Estudio del Problema del Matrimonio Estable: Presentación del Algoritmo de Gale-Shapley, Análisis de Variantes e Introducción de la Variante Preferencia por Características

Pablo Quintana

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ciencias y Humanidades

Estudio del Problema del Matrimonio Estable: Presentación del Algoritmo de Gale-Shapley, Análisis de Variantes e Introducción de la Variante Preferencia por Características

Trabajo de graduación en modalidad de tesis presentado por
Pablo Quintana
para optar al grado académico de Licenciado en Matemática Aplicada

Guatemala
2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ciencias y Humanidades

Estudio del Problema del Matrimonio Estable: Presentación del Algoritmo de Gale-Shapley, Análisis de Variantes e Introducción de la Variante Preferencia por Características

Trabajo de graduación en modalidad de tesis presentado por
Pablo Quintana
para optar al grado académico de Licenciado en Matemática Aplicada

Guatemala
2025

Vo.Bo.:

(f) _____

Alan Gerardo Reyes Figueroa

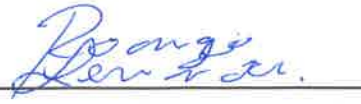
Tribunal Examinador:

(f) _____

Alan Gerardo Reyes Figueroa

(f) _____

Dorval José Carías Samayoa



(f) _____

Luis Rodrigo Leonardo Castellanos

Fecha de aprobación: Guatemala, 07 de enero de 2025.

Prefacio

A lo largo de la carrera de Matemática Aplicada los estudiantes logramos observar dos mundos distintos, el de la teoría y el de las aplicaciones de esta. El mundo de la teoría lleva a conocer sobre las demostraciones y la escritura formal de argumentos para probar un enunciado. Mientras que en el área de aplicación se estudia infinidad de aplicaciones derivadas de la teoría, entre estas, los algoritmos.

Cuando uno va creciendo y se va formando se tiene noción de qué es un algoritmo, a pesar de eso, es difícil de definirlo formalmente. Esta serie de pasos finitos que llamamos algoritmo es utilizada en muchas áreas de la matemática, aun así, durante la carrera es poco frecuente desarrollar teoría basada en un algoritmo en específico. Desde que conocí el Problema del Matrimonio Estable y que su solución se desarrollaba con base a una serie de instrucciones diseñados por David Gale y Lloyd Shapley me vi intrigado, debido a que nace una teoría distinta a mucho de lo que había estudiado antes.

El problema, el cuál se basa en encontrar un emparejamiento de dos conjuntos basado en preferencias, parece sencillo, debido a que es basado en conjuntos discretos, pero como se presenta en el trabajo, es necesario desarrollar bastante para asegurar su solución. Además, al trabajar con el problema surgen variantes de cómo esto se puede complicar al momento de trasladarlo a una situación rutinaria, por eso también nace la motivación a mostrar variantes famosas e introducir una nueva variante: Preferencias por Características.

Agradecimientos

Primero, es un honor agradecer a mi familia, quienes siempre me han dado su amor y su apoyo incondicional. Además, sin ellos no hubiera logrado llegar a estas alturas de mi carrera profesional. Son las personas más especiales en mi vida y uno de mis objetivos es lograr ser un respaldo para ellos, como ellos lo han sido para mí desde mi nacimiento.

Quiero darle un especial agradecimiento también a mis amigos, con los que he crecido y los que he conocido más adelante en mi vida. Estas personas agregan un valor imprescindible en mi vida, agregan esas sonrisas y momentos que recordaré por siempre. El aprecio que les tengo es incalculable.

En el momento que empecé a estudiar esta carrera llegué con una ilusión enorme, pero también con un temor que a veces superaba esta ilusión. Mientras fui conociendo a mis catedráticos y observando como disfrutan de su profesión ese temor fue desvaneciéndose. Les doy un agradecimiento muy especial a todos los catedráticos que formaron parte de este camino.

Índice

Resumen

La presente tesis busca enunciar el Problema del Matrimonio Estable, mostrar su solución con sus respectivas demostraciones y ejemplos. Luego, mostrar tres variantes de este problema: Conjuntos Desiguales, Preferencias Incompletas, Preferencias por Características. Para cada una se desarrolla la teoría correspondiente y necesaria. Además, se muestran conceptos previos para tener a la mano las herramientas requeridas para tener contexto del problema, sus variantes, los algoritmos y las pruebas que aseguran su solución. Este problema nace al tener el caso donde existan dos conjuntos del mismo tamaño cuyo deseo es emparejarse con el otro conjunto, es decir, para cada elemento encontrar una pareja del otro conjunto. Lo que agrega el enunciado es que cada elemento cuenta con una lista de preferencias. Con base en esta lista entra el concepto de emparejamiento inestable. Este busca emular qué pasaría si dos elementos que no están emparejados se prefieren mutuamente antes que a sus parejas. Si esto sucede se dice que el emparejamiento es inestable. La solución muestra que bajo ciertas condiciones siempre se puede encontrar un emparejamiento no inestable, o emparejamiento estable, además, también se muestra para las respectivas variantes.

CAPÍTULO 1

Introducción

El Problema del Matrimonio Estable consiste en encontrar una forma de emparejar a los elementos de dos conjuntos disjuntos, usualmente del mismo tamaño, de modo que no existan pares que prefieran estar entre sí antes que con sus respectivas asignaciones. Cada elemento cuenta con una lista de preferencias ordenadas sobre los elementos del conjunto opuesto, y el objetivo es evitar emparejamientos inestables, es decir, situaciones en las que dos elementos no emparejados mutuamente preferirían estar juntos antes que con sus respectivas parejas. Este problema, formulado originalmente por Gale y Shapley en 1962, no solo cuenta con una solución garantizada bajo ciertas condiciones, sino que también ha dado lugar a una vasta literatura que profundiza en sus variantes y aplicaciones.

La presente tesis desarrolla rigurosamente el algoritmo de Gale-Shapley, encargado de encontrar un emparejamiento estable, y demuestra formalmente sus propiedades fundamentales mediante una serie de definiciones, lemas y teoremas. Para comprender adecuadamente la formulación y resolución del problema, se introducen nociones clave de teoría de conjuntos, funciones, álgebra lineal, teoría de grafos y orden lexicográfico. Sobre esta base teórica, se analizan tres variantes del problema original que reflejan condiciones más generales o realistas: conjuntos de distinto tamaño, preferencias incompletas y preferencias basadas en características.

Particular atención se otorga a la última de estas variantes, donde las preferencias no se establecen sobre individuos específicos, sino sobre atributos que caracterizan a cada elemento. Este enfoque resulta especialmente útil en escenarios donde no existe información directa entre los conjuntos, pero sí se dispone de criterios objetivos para evaluar compatibilidades. Gracias a una adaptación adecuada del algoritmo de Gale-Shapley y al uso del orden lexicográfico, es posible garantizar nuevamente la estabilidad del emparejamiento. Esta tesis, por tanto, no solo reafirma la robustez del problema clásico, sino que extiende su aplicabilidad a contextos más amplios y actuales.

CAPÍTULO 2

Objetivos

2.1. Objetivos generales

- Analizar y presentar el problema del matrimonio estable y sus variantes clásicas, destacando su relevancia teórica y práctica.
- Abordar una nueva variante del problema del matrimonio estable basada en preferencias por características, proponiendo un algoritmo adaptado y explorando sus aplicaciones potenciales.

2.2. Objetivos específicos

- Describir en detalle el problema original del matrimonio estable, incluyendo su formulación y el algoritmo estándar de Gale-Shapley.
- Explorar y analizar variantes clásicas del problema del matrimonio estable, en particular conjuntos desiguales y preferencias incompletas.
- Identificar y discutir aplicaciones prácticas del problema del matrimonio estable y sus variantes en contextos reales, como la asignación de plazas escolares y la contratación laboral.
- Proponer una variante del problema del matrimonio estable en la que las preferencias se basen en características representadas por vectores binarios y desarrollar un algoritmo adaptado para resolver esta variante.

CAPÍTULO 3

Justificación

Esta tesis examina el problema del matrimonio estable y sus variantes, incluyendo una propuesta original centrada en la preferencia por características. Este estudio combina teoría y práctica, ofreciendo una comprensión profunda de un problema clásico y proponiendo soluciones novedosas aplicables a contextos reales como la selección de personal y la organización de equipos. Al presentar una versión original del algoritmo adaptado a preferencias basadas en características, la investigación demuestra creatividad e innovación. Además, la flexibilidad de esta nueva versión permite su extensión a otras variantes del problema, lo que abre la puerta a futuras investigaciones que desarrollen más variantes basadas en preferencias por características. Esto amplía su aplicabilidad y contribuye significativamente al campo de la optimización combinatoria y la teoría de juegos, potenciando futuras investigaciones en estas áreas.

CAPÍTULO 4

Antecedentes

En el trabajo de Santamaría Manteca Paula [10], se aborda exhaustivamente el problema del matrimonio estable y algunas variantes, proporcionando soluciones a través del Algoritmo de Gale-Shapley. Este algoritmo garantiza que siempre existe un emparejamiento estable cuando el número de hombres y mujeres es igual, y se ha utilizado como base para diversas aplicaciones prácticas y adaptaciones en problemas de asignación. Las soluciones presentadas en este trabajo se enfocan en variantes del problema original, como listas incompletas o inaceptables, y listas con indiferencias, adaptando el algoritmo para manejar estas diferencias. Por ejemplo, en el caso de listas incompletas, se realizan modificaciones al algoritmo de Gale-Shapley para asegurar que el proceso termine cuando todos los proponentes estén emparejados o cuando no queden más parejas aceptables. Además de resolver el problema base, el documento explora aplicaciones prácticas en contextos reales como la asignación de estudiantes a universidades y la organización de equipos en hospitales. Estas aplicaciones demuestran la flexibilidad y utilidad del algoritmo y sus variantes para abordar problemas complejos de asignación en diversas áreas. Por ejemplo, el problema de los hospitales y residentes se adapta del problema del matrimonio estable para asignar estudiantes a universidades, considerando las capacidades limitadas y las preferencias incompletas de las universidades. Otra aplicación notable es la asignación de usuarios a servidores en servicios de Internet distribuidos, optimizando la cercanía geográfica y el costo de servicio. Estas variantes y aplicaciones muestran cómo el problema del matrimonio estable y sus soluciones pueden extenderse y aplicarse en diversos campos.

La tesis de Joe Hidakatsu[3], titulada "Structure of the Stable Marriage and Stable Roommate Problems and Applications", proporciona una solución al problema del matrimonio estable utilizando el conocido algoritmo de Gale-Shapley, el cual siempre produce el mismo matrimonio estable independientemente de cómo se ejecute el algoritmo. Además, los autores Robert Irving y Paul Leather construyeron el poset de rotaciones, cuyas colecciones cerradas hacia abajo tienen una correspondencia uno a uno con el conjunto de asignaciones de matrimonio estable. El trabajo discute cómo utilizar el poset de rotaciones para encontrar la k-óptima coincidencia y demuestra que una k-óptima coincidencia es la misma que una coincidencia de mínimo arrepentimiento para un k suficientemente alto. Dan Gusfield define el poset de rotaciones para el problema del compañero de cuarto estable y lo utiliza para enumerar eficientemente todas las asignaciones de compañeros de cuarto estables. El estudio también abarca variantes del problema del matrimonio estable, como el problema del compañero de cuarto estable, en el cual en lugar de tener dos grupos que se emparejan

entre sí, hay un solo grupo y se busca emparejar personas dentro de ese grupo. Esta generalización lleva a la capacidad de enumerar todas las coincidencias estables posibles de manera relativamente eficiente. Sin embargo, dependiendo del número de participantes en el proceso de emparejamiento, el número de coincidencias estables posibles puede ser extremadamente grande, lo que implica que encontrar todas las coincidencias estables puede tomar mucho tiempo debido a la gran cantidad de coincidencias posibles. A pesar de esto, los resultados estructurales de Irving, Leather y Gusfield pueden llevar a la capacidad de encontrar la coincidencia estable correcta para la aplicación en mente, reconociendo que no todas las coincidencias estables son apropiadas.

El artículo "Hard variants of stable marriage"de David F. Manlove [5], Robert W. Irving, Kazuo Iwama, Shuichi Miyazaki y Yasufumi Morita aborda el problema del matrimonio estable y sus diversas variantes. Este trabajo se destaca por ser el primer estudio exhaustivo sobre variantes del problema en las que las listas de preferencias de los participantes no son necesariamente completas ni totalmente ordenadas. Los autores muestran que, bajo ciertas suposiciones restrictivas, varias de estas variantes son difíciles de resolver y difíciles de aproximar. En contraste con el caso donde las listas de preferencias son completas o estrictamente ordenadas, una instancia del problema puede admitir emparejamientos estables de diferentes tamaños. El artículo también proporciona un algoritmo de aproximación con un factor de 2 para los problemas de encontrar un emparejamiento estable de tamaño máximo o mínimo. Además, discute las implicaciones significativas de estos resultados para los esquemas prácticos de emparejamiento. Entre los problemas difíciles identificados se incluyen encontrar un emparejamiento estable de tamaño máximo o mínimo, determinar si una pareja dada es estable, y encontrar o aproximar tanto un emparejamiento estable 'igualitario' como uno de 'mínimo arrepentimiento'.

El orden lexicográfico [4] es una relación de orden definida sobre el producto cartesiano de conjuntos ordenados, utilizada comúnmente para ordenar y comparar cadenas de caracteres. En la práctica, se aplica frecuentemente en el desarrollo de aplicaciones, como al comparar valores de tipo String en Swift. Este método de ordenación es análogo a la forma en que se ordenan los términos en un diccionario, comparando elementos secuencialmente según su posición en el alfabeto.

En la tesis de Villalba Rodríguez [15], se aborda un análisis detallado de los grafos y sus aplicaciones en la teoría de la combinatoria. La autora proporciona una introducción a los conceptos básicos de la teoría de grafos, explicando la importancia de las estructuras de emparejamiento y su relevancia en diversos campos matemáticos y prácticos. Se destacan las definiciones fundamentales, como grafos bipartitos, y se establece la base para entender cómo estas estructuras se utilizan para modelar problemas complejos de asignación y emparejamiento. En cuanto a los teoremas de Hall y de Tutte, la tesis se enfoca en su demostración y aplicaciones. El teorema de Hall, clave para la existencia de emparejamientos perfectos en grafos bipartitos, se presenta con una prueba clara y ejemplos ilustrativos que muestran su aplicación en problemas reales de emparejamiento. Por otro lado, el teorema de Tutte, que generaliza el concepto de emparejamiento en grafos no bipartitos, se aborda en términos de su condición necesaria y suficiente para la existencia de emparejamientos perfectos. La autora no solo presenta la demostración teórica, sino que también examina sus implicaciones prácticas y la relación entre ambos teoremas en la teoría de emparejamientos.

CAPÍTULO 5

Conceptos previos

5.1. Teoría de Conjuntos

Definición 5.1

A la cantidad de elementos que tiene un conjunto se le conoce como Cardinalidad, sea el conjunto A, su Cardinalidad es denotada por $\#A$.

Nota 5.1

Si A es un conjunto finito, entonces $\#A \in \mathbb{Z}^+$.

Definición 5.2

Sean A y B conjuntos, la Unión de A y B es:

$$A \cup B = \{x : x \in A \ o \ x \in B\}$$

.

Definición 5.3

Sean A y B conjuntos, la Intersección entre A y B es:

$$A \cap B = \{x : x \in A \ \& \ x \in B\}$$

.

Definición 5.4

Sean A y B conjuntos, la Diferencia entre A y B es:

$$A - B = \{x : x \in A \ \& \ x \notin B\}$$

.

Definición 5.5

Sean A y B conjuntos, se dice que A está contenido en B ssi:

$$\forall a \in A \rightarrow a \in B$$

.

Se denota como $A \subseteq B$

Definición 5.6

Sean A y B conjuntos, el producto cartesiano entre A y B se define como:

$$A \times B = \{(a, b) : a \in A, b \in B\}$$

.

Propiedad 5.1

Si A y B son conjuntos finitos entonces:

$$\#(A \times B) = \#A * \#B$$

.

Demostración. Sea $b \in B$. Nótese que para cada $a \in A$, se puede formar un par (a, b), donde cada par es distinto debido a que se utilizan elementos distintos de A. Por lo tanto, para cada $b \in B$ se tienen $\#A$ pares, ya que existen $\#B$ elementos en B, entonces:

$$\#(A \times B) = \#A * \#B$$

. □

Nota 5.2

En este trabajo si hay un conjunto A, se denomina como:

$$A^n = \underbrace{A \times A \times \cdots \times A}_{n \ veces} = \{(a_1, a_2, ..., a_n) : a_i \in A \forall i \in \{1, 2, ..., n\}\}$$

El cuál es el n-ésimo producto de A sobre sí mismo.

5.2. Teoría de Funciones

Definición 5.7

Sean X, Y conjuntos. Se le llama a $f : X \rightarrow Y$ mapeo, donde X se le denomina como dominio y Y como contradominio. f es la regla de asignación que al aplicarla a un elemento de X mapea a un elemento de Y. Este mapeo es función ssi:

- $\forall x \in X$ existe un único $y \in Y$ tal que $f(x) = y$.

Definición 5.8

Sean $A \subseteq X$, Y un conjunto y $f : X \to Y$, la imagen del conjunto A bajo f es:

$$f(A) = \{f(x) : x \in A\}.$$

Nótese que $f(A) \subseteq Y$

Definición 5.9

Sean $B \subseteq Y$, X un conjunto y $f : X \to Y$, la preimagen del conjunto B bajo f es:

$$f^{-1}(B) = \{x : \exists b \in B \ni f(x) = b\}.$$

Nótese que $f^{-1}(B) \subseteq X$.

Propiedad 5.2

Para una función $f : X \to Y$, y $A \subseteq X$ se tiene que:

$$A \subseteq f^{-1}(f(A)).$$

Demostración. Sea $x \in A$ entonces por definición de imagen, $f(x) = y \in f(A)$. Ahora, por definición de preimagen, ya que existe $y \in f(A)$ tal que $f(x) = y$, entonces, $x \in f^{-1}(f(A))$, lo cuál implica que:

$$A \subseteq f^{-1}(f(A)).$$

□

Propiedad 5.3

Para una función $f : X \to Y$, y $B \subseteq Y$ se tiene que:

$$f(f^{-1}(B)) \subseteq B.$$

Demostración. Sea $y \in f(f^{-1}(B))$, entonces existe $x \in f^{-1}(B)$ tal que $f(x) = y$, ya que $x \in f^{-1}(B)$ entonces, $y = f(x)$ es, tal que $y \in B$, por definición de preimagen. Por lo tanto,

$$f(f^{-1}(B)) \subseteq B.$$

□

Definición 5.10

Una función $f : X \to Y$ se dice que es inyectiva ssi:

$\forall x_1, x_2 \in X$ tal que:

$$f(x_1) = f(x_2).$$

Entonces:

$$x_1 = x_2.$$

Es decir, si dos elementos de X tienen la misma imagen, entonces son el mismo elemento.

