{v}))$, with $\overline{u}_i = u_i/(n+1)\ \forall i$ and $\overline{v}_j = v_j/(n+1)\ \forall j$, satisfies $(1/(n+1))$-complementary slackness for the original problem. Therefore, the solution is optimal for the linear assignment problem.

Mack's algorithm is based on two trivial observations for the linear assignment problem:

1. The cost matrix can be reduced without influencing the optimal solution.
2. An optimal solution is found if, for a certain reduced cost matrix, the row minima occur in different columns.

As long as 2 is not fulfilled, Mack's algorithm adjusts the reduced costs in such a way that the row minima are spread over more columns. Construct the auxiliary network and determine an alternating path of minimal total reduced cost, and use it to augment the solution. Then, adjust the dual solution to restore complementary slackness and go to step 2.

3. Variants and computational support

We introduce some variants of the assignment problem and its mathematical formulation, as well as its applications. Finally, we provide computational support to solve the linear sum assignment problem with the help of various computer programs.

References

[1] Burkard, R.E., Dell'Amico, M. y Martello, S. *Assignment Problems*. SIAM, Philadelphia, 2009.

[2] Burkard, R.E. y Çela, E. Linear assignment problems and extensions. En Du, D.Z. y Pardalos, P.M. (editores) *Handbook of Combinatorial Optimization*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999, pp. 75-149.

[3] Alexander Schrijver. On the history of combinatorial optimization (till 1960). En Aardal, K. Nemhauser, G.L. y Weismantel, R. (editores) *Discrete Optimization*, vol.12 of *Handbooks in Operations Research and Management Science*, Elsevier, Amsterdam, 2005, pp. 1-68.