

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

TESIS DE LICENCIATURA



Matchings Estables Tridimensionales sobre Tres Conjuntos

Análisis del problema de hallar un matching estable tridimensional sobre tres conjuntos como una extensión del problema de los matrimonios estables

Tesistas: Cecilia Lasecki - L.U. 378/89
María Eugenia Riggi - L.U.

124/89

Elizabeth Solmesky - L.U. 1323/88

Director: Dr. Guillermo Durán
Codirectora: Lic. Paula Zabala

Pabellón 1 - Planta Baja - Ciudad Universitaria
(1428) Buenos Aires - Argentina
<http://www.dc.uba.ar>

Resumen

El problema de los matrimonios estables consiste en encontrar una correspondencia o matching entre agentes de dos conjuntos disjuntos que satisfaga a todos los participantes basándose en sus preferencias de tal forma que no exista dos agentes que hubieran preferido estar juntos antes que con sus asignaciones en el matching. Este problema y sus variantes fueron ampliamente estudiados y existen algoritmos que encuentran solución en tiempo polinomial si la misma existe [GS/62].

Una extensión de este problema es el de matching estable de tres dimensiones sobre tres conjuntos donde se trata de encontrar un matching conformado por ternas de agentes en tres conjuntos disjuntos que satisfaga la condición de estabilidad. El mismo fue planteado como un problema abierto en [Knu/76] y fue analizado posteriormente en [NH/95] donde se demuestra que determinar si una instancia dada tiene solución es NP-Completo. Veremos, en el transcurso de este trabajo, que el conjunto de instancias analizadas por [NH/95] forma parte de un universo más amplio. Definiremos y clasificaremos este universo y trabajaremos sobre subconjuntos acotados donde analizaremos la existencia de un matching estable, la búsqueda del mismo y la estructura del conjunto de los matchings estables.

Abstract

The stable marriage problem consists in searching an assignment or matching between agents in two different sets that satisfies all the participants based on the preferences between the agents, where there are no two agents that prefer each other to their assignments on the matching. This problem and its variants were widely studied in the past and there are algorithms that find a solution in polynomial time if such a solution exists [GS/62].

An extension of this problem is the 'tridimensional stable matching problem' where the purpose is to find a matching composed of groups of three agents between three disjointed sets that satisfy the stability condition. This problem was introduced as an open problem by Knuth in [Knu/76] and after that was analyzed in [NH/95] where it is proved that to know if an instance has a solution it is a NP-Complete problem.

In this work we will show, that the set of instances studied by [NH/95] is part of a wider universe. We will define and classify this universe and we will work in restricted subsets where the existence of a stable matching could be found, where to search for this matching, and the structure of the complete stable matching set.

Indice

CAPÍTULO 1: INTRODUCCIÓN	1
Problemática y motivación	1
Desarrollo de esta tesis	2
CAPÍTULO 2: ANTECEDENTES	4
2.1 El problema de los Matrimonios Estables	4
2.1.1 El Algoritmo de Gale y Shapley.....	5
2.1.2 El conjunto de todos los Matchings Estables	6
2.1.3 Variantes.....	8
2.2 El problema de Hospitales-Residentes.....	9
2.3 El problema de los Compañeros de Cuarto.....	10
2.4 Problemas de Matching Estable Tridimensional.....	10
2.5 El problema de los Matchings 'muchos a muchos'	12
CAPÍTULO 3: MATCHING ESTABLE TRIDIMENSIONAL SOBRE TRES CONJUNTOS (MET3)	14
3.1 Definiciones Básicas y Notación.....	14
3.1.1 Matrimonios Estables.....	14
3.1.2 Matchings Estables Tridimensionales sobre tres Conjuntos	15
3.1.2.1 Algunas definiciones adicionales	16
3.2 Propiedades	18
3.3 Clasificación	19
3.3.1 Con Listas de Preferencias Unicas (LU)	19
3.3.2 Con Listas de Preferencias Simples (LS).....	20
3.3.3 Con Listas de Preferencias por Pares (LP).....	21
3.3.4 Con Listas de Preferencias Combinadas (LC)	21
3.4 El conjunto de todos los Matchings Estables	21
3.4.1 El conjunto de todos los Matchings Estables Tridimensionales sobre tres Conjuntos	22
3.4.1.1 Representación del conjunto de todos los matchings estables	22
3.4.1.2 Formulación del problema MET3 como un problema de Programación Lineal Entera.....	23
CAPÍTULO 4: MATCHING ESTABLE TRIDIMENSIONAL SOBRE TRES CONJUNTOS CON LISTAS ÚNICAS	25
4.1 Introducción	25
4.2 Con Listas de Preferencias Mutuas (LUM)	26
4.2.1 Matching y Estabilidad.....	26