Definición 5.11

Una función $f : X \to Y$ se dice que es sobreyectiva ssi:

$$\forall y \in Y \exists x \in X \ni f(x) = y.$$

Es decir, todos los elementos de Y tienen una preimagen no vacía.

Definición 5.12

Una función $f : X \to Y$ se dice que es biyectiva ssi es inyectiva y sobreyectiva.

5.3. Álgebra Lineal

Definición 5.13

Sea F un conjunto, y $+, *$, operaciones binarias, en este caso denominadas como suma y multiplicación, entonces, si se cumple:

Bajo la adición:

- Cerradura: $\forall a, b \in F$, $(a + b) \in F$.

- Asociatividad: $\forall a, b, c \in F$, $(a + b) + c = a + (b + c)$.
- Neutro: Existe $0 \in F$ tal que: $\forall a \in F$, $a + 0 = a$.
- Inverso: $\forall a \in A$ existe $-a \in F$ tal que: $a + (-a) = 0$.
- Conmutatividad: $\forall a, b \in F$, $a + b = b + a$.

Bajo la multiplicación:

- Cerradura: $\forall a, b \in F$, $(a * b) \in F$.
- Asociatividad: $\forall a, b, c \in F$, $(a * b) * c = a * (b * c)$.
- Neutro: Existe $1 \in F$ tal que: $\forall a \in F$, $a * 1 = a$.
- Inverso: $\forall a \in F$ existe $a^{-1} \in F$ tal que $a * a^{-1} = 1$.
- Conmutatividad: $\forall a, b \in F$, $a * b = b * a$.

Bajo ambas:

- Distributividad: $\forall a, b, c \in F$, $a * (b + c) = (a * b) + (a * c)$.

Entonces F es un campo.

Definición 5.14

Sean V un conjunto, F un campo y:

- $+ : V \times V \to V$.
- $* : F \times V \to V$.

operaciones binarias, si se cumplen las siguientes:

- Asociatividad: $\forall u, v, w \in V$, $u + (v + w) = (u + v) + w$.
- Conmutatividad: $\forall u, v \in V$, $u + v = v + u$.
- Neutro: Existe $0 \in V$ tal que $\forall v \in V$ $v + 0 = v$.
- Inverso: $\forall v \in V$ existe $-v \in V$ tal que $v + (-v) = 0$.
- Compatibilidad: $\forall a, b \in F$, $v \in V$ se tiene que $a * (b * v) = (a * b) * v$.
- Neutro de escalares: Existe $1 \in F$ tal que $1 * v = v$, $\forall v \in V$.
- Distributividad 1: $\forall a \in F$, $u, v \in V$ se tiene que $a * (u + v) = a * u + a * v$.
- Distributividad 2: $\forall a, b \in F$, $v \in V$ se tiene que $(a + b) * v = a * v + b * v$.

Entonces: $(V, +, *, F)$ es un Espacio Vectorial.

Definición 5.15 Sea $(V, +, *, F)$ un Espacio Vectorial, y $v \in V$, entones v es un vector.

Nota 5.3 $(\mathbb{R}^n, +, *, \mathbb{R})$ es un espacio vectorial, sus vectores son representados como:

$$v = (v_1, v_2, ..., v_n)$$

donde $v_i \in \mathbb{R} \forall i \in \{1, 2, ..., n\}$.

Definición 5.16

Sea F un campo, un conjunto bidimensional de elementos de F ordenados en filas y columnas se le conoce como Matriz. Se le denomina como Matriz M de tamaño $m \times n$, es decir M cuenta con m filas y n columnas, donde $m, n \in \mathbb{Z}^+$. Cada elemento de la Matriz es representado como:

$$M_{i,j}$$

donde: $i \in \{1, 2, ..., m\}$, $j \in \{1, 2, ..., n\}$ y $M_{i,j} \in F$.

Definición 5.17

Sea $S = \{1, 2, ..., n\}$ donde $n \in \mathbb{Z}^+$. Una permutación de S es una función $\sigma : S \to S$ tal que σ es biyectiva.

Definición 5.18

Sea M una matriz de tamaño $n \times n$, entonces se define el determinante de M como:

$$det(M) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} M_{i,\sigma(i)}$$

donde S_n es el conjunto de todas las permutaciones de $S = \{1, 2, ..., n\}$ y $sgn : S_n \to \{-1, 1\}$ es la función que asigna cada permutación a un signo positivo o negativo.

5.4. Orden lexicográfico

Definición 5.19

Sean A y B conjuntos no vacíos. Sea $R \subseteq A \times B$ tal que:

$$R = \{(a, b) : (a, b) \ cumple \ con \ p\}$$

donde p es una condición. A R se le llama una relación binaria entre A y B. La notación para decir que $(a, b) \in R$ es aRb.

Definición 5.20

Sea A un conjunto no vacío, y sea R una relación binaria entre A y A. Entonces, si se cumple que:

- Reflexividad: $\forall x \in A$ entonces xRx.
- Antisimetría: Sean $x, y \in A$, si xRy y yRx entonces $x = y$.

- Transitividad: Sean $x, y, z \in A$, si xRy y yRz entonces xRz.

Entonces, R es una relación de orden parcial, y A es un conjunto ordenado bajo la relación R.

Definición 5.21

Sean $(A, \leq_A)$ y $(B, \leq_B)$ conjuntos ordenados bajo las relaciones $\leq_A$, $\leq_B$ respectivamente. Entonces el orden lexicográfico para $A \times B$ se define como:

$$(a, b) \leq_{A,B} (a', b') \text{ ssi } (a \leq_A a' \text{ y } a \neq a') \text{ ó } (a = a \text{ y } b \leq_B b').$$

Propiedad 5.4

Sean Sean $(A, \leq_A)$ y $(B, \leq_B)$ conjuntos ordenados bajo las relaciones $\leq_A$, $\leq_B$ respectivamente. El orden lexicográfico para $A \times B$ es una relación de orden parcial.

Demostración. Sean Sean $(A, \leq_A)$ y $(B, \leq_B)$ conjuntos ordenados bajo las relaciones $\leq_A$, $\leq_B$ respectivamente.

- Reflexividad: Sea $(a, b) \in A \times B$, Nótese que $a = a$ y $b \leq_B b$. Por lo tanto, se cumple con que $(a = a \text{ y } b \leq_B b)$, entonces $(a, b) \leq_{A,B} (a, b)$.
- Antisimetría: Sean $(a, b), (a', b') \in A \times B$, tal que $(a, b) \leq_{A,B} (a', b')$ y $(a', b') \leq_{A,B} (a, b)$. Nótese esto implica que $a \leq_A a'$ y que $a' \leq_A a$, por lo tanto, ya que $\leq_A$ es relación de orden entonces: $a = a'$. Ahora, sucede también que $b \leq_B b'$, $b' \leq_B b$ y ya que $\leq_B$ es relación de orden, entonces $b = b'$, por lo tanto, $(a, b) = (a', b')$.
- Transitividad: Sean $(a, b), (a', b'), (a'', b'') \in A \times B$ tal que $(a, b) \leq_{A,B} (a', b')$ y $(a', b') \leq_{A,B} (a'', b'')$. Esto significa que $a \leq_A a' \leq_A a''$, por lo tanto, $a \leq_A a''$, ya que $\leq_A$ es una relación de orden. Ahora, si $a \neq a''$ entonces se tiene que $(a, b) \leq_{A,B} (a'', b'')$, si $a = a''$ entonces, $a = a' = a''$. Por lo tanto, $b \leq_B b'$ y $b' \leq_B b''$, lo cuál implica que $b \leq_B b''$. Entonces, $(a, b) \leq_{A,B} (a'', b'')$

Por lo tanto, $\leq_{A,B}$ (orden lexicográfico) es una relación de orden parcial entre $A \times B$ y $A \times B$. □

5.5. Teoría de grafos

Definición 5.22

Un Grafo Dirigido es un par ordenado $G = (V, E)$ donde:

- V es un conjunto no vacío, el cuál será llamado conjunto de vértices.
- $E \subseteq V \times V$, el cuál será llamado el conjunto de aristas.

A $(v_1, v_2) \in E$ se le llama una arista, en un grafo dirigido, que $(v_1, v_2) \in E$ no implica que $(v_2, v_1) \in E$.

Definición 5.23

Un Grafo No Dirigido es un par ordenado $G = (V, E)$ donde:

- V es un conjunto no vacío, el cual será llamado conjunto de vertices.
- $E \subseteq V \times V$, el cuál será llamado el conjunto de aristas.

A $(v_1, v_2) \in E$ se le llama una arista, la cual es bidireccional en un grafo no dirigido. Por lo tanto es necesario que $(v_2, v_1) \in E$. Es decir que para un grafo no dirigido si $(v_1, v_2) \in E$ entonces $(v_2, v_1) \in E$.

5.6. Algoritmos

Definición 5.24

Una iteración es un conjunto finito de instrucciones, las cuales se siguen en orden. Este conjunto de instrucciones es diseñado para participar en un proceso iterativo, es decir, son diseñadas para que se puedan repetir varias veces hasta que termine el proceso.

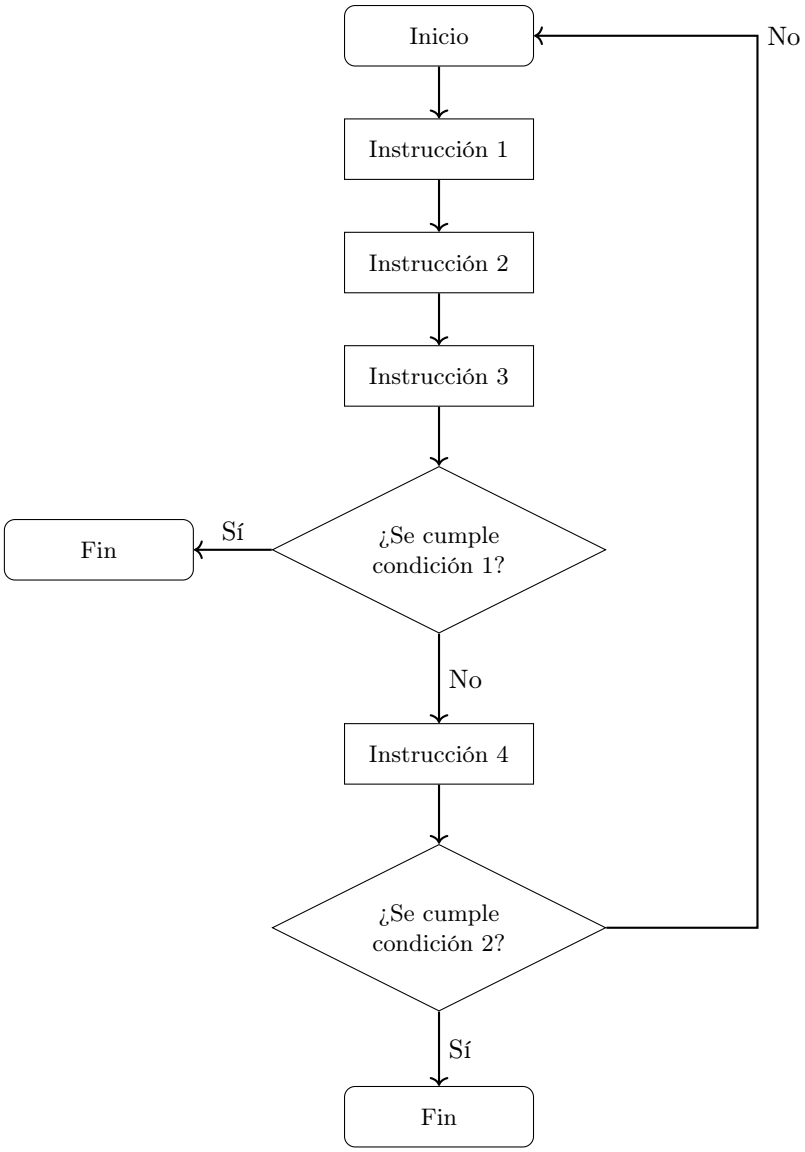
Definición 5.25

A este proceso de repetir el conjunto de instrucciones llamado iteración se le conoce como Algoritmo. Un algoritmo debe contar con ciertas condiciones:

- Debe tener un criterio de finalización, el cuál si se cumple, el algoritmo para, si no se cumple, se repite la iteración.
- En cada iteración debe verificarse si el criterio se cumple o no.
- El criterio debe ser tal que el número de iteraciones sea finito.

Diagrama de flujo

Para representar una iteración y un algoritmo se puede utilizar un diagrama de flujo, donde se representa en cada figura una instrucción, además de representarse la verificación del criterio de finalización. Así como se muestra a continuación:



Nótese en este caso que el criterio de decisión es si se cumple alguna de las dos condiciones presentadas en un rombo, si alguna se cumple el algoritmo finaliza, de lo contrario, sigue iterando. Estas condiciones deben ser diseñadas tal que se asegure que en algún momento alguna de las dos se cumple, así el algoritmo termina.

CAPÍTULO 6

Emparejamiento desde la perspectiva de grafos

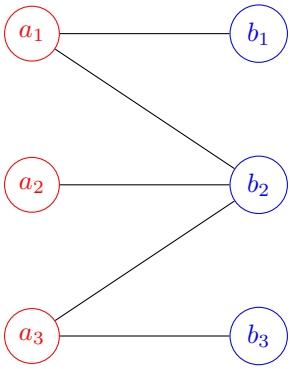
Definición 6.1: Sea $G = (V, E)$ un grafo no dirigido. Esto es:

- V es un conjunto tal que $\#V = \mathbb{Z}^+$.
- $E \subseteq \{(u, v) : u, v \in V\}$.

El grafo G se dice que es bipartito si existen A, B conjuntos no vacíos tal que $A \cup B = V$, $A \cap B = \emptyset$, y $\forall (u, w) \in E$ $u \in A$ y $w \in B$.

Ejemplo a Definición 6.1

Sean $V = \{a_1, a_2, a_3, b_1, b_2, b_3\}$, $E = \{(a_1, b_1), (a_1, b_2), (a_2, b_2), (a_3, b_2), (a_3, b_3)\}$. Nótese que $G = (V, E)$ es un grafo no dirigido. Ahora, sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$ conjuntos no vacíos. Nótese que $A \cup B = V$, $A \cap B = \emptyset$, además $\forall (a, b) \in E$ $a \in A$, $b \in B$. Por lo tanto, G es un grafo bipartito. Como lo muestra la siguiente figura:



Definición 6.2: Sea $G = (V, E)$ un grafo no dirigido tal que $\#V = n$ con $n \in \mathbb{Z}^+$, entonces $V = \{v_1, v_2, ..., v_n\}$. Sea la matriz M de tamaño $n \times n$ tal que:

- $M_{i,j} = 1$, si $(v_i, v_j) \in E$ o $(v_j, v_i) \in E$.
- $M_{i,j} = 0$, si $(v_i, v_j) \notin E$ y $(v_j, v_i) \notin E$.

M es la matriz de Adyacencia de G.

Ejemplo a Definición 6.2

Sea $V = \{v_1, v_2, v_3, v_4\}$ y $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_1, v_4)\}$. Nótese que $G = (V, E)$ es un grafo no dirigido y

$$M = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

es la matriz de adyacencia de G, debido a que cada 1 de la matriz corresponde a la tupla (v_i, v_j) o (v_j, v_i) que aparece en E.

Definición 6.3: Sea $G = (V, E)$ un grafo bipartito donde la partición de V está dada por A y B, donde $\#A = n$, $\#B = m$, con $A = \{a_i, a_2, ..., a_n\}$, $B = \{b_1, b_2, ..., b_n\}$. Sea M_B una matriz de tamaño $n \times m$ tal que:

- $M_{Bi,j} = 1$, si $(a_i, b_j) \in E$.
- $M_{Bi,j} = 0$, si $(a_i, b_j) \notin E$.

M es la matriz de Biadyacencia de G.

Ejemplo a Definición 6.3

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, $V = A \cup B$,$E = \{(a_1, b_1), (a_1, b_2), (a_2, b_2), (a_3, b_2), (a_3, b_3)\}$. Nótese que $G = (V, E)$ es un grafo bipartito, y sea:

$$M_B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

M_B es la matriz de biadyacencia del grafo G.

Definición 6.4 Sea $G = (V, E)$ un grafo bipartito donde la partición de V está dada por A y B, donde $\#A = n = \#B$. Entonces un Emparejamiento de G es $P \subseteq E$ tal que:

- $\#P = n$.
- $\forall a \in A$ no existen b_i, b_j con $b_i \neq b_j$ tal que $(a, b_i), (a, b_j) \in P$.
- $\forall b \in B$ no existen a_i, a_j con $a_i \neq a_j$ tal que $(a_i, b), (a_j, b) \in P$.

Es decir, existe $f : A \to B$ biyectiva tal que si $(u, v) \in P$ entonces $v = f(u)$.

Ejemplo a Definición 6.4

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, $V = A \cup B$,$E = \{(a_1, b_1), (a_1, b_2), (a_2, b_2), (a_3, b_2), (a_3, b_3)\}$. Nótese que $G = (V, E)$ es un grafo bipartito, y sea:

$$P = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}.$$

Nótese que P cumple con las condiciones propuestas en la definición 6.4, por lo tanto, P es un Emparejamiento de G.

Teorema 6.1: Sean A, B tal que $\#A = \#B = n$, donde $n \in \mathbb{Z}^+$. Además, sean $V = A \cup B$, $E = \{(a, b) : a \in A, b \in B\}$, $G = (V, E)$ un grafo bipartito. Entonces existe un emparejamiento de G ssi $\forall S \subseteq A \ \#N(S) \geq \#S$, donde $N(S) = \{b \in B : \exists a \in S \ni (a, b) \in E\}$, el conjunto de vecinos de S.

Demostración. Sean A, B tal que $\#A = \#B = n$, donde $n \in \mathbb{Z}^+$. Además, sean $V = A \cup B$, $E = \{(a, b) : a \in A, b \in B\}$, $G = (V, E)$ un grafo bipartito.

($\Rightarrow$) Supóngase que existe P, un emparejamiento de G. y supóngase por reducción al absurdo que existe $S \subseteq A$ tal que $\#N(S) < \#S$. Por lo tanto, existe $a \in S$ tal que no existe $b \in B$ donde $(a, b) \in P$ ($\rightarrow\leftarrow$), lo cual es una contradicción ya que si eso pasa P no es un emparejamiento de G.

($\Leftarrow$) Supóngase que $\forall S \subseteq A \ \#N(S) \geq \#S$. Sea $\psi : A \rightarrow \mathbb{Z}^+$, tal que $\psi(a) = \#N(\{a\})$, ahora, hágase una lista de los elementos de A $L = \{a_1, a_2, ..., a_n\}$ tal que: $\psi(a_1) \leq \psi(a_2) \leq ... \leq \psi(a_n)$. Ahora, considérese el siguiente algoritmo de n iteraciones.

Se comienza con $P = \{\}$.

Para cada iteración:

1. Se tiene una lista $A' \subseteq$ A, tal que $A' = \{a \in A : \exists b \in B \ni (a, b) \in P\}$.
2. Se tiene una lista $B' \subseteq$ B, tal que $B' = \{b \in B : \exists a \in A \ni (a, b) \in P\}$.
3. Se toma el primer elemento de $L - A'$, llámese a.
4. Tómese un elemento $b \in N(\{a\}) - B'$.
5. $P = P \cup \{(a, b)\}$.

Nótese que el único paso que puede generar problemas es el paso 3 del algoritmo. Ya que se debe asegurar que $(N(\{a\}) - B') \neq \emptyset$. Sea $S = A' \cup \{a\}$, nótese que $\#N(S) \geq \#A' + 1$. Ahora, nótese que $\#B' = \#A'$. Por lo tanto, $N(S) - B'$ tiene al menos un elemento que no está emparejado. Por lo tanto, $(N(\{a\}) - B') \geq 1$. Ahora, ya que este algoritmo itera n veces y ningún elemento de A y B se elige dos veces, entonces P es un emparejamiento de G.

□

Definición 6.5: Sea $G = (V, E)$ un grafo bipartito donde la partición de V está dada por A y B, donde $\#A = n$, $\#B = m$, entonces $A = \{a_i, a_2, ..., a_n\}$, $B = \{b_1, b_2, ..., b_n\}$. Sea M_T una matriz de tamaño $n \times m$ tal que:

- $M_{T\,i,j} = x_{i,j}$, si $(a_i, b_j) \in E$.
- $M_{T\,i,j} = 0$, si $(a_i, b_j) \notin E$.

donde $x_{i,j}$ es una variable en los reales. A esta matriz se le conoce como la Matriz de Tutte de G.

Ejemplo a Definición 6.5

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, $V = A \cup B$,$E = \{(a_1, b_1), (a_1, b_2), (a_2, b_2), (a_3, b_2), (a_3, b_3)\}$. Nótese que $G = (V, E)$ es un grafo bipartito, y sea:

$$M_T = \begin{bmatrix} x_{1,1} & x_{1,2} & 0 \\ 0 & x_{2,2} & 0 \\ 0 & x_{3,2} & x_{3,3} \end{bmatrix}.$$

M_T es la matriz de Tutte del grafo G.

Teorema 6.2: Sea $G = (V, E)$ un grafo bipartito donde la partición de V está dada por A y B, donde $\#A = n$, $\#B = m$, entonces $A = \{a_i, a_2, ..., a_n\}$, $B = \{b_1, b_2, ..., b_n\}$. Sea M_T la matriz de Tutte de G. Entonces G tiene un emparejamiento ssi $det(M_T) \neq 0$.

Demostración. Sea $G = (V, E)$ un grafo bipartito donde la partición de V está dada por A y B, donde $\#A = n$, $\#B = m$, entonces $A = \{a_i, a_2, ..., a_n\}$, $B = \{b_1, b_2, ..., b_n\}$. Sea M_T la matriz de Tutte de G.

($\Rightarrow$) Supóngase que existe P, un emparejamiento de G. Por lo tanto, sea S_n el conjunto de permutaciones sobre $[n]$. Ahora, ya que P es un emparejamiento, existe un $\sigma \in S_n$ tal que $\{(a_i, b_{\sigma(i)}) : a_i \in A, b_{\sigma(i)} \in B, i \in \{1, 2, ..., n\}\} = P$. Entonces: $\prod_{i=1}^{n} m_{i,\sigma(i)} = x_{1,\sigma(1)} x_{2,\sigma(2)} ... x_{n,\sigma(n)} \neq 0$. Ahora, ya que no hay otra permutación que genere las mismas variables, entonces no existe otro producto que al restar anule el término distinto de 0. Por lo tanto,

$$det(M_T) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} M_{T\, i,\sigma(i)} \neq 0.$$

($\Leftarrow$) Supóngase que $det(M_T) \neq 0$. Por lo tanto existe $\sigma \in S_n$ tal que $\prod_{i=i}^{n} m_{i,\sigma(i)} \neq 0$. Eso por construcción de la Matriz de Tutte de G existe $P = \{(a_1, b_{\sigma(1)}), (a_2, b_{\sigma(2)}), ..., (a_n, b_{\sigma(n)})\}$, nótese que $\#P = n$, y debido a que σ es una permutación entonces ningún elemento de A o de B está emparejado dos veces con dos elementos distintos. Entonces P es un emparejamiento de G.

□

Definición 6.6 Sea M una matriz de tamaño $n \times n$. Sea S_n el conjunto de permutaciones de $\{1, 2, ..., n\}$:

$$perm(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} M_{i,\sigma(i)}.$$

$perm(M)$ se llama el permanente de M.

Ejemplo a Definición 6.6

Sea:

$$M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Entonces $S_n = \{\{1,2,3\},\{1,3,2\},\{2,1,3\},\{2,3,1\},\{3,1,2\},\{3,2,1\}\}$. Por lo tanto:

$$perm(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} M_{i,\sigma(i)} =$$

$$= M_{1,1}M_{2,2}M_{3,3} + M_{1,1}M_{2,3}M_{3,2} + M_{1,2}M_{2,1}M_{3,3} + M_{1,2}M_{2,3}M_{3,1} + M_{1,3}M_{2,1}M_{3,2} + M_{1,3}M_{2,2}M_{3,1} =$$

$$perm(M) = 0 + 1 + 1 + 0 + 0 + 0 = 2$$

Teorema 6.3 Sean A, B tal que $\#A = \#B = n$, donde $n \in \mathbb{Z}^+$. Además, sean $V = A \cup B$, $E = \{(a,b) : a \in A, b \in B\}$, $G = (V,E)$ un grafo bipartito y sea M_B la matriz de biadyacencia de G. Entonces existen $perm(M_B)$ emparejamientos distintos en G.

Demostración. Sea $S \subseteq S_n$ tal que $\forall \sigma \in S'$ $P = \{(a_1, b_{\sigma(1)}), (a_2, b_{\sigma(2)}), ..., (a_n, b_{\sigma(n)})\}$ es un emparejamiento, nótese que por construcción de la matriz de adyacencia, $M_{B\,i,\sigma(i)} = 1$, entonces: $\prod_{i=1}^{n} M_{i,\sigma(i)} = 1$. Por lo tanto, si para todo $\sigma \notin S'$ no produce un emparejamiento entonces:

$$perm(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} M_{i,\sigma(i)} = \sum_{\sigma \in S'} \prod_{i=1}^{n} M_{i,\sigma(i)} = \#S'.$$

y este valor corresponde a la cantidad de emparejamientos de G distintos. □

CAPÍTULO 7

El problema del matrimonio estable y su solución

El problema del matrimonio estable, formulado por David Gale y Lloyd Shapley en 1962, es un problema clásico que busca determinar cómo emparejar dos conjuntos de individuos de manera que el emparejamiento resultante sea estable. En este capítulo, se presentará una descripción detallada de este problema, así como el algoritmo de Gale-Shapley, que garantiza una solución estable, y las propiedades derivadas del emparejamiento resultante.

Para facilitar la comprensión del algoritmo y las demostraciones posteriores, primero se presentarán cinco definiciones clave relacionadas con el problema. Cada una de estas definiciones será acompañada de ejemplos ilustrativos que ayudarán a aclarar los conceptos fundamentales. Estas definiciones son cruciales para establecer las bases del problema y preparar el terreno para las demostraciones que siguen.

A continuación, se explicará el algoritmo de Gale-Shapley en términos generales, seguido de un ejemplo concreto que muestra su funcionamiento en la práctica. Este algoritmo se centra en la interacción entre dos conjuntos, A y B, donde uno de los conjuntos hace las propuestas de emparejamiento y el otro responde de acuerdo a sus preferencias. El proceso iterativo garantiza que se obtenga un emparejamiento estable.

Posteriormente, se abordarán diversas propiedades y lemas derivados del algoritmo. Estas propiedades refuerzan la estructura para asegurar la estabilidad del emparejamiento. Finalmente, todas estas herramientas permitirán demostrar el Teorema del Matrimonio Estable.

Este problema se presentará desde un punto de vista conjuntista y de funciones, pero al traducirlo al idioma de grafos presentado en el capítulo 3, se busca optimizar el emparejamiento de un Grafo Bipartito cuya matriz de Biadyacencia es cuadrada y todos sus valores son 1.

Definición 7.1: Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$. Se le llama un emparejamiento entre A y B a:

$$P = \{(a, b) : a \in A, b \in B\}$$

donde:

- $\#P = n$.
- $\forall a_i \in A$ existe $b_j \in B$ tal que $(a_i, b_j) \in P$.
- $\forall b_i \in B$ existe $a_j \in A$ tal que $(a_j, b_i) \in P$.

Ejemplo a Definición 7.1: Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$ y sea:

$$P = \{(a_1, b_3), (a_2, b_1), (a_4, b_2), (a_3, b_4)\}.$$

Entonces:

- Nótese que $\#P = 4 = \#A = \#B$
- Para el conjunto A:
  - Para a_1, $(a_1, b_3) \in P$
  - Para a_2, $(a_2, b_1) \in P$
  - Para a_3, $(a_3, b_4) \in P$
  - Para a_4, $(a_4, b_2) \in P$.
- Para el conjunto B:
  - Para b_1, $(a_2, b_1) \in P$
  - Para b_2, $(a_4, b_2) \in P$
  - Para b_3, $(a_1, b_3) \in P$
  - Para b_4, $(a_3, b_4) \in P$.

Por lo tanto, P es un emparejamiento entre A y B.

Definición 7.2: Sean dos conjuntos A, B tal que $\#B = n$ donde $n \in \mathbb{Z}^+$. Sea $a \in A$. La función biyectiva:

$$\psi_a : B \longrightarrow \{1, 2, ..., n-1, n\}$$

es un ranking de a para B.

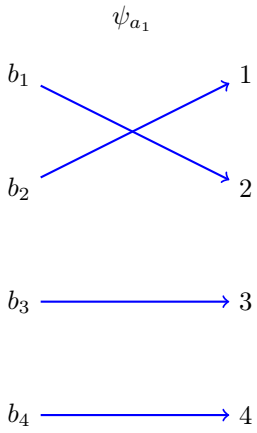
Asimismo, una lista de preferencias de a para B está dada por:

$$B_a = \{b_i, ..., b_j\}$$

donde el primer elemento corresponde al que hace que $\psi_a(b_i) = 1$, así sucesivamente hasta el último elemento que corresponde al que hace que $\psi_a(b_j) = n$.

De manera idéntica se definen un ranking de b para A y una lista de preferencias de b para A con $b \in B$.

Ejemplo a definición 7.2: Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$. Un ranking de a_1 para B sería dado por $\psi_{a_1} : B \longrightarrow \{1, 2, 3, 4\}$, tal que:



Por lo tanto, la lista de preferencias de a_1 para B se describe: $B_{a_1} = \{b_2, b_1, b_3, b_4\}$

Definición 7.3: Sean dos conjuntos A, B, sean $a \in A$, $b_i, b_j \in B$, y sea ψ_a un ranking de a para B entonces b_i es mejor emparejamiento para a que b_j si y solo si

$$\psi_a(b_i) < \psi_a(b_j).$$

Ejemplo a definición 7.3:

Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$. Y sea ψ_{a_1} la función presentada en el Ejemplo a definición 5.2, entonces:

- $\psi_{a_1}(b_1) = 2$
- $\psi_{a_1}(b_2) = 1$
- $\psi_{a_1}(b_3) = 3$
- $\psi_{a_1}(b_4) = 4$

Por lo tanto, b_2 es mejor emparejamiento para a_1 que b_1, ya que:

$$1 = \psi_{a_1}(b_2) < \psi_{a_1}(b_1) = 2.$$

Definición 7.4: Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A y sea P un emparejamiento entre A y B, P es un emparejamiento inestable si y solo si existen $(a, b), (a', b') \in P$ tal que:

- b' es mejor emparejamiento para a que b.
- a es mejor emparejamiento para b' que a'.

Asimismo, un emparejamiento estable es un emparejamiento que no es emparejamiento inestable.

Ejemplo 1 a definición 7.4:

Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_4, b_2\}$
  - a_2: $\{b_4, b_2, b_1, b_3\}$
  - a_3: $\{b_1, b_2, b_3, b_4\}$
  - a_4: $\{b_2, b_3, b_4, b_1\}$
- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2, a_4\}$
  - b_2: $\{a_3, a_2, a_4, a_1\}$
  - b_3: $\{a_1, a_2, a_4, a_3\}$
  - b_4: $\{a_4, a_1, a_2, a_3\}$

Ahora, sea $P = \{(a_1, b_4), (a_4, b_3), (a_2, b_1), (a_3, b_2)\}$. P es un emparejamiento, nótese el caso de (a_1, b_4), (a_4, b_3):

- b_3 es mejor emparejamiento para a_1 que b_4
- a_1 es mejor emparejamiento para b_3 que a_4.

Por lo tanto, P es un emparejamiento inestable.

Ejemplo 2 a definición 7.4:

Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_4, b_2\}$
  - a_2: $\{b_4, b_2, b_1, b_3\}$
  - a_3: $\{b_1, b_2, b_3, b_4\}$
  - a_4: $\{b_2, b_3, b_4, b_1\}$

- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2, a_4\}$
  - b_2: $\{a_3, a_2, a_4, a_1\}$
  - b_3: $\{a_1, a_2, a_4, a_3\}$
  - b_4: $\{a_4, a_1, a_2, a_3\}$

Ahora, sea $P = \{(a_1, b_3), (a_2, b_4), (a_3, b_1), (a_4, b_2)\}$. P es un emparejamiento y además cada elemento de A está con su mejor emparejamiento, es decir, que para estos elementos no existe un mejor emparejamiento, entonces no es un emparejamiento inestable, por lo tanto, P es un emparejamiento estable.

7.1. Algoritmo Gale-Shapley

Algoritmo: Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A y sea P un emparejamiento entre A y B. Por lo tanto, todos los elemendos de A y B presentan listas de preferencias para el otro conjunto.

Para este algoritmo un conjunto será denominado como conjunto activo, en este caso el conjunto A. Mientras que el otro conjunto será denominado como conjunto pasivo, en este caso el conjunto B. Esto debido a que el conjunto A será el encargado de realizar las propuestas y el conjunto B el encargado de aceptar o rechazar estas propuestas.

Para cada iteración cada elemento a del conjunto A tendrá la siguiente información:

- E_a, emparejamiento actual del elemento a, puede referirse a un elemento del conjunto B, o estar vacío. La notación para hacer referencia a que este es vacío es: $E_a = \_$, de lo contrario $E_a \in B$
- $L_a = \{b_i, ..., b_j\}$, lista de preferencias, esta lista está dada por ψ_a, el ranking de B para a. Esta lista es constante, no cambia durante las iteraciones.
- $R_a = \{\}$, lista de rechazos, esta es un subconjunto de B. Contiene todos los elementos de B a los cuales a les propuso y no están en E_a en la iteración actual.

Para cada iteración cada elemento b del conjunto B tendrá la siguiente información:

- E_b, emparejamiento actual del elemento b, puede referirse a un elemento del conjunto A, o estar vacío. La notación para hacer referencia a que este es vacío es: $E_a = \_$, de lo contrario $E_a \in B$
- $L_b = \{b_i, ..., b_j\}$, lista de preferencias, esta lista está dada por ψ_b, el ranking de A para b. Esta lista es constante, no cambia durante las iteraciones.
- $C_b = \{\}$ lista de propuestas actuales. Subconjunto de A, el cual comprende los elementos de A que propusieron a b en la iteración actual.

Cada iteración comprende los siguientes pasos:

1. Se identifican los elementos de A cuyo emparejamiento actual E_a sea vacío. Llámese a este conjunto $\hat{A}$.

2. Para cada $a \in \hat{A}$ se obtiene una lista de sus posibles parejas, dada por $(B - R_a)$.
3. Para cada $a \in \hat{A}$ se obtiene su mejor emparejamiento: $b_a = \psi_a^{-1}(min(\psi_a(B - R_a)))$.
4. Cada $a \in \hat{A}$ entra al conjunto C_{b_a} del elemento de b_a.
5. Se identifican los elementos de B cuya lista de propuestas actuales C_b no sea vacía. Llámese a este conjunto $\hat{B}$.
6. Para cada $b \in \hat{B}$ se obtiene una lista de posibles parejas, dada por $C_b \cup \{E_b\}$.
7. Para cada $b \in \hat{B}$ se obtiene al mejor emparejamiento $a_b = \psi_b^{-1}(min(\psi_b(C_b \cup \{E_b\})))$.
8. Para cada $b \in \hat{B}$ $E_b = a_b$ y $E_{a_b} = b$.
9. Para cada $a \in \hat{A}$ rechazado $E_a = \_$.
10. Para cada $b \in \hat{B}$, y para todo $a \in (C_b - \{E_b\})$, $R_a = R_a \cup \{b\}$.
11. Para cada $b \in \hat{B}$ $C_b = \emptyset$.

El algoritmo termina cuando al intentar realizar una iteración $\hat{A}$ sea vacío.

Ejemplo del Algoritmo:

Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_4, b_2\}$
  - a_2: $\{b_4, b_2, b_1, b_3\}$
  - a_3: $\{b_3, b_1, b_2, b_4\}$
  - a_4: $\{b_2, b_3, b_4, b_1\}$
- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2, a_4\}$
  - b_2: $\{a_3, a_2, a_4, a_1\}$
  - b_3: $\{a_1, a_2, a_4, a_3\}$
  - b_4: $\{a_4, a_1, a_2, a_3\}$

El conjunto A será el conjunto activo y el conjunto B será el conjunto pasivo:

Primera iteración:

- $\hat{A} = \{a_1, a_2, a_3, a_4\}$
- El mejor emparejamiento para cada elemento es:
  - $a_1 : b_3$
  - $a_2 : b_4$
  - $a_3 : b_3$
  - $a_4 : b_2$

a_1	(b_3)	b_1	b_4	b_2
a_2	(b_4)	b_2	b_1	b_3
a_3	(b_3)	b_1	b_2	b_4
a_4	(b_2)	b_3	b_4	b_1

b_1	a_1	a_3	a_2	a_4
b_2	a_3	a_2	(a_4)	a_1
b_3	(a_1)	a_2	a_4	(a_3)
b_4	a_4	a_1	(a_2)	a_3

- Como muestra la figura b_2 se empareja con a_4, b_3 tiene como opciones a a_1, a_3, b_4 se empareja con a_2.
- Ya que $1 = \psi_{b_3}(a_1) < \psi_{b_3}(a_3) = 4$, entonces b_3 se empareja con a_1.
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_3$
  - $E_{a_2} = b_4$
  - $E_{a_3} = \_$
  - $E_{a_4} = b_2$
  - $E_{b_1} = \_$
  - $E_{b_2} = a_4$
  - $E_{b_3} = a_1$
  - $E_{b_4} = a_2$
- Para a_3 su lista de rechazos pasa a ser $R_{a_3} = \{b_3\}$.

Segunda iteración:

- $\hat{A} = \{a_3\}$
- Las opciones para a_3 son $\{b_1, b_2, b_4\}$, de estas su mejor emparejamiento es b_1

a_1	(b_3)	b_1	b_4	b_2
a_2	(b_4)	b_2	b_1	b_3
a_3	~~b_3~~	(b_1)	b_2	b_4
a_4	(b_2)	b_3	b_4	b_1

b_1	a_1	(a_3)	a_2	a_4
b_2	a_3	a_2	(a_4)	a_1
b_3	(a_1)	a_2	a_4	~~a_3~~
b_4	a_4	a_1	(a_2)	a_3

- Ya que a_3 es la única opción para b_1, se emparejan
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_3$
  - $E_{a_2} = b_4$
  - $E_{a_3} = b_1$
  - $E_{a_4} = b_2$
  - $E_{b_1} = a_3$
  - $E_{b_2} = a_4$
  - $E_{b_3} = a_1$
  - $E_{b_4} = a_2$

Ya que si se hiciera otra iteración $\hat{A}$ sería vacío, entonces el algoritmo termina. Se obtiene así el emparejamiento:

$$P = \{(a_1, b_3), (a_2, b_4), (a_3, b_1), (a_4, b_2)\}.$$

Propiedad 7.1: El Algoritmo Gale-Shapley itera a lo más n^2 veces.

Demostración. Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo.

Sea $a \in A$, nótese que puede hacer una propuesta a lo más a n elementos de B, ya que $\#B = n$ y a no puede ser rechazado dos veces por el mismo elemento. Además, a solo puede ser rechazado por aquellos elementos que tengan emparejamiento. Por principio del palomar, solo $n-1$ elementos de B pueden tener una pareja que no sea a. Por lo tanto, el elemento a debe ser aceptado si participa en n iteraciones. Es decir, que después de n iteraciones donde a participe se asegura para la siguiente que $a \notin \hat{A}$. En conclusión a participa a lo más en n iteraciones.

Por lo tanto, si cada elemento de A puede participar máximo en n iteraciones, y existen n elementos de A. Entonces el algoritmo itera a lo más n^2 veces.

□

Lema 7.1: El Algoritmo Gale-Shapley devuelve un emparejamiento entre A y B.

Demostración. Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo.

Ya que el algoritmo no para hasta que el conjunto $\hat{A}$ sea vacío, el cual ingresa elementos de A que no tienen pareja, entonces al finalizar el algoritmo $\forall a \in A$ existe $b \in B$ tal que $E_a = b$. Además, debido que los elementos de B no pueden tener dos emparejamientos al mismo tiempo y que $\#A = \#B$. Por principio del palomar, no existen $a_i, a_j \in A$ con $a_i \neq a_j$ tal que $E_{a_i} = E_{a_j}$. Por lo tanto, $B = \{E_{a_1}, E_{a_2}, ..., E_{a_n}\}$. Defínase entonces:

$$P = \{(a_i, E_{a_i}) : i \in \{1, 2, ..., n\}, a_i \in A\}.$$

Ya que $B = \{E_{a_1}, E_{a_2}, ..., E_{a_n}\}$ y todos los E_{a_i} son distintos a pares, entonces P es un emparejamiento entre A y B.

□

7.2. Teorema del Matrimonio Estable

Teorema 7.1: Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Entonces, existe al menos un emparejamiento estable entre A y B.

Demostración. Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A

como activo y el conjunto B como pasivo para correr el algoritmo. Por Lema 7.1, se obtiene P, un emparejamiento entre A y B.

Por reducción al absurdo dígase que el emparejamiento P es inestable, es decir, existen $a, a' \in A$, $b, b' \in B$ tal que $(a, b), (a', b') \in P$ con $a \neq a'$, $b \neq b'$ y $\psi_a(b') < \psi_a(b)$, $\psi_{b'}(a) < \psi_{b'}(a')$. Entonces, b' es mejor emparejamiento para a que b, por lo tanto, está antes en su lista de preferencias.

Por diseño del algoritmo a propuso antes a b' que a b. Ahora, sea $H \subseteq A$ el conjunto de los elementos de A que propusieron a b'. Nótese que $a \in H$ ya que a propuso a b' antes que a b y también $a' \in H$, ya que $E_{b'} = a'$. Entonces, por diseño del algoritmo, se tiene que $\psi_{b'}(a') = min(\psi_{b'}(H))$. Entonces $\psi_{b'}(a') \leq \psi_{b'}(a)$. Pero $\psi_{b'}$ es biyectiva por definición, entonces $\psi_{b'}(a') < \psi_{b'}(a)$ $(\rightarrow\leftarrow)$ Lo cual contradice la hipótesis que $\psi_{b'}(a') < \psi_{b'}(a)$.

En conclusión, P no es un emparejamiento inestable, por lo tanto, P es un emparejamiento estable entre A y B. Además, por Propiedad 7.1, el algoritmo termina, por lo tanto el algoritmo de Gale-Shapley siempre devuelve un emparejamiento estable entre A y B.

□

CAPÍTULO 8

Variantes clásicas del Problema del Matrimonio Estable

En este capítulo se abordan dos variantes del problema del Matrimonio Estable presentado en el capítulo anterior. Las dos variantes se definen como la variante de conjuntos desiguales, es decir, que la cardinalidad del conjunto que propone y el conjunto pasivo no sean iguales y la variante de preferencias incompletas, donde las listas de preferencias no corresponden en tamaño a la cardinalidad del conjunto objetivo.

Traduciéndolo al idioma de grafos presentado en el capítulo 3, la variante de conjuntos desiguales se presentaría como el grafo bipartito donde su matriz de Biadyacencia esté llena de $1's$ pero que esta matriz no es cuadrada. Además, realizando el mismo ejercicio, la variante de preferencias incompletas se presenta como el grafo bipartito donde su matriz de Biadyacencia es cuadrada pero no necesariamente todos sus valores son 1 a diferencia de la variante clásica.

8.1. Conjuntos Desiguales

En esta sección se abordará la variante de Conjuntos Desiguales, esta variante busca generalizar el problema del Matrimonio Estable quitando la restricción que condiciona a que los conjuntos que participan deben ser de igual cardinalidad. La variante se abordará para los casos donde A es el conjunto que propone y es de menor cardinalidad al conjunto pasivo B. Asimismo, si A, el conjunto que propone, es de mayor cardinalidad que B. El caso donde son iguales se desarrolla en el capítulo 4.

Antes de iniciar con el desarrollo teórico de las variantes del Problema el Matrimonio Estable vale la pena redefinir conceptos del capítulo anterior, esto con el fin de poder generalizar el algoritmo para estas variantes. En particular se deben actualizar las definiciones 4.1 y 4.4, las cuales de ahora en adelante serán reemplazadas por las definiciones 5.1 y 5.3 correspondientemente.

8.1.1. $\#A < \#B$

Definición 8.1: Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$. Se le llama un emparejamiento entre A y B a:

P = {(a,b): $\in A, b \in B$}

donde:

- $\#P = min\{n, m\}$.
- $\forall a \in A$ no existen $b_i, b_j \in B$ tal que $b_i \neq b_j$ y $(a, b_i), (a, b_j) \in P$.
- $\forall b \in B$ no existen $a_i, a_j \in A$ tal que $a_i \neq a_j$ y $(a_i, b), (a_j, b) \in P$.

Ejemplo a Definición 8.1:

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$ y sea:

$$P = \{(a_1, b_2), (a_2, b_1), (a_3, b_4)\}.$$

Nótese que:

- $\#P = min\{3, 4\} = min\{\#A, \#B\} = 3$.
- $\forall a \in A$ no existen $b_i, b_j \in B$ tal que $b_i \neq b_j$ y $(a, b_i), (a, b_j) \in P$.
- $\forall b \in B$ no existen $a_i, a_j \in A$ tal que $a_i \neq a_j$ y $(a_i, b), (a_j, b) \in P$.

Por lo tanto, P es un emparejamiento.

Definición 8.2:

Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$, P un emparejamiento entre A y B, si $n < m$ se dice que $b \in B$ está desemparejado si no existe $a \in A$ tal que $(a, b) \in P$

Ejemplo a Definición 8.2

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$ y sea:

$$P = \{(a_1, b_2), (a_2, b_1), (a_3, b_4)\}$$

Nótese que P es un emparejamiento, por lo tanto, b_2 está desemparejado.

Definición 8.3:

Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A y sea P un emparejamiento entre A y B, P es un emparejamiento inestable si y solo si existen $(a, b), (a', b') \in P$ tal que:

- b' es mejor emparejamiento para a que b.
- a es mejor emparejamiento para b' que a'.

ó, si pasa que existe $(a, b) \in P$ tal que: si $n < m$ y existe $b' \in B$ desemparejado, donde b' es mejor emparejamiento para a que b. Asimismo, si $m < n$ y existe $a' \in A$ desemparejado donde a' es mejor emparejamiento para b que a.

Asimismo, un emparejamiento estable es un emparejamiento que no es emparejamiento inestable.

Ejemplo 1 a Definición 8.3:

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_4, b_2\}$
  - a_2: $\{b_4, b_2, b_1, b_3\}$
  - a_3: $\{b_1, b_2, b_3, b_4\}$.
- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2\}$
  - b_2: $\{a_3, a_2, a_1\}$
  - b_3: $\{a_1, a_2, a_3\}$
  - b_4: $\{a_1, a_2, a_3\}$.

Ahora, sea $P = \{(a_1, b_1), (a_2, b_3), (a_3, b_2)\}$. P es un emparejamiento, nótese el caso de (a_1, b_1), (a_2, b_3):

- b_3 es mejor emparejamiento para a_1 que b_1.
- a_1 es mejor emparejamiento para b_3 que a_2.

Por lo tanto, P es un emparejamiento inestable.

Ejemplo 2 a Definición 8.3:

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_4, b_2\}$
  - a_2: $\{b_4, b_2, b_1, b_3\}$
  - a_3: $\{b_1, b_2, b_3, b_4\}$.
- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2\}$
  - b_2: $\{a_3, a_2, a_1\}$
  - b_3: $\{a_1, a_2, a_3\}$

- b_4: $\{a_1, a_2, a_3\}$.

Ahora, sea $P = \{(a_1, b_4), (a_2, b_3), (a_3, b_2)\}$. P es un emparejamiento, nótese el caso de (a_1, b_4), b_1:

- b_1 es mejor emparejamiento para a_1 que b_4 y b_1 está desemparejado.

Por lo tanto, P es un emparejamiento inestable.

Ejemplo 3 a Definición 8.3:

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_4, b_2\}$
  - a_2: $\{b_4, b_2, b_1, b_3\}$
  - a_3: $\{b_1, b_2, b_3, b_4\}$.
- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2\}$
  - b_2: $\{a_3, a_2, a_1\}$
  - b_3: $\{a_1, a_2, a_3\}$
  - b_4: $\{a_1, a_2, a_3\}$.

Ahora, sea $P = \{(a_1, b_3), (a_2, b_4), (a_3, b_1)\}$. P es un emparejamiento y además cada elemento de A está con su mejor emparejamiento, es decir, que para estos elementos no existe un mejor emparejamiento, entonces no es un emparejamiento inestable, por lo tanto, P es un emparejamiento estable.

Algoritmo para el caso

Se utiliza el algoritmo Gale-Shapley presentado en el capítulo 4.

Ejemplo al Algoritmo para el caso $\#A < \#B$

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_4, b_2\}$
  - a_2: $\{b_4, b_2, b_1, b_3\}$
  - a_3: $\{b_3, b_1, b_2, b_4\}$.
- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2\}$
  - b_2: $\{a_3, a_2, a_1\}$
  - b_3: $\{a_1, a_2, a_3\}$

- b_4: $\{a_1, a_2, a_3\}$.

El conjunto A será el conjunto activo y el conjunto B será el conjunto pasivo:

Primera iteración:

- $\hat{A} = \{a_1, a_2, a_3\}$
- El mejor emparejamiento para cada elemento es:
  - $a_1 : b_3$
  - $a_2 : b_4$
  - $a_3 : b_3$

a_1	(b_3)	b_1	b_4	b_2
a_2	(b_4)	b_2	b_1	b_3
a_3	(b_3)	b_1	b_2	b_4

b_1	a_1	a_3	a_2
b_2	a_3	a_2	a_1
b_3	(a_1)	a_2	(a_3)
b_4	a_1	(a_2)	a_3

- Como muestra la figura b_3 tiene como opciones a a_1, a_3, b_4 se empareja con a_2
- Ya que $1 = \psi_{b_3}(a_1) < \psi_{b_3}(a_3) = 3$, entonces b_3 se empareja con a_1
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_3$
  - $E_{a_2} = b_4$
  - $E_{a_3} = \_$
  - $E_{b_1} = \_$
  - $E_{b_2} = \_$
  - $E_{b_3} = a_1$
  - $E_{b_4} = a_2$
- Para a_3 su lista de rechazos pasa a ser $R_{a_3} = \{b_3\}$.

Segunda iteración:

- $\hat{A} = \{a_3\}$
- Las opciones para a_3 son $\{b_1, b_2, b_4\}$, de estas su mejor emparejamiento es b_1

a_1	(b_3)	b_1	b_4	b_2
a_2	(b_4)	b_2	b_1	b_3
a_3	~~b_3~~	(b_1)	b_2	b_4

b_1	a_1	(a_3)	a_2
b_2	a_3	a_2	a_1
b_3	(a_1)	a_2	~~a_3~~
b_4	a_1	(a_2)	a_3

- Ya que a_3 es la única opción para b_1, se emparejan.
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_3$
  - $E_{a_2} = b_4$
  - $E_{a_3} = b_1$
  - $E_{b_1} = a_3$
  - $E_{b_2} = \_$
  - $E_{b_3} = a_1$
  - $E_{b_4} = a_2$

Ya que si se hiciera otra iteración $\hat{A}$ sería vacío, entonces el algoritmo termina. Se obtiene así el emparejamiento:

$$P = \{(a_1, b_3), (a_2, b_4), (a_3, b_1)\}.$$

Propiedad 8.1 El Algoritmo Gale-Shapley itera a lo más $m * n$ veces para el caso $\#A < \#B$.

Demostración. Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$, donde $\forall$ $a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall$ $b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo.

Sea $a \in A$, nótese que puede hacer una propuesta a lo más a m elementos de B, ya que $\#B = m$ y a no puede ser rechazado dos veces por el mismo elemento. Además, a solo puede ser rechazado por aquellos elementos que tengan emparejamiento. Por principio del palomar, solo $m - 1$ elementos de B pueden tener una pareja que no sea a. Por lo tanto, el elemento a debe ser aceptado si participa en m iteraciones. Es decir, que después de m iteraciones donde a participe se asegura para la siguiente que $a \notin \hat{A}$. En conclusión a participa a lo más en m iteraciones.

Por lo tanto, si cada elemento de A puede participar máximo en m iteraciones, y existen n elementos de A. Entonces el algoritmo itera a lo más $m * n$ veces. □

Lema 8.1: El Algoritmo Gale-Shapley devuelve un emparejamiento entre A y B para el caso $\#A < \#B$.

Demostración. Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$ y $n < m$, donde $\forall$ $a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall$ $b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo.

Ya que el algoritmo no para hasta que el conjunto $\hat{A}$ sea vacío, el cual ingresa elementos de A que no tienen pareja, entonces al finalizar el algoritmo $\forall a \in A$ existe $b \in B$ tal que $E_a = b$. Sea $B' = \{E_{a_i} : a_i \in A\}$. Nótese que no existen $a_i, a_j \in A$ con $a_i \neq a_j$ tal que $E_{a_i} = E_{a_j}$. Por lo tanto, $\#B' = n$. Ahora, defínase:

$$P = \{(a_i, E_{a_i}) : i \in \{1, 2, ..., n\}, a_i \in A\}.$$

Nótese entonces que $\#P = n = \#A = min\{m, n\}$, ya que los elementos de A no aparecen repetidos. Además, ya que $\#P = n$ y todos los elementos de A aparecen en alguna tupla de P, entonces por principio del palomar no existen $b_i, b_j \in B'$ con $b_i \neq b_j$ tal que $(a, b_i), (a, b_j) \in P$

para algún $a \in A$. Además, todos los elementos de B' aparecen en P, por lo tanto, también por el principio del palomar no existen $a_i, a_j \in A$ con $a_i \neq a_j$ tal que $(a_i, b), (a_j, b) \in P$ para algún $b \in B$. Entonces, P es un emparejamiento.

□

Teorema 8.1: Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$, $n < m$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Entonces, existe al menos un emparejamiento estable entre A y B.

Demostración. Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$ y $n < m$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo. Por Lema 8.1, se obtiene P, un emparejamiento entre A y B.

Por reducción al absurdo dígase que el emparejamiento es inestable, es decir, existen $a, a' \in A$, $b, b' \in B$ tal que $(a, b), (a', b') \in P$ con $a \neq a'$, $b \neq b'$ y $\psi_a(b') < \psi_a(b)$, $\psi_{b'}(a) < \psi_{b'}(a')$.

Por diseño del algoritmo a propuso antes a b' que a b. Ahora, sea $H \subseteq A$ el conjunto de los elementos de A que propusieron a b'. Nótese que $a \in H$ ya que a propuso a b' antes que a b y también $a' \in H$, ya que $E_{b'} = a'$. Entonces, por diseño del algoritmo, se tiene que $\psi_{b'}(a') = min(\psi_{b'}(H))$. Entonces $\psi_{b'}(a') \leq \psi_{b'}(a)$. Pero $\psi_{b'}$ es biyectiva por definición, entonces $\psi_{b'}(a') < \psi_{b'}(a)$ $(\rightarrow\leftarrow)$ Lo cual contradice la hipótesis que $\psi_{b'}(a') < \psi_{b'}(a)$.

Ahora, tomando la segunda opción de la definición 8.4 para que el emparejamiento sea inestable, propóngase que existen $a \in A$, $b, b' \in B$ tal que $(a, b) \in P$, b' desemparejado y $\psi_a(b') < \psi_a(b)$. Por diseño de algoritmo b' tuvo que proponer antes a a que a b, pero ya que b' está desemparejado significa que rechazó a a sin tener pareja $(\rightarrow\leftarrow)$, lo cual contradice el diseño del algoritmo ya que si se propone a un elemento y este no tiene pareja, debe aceptar.

En conclusión, P no es un emparejamiento inestable, por lo tanto, P es un emparejamiento estable. Además, por Propiedad 8.1, el algoritmo termina, por lo tanto el algoritmo de Gale-Shapley siempre devuelve un emparejamiento estable entre A y B.

□

8.1.2. $\#A > \#B$

Algoritmo para el caso $\#A > \#B$

Se ajusta el algoritmo Gale-Shapley presentado en el capítulo 4, en este caso existe un nuevo criterio para que el algoritmo se detenga, el diseño ajustado se presenta a continuación:

Algoritmo: Sean dos conjuntos A, B tal que $\#A = n$,$\#B = m$ donde $n, m \in \mathbb{Z}^+$, donde $\forall$ $a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A y sea P un emparejamiento entre A y B. Por lo tanto, todos los elemendos de A y B presentan listas de preferencias para el otro conjunto.

Para este algoritmo un conjunto será denominado como conjunto activo, en este caso el conjunto A. Mientras que el otro conjunto será denominado como conjunto pasivo, en este caso el conjunto B. Esto debido a que el conjunto A será el encargado de realizar las propuestas y el conjunto B el encargado de aceptar o rechazar estas propuestas.

Para cada iteración cada elemento a del conjunto A tendrá la siguiente información:

- $E_a = \_$, emparejamiento actual, puede tener a lo más un elemento del conjunto B, o estar vacío.
- $L_a = \{b_i, ..., b_j\}$, lista de preferencias, esta lista está dada por ψ_a, el ranking de B para a. Esta lista es constante, no cambia durante las iteraciones.
- $R_a = \{\}$, lista de rechazos, esta es un subconjunto de B. Contiene todos los elementos de B a los cuales a les propuso y no están en E_a en la iteración actual.

Para cada iteración cada elemento b del conjunto B tendrá la siguiente información:

- $E_b = \_$, emparejamiento actual, puede tener a lo más un elemento del conjunto A, o estar vacío.
- $L_b = \{b_i, ..., b_j\}$, lista de preferencias, esta lista está dada por ψ_b, el ranking de A para b. Esta lista es constante, no cambia durante las iteraciones.
- $C_b = \{\}$ lista de propuestas actuales. Subconjunto de A, el cual comprende los elementos de A que propusieron a b en la iteración actual.

Además se tiene en general el conjunto U, donde $U \subset A$, a este entran los elementos de A que ya fueron rechazados por todos los elementos de B, es decir, los elementos desemparejados.

Cada iteración comprende los siguientes pasos:

1. Se identifican los elementos de A cuyo emparejamiento actual E_a sea vacío. Llámese a este conjunto $\hat{A}$.
2. $\hat{A} = \hat{A} \cap (U^c)$.
3. Para cada $a \in \hat{A}$ se obtiene una lista de sus posibles parejas, dada por $(B - R_a)$.
4. Para cada $a \in \hat{A}$ se obtiene su mejor emparejamiento: $b_a = \psi_a^{-1}(min(\psi_a(B - R_a)))$.
5. Cada $a \in \hat{A}$ entra al conjunto C_{b_a} del elemento de b_a.
6. Se identifican los elementos de B cuya lista de propuestas actuales C_b no sea vacía. Llámese a este conjunto $\hat{B}$.
7. Para cada $b \in \hat{B}$ se obtiene una lista de posibles parejas, dada por $C_b \cup \{E_b\}$.
8. Para cada $b \in \hat{B}$ se obtiene al mejor emparejamiento $a_b = \psi_b^{-1}(min(\psi_b(C_b \cup \{E_b\})))$.
9. Para cada $b \in \hat{B}$ $E_b = a_b$ y $E_{a_b} = b$.
10. Para cada $a \in \hat{A}$ rechazado $E_a = \_$.
11. Para cada $b \in \hat{B}$, y para todo $a \in (C_b - \{E_b\})$, $R_a = R_a \cup \{b\}$.
12. Para cada $b \in \hat{B}$ $C_b = \emptyset$.
13. Para cada $a \in \hat{A}$, si $\#R_a = m$ entonces: $U = U \cup \{a\}$.

El algoritmo termina cuando al terminar una iteración $\#U = n - m$.

Ejemplo al Algoritmo para el caso $\#\mathbf{A} > \#\mathbf{B}$

Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3\}$ y considérense las listas de preferencias correspondientes a los rankings de cada elemento:

- Listas de preferencias para B de los elementos de A
  - a_1: $\{b_3, b_1, b_2\}$
  - a_2: $\{b_2, b_1, b_3\}$
  - a_3: $\{b_3, b_1, b_2\}$
  - a_4: $\{b_2, b_3, b_1\}$.
- Listas de preferencias para A de los elementos de B
  - b_1: $\{a_1, a_3, a_2, a_4\}$
  - b_2: $\{a_3, a_2, a_4, a_1\}$
  - b_3: $\{a_1, a_2, a_4, a_3\}$.

El conjunto A será el conjunto activo y el conjunto B será el conjunto pasivo:

Primera iteración:

- $\hat{A} = \{a_1, a_2, a_3, a_4\}$
- El mejor emparejamiento para cada elemento es:
  - $a_1 : b_3$
  - $a_2 : b_2$
  - $a_3 : b_3$
  - $a_4 : b_2$

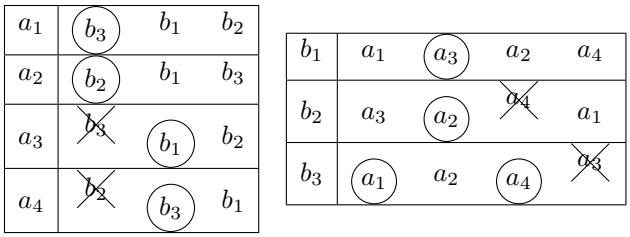
a_1	(b_3)	b_1	b_2
a_2	(b_2)	b_1	b_3
a_3	(b_3)	b_1	b_2
a_4	(b_2)	b_3	b_1

b_1	a_1	a_3	a_2	a_4
b_2	a_3	(a_2)	(a_4)	a_1
b_3	(a_1)	a_2	a_4	(a_3)

- Ya que $2 = \psi_{b_2}(a_2) < \psi_{b_2}(a_4) = 3$, entonces b_2 se empareja con a_2
- Ya que $1 = \psi_{b_3}(a_1) < \psi_{b_3}(a_3) = 4$, entonces b_3 se empareja con a_1
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_3$
  - $E_{a_2} = b_2$
  - $E_{a_3} = \_$
  - $E_{a_4} = \_$
  - $E_{b_1} = \_$
  - $E_{b_2} = a_2$
  - $E_{b_3} = a_1$
- Para a_3 su lista de rechazos pasa a ser $R_{a_3} = \{b_3\}$ y para a_4 su lista de rechazos pasa a ser $R_{a_4} = \{b_2\}$.

Segunda iteración:

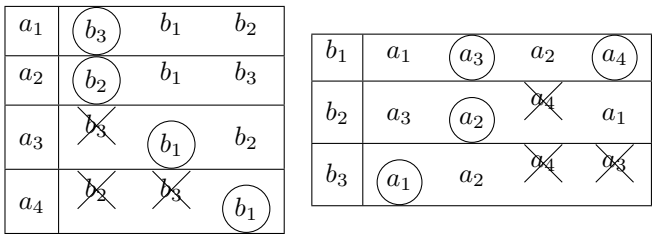
- $\hat{A} = \{a_3, a_4\}$
- Las opciones para a_3 son $\{b_1, b_2\}$, de estas su mejor emparejamiento es b_1
- Las opciones para a_4 son $\{b_3, b_1\}$, de estas su mejor emparejamiento es b_3



- Ya que a_3 es la única opción para b_1, se emparejan
- Ya que a_1 es mejor emparejamiento para b_3 que a_4, entonces b_3 se mantiene con su emparejamiento actual
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_3$
  - $E_{a_2} = b_2$
  - $E_{a_3} = b_1$
  - $E_{a_4} = \_$
  - $E_{b_1} = a_3$
  - $E_{b_2} = a_2$
  - $E_{b_3} = a_1$
- $R_{a_4} = \{b_2, b_3\}$.

Tercera iteración:

- $\hat{A} = \{a_4\}$
- Las opciones para a_4 son $\{b_1\}$, de estas su mejor emparejamiento es b_1



- Ya que a_3 es mejor emparejamiento para b_1 que a_4, entonces b_1 se mantiene con su emparejamiento actual
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_3$
  - $E_{a_2} = b_2$

- $E_{a_3} = b_1$
- $E_{a_4} = \_$
- $E_{b_1} = a_3$
- $E_{b_2} = a_2$
- $E_{b_3} = a_1$

- $R_{a_4} = \{b_2, b_3, b_1\}$
- Ya que $\#R_{a_4} = m = \#B$, entonces $U = \{a_4\}$.

Ya que $\#U = 1 = 4 - 3 = n - m$, el algoritmo termina y el emparejamiento obtenido es:

$$P = \{(a_1, b_3), (a_2, b_2), (a_3, b_1)\}.$$

Propiedad 8.2 El Algoritmo Gale-Shapley ajustado para el caso $\#A > \#B$ itera a lo más $m * n$ veces.

Demostración. Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$, $n > m$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo.

Sea $a \in A$, nótese que puede hacer una propuesta a lo más a m elementos de B, ya que $\#B = m$ y a no puede ser rechazado dos veces por el mismo elemento. Además, a solo puede ser rechazado por aquellos elementos que tengan emparejamiento. Por principio del palomar, solo m elementos de B pueden tener una pareja que no sea a. Por lo tanto, el elemento a debe ser aceptado si participa en m iteraciones o quedar en el conjunto de desemparejados. Es decir, que después de m iteraciones donde a participe se asegura para la siguiente que $a \notin \hat{A}$ ó $a \in U$, y si está en U ya no participa en la siguiente iteración. En conclusión a participa a lo más en m iteraciones.

Por lo tanto, si cada elemento de A puede participar máximo en m iteraciones, y existen n elementos de A. Entonces el algoritmo itera a lo más $m * n$ veces. □

Lema 8.2: El Algoritmo Gale-Shapley devuelve un emparejamiento entre A y B para el caso $\#A > \#B$.

Demostración. Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$ y $n > m$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo.

Ya que el algoritmo no para hasta que $\#U = n - m$, el cual ingresa elementos de A que ya fueron rechazados por todos los elementos de B, entonces al finalizar el algoritmo $\forall b \in B$ existe $a \in A$ tal que $E_b = a$. Sea $A' = \{E_{b_i} : b_i \in B\}$. Nótese que no existen $b_i, b_j \in B$ con $b_i \neq b_j$ tal que $E_{b_i} = E_{b_j}$. Por lo tanto, $\#A' = m$. Ahora, defínase:

$$P = \{(E_{b_i}, b_i) : i \in \{1, 2, ..., m\}, b_i \in B\}$$

Nótese entonces que $\#P = m = \#B = minm, n$, ya que los elementos de B no aparecen repetidos. Además, ya que $\#P = m$ y todos los elementos de B aparecen en alguna tupla de P, entonces por principio del palomar no existen $a_i, a_j \in A'$ con $a_i \neq a_j$ tal que $(a_i, b), (a_j, b) \in P$ para algún $b \in B$. Además, todos los elementos de A' aparecen en P, por lo tanto, también por el

principio del palomar no existen $b_i, b_j \in B$ con $b_i \neq b_j$ tal que $(a, b_i), (a, b_j) \in P$ para algún $b \in B$. Entonces, P es un emparejamiento.

□

Teorema 8.2: Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$, $n > m$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Entonces, existe al menos un emparejamiento estable entre A y B.

Demostración. Sean dos conjuntos A, B tal que $\#A = n$, $\#B = m$ donde $m, n \in \mathbb{Z}^+$ y $n < m$, donde $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j} ranking de b_j para A. Tómese el conjunto A como activo y el conjunto B como pasivo para correr el algoritmo Gale-Shapley ajustado. Por Lema 8.2, se obtiene P, un emparejamiento entre A y B.

Por reducción al absurdo dígase que el emparejamiento es inestable, es decir, existen $a, a' \in A$, $b, b' \in B$ tal que $(a, b), (a', b') \in P$ con $a \neq a'$, $b \neq b'$ y $\psi_a(b') < \psi_a(b)$, $\psi_{b'}(a) < \psi_{b'}(a')$.

Por diseño del algoritmo a propuso antes a b' que a b. Ahora, sea $H \subseteq A$ el conjunto de los elementos de A que propusieron a b'. Nótese que $a \in H$ ya que a propuso a b' antes que a b y también $a' \in H$, ya que $E_{b'} = a'$. Entonces, por diseño del algoritmo, se tiene que $\psi_{b'}(a') = min(\psi_{b'}(H))$. Entonces $\psi_{b'}(a') \leq \psi_{b'}(a)$. Pero $\psi_{b'}$ es biyectiva por definición, entonces $\psi_{b'}(a') < \psi_{b'}(a)$ $(\rightarrow\leftarrow)$. Lo cual contradice la hipótesis que $\psi_{b'}(a') < \psi_{b'}(a)$.

Ahora, tomando la segunda opción de la definición 8.4 para que el emparejamiento sea inestable, propóngase que existen $a, a' \in A$, $b \in B$ tal que $(a, b) \in P$, a' desemparejado y $\psi_b(a') < \psi_b(a)$. Por diseño de algoritmo a' tuvo que proponer antes a b que a a, pero a la vez $a' \in U$, el conjunto de los desemparejados, por lo tanto, $\forall b' \in B$, b' rechazó a a', en particular b rechazó a a' $(\rightarrow\leftarrow)$, lo cual contradice el diseño del algoritmo ya que si a' es mejor emparejamiento para b que a, b está obligado a emparejarse con a' sobre a, si a' le propone.

En conclusión, P no es un emparejamiento inestable, por lo tanto, P es un emparejamiento estable, además, por Propiedad 8.2, el algoritmo termina, por lo tanto, el algoritmo de Gale-Shapley ajustado siempre devuelve un emparejamiento estable entre A y B.

□

8.2. Preferencias incompletas

En esta sección se tratará el caso donde los elementos de A y B no cuentan con una lista completa de preferencias. Primero se trata el caso donde estas listas no presentan un orden en específico. Es decir para un elemento $a \in A$ todos los elementos $b \in B$ que aparezcan en su lista son equivalentes. Este caso se tratará como un grafo bipartito y así se podrán aprovechar los conceptos y propiedades obtenidos en el capítulo 3. Luego se expondrá el caso donde además de preferencias incompletas, estas preferencias están ordenadas.

8.2.1. Caso donde hay lista de preferencias incompletas

Para este caso se utilizará la perspectiva desde los grafos. Ahora se tienen los conjuntos A y B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$. Además cada elemento $a \in A$ presenta un conjunto $B_a \subseteq B$ de preferencias, y también cada elemento $b \in B$ presenta un conjunto $A_b \subseteq A$ de preferencias. Para traducirlo a grafos, dígase que $V = A \cup B$ es el conjunto de vértices y $E = \{(a, b) : a \in A_b, b \in B_a\}$,

es decir, la arista existe si a está en las preferencias de b y viceversa. Así se obtiene un grafo bipartito y se pueden aprovechar las propiedades de este y de su Matriz de Biadyacencia correspondiente.

Definición 8.4 Sean A y B tal que $\#A = \#B = n$, donde $n \in \mathbb{Z}^+$. Además, cada elemento $a \in A$ y $b \in B$ cuenta con sus conjuntos de preferencias B_a, A_b correspondientes. Y sea M una matriz de $n \times n$ tal que:

- $M_{i,j} = 1$, si $a_i \in A_{b_j}$ y $b_j \in A_{a_i}$.
- $M_{i,j} = 0$, en caso contrario.

Esta matriz es la matriz de Biadyacencia entre A y B.

Ejemplo a Definición 8.4

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, donde cada elemento tiene su conjunto de preferencia correspondiente:

- $B_{a_1} = \{b_1, b_2, b_3\}$
- $B_{a_2} = \{b_2\}$
- $B_{a_3} = \{b_2, b_3\}$

- $A_{b_1} = \{a_1, a_2\}$
- $A_{b_2} = \{a_1, a_2, a_3\}$
- $A_{b_3} = \{a_3\}$.

Nótese entonces que:

$$M_{1,1} = M_{1,2} = M_{2,2} = M_{3,2} = M_{3,3} = 1.$$

$$M_{1,3} = M_{2,1} = M_{2,3} = M_{3,1} = 0.$$

Entonces:

$$M_{=} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

es la matriz de Biadyacencia entre A y B.

Teorema 8.3: Sean A y B tal que $\#A = \#B = n$, donde $n \in \mathbb{Z}^+$. Además, cada elemento $a \in A$ y $b \in B$ cuenta con sus conjuntos de preferencias B_a, A_b correspondientes. Entonces existe un emparejamiento P entre A y B tal que $\forall (a, b) \in P$ $a \in A_b$, $b \in B_a$ ssi $perm(M) > 0$.

Demostración. Sean A y B tal que $\#A = \#B = n$, donde $n \in \mathbb{Z}^+$. Además, cada elemento $a \in A$ y $b \in B$ cuenta con sus conjuntos de preferencias B_a, A_b correspondientes.

($\Rightarrow$) Sea P un emparejamiento entre A y B tal que $\forall(a,b) \in P$ $a \in A_b$, $b \in B_a$. Nótese entonces que existe $\sigma \in S_n$ tal que $P = \{(a_i, b_{\sigma(i)}) : i \in \{1,2,...,n\}\}$, esto debido a que P es un emparejamiento y $\#P = n$. Nótese entonces que $\prod_{i=1}^{n} M_{i,\sigma(i)} = \prod_{i=1}^{n} 1 = 1$. Ya que todos los términos que entran a la sumatoria en la fórmula del permanente de M son no negativos, entonces:

$$perm(M) \sum_{\sigma \in \S_n} \prod_{i=1}^{n} M_{i,\sigma(i)} \geq 1 > 0.$$

($\Leftarrow$) Supóngase que $perm(M) > 0$. Por definición de permanente y de matriz de Biadyacencia entre A y B existe $\sigma \in S_n$ tal que $\prod_{i=1}^{n} M_{i,\sigma(i)} = 1$. Sea $P = \{(a_i, b_{\sigma(i)}) : i \in \{1,2,...,n\}\}$. Nótese que $\forall i \in \{1,2,...,n\}$ $M_{i,\sigma(i)} = 1$. Es decir, $a_i \in A_{b_{\sigma(i)}}$ y $b_{\sigma(i)} \in B_{a_i}$ $\forall i \in \{1,2,...,n\}$. Además P es un emparejamiento entre A y B.

□

Ejemplo a Teorema 8.3

Tómese el ejemplo a Definición 8.4, por lo tanto se tiene la Matriz de Biadyacencia entre A y B:

$$M_{=} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Ahora:

- $P = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$ es un emparejamiento entre A y B y se cumple que cada elemento está en el conjunto de preferencia de pareja correspondiente.
- $perm(M) = 1 > 0$.

Propiedad 8.3 Sean A y B tal que $\#A = \#B = n$, donde $n \in \mathbb{Z}^+$. Además, cada elemento $a \in A$ y $b \in B$ cuenta con sus conjuntos de preferencias B_a, A_b correspondientes. Entonces si $perm(M) > 0$, se puede encontrar un emparejamiento entre A y B donde cada elemento esté en el conjunto de preferencia de su pareja correspondiente a lo más en $n!$ iteraciones.

Demostración. Si $perm(M) > 0$. Por Teorema 8.3 existe $P = \{(a_i, b_{\sigma(i)}) : i \in \{1,2,...,n\}\}$ emparejamiento entre A y B donde cada elemento esté en el conjunto de preferencia de su pareja correspondiente. Ahora, defínase el algoritmo donde cada iteración corresponde a verificar si $\sigma \in S_n$ y $\prod_{i=1}^{n} M_{i,j} = 1$. Ya que hay $\#S_n = n!$ Entonces encontrar la permutación $\sigma \in S_n$ que genera el emparejamiento P toma a lo más $n!$ iteraciones. □

Nota 8.1: Nótese que por el teorema 8.3 y la propiedad 8.3 existe un procedimiento para saber si existe un emparejamiento entre A y B con preferencias incompletas donde los elementos están emparejados con un elemento en su conjunto de preferencias se necesita obtener la matriz de Biadyacencia entre A y B y conocer si su permanente es positivo. En ese caso existe un algoritmo para obtener el emparejamiento mencionado.

8.2.2. Caso donde hay lista de preferencias ordenadas además de preferencias incompletas

En este caso se combina el problema clásico del Matrimonio Estable con las preferencias incompletas, agregando la complejidad de tener un orden en las preferencias. Con estas condiciones se agrega la complicación de obtener un emparejamiento estable, bajo su definición en la variante clásica, con listas de preferencias incompletas. Para abordar esto se necesita redefinir el ranking y la lista de preferencias, lo cual se realiza en la Definición 8.4, además se necesita definir el concepto de emparejamiento aceptable, el cual busca emular el emparejamiento entre dos conjuntos pero que se cumpla la condición que cada elemento está en la lista de preferencias de su pareja.

Definición 8.4: Sean dos conjuntos A, B tal que $\#B = n$ donde $n \in \mathbb{Z}^+$, sea $m \in \mathbb{Z}^+$ tal que $m \leq n$. Sea $a \in A$, La función biyectiva:

$$\psi_a : B' \longrightarrow \{1, 2, ..., m-1, m\}$$

donde $B' \subseteq B$, es un ranking de a para B.

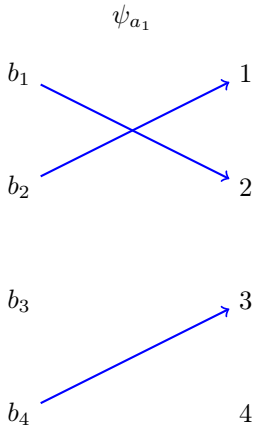
Asimismo, una lista de preferencias de a para B está dada por:

$$B_a = \{b_i, ..., b_j\}$$

donde el primer elemento corresponde al que hace que $\psi_a(b_i) = 1$, así sucesivamente hasta el último elemento que corresponde al que hace que $\psi_a(b_j) = m$.

De manera idéntica se definen un ranking de b para A y una lista de preferencias de b para A con $b \in B$.

Ejemplo a definición 8.4: Sean $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$. Un ranking de a_1 para B sería dado por $\psi_{a_1} : \{b_1, b_2, b_4\} \longrightarrow \{1, 2, 3\}$, tal que:



Por lo tanto, la lista de preferencias de a_1 para B se describe: $B_{a_1} = \{b_2, b_1, b_4\}$.

Definición 8.5 Sean dos conjuntos A, B tal que $\#A = \#B = n$ donde $n \in \mathbb{Z}^+$ y $\forall\ a_i \in A$ existe ψ_{a_i} ranking de a_i para B. Además, $\forall\ b_j \in B$ existe ψ_{b_j}. Se le llama un emparejamiento aceptable entre A y B a:

$$P = \{(a, b) : a \in A, b \in B\}$$

donde:

- $\#P = n$,
- $\forall a \in A$ existe $b \in B$ tal que $(a, b) \in P$ y existe $k \in \mathbb{Z}^+$ tal que $\psi_a(b) = k$,
- $\forall b \in B$ existe $a \in A$ tal que $(a, b) \in P$ y existe $k \in \mathbb{Z}^+$ tal que $\psi_b(a) = k$.

Ejemplo a Definición 8.5

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, donde cada elemento tiene su conjunto de preferencia correspondiente:

- $B_{a_1} = \{b_1, b_2, b_3\}$
- $B_{a_2} = \{b_2\}$
- $B_{a_3} = \{b_2, b_3\}$

- $A_{b_1} = \{a_1, a_2\}$
- $A_{b_2} = \{a_3, a_2, a_1\}$
- $A_{b_3} = \{a_3\}$.

Nótese entonces que:

$$P = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$$

es un emparejamiento aceptable entre A y B.

Ejemplo Algoritmo Gale-Shapley

Sean $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, donde cada elemento tiene su conjunto de preferencia correspondiente:

- $B_{a_1} = \{b_1, b_2, b_3\}$
- $B_{a_2} = \{b_2\}$
- $B_{a_3} = \{b_2, b_3\}$

- $A_{b_1} = \{a_1, a_2\}$
- $A_{b_2} = \{a_3, a_2, a_1\}$
- $A_{b_3} = \{a_3\}$.

El conjunto A será el conjunto activo y el conjunto B será el conjunto pasivo.

Primera iteración:

- $\hat{A} = \{a_1, a_2, a_3\}$

- El mejor emparejamiento para cada elemento es:
  - $a_1 : b_1$
  - $a_2 : b_2$
  - $a_3 : b_2$

a_1	(b_1)	b_2	b_3
a_2	(b_2)		
a_3	(b_2)	b_3	

b_1	(a_1)	a_2	
b_2	(a_3)	(a_2)	a_1
b_3	a_3		

- Como muestra la figura b_2 tiene como opciones a a_2, a_3, b_1 se empareja con a_1.
- Ya que $1 = \psi_{b_2}(a_3) < \psi_{b_2}(a_2) = 2$, entonces b_2 se empareja con a_2.
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_1$
  - $E_{a_2} = \_$
  - $E_{a_3} = b_2$
  - $E_{b_1} = a_1$
  - $E_{b_2} = a_3$
  - $E_{b_3} = \_$
- Para a_3 su lista de rechazos pasa a ser $R_{a_3} = \{b_3\}$.

Ahora, nótese que a_2 no está emparejado, pero tampoco tiene a nadie disponible en su conjunto de preferencias, por lo tanto, al intentar correr nuevamente el algoritmo a_2 siempre será desemparejado, ya que no tiene a quien proponerle. Por lo tanto, el algoritmo no devuelve un emparejamiento aceptable.

Discusión del caso

El ejemplo del uso del algoritmo junto con el ejemplo a la Definición 8.5 deja en evidencia que a pesar de tener asegurado un emparejamiento aceptable entre los conjuntos A y B no necesariamente se puede encontrar un emparejamiento estable, es más, utilizando el algoritmo que promete un emparejamiento estable en su variante clásica ni siquiera logra un emparejamiento aceptable. Se puede notar que el hecho que las preferencias estén en orden agrega mucha complejidad al problema de las preferencias incompletas. Encontrar las condiciones que aseguren un emparejamiento estable con un respectivo algoritmo es una tarea de interés para un trabajo posterior a este, debido a que se pueden obtener muchos beneficios y se pueden encontrar muchas aplicaciones para este caso tan general.

CAPÍTULO 9

Variante Preferencias por carecterísticas

Definición 9.1: Sean A, $a \in A$ y sea $v_a \in \{0,1\}^n$, donde $n \in \mathbb{Z}^+$. A este vector se le llama el vector de características de a.

Ejemplo a Definición 9.1

Sean $A = \{a_1, a_2, a_3\}$ y sea:

$$v_{a_1} = (1, 0, 1, 1)$$

Este es el vector de características de a_1.

Definición 9.2 Sean A, $a_1, a_2 \in A$, $v_{a_1}, v_{a_2} \in \{0,1\}^n$ tal que v_{a_1} es el vector de características de a_1 y v_{a_2} es el vector de características de a_2. Se dice que:

$$v_{a_1} = (v_1, v_2, ..., v_n) >_L (v'_1, v'_2, ..., v'_n) = v_{a_2}$$

ssi $v_1 > v'_1$ ó $(v_2, ..., v_n) >_L (v'_2, ..., v'_n)$.

Asimismo, se dice que:

$$v_{a_1} =_L v_{a_2}$$

ssi $v_{a_1} = v_{a_2}$.

Ahora,

$$v_{a_1} \geq_L v_{a_2}$$

ssi $v_{a_1} >_L v_{a_2}$ o $v_{a_1} =_L v_{a_2}$.

Además se dice que:

$$v_{a_1} <_L v_{a_2}$$

ssi $v_{a_1} \not\geq_L v_{a_2}$.

Ejemplo 1 a Definición 9.2

Sean A, $a_1, a_2 \in A$, $v_{a_1} = (v_1, v_2, v_3, v_4) = (1, 0, 0, 1)$, $v_{a_2} = (v'_1, v'_2, v'_3, v'_4) = (1, 0, 1, 1)$

Ahora, se tiene que $v_1 = v'_1$. Por lo tanto se procede con $(v_2, v_3, V_4) = (0, 0, 1)$, $(v'_2, v'_3, v'_4) = (0, 1, 1)$

Nótese que $v_2 = v'_2$. Por lo tanto se procede con $(v_3, v_4) = (0, 1)$, $(v'_3, v'_4) = (1, 1)$

Nótese que $v'_3 = 1 > 0 = v_3$. Por lo tanto:

$$v_{a_2} >_L v_{a_1}.$$

Entonces:

$$v_{a_2} \geq_L v_{a_1}.$$

Ejemplo 2 a Definición 9.2

Sean A, $a_1, a_2 \in A$, $v_{a_1} = (v_1, v_2, v_3, v_4) = (1, 0, 0, 1)$, $v_{a_2} = (v'_1, v'_2, v'_3, v'_4) = (1, 0, 0, 1)$.

Nótese que $v_{a_1} = v_{a_2}$, entonces: $v_{a_1} =_L v_{a_2}$.

Por lo tanto:

$$v_{a_1} \geq_L v_{a_2}$$

Propiedad 9.1: $\geq_L$ es una relación de orden parcial en $\{0, 1\}^n$ donde $n \in \mathbb{Z}^+$.

Demostración. (Reflexividad) Sea $v \in \{0, 1\}^n$. Nótese que si $v = v$, entonces $v =_L v$. Por lo tanto, $v \geq_L v$.

(Antisimetría) Sean $v_1, v_2 \in \{0, 1\}^n$ tal que $v_1 \geq_L v_2$, $v_2 \geq_L v_1$.

Ya que $v_1 \geq_L v_2$ entonces existen dos casos:

- $v_1 =_L v_2$, entonces: $v_1 = v_2$.
- $v_1 >_L v_2$. Por lo tanto, existe $k \in \{1, 2, ..., n\}$, tal que $\forall k' \in \mathbb{Z}^+$ donde $k' < k$ se tiene $v_{1'_k} = v_{2'_k}$ y $v_{1_k} > v_{2_k}$, entonces se tiene que $v_1 \neq v_2$. Además, nótese que $v_2 <_L v_1$, por lo tanto, por definición $v_2 \not\geq_L v_1$, pero la hipótesis es que $v_2 \geq_L v_1$. Por lo tanto, este caso no es viable.

En cualquiera de los dos casos se tiene que: $v_1 =_L v_2$.

(Transitividad) Sean $v_1, v_2, v_3 \in \{0,1\}^n$ tal que $v_1 \geq_L v_2$ y $v_2 \geq v_3$. Para esto existen 4 casos posibles:

- $v_1 =_L v_2 =_L v_3$. Entonces: $v_1 = v_2 = v_3$, por lo tanto, $v_1 =_L v_3$, entonces: $v_1 \geq_L v_3$.
- $v_1 =_L v_2$ y $v_2 >_L v_3$. Por lo tanto, $v_1 = v_2$ entonces $v_1 >_L v_3$, entonces: $v_1 \geq_L v_3$.
- $v_1 >_L v_2$ y $v_2 =_L v_3$. Por lo tanto, $v_2 = v_3$ entonces $v_1 >_L v_3$, entonces: $v_1 \geq_L v_3$.
- $v_1 >_L v_2 >_L v_3$. Entonces existen $k_1, k_2 \in \{1, 2, ..., n\}$, tal que $\forall k_1' \in \mathbb{Z}^+$ donde $k_1' < k_1$ y $\forall k_2' \in \mathbb{Z}^+$ donde $k_2' < k_2$ se tiene $v_{1_{k_1'}} = v_{2_{k_1'}}$, $v_{2_{k_2'}} = v_{3_{k_2'}}$, $v_{1_{k_1}} > v_{2_{k_1}}$ y $v_{2_{k_2}} > v_{3_{k_2}}$. Nótese que $k_1 \neq k_2$, ya que el vector es binario, entonces no se puede tener $v_{1_{k_1}} > v_{2_{k_1}} > v_{3_{k_1}}$. Por lo tanto, hay dos subcasos:
  - $k_1 > k_2$, entonces $v_{1_{k_2}} = v_{2_{k_2}} > v_{3_{k_2}}$. Por lo tanto, $v_1 >_L v_3$, entonces: $v_1 \geq_L v_3$.
  - $k_1 < k_2$, entonces $v_{1_{k_1}} > v_{2_{k_1}} = v_{3_{k_1}}$. Por lo tanto, $v_1 >_L v_3$, entonces: $v_1 \geq_L v_3$.

Entonces: $\geq_L$ es relación de orden parcial en $\{0,1\}^n$.

□

Definición 9.3 Sean A y B tal que $\#A = \#B = m$. Sea C_B tal que $\#C_B = n$, el conjunto C_B se le llama conjunto de características de B. Ahora, sea $a \in A$, entonces hágase un ranking del conjunto C_B para a, y sea $c_a = \{c_1, c_2, ..., c_n\}$ la lista de preferencias de C_B para a, donde: $\psi_a(c_i) < \psi_a(c_j)$ si $i < j$ y para cada c_i, $c_i \in C_B$.

Definición 9.4 Sea A tal que $\#A = m$ y C_A el conjunto de características de A. Sea $a \in A$ entonces $h_a :_A \rightarrow \{0,1\}$ es la función de características de a. Donde se dice que si a cuenta con la característica c entonces $h_a(c) = 1$, de lo contrario $h_a(c) = 0$.

Definición 9.5 Sean A y B tal que $\#A = \#B = m$. Sea C_b el conjunto de características de B tal que $\#C_B = n$. Sean $a \in A$, $b \in B$, $c_a = \{c_1, c_2, ..., c_n\}$ la lista de preferencias de C_B para a, h_b la función de características de b y $v_{(a,b)} \in \{0,1\}^n$ tal que $v_{(a,b)} = (h_b(c_1), h_b(c_2), ..., h_b(c_n))$ es el vector de características de b asociado a a.

Definición 9.6 Sean A y B tal que $\#A = \#B = m$. Sea $a \in A$, y sea $\forall b \in B$ $v_{(a,b)}$ el vector de características de b asociado a a. Entonces sea:

$$B_a = \{b_1, b_2, ..., b_m\}$$

es la lista de preferencias de B para a

donde si $i < j$ entonces $v_{(a,b_i)} \geq_L v_{(a,b_j)}$.

Nota 9.1 Nótese que ya que $\geq_L$ es una relación de orden parcial siempre se podrá encontrar la lista de preferencias definida en la Definición 9.6 que cumpla con la condición de orden.

Ejemplo a Definición 9.6

Sean $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, y $C_A = \{c_1, c_2\}$, $C_B = \{c_1', c_2', c_3'\}$ los conjuntos de características de A y B correspondientemente.

Ahora, las listas de preferencias de C_B para los elementos de a son:

- $a_1 = \{c_2', c_3', c_1'\}$

- $a_2 = \{c'_1, c'_2, c'_3\}$

y las listas de preferencias de C_A para los elementos de b son:

- $b_1 = \{c_2, c_1\}$
- $b_2 = \{c_1, c_2\}$.

Ahora, se dice que a_1 cuenta con las características c_1, a_2 cuenta con la característica c_2, b_1 cuenta con las características c'_1, c'_2 y b_2 cuenta con las características c'_2, c'_3. Por lo tanto, los vectores asociados de los elementos de B para los elementos de A quedan como:

- $v_{(a_1,b_1)} = (h_{b_1}(c'_2), h_{b_1}(c'_3), h_{b_1}(c'_1)) = (1, 0, 1)$
- $v_{(a_1,b_2)} = (h_{b_2}(c'_2), h_{b_2}(c'_3), h_{b_2}(c'_1)) = (1, 1, 0)$
- $v_{(a_2,b_1)} = (h_{b_1}(c'_1), h_{b_1}(c'_2), h_{b_1}(c'_3)) = (1, 1, 0)$
- $v_{(a_2,b_2)} = (h_{b_2}(c'_1), h_{b_2}(c'_2), h_{b_2}(c'_3)) = (0, 1, 1)$.

Ahora, los vectores asociados de los elementos de A para los elementos de B se obtienen:

- $v_{(b_1,a_1)} = (h_{a_1}(c_2), h_{a_1}(c_1)) = (0, 1)$
- $v_{(b_1,a_2)} = (h_{a_2}(c_2), h_{a_2}(c_1)) = (1, 0)$
- $v_{(b_2,a_1)} = (h_{a_1}(c_1), h_{a_1}(c_2)) = (1, 0)$
- $v_{(b_2,a_2)} = (h_{a_2}(c_1), h_{a_2}(c_2)) = (0, 1)$.

Ya que se obtuvieron los vectores, para cada elemento se puede verificar su orden:

- a_1: $(1, 1, 0) \geq_L (1, 0, 1) \rightarrow v_{(a_1,b_2)} \geq_L v_{(a_1,b_1)}$
- a_2: $(1, 1, 0) \geq_L (0, 1, 1) \rightarrow v_{(a_2,b_1)} \geq_L v_{(a_2,b_2)}$
- b_1: $(1, 0) \geq_L (0, 1) \rightarrow v_{(b_1,a_2)} \geq_L v_{(b_1,a_1)}$
- b_2: $(1, 0) \geq_L (0, 1) \rightarrow v_{(b_2,a_1)} \geq_L v_{(b_2,a_2)}$.

Ahora, con los órdenes de los vectores según la relación de orden obtenida a partir de la Definición 9.2 se obtienen las listas de preferencias para los elementos de A y B:

- $B_{a_1} = \{b_2, b_1\}$
- $B_{a_2} = \{b_1, b_2\}$
- $A_{b_1} = \{a_2, a_1\}$
- $A_{b_2} = \{a_1, a_2\}$.

9.1. Algoritmo

El objetivo de este capítulo como el de los capítulos anteriores es encontrar un emparejamiento estable entre dos conjuntos de elementos A y B, en este caso, la diferencia es que esto se quiere realizar con base a características que tienen los elementos y los gustos por las características de los elementos del conjunto contrario. Para este problema no se necesita modificar el algoritmo Gale-Shapley, simplemente se necesita agregar un paso previo, el cual consiste en obtener la lista de preferencias del conjunto contrario para cada elemento. Esto anterior se realiza aprovechándose de los conceptos definidos en este capítulo, el Ejemplo a la Definición 9.6 muestra paso a paso como se pueden obtener las listas de preferencias que permiten utilizar el Algoritmo Gale-Shapley, y la Propiedad 9.1 asegura que estas listas siempre existen, por lo tanto, siempre se podrá utilizar el algoritmo, el cual es inicializado cumpliendo las hipótesis presentadas en el capítulo 4. Entonces, por Teorema 7.1 siempre se encontrará un emparejamiento estable.

Ejemplo de uso de Algoritmo

Por lo tanto, el emparejamiento obtenido es:

Sean $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, y $C_A = \{c_1, c_2\}$, $C_B = \{c'_1, c'_2, c'_3\}$ los conjuntos de características de A y B correspondientemente.

Ahora, las listas de preferencias de C_B para los elementos de a son:

- $a_1 = \{c'_2, c'_3, c'_1\}$
- $a_2 = \{c'_1, c'_2, c'_3\}$

y las listas de preferencias de C_A para los elementos de b son:

- $b_1 = \{c_2, c_1\}$
- $b_2 = \{c_1, c_2\}$.

Ahora, se dice que a_1 cuenta con las características c_1, a_2 cuenta con la característica c_2, b_1 cuenta con las características c'_1, c'_2 y b_2 cuenta con las características c'_2, c'_3. Por lo tanto, los vectores asociados de los elementos de B para los elementos de A quedan como:

- $v_{(a_1,b_1)} = (h_{b_1}(c'_2), h_{b_1}(c'_3), h_{b_1}(c'_1)) = (1, 0, 1)$
- $v_{(a_1,b_2)} = (h_{b_2}(c'_2), h_{b_2}(c'_3), h_{b_2}(c'_1)) = (1, 1, 0)$
- $v_{(a_2,b_1)} = (h_{b_1}(c'_1), h_{b_1}(c'_2), h_{b_1}(c'_3)) = (1, 1, 0)$
- $v_{(a_2,b_2)} = (h_{b_2}(c'_1), h_{b_2}(c'_2), h_{b_2}(c'_3)) = (0, 1, 1)$.

Ahora, los vectores asociados de los elementos de A para los elementos de B se obtienen:

- $v_{(b_1,a_1)} = (h_{a_1}(c_2), h_{a_1}(c_1)) = (0, 1)$
- $v_{(b_1,a_2)} = (h_{a_2}(c_2), h_{a_2}(c_1)) = (1, 0)$
- $v_{(b_2,a_1)} = (h_{a_1}(c_1), h_{a_1}(c_2)) = (1, 0)$
- $v_{(b_2,a_2)} = (h_{a_2}(c_1), h_{a_2}(c_2)) = (0, 1)$.

Ya que se obtuvieron los vectores, para cada elemento se puede verificar su orden:

- a_1: $(1,1,0) \geq_L (1,0,1) \rightarrow v_{(a_1,b_2)} \geq_L v_{(a_1,b_1)}$
- a_2: $(1,1,0) \geq_L (0,1,1) \rightarrow v_{(a_2,b_1)} \geq_L v_{(a_2,b_2)}$
- b_1: $(1,0) \geq_L (0,1) \rightarrow v_{(b_1,a_2)} \geq_L v_{(b_1,a_1)}$
- b_2: $(1,0) \geq_L (0,1) \rightarrow v_{(b_2,a_1)} \geq_L v_{(b_2,a_2)}$.

Ahora, con los órdenes de los vectores según la relación de orden obtenida a partir de la Definición 9.2 se obtienen las listas de preferencias para los elementos de A y B:

- $B_{a_1} = \{b_2, b_1\}$
- $B_{a_2} = \{b_1, b_2\}$
- $A_{b_1} = \{a_2, a_1\}$
- $A_{b_2} = \{a_1, a_2\}$.

Ahora, ya que existen listas de preferencias se procede a utilizar el Algoritmo Gale-Shapley, donde el conjunto que propone es A y el conjunto pasivo es B:

Primera iteración:

- $\hat{A} = \{a_1, a_2\}$
- El mejor emparejamiento para cada elemento es:
  - $a_1 : b_2$
  - $a_2 : b_1$

a_1	b_2	b_1
a_2	b_1	b_2

b_1	a_2	a_1
b_2	a_1	a_2

- Como muestra la figura b_1 se empareja con a_2, b_2 se empareja con a_1.
- Entonces los emparejamientos quedan:
  - $E_{a_1} = b_2$
  - $E_{a_2} = b_1$
  - $E_{b_1} = a_2$
  - $E_{b_2} = a_1$.

Ya que si se intenta realizar una nueva iteración $\hat{A} = \emptyset$ entonces, el algoritmo finaliza y se obtiene el emparejamiento estable entre A y B:

$$P = \{(a_1, b_2), (a_2, b_1)\}.$$

CAPÍTULO 10

Aplicaciones del teorema del matrimonio estable y sus variantes

Una pregunta que puede surgir después de todo el desarrollo teórico presentado del Problema del Matrimonio Estable y sus variantes es en qué casos pueden utilizarse los algoritmos y como alguien puede aprovecharse de estos al encontrarse problemas en su rutina. Estos algoritmos brillan en los problemas de asignación, debido a que dan las herramientas para optimizar los emparejamientos entre dos conjuntos. Para cada variante presentada se presentará un ejemplo generalizable que puede encontrarse fácilmente en diferentes campos donde se utilizará la teoría desarrollada en los capítulos anteriores.

10.1. Problema original

El problema original da muchas herramientas al momento en el que se tengan que emparejar dos conjuntos, además de asegurar que existe un emparejamiento estable al momento de tener listas de preferencias, también da el algoritmo de como obtenerlo, a continuación se presenta un ejemplo de como se puede utilizar el algoritmo en un problema de asignación entre alumnos y escuelas.

Emparejamiento estable para alumnos y plazas escolares

Para este ejemplo práctico se tomará el caso donde un programa de becas internacionales elige a 10 alumnos para colocar en 10 distintas escuelas. El programa al conocer que existe el algoritmo de Gale-Shapley solicita a los estudiantes una lista de preferencias ordenada de las escuelas, además, hace la misma solicitud a los directores de las escuelas. El conjunto de alumnos será:

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}\}$$

Mientras que el conjunto de las escuelas será:

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$$

Las listas de preferencias de los alumnos son:

- a_1: $\{e_3, e_4, e_7, e_2, e_9, e_{10}, e_1, e_8, e_5, e_6\}$
- a_2: $\{e_1, e_3, e_6, e_{10}, e_5, e_2, e_7, e_9, e_4, e_8\}$
- a_3: $\{e_{10}, e_2, e_5, e_8, e_7, e_4, e_9, e_3, e_6, e_1\}$
- a_4: $\{e_7, e_{10}, e_6, e_4, e_9, e_8, e_2, e_1, e_3, e_5\}$
- a_5: $\{e_{10}, e_3, e_8, e_9, e_7, e_1, e_6, e_2, e_4, e_5\}$
- a_6: $\{e_5, e_7, e_8, e_3, e_4, e_9, e_6, e_1, e_{10}, e_2\}$
- a_7: $\{e_1, e_6, e_2, e_4, e_9, e_3, e_7, e_{10}, e_8, e_5\}$
- a_8: $\{e_6, e_7, e_1, e_{10}, e_5, e_8, e_9, e_2, e_4, e_3\}$
- a_9: $\{e_4, e_{10}, e_1, e_2, e_9, e_6, e_7, e_5, e_8, e_3\}$
- a_{10}: $\{e_9, e_1, e_5, e_{10}, e_7, e_4, e_3, e_6, e_2, e_8\}$

Mientras que las listas de preferencias de las escuelas son:

- e_1: $\{a_{10}, a_4, a_7, a_5, a_2, a_1, a_3, a_9, a_8, a_6\}$
- e_2: $\{a_5, a_7, a_4, a_6, a_1, a_3, a_9, a_2, a_8, a_{10}\}$
- e_3: $\{a_5, a_3, a_4, a_6, a_9, a_2, a_1, a_{10}, a_7, a_8\}$
- e_4: $\{a_8, a_9, a_4, a_{10}, a_5, a_6, a_7, a_1, a_2, a_3\}$
- e_5: $\{a_9, a_3, a_1, a_{10}, a_5, a_6, a_2, a_7, a_8, a_4\}$
- e_6: $\{a_7, a_6, a_{10}, a_8, a_9, a_4, a_2, a_3, a_1, a_5\}$
- e_7: $\{a_3, a_6, a_{10}, a_2, a_8, a_4, a_5, a_7, a_1, a_9\}$
- e_8: $\{a_9, a_7, a_{10}, a_4, a_8, a_2, a_6, a_3, a_1, a_5\}$
- e_9: $\{a_{10}, a_6, a_4, a_1, a_5, a_7, a_8, a_3, a_2, a_9\}$
- e_{10}: $\{a_3, a_6, a_4, a_5, a_8, a_9, a_1, a_{10}, a_7, a_2\}$

Con esta información ya se cumplen las condiciones para asegurar un emparejamiento estable, es decir que se tendrá un emparejamiento entre los alumnos y las escuelas donde no existan dos parejas tal que un alumno y una escuela se prefieran antes que a su pareja. Esto simplemente aplicando el algoritmo explicado en el capítulo 4 e implementado en Python.

Al programa de becas le interesa siempre dar una buena imagen a las escuelas que participan en el programa, por lo tanto, coloca el conjunto de escuelas como el conjunto que propone y el conjunto de los alumnos como el conjunto pasivo. Así además de asegurar un emparejamiento estable, logra que las escuelas propongan a los alumnos de su preferencia primero.

Corriendo el algoritmo se obtiene el emparejamiento entre alumnos y escuelas:

$$P = \{(e_1, a_7), (e_2, a_1), (e_3, a_5), (e_4, a_9), (e_5, a_6), (e_6, a_8), (e_7, a_2), (e_8, a_4), (e_9, a_{10}), (e_{10}, a_3)\}.$$

El Teorema 7.1 nos asegura que es un emparejamiento estable entre E y A.

10.2. Variante de conjuntos desiguales

Muchos casos pueden presentar el problema que no necesariamente los conjuntos que se intentan emparejar son de la misma cardinalidad. Debido al desarrollo obtenido en el capítulo 5 sección 1 se tienen herramientas para abordar este tipo de problemas, los cuales pueden ser muy comunes, por lo tanto, este desarrollo puede aplicarse a muchos problemas. Por esa razón se presenta a continuación un ejemplo de cómo se puede ajustar esta variante a un problema del mundo laboral.

Emparejamiento estable para plazas laborales y posibles empleados

Para este ejemplo se tomará un caso donde una empresa tiene 14 aplicantes competentes para 7 plazas distintas. La empresa conociendo que existe el algoritmo Gale-Shapley y además, este se puede ajustar a conjuntos desiguales realiza para cada plaza una lista de preferencia de aplicantes. Además a los aplicantes se les explica la situación y les piden una lista de preferencia de las plazas disponibles. Por lo tanto se tienen los conjuntos de,

aplicantes:

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}\}$$

y plazas:

$$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}.$$

Ahora, las listas de preferencias de los aplicantes son:

- a_1: $\{p_2, p_6, p_4, p_7, p_5, p_1, p_3\}$
- a_2: $\{p_7, p_1, p_2, p_4, p_5, p_3, p_6\}$
- a_3: $\{p_2, p_1, p_3, p_7, p_5, p_6, p_4\}$
- a_4: $\{p_1, p_5, p_6, p_2, p_4, p_7, p_3\}$
- a_5: $\{p_1, p_6, p_5, p_7, p_4, p_3, p_2\}$
- a_6: $\{p_3, p_1, p_7, p_5, p_4, p_2, p_6\}$
- a_7: $\{p_4, p_1, p_6, p_7, p_2, p_5, p_3\}$
- a_8: $\{p_5, p_3, p_7, p_1, p_4, p_2, p_6\}$
- a_9: $\{p_2, p_5, p_1, p_3, p_4, p_6, p_7\}$
- a_{10}: $\{p_1, p_2, p_5, p_3, p_7, p_4, p_6\}$
- a_{11}: $\{p_3, p_1, p_7, p_2, p_5, p_6, p_4\}$
- a_{12}: $\{p_7, p_1, p_4, p_3, p_2, p_5, p_6\}$
- a_{13}: $\{p_2, p_6, p_5, p_1, p_4, p_3, p_7\}$
- a_{14}: $\{p_6, p_5, p_4, p_7, p_2, p_1, p_3\}$.

mientras que las listas de preferencias para cada plaza son:

- p_1: $\{a_{14}, a_1, a_{10}, a_7, a_2, a_{13}, a_3, a_{12}, a_6, a_4, a_{11}, a_5, a_9, a_8\}$
- p_2: $\{a_{10}, a_1, a_{14}, a_{13}, a_{12}, a_6, a_3, a_9, a_8, a_2, a_5, a_7, a_4, a_{11}\}$
- p_3: $\{a_{10}, a_1, a_{14}, a_7, a_{11}, a_9, a_5, a_2, a_8, a_4, a_{12}, a_3, a_{13}, a_6\}$
- p_4: $\{a_{14}, a_1, a_{10}, a_5, a_6, a_{11}, a_{13}, a_2, a_4, a_8, a_{12}, a_9, a_3, a_7\}$
- p_5: $\{a_{10}, a_1, a_{14}, a_6, a_5, a_{13}, a_{12}, a_3, a_9, a_4, a_2, a_7, a_8, a_{11}\}$
- p_6: $\{a_1, a_{10}, a_{14}, a_5, a_{11}, a_{12}, a_3, a_9, a_6, a_2, a_8, a_4, a_7, a_{13}\}$
- p_7: $\{a_1, a_{10}, a_{14}, a_4, a_{13}, a_{12}, a_5, a_8, a_2, a_7, a_9, a_3, a_{11}, a_6\}$.

Con esta información se puede correr el algoritmo Gale-Shapley ajustado para conjuntos desiguales, así como se revela en el capítulo 5, sección 1, además este asegura un emparejamiento estable sin importar cual conjunto propone y cual conjunto es pasivo.

Ahora, debido a la naturaleza del problema la empresa se ve obligada a que el conjunto de las plazas sea el conjunto que propone mientras que los aplicantes sean el conjunto pasivo, debido a que es de su interés que las plazas busquen a su preferencia primero. Por lo tanto, se realiza el ejercicio con el conjunto P como el conjunto que propone y A como el conjunto pasivo, ya que $\#P < \#A$, entonces se usa el algoritmo ajustado para que el conjunto pasivo tenga mayor cardinalidad que el conjunto que propone.

Corriendo el algoritmo se obtiene el emparejamiento entre P y A:

$$P = \{(p_1, a_{10}), (p_2, a_1), (p_3, a_7), (p_4, a_5), (p_5, a_6), (p_6, a_{14}), (p_7, a_4)\}.$$

En este caso $a_2, a_3, a_8, a_9, a_{11}, a_{12}, a_{13}$ se quedan desemparejados, es decir, no entran a la empresa.

10.3. Variante de preferencias incompletas

La variante de Preferencias Incompletas es perfecta para encontrar un emparejamiento donde existan parejas que por compatibilidad no pueden suceder, un ejemplo sencillo es la compatibilidad entre los tipos de sangre de los humanos, no todos los tipos de sangre pueden donar a todos. Esto hace que las listas sean incompletas, lo cual hace que todo el desarrollo del capítulo 5 sección 2 se ajuste a este tipo de problemas y se pueda encontrar un emparejamiento aceptable.

Emparejamiento aceptable para sangre y pacientes de un hospital

Para esta sección se abordara un caso que puede presentarse comunmente en los hospitales, donde existen pacientes que necesitan donación de sangre y existen litros de sangre disponibles para estos pacientes. Esto se puede complicar debido a que no todos los tipos de sangre pueden donar a todos y viceversa. Entonces esto se adecua de manera perfecta a la variante de Preferencias Incompletas, ya que si un tipo de sangre no puede donar a otra, esta no aparece en su lista de preferencias. Se realizarán dos ejemplos, uno donde no se encuentra un emparejamiento aceptable, según definido en la sección 2 del capítulo 5 y otro donde si se encuentra.

Ejemplo 1

Para este ejemplo dígase que hay pacientes que necesitan sangre con los siguientes tipos de sangre respectivamente: $R = \{O^-, A^-, A^+, A^+, B^+, AB^+\}$, mientras que disponibles para donación hay litros de sangre de $D = \{O^-, O^+, A^+, B^+, AB^-, AB^+\}$. Ahora, la lista de los donadores y a quienes les pueden dar, está dada por las siguientes listas de preferencias:

- O^-: $\{O^-, A^-, A^+, B^+, AB^+\}$
- O^+: $\{A^+, B^+, AB^+\}$
- A^+: $\{A^+, AB^+\}$
- B^+: $\{B^+, AB^+\}$
- AB^-: $\{AB^+\}$
- AB^+: $\{AB^+\}$

y la lista de los receptores y de quienes pueden recibir están dadas por:

- O^-: $\{O^-\}$
- A^-: $\{O^-\}$
- A^+: $\{O^-, O^+, A^+\}$
- A^+: $\{O^-, O^+, A^+\}$
- B^+: $\{O^-, O^+, B^+\}$
- AB^+: $\{O^-, O^+, A^+, B^+, AB^-, AB^+\}$

Ahora, que se tienen las listas de preferencias se puede obtener la matriz de Biadyacencia entre D y R:

	O^-	O^+	A^+	A^+	B^+	AB^+
O^-	1	1	1	1	1	1
O^+	0	0	1	1	1	1
A^+	0	0	1	1	0	1
B^+	0	0	0	0	1	1
AB^-	0	0	0	0	0	1
AB^+	0	0	0	0	0	1

donde los elementos verticales corresponden al conjunto D y los horizontales al conjunto R.

Ahora, aprovechándonos del Teorema 8.3, existe un emparejamiento aceptable si el permanente de esta matriz es mayor a 0. Corriendo un código en Python que calcula el permanente y si este es mayor a 0, da todos los emparejamientos aceptables. En este caso el permanente de la Matriz de Biadyacencia es igual a 0. Por lo tanto, en este caso no se pueden emparejar a todos los pacientes con sangre compatible.

Ejemplo 2

Para este ejemplo dígase que hay pacientes que necesitan sangre con los siguientes tipos de sangre respectivamente: $R = \{O^-, O^+, A^+, B^+, AB^-, AB^+\}$, mientras que disponibles para donación hay litros de sangre de $D = \{O^-, O^+, A^+, B^+, AB^-, AB^+\}$. Ahora, la lista de los donadores y a quienes les pueden dar, está dada por las siguientes listas de preferencias:

- O^-: $\{O^-, O^+, A^+, B^+, AB^-, AB^+\}$
- O^+: $\{O^+, A^+, B^+, AB^+\}$
- A^+: $\{A^+, AB^+\}$

- B^+: $\{B^+, AB^+\}$
- AB^-: $\{AB^-, AB^+\}$
- AB^+: $\{AB^+\}$

y la lista de los receptores y de quienes pueden recibir están dadas por:

- O^-: $\{O^-\}$
- O^+: $\{O^-, O^+\}$
- A^+: $\{O^-, O^+, A^+\}$
- B^+: $\{O^-, O^+, B^+\}$
- AB^-: $\{O^-, AB^-\}$
- AB^+: $\{O^-, O^+, A^+, B^+, AB^-, AB^+\}$

Ahora, que se tienen las listas de preferencias se puede obtener la matriz de Biadyacencia entre D y R:

	O^-	O^+	A^+	B^+	AB^-	AB^+
O^-	1	1	1	1	1	1
O^+	0	1	1	1	0	1
A^+	0	0	1	0	0	1
B^+	0	0	0	1	0	1
AB^-	0	0	0	0	1	1
AB^+	0	0	0	0	0	1

donde los elementos verticales corresponden al conjunto D y los horizontales al conjunto R.

Ahora, aprovechándonos del Teorema 8.3, existe un emparejamiento aceptable si el permanente de esta matriz es mayor a 0. Corriendo un código en Python que calcula el permanente y si este es mayor a 0, da todos los emparejamientos aceptables. En este caso se tiene que el permanente es 1. Entonces se tiene solo un emparejamiento aceptable, en este caso es:

$$P = \{(O^-, O^-), (O^+, O^+), (A^+, A^+), (B^+, B^+), (AB^-, AB^-), (AB^+, AB^+)\}.$$

10.4. Variante de preferencias por características

Esta variante del problema se ajusta perfectamente para obtener emparejamientos en casos donde se conozca el contexto de los elementos de los conjuntos y además se conozca que características del otro conjunto son deseables para cada elemento. Esto se puede ajustar muy bien en el mundo real, debido a que es más sencillo hacer preguntas sobre características binarias antes que pedir una lista ordenada del otro conjunto. Por ejemplo, si se desea obtener una lista de preferencias entre doctores y hospitales, es más fácil preguntar a los doctores si prefiere si su hospital esté en la ciudad o no, que sea público o privado, etc. Además, puede resultar más generalizable tener que ordenar estas características en importancia que ordenar los hospitales en sí. En el siguiente ejemplo se le dará seguimiento a lo comentado en esta introducción, con características generales pero siempre siguiendo en la línea de asignación de doctores a hospitales.

Emparejamiento estable para doctores y hospitales

En este ejemplo se manejará la asignación de 5 doctores y 5 hospitales. A los doctores les interesan 6 características de los hospitales y a los hospitales le interesan 4 características de los doctores.

Llámese al conjunto de Doctores $D = \{d_1, d_2, d_3, d_4, d_5\}$, al conjunto de hospitales $H = \{h_1, h_2, h_3, h_4, h_5\}$. Las características de los doctores que interesan a los hospitales son $C_D = \{c_1, c_2, c_3, c_4\}$ y las características de los hospitales que interesan a los doctores son: $C_H = \{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6\}$.

Además para cada conjunto se tiene la lista de preferencias correspondiente a cada elemento, para el conjunto D:

- d_1: $\{c'_2, c'_1, c'_3, c'_5, c'_6, c'_4\}$
- d_2: $\{c'_3, c'_2, c'_1, c'_6, c'_4, c'_5\}$
- d_3: $\{c'_5, c'_1, c'_3, c'_2, c'_6, c'_4\}$
- d_4: $\{c'_1, c'_5, c'_6, c'_2, c'_3, c'_4\}$
- d_5: $\{c'_1, c'_4, c'_2, c'_5, c'_3, c'_6\}$

Para el conjunto H:

- h_1: $\{c_1, c_3, c_4, c_2\}$
- h_2: $\{c_4, c_1, c_2, c_3\}$
- h_3: $\{c_2, c_1, c_4, c_3\}$
- h_4: $\{c_4, c_2, c_3, c_1\}$
- h_5: $\{c_1, c_4, c_2, c_3\}$.

Además, haciendo uso de la función de caracteterísticas de cada elemento se sabe con que características cuenta cada elemento, en este caso para el conjunto D se tiene que:

- d_1: $h_{d_1}(c_1) = 1$, $h_{d_1}(c_2) = 0$, $h_{d_1}(c_3) = 0$, $h_{d_1}(c_4) = 0$
- d_2: $h_{d_2}(c_1) = 0$, $h_{d_2}(c_2) = 0$, $h_{d_2}(c_3) = 1$, $h_{d_2}(c_4) = 0$
- d_3: $h_{d_3}(c_1) = 1$, $h_{d_3}(c_2) = 0$, $h_{d_3}(c_3) = 0$, $h_{d_3}(c_4) = 0$
- d_4: $h_{d_4}(c_1) = 1$, $h_{d_4}(c_2) = 1$, $h_{d_4}(c_3) = 1$, $h_{d_4}(c_4) = 0$
- d_5: $h_{d_5}(c_1) = 1$, $h_{d_5}(c_2) = 0$, $h_{d_5}(c_3) = 0$, $h_{d_5}(c_4) = 1$

mientras que para el conjunto H se tiene que:

- h_1: $h_{h_1}(c'_1) = 0$, $h_{h_1}(c'_2) = 1$, $h_{h_1}(c'_3) = 1$, $h_{h_1}(c'_4) = 1$, $h_{h_1}(c'_5) = 0$, $h_{h_1}(c'_6) = 0$
- h_2: $h_{h_2}(c'_1) = 1$, $h_{h_2}(c'_2) = 1$, $h_{h_2}(c'_3) = 0$, $h_{h_2}(c'_4) = 1$, $h_{h_2}(c'_5) = 1$, $h_{h_2}(c'_6) = 0$
- h_3: $h_{h_3}(c'_1) = 1$, $h_{h_3}(c'_2) = 0$, $h_{h_3}(c'_3) = 1$, $h_{h_3}(c'_4) = 1$, $h_{h_3}(c'_5) = 1$, $h_{h_3}(c'_6) = 0$
- h_4: $h_{h_4}(c'_1) = 1$, $h_{h_4}(c'_2) = 0$, $h_{h_4}(c'_3) = 1$, $h_{h_4}(c'_4) = 1$, $h_{h_4}(c'_5) = 1$, $h_{h_4}(c'_6) = 0$

- h_5: $h_{h_5}(c'_1) = 0$, $h_{h_5}(c'_2) = 1$, $h_{h_5}(c'_3) = 1$, $h_{h_5}(c'_4) = 1$, $h_{h_5}(c'_5) = 1$, $h_{h_5}(c'_6) = 1$.

Ahora, al correr un algoritmo que genera los vectores de características, los ordena y así genera las listas de preferencia, como se realiza en el algoritmo del capítulo 6, se obtienen las listas de preferencias que ingresan al algoritmo Gale-Shapley.

- d_1: $\{h_2, h_5, h_1, h_3, h_4\}$
- d_2: $\{h_5, h_1, h_3, h_4, h_2\}$
- d_3: $\{h_3, h_4, h_2, h_5, h_1\}$
- d_4: $\{h_2, h_3, h_4, h_5, h_1\}$
- d_5: $\{h_2, h_3, h_4, h_5, h_1\}$

- h_1: $\{d_4, d_1, d_3, d_2, d_5\}$
- h_2: $\{d_5, d_4, d_1, d_3, d_2\}$
- h_3: $\{d_4, d_1, d_3, d_5, d_2\}$
- h_4: $\{d_5, d_4, d_2, d_1, d_3\}$
- h_5: $\{d_4, d_1, d_3, d_5, d_2\}$.

Corriendo el algoritmo Gale-Shapley utilizado en el primer ejemplo de este capítulo se obtiene el siguiente emparejamiento, en este caso supóngase que interesa más obtener un emparejamiento con los hositales como el conjunto que propone y los doctores como el conjunto pasivo:

$$P = \{(h_1, d_3), (h_2, d_5), (h_3, d_4), (h_4, d_2), (h_5, d_1)\}.$$

El cual es un emparejamiento estable, ya que se utiliza el algoritmo Gale-Shapley y las listas de preferencias son generadas con una relación de orden.

CAPÍTULO 11

Conclusiones

- Al leer el enunciado del Problema del Matrimonio Estable se pueden reconocer de manera sencilla situaciones en las que su solución pueda ser útil. Con base en esto el algoritmo Gale-Shapley toma mucho valor, esto se puede observar en la facilidad con la que se obtienen emparejamientos en situaciones que a priori pueden ser complicadas de plantear, como optimizar las parejas en un problema de asignación. Asimismo, las soluciones de las variantes funcionan como un auxiliar en caso se cuenta con información distinta o incluso incompleta.
- La variante de Preferencias por Características logra dar un enfoque distinto al problema original. Esto debido a que se pueden considerar situaciones donde los elementos de los conjuntos no se conocen entre sí, por lo tanto, no es posible que den listas de preferencias ya que lo harían a ciegas. Mientras que sí pueden ordenar características, hace que a pesar del desconocimiento de los elementos entre sí, un tercero que tenga información sobre ambos conjuntos y las preferencias de sus características pueda realizar el emparejamiento aprovechándose también del algoritmo Gale-Shapley.
- El algoritmo Gale-Shapley logra con éxito devolver un emparejamiento estable entre dos conjuntos del mismo tamaño, esto además del algoritmo, lo revela la teoría descrita derivada a este. Además, ya que se asegura que este termina el emparejamiento es seguro siempre y cuando se tengan conjuntos del mismo tamaño, cada elemento tenga una lista de preferencias del otro conjunto y se disponga un conjunto como activo y otro como pasivo.
- Existen dos variantes famosas del problema del Matrimonio Estable: Conjuntos Desiguales y Preferencias Incompletas. Estas dos variantes dan otra perspectiva y dan una solución más general. En cuanto a la primera variante se encontró que modificando sutilmente el algoritmo Gale-Shapley se asegura también un emparejamiento estable. Paralelo a esto la variante de Preferencias Incompletas puede dividirse en casos, uno donde se tengan preferencias incompletas no ordenadas, es decir, los elementos solo están interesados en que su pareja esté en su lista pero adentro de su lista no tienen rango. Este caso se aprovecha de los grafos no dirigidos y la matriz de biadyacencia para encontrar un emparejamiento que cumpla con la condición. Agregándo la complicación donde estas listas incompletas también están ordenadas revela el segundo caso, este complica el problema a tal punto que debe buscarse un algoritmo distinto y reconocer bajo que características se puede encontrar un emparejamiento estable.
- Existen muchas aplicaciones para este problema y sus variantes. En el capítulo 10 se presentan algunos ejemplos de como el algoritmo y sus variantes pueden ser utilizados en problemas de

asignación en el mundo laboral. Además, el ejercicio ayuda a saber cuál de las variantes utilizar según la información que se tenga disponible.

- El concepto de Orden Lexicográfico hizo posible dar una relación de orden para los vectores de características en la última variante, lo cual fue clave para poder diseñar la solución al problema. Luego de obtener esta relación de orden y por ende, un conjunto ordenado ya se puede utilizar el algoritmo Gale-Shapley. Esta variante de Preferencias por Características también es compatible con otras variantes, lo cual puede dar un valor añadido, en caso se desee estudiar en un trabajo de seguimiento a este.

CAPÍTULO 12

Recomendaciones

Una posible oportunidad de seguimiento del desarrollo se presenta en la variante de preferencias incompletas. Como fue mencionado en la discusión de la sección 8.2 al añadir la hipótesis que menciona que los elementos tienen listas de preferencias incompletas y además ordenadas se genera un caso bastante complejo, ya que no es trivial como ajustar el algoritmo Gale-Shapley a esta situación. En un trabajo posterior se podría inverstigar si es posible ajustar el algoritmo a esta variante sin realizar muchos cambios o si sería adecuado diseñar un algoritmo distinto que se adecue a las hipótesis.

En la variante de preferencias por características se utiliza un orden que permite ordenar vectores donde como condición se toma que los conjuntos de los cuales provienen los elementos que componen al vector deben tener una relación de orden total o parcial. Al cumplir esto se puede utilizar el orden lexicográfico. En este trabajo las características se presentaron como vectores binarios, para un trabajo posterior puede ser interesante utilizar otro conjunto para medir la compatibilidad de las características aprovechándose de la flexibilidad del orden lexicográfico.

Siguiendo con la variante mencionada en el párrafo anterior puede investigarse otra manera de ordenar los vectores para obtener las listas de preferencias. Si se deseara explorar otro tipo de ordenamiento, quizás basado en el recuento de características o en una ponderación, sería interesante que un trabajo posterior se investigaran otras metodologías para esta variante.

CAPÍTULO 13

Bibliografía

[1] G. Bonanno. *Game theory 1, basic concepts*. Createspace Independent Publishing Platform, 2018.

[2] J. González, I. García-Jurado y M. G. Fiestras-Janeiro. *An introductory course on mathematical game theory and applications*. Graduate Studies in Mathematics. American Mathematical Society, 2023.

[3] J. Hidakatsu. "Structure of the Stable Marriage and Stable Roommate Problems and Applications". Tesis doct. University of Michigan, 2014.

[4] Lafactoriaapple. *Orden Lexicográfico*. Online. Disponible en: `https://www.lafactoriaapple.com/ciencias-de-la-computacion/orden-lexicografico.php`. Ene. de 2021.

[5] D. F. Manlove et al. "Hard variants of stable marriage". En: *Theoretical Computer Science* 276.1-2 (2002), págs. 261-279.

[6] A. Mas-Colell, M. D. Whinston y J. R. Green. *Microeconomic theory*. Oxford University Press, 1995.

[7] O. Muiño Rodríguez. *El problema de asignación*. Trabajo de Fin de Máster, Universidad de Santiago de Compostela, Universidad de A Coruña, Universidad de Vigo. 2016.

[8] M. Osborne y A. Rubinstein. *A course in game theory*. MIT Press, 1994.

[9] J. Pérez, J. L. Jimeno y E. Cerda. *Teoría de juegos*. Madrid, España: Pearson Educación, 2004.

[10] P. Santamaría Manteca. *Emparejamientos Estables (Stable Matchings)*. Trabajo de fin de grado para acceder al grado en Matemáticas, Facultad de Ciencias, Universidad de Oviedo. Dirigido por Daniel Sadornil Renedo. 2020.

[11] Y. Shoham y K. Leyton-Brown. *Multiagent systems (algorithmic, game-theoretic, and logical foundations)*. Cambridge University Press, 2008.

[12] A. N. Siegel. *Combinatorial Game Theory*. Vol. 146. Graduate Studies in Mathematics. American Mathematical Society, 2013.

[13] H. R. Varian. *Intermediate Microeconomics: A Modern Approach*. 9th, Media Update. W. W. Norton & Company, 2019.

[14] H. R. Varian. *Microeconomic Analysis*. 3rd. W. W. Norton & Company, 1992.

[15] Cristina Villalba Rodríguez. *Análisis de grafos y aplicaciones: Teoremas de Hall y Tutte*. Trabajo de Fin de Grado, Universidad de Sevilla. 2021. URL: https://idus.us.es/bitstream/handle/11441/134402/Villalba%20Rodr%C3%ADguez%20Cristina%20TFG.pdf?sequence=1&isAllowed=y.

CAPÍTULO 14

Anexos

Listing 14.1: Función para implementación de Algoritmo Gale-Shapley

```
def gs(A_preferences, B_preferences):

    # Obtener los conjuntos
    A = list(A_preferences.keys())
    B = list(B_preferences.keys())

    # Hacer diccionarios para ingresar la informacion que requiere el algoritmo
    A_dicts = {}
    B_dicts = {}

    # Ciclo para llenar la informacion inicial del conjunto A
    for a_el in A:
        E_a = ''
        L_a = A_preferences[a_el]
        R_a = []
        A_dicts[a_el] = [E_a, L_a, R_a]

    # Ciclo para llenar la informacion inicial del conjunto B
    for b_el in B:
        E_b = ''
        L_b = B_preferences[b_el]
        C_b = []
        B_dicts[b_el] = [E_b, L_b, C_b]

    # Algoritmo
    while True:

        # Creaci n del conjunto hat_A y revisi n de si este es vac o
        hat_A = []
        for a_el in A:
            if A_dicts[a_el][0] == '':
```

```
                hat_A.append(a_el)

        # Break del ciclo en caso hat_A sea vac o
        if not hat_A:
            break


        # Creacion del conjunto hat_B
        hat_B = []

        # Los elementos del conjunto hat_A proponen
        for a_el in hat_A:
            D = [item for item in A_dicts[a_el][1] if item not in A_dicts[a_el][2]]
            b_a = D[0]
            B_dicts[b_a][2].append(a_el)
            A_dicts[a_el][0] = b_a
            hat_B.append(b_a)

        # Hacer que hat_B sea una lista con todos sus elementos distintos
        hat_B = list(set(hat_B))


        # Los elementos de hat_B escogen a su pareja
        for b_el in hat_B:
            if B_dicts[b_el][0] != '':
                posibles = list(set(B_dicts[b_el][2]) | set([B_dicts[b_el][0]]))
            else:
                posibles = B_dicts[b_el][2]
            valor_minimo = min(posibles, key=lambda x: B_dicts[b_el][1].index(x))
            for pos in posibles:
                A_dicts[pos][0] = ''

            B_dicts[b_el][0] = valor_minimo
            A_dicts[valor_minimo][0] = b_el

            posibles.remove(valor_minimo)
            for a_r in posibles:
                A_dicts[a_r][2].append(b_el)

            B_dicts[b_el][2] = []



    # Crear el emparejamiento
    P = []
    for a_el in A:
        P.append((a_el, A_dicts[a_el][0]))

    return P
```

Listing 14.2: Función para variante Preferencias Incompletas

```
def emp_aceptable(A_preferences, B_preferences):
```

```
    # Obtener los conjuntos
    A = list(A_preferences.keys())
    B = list(B_preferences.keys())

    # Obtener la longitud de los conjuntos
    n = len(A)

    # Hacer la matriz de biadyacencia
    matriz = [[0 for i in range(n)] for j in range(n)]
    for i in range(n):
        for j in range(n):
            if (A[i] in B_preferences[B[j]]) and (B[j] in A_preferences[A[i]]):
                matriz[i][j] = 1

    # Realizar las permutaciones
    conjunto = list(range(n))
    permutaciones = list(itertools.permutations(conjunto))

    # Obtener el permanente
    suma = 0
    perm_emparejamientos = []

    for permutacion in permutaciones:
        valor = 1
        for k in range(n):
            valor = valor * matriz[k][permutacion[k]]

        suma = suma + valor
        if valor == 1:
            perm_emparejamientos.append(permutacion)


    # Llenar la lista de los emparejamientos posibles
    P = []

    for pert in perm_emparejamientos:
        pairing = []
        for k in range(n):
            pairing.append((A[k], B[pert[k]]))

        P.append(pairing)

    return P
```

Listing 14.3: Funciones para variante Preferencias por Características

```
def preferencias(A_preferences, B_preferences, A_dict, B_dict):

    # Obtener los conjuntos
    A = list(A_preferences.keys())
    B = list(B_preferences.keys())
```

```
    # Declarar los diccionarios que se devuelven
    A_pref = {}
    B_pref = {}


    # Ciclo para ordenar preferencias para cada elemento de A
    for a_el in A:
        lista = []

        # Obtener el vector para cada elemento del conjunto B
        for b_a in B:
            vector = [B_dict[b_a][c] for c in A_preferences[a_el]]
            lista.append(vector)


        # Obtener los vectores ordenados y ordenar preferencias
        indices_ordenados = lexicografico(lista)
        B_lista = [B[indice] for indice in indices_ordenados]
        A_pref[a_el] = B_lista


    # Ciclo para ordenar preferencias para cada elemento de B
    for b_el in B:
        lista = []

        # Obtener el vector para cada elemento del conjunto A
        for a_b in A:
            vector = [A_dict[a_b][c] for c in B_preferences[b_el]]
            lista.append(vector)

        # Obtener los vectores ordenados y ordenar preferencias
        indices_ordenados = lexicografico(lista)
        A_lista = [A[indice] for indice in indices_ordenados]
        B_pref[b_el] = A_lista

    return A_pref, B_pref


def lexicografico(vectores):

    # Enumerar vectores para conservar los   ndices
    indices = list(range(len(vectores)))
    enumerados = list(enumerate(vectores))

    # Ordenar enumerados de mayor a menor
    enumerados_ordenados = sorted(enumerados, key=lambda x: x[1], reverse=True)

    # Separar los indices y los vectores ordenados
    indices_ordenados, vectores_ordenados = zip(*enumerados_ordenados)

    return list(indices_ordenados)
```