4.2.2	Búsqueda de un Matching Estable	27
4.2.3	El conjunto de todos los Matchings Estables	33
4.3	Con Listas de Preferencias Circulares (LUC)	43
4.3.1	Matching y Estabilidad.....	43
4.3.2	Búsqueda de un Matching Estable	46
4.3.3	El conjunto de todos los Matchings Estables	55
CAPÍTULO 5: MATCHING ESTABLE TRIDIMENSIONAL SOBRE TRES CONJUNTOS CON LISTAS SIMPLES		60
5.1	Matching y Estabilidad.....	60
5.2	Búsqueda de un Matching Estable	63
5.3	El conjunto de todos los Matchings Estables	66
CAPÍTULO 6: MATCHING ESTABLE TRIDIMENSIONAL SOBRE TRES CONJUNTOS CON LISTAS POR PARES		70
6.1	Matching y Estabilidad.....	70
6.2	Búsqueda de un Matching Estable	72
6.3	El conjunto de todos los Matching Estables.....	76
6.4	Subconjuntos de instancias.....	78
6.4.1	Listas por Pares Consistentes (LPC)	79
6.4.1.1	Relación con Instancias de Listas Simples.....	80
6.4.2	Listas con Prioridad	82
6.4.2.1	Esquema de Prioridad Mutua (LPPM)	84
6.4.2.1.1	Búsqueda de un Matching Estable en LPPM	84
6.4.2.1.2	El conjunto de todos los Matchings Estables.....	87
6.4.2.2	Esquema de Prioridad Circular (LPPC)	89
6.4.2.2.1	El conjunto de todos los Matchings Estables.....	90
6.4.2.3	Relación con Instancias de Listas Unicas.....	90
6.4.2.3.1	Búsqueda de un Matching Estable	93
CAPÍTULO 7: MATCHING ESTABLE TRIDIMENSIONAL SOBRE TRES CONJUNTOS CON LISTAS COMBINADAS		97
7.1	Introducción	97
7.2	USP.....	98
7.2.1	UsSP.....	98
7.2.1.1	Búsqueda de un Matching Estable	99
7.2.1.2	Subconjuntos de instancias: UsSP con prioridad.....	101
7.2.2	UpSP.....	101
7.2.2.1	Subconjuntos de instancias: UpSP con prioridad	102
7.2.3	El conjunto de todos los Matchings Estables	105
7.3	UUS	106
7.3.1	Búsqueda de un Matching Estable	108
7.3.2	El conjunto de todos los Matchings Estables	110

7.4	UUP	111
7.4.1	U _U U _U P	111
7.4.1.1	Búsqueda de un Matching Estable	112
7.4.1.2	Subconjuntos de instancias: U _U U _U P con prioridad	113
7.4.2	U _U U _P P	114
7.4.2.1	Subconjuntos de instancias: U _U U _P P con prioridad	115
7.4.2.1.1	U _U U _P P _{U_U}	116
7.4.2.1.2	U _U U _P P _{U_P}	116
7.4.3	U _P U _P P	118
7.4.3.1	Subconjuntos de instancias: U _P U _P P con prioridad	119
7.4.4	El conjunto de todos los Matchings Estables	121
7.5	SSU	122
7.5.1	Búsqueda de un Matching Estable	123
7.5.2	El conjunto de todos los Matchings Estables	124
7.6	SSP	125
7.6.1	Subconjuntos de instancias: SSP con prioridad	126
7.6.2	El conjunto de todos los Matchings Estables	128
7.7	PPU	128
7.7.1	Subconjuntos de instancias: PPU con prioridad	130
7.7.1.1	P _P P _P U	130
7.7.1.2	P _U P _U U	132
7.7.1.3	P _P P _U U	134
7.7.1.3.1	P _P P _U U _{P_U}	134
7.7.1.3.2	P _P P _U U _{P_P}	136
7.7.2	El conjunto de todos los Matchings Estables	138
7.8	PPS	139
7.8.1	Subconjuntos de instancias: PPS con prioridad	140
7.8.2	El conjunto de todos los Matchings Estables	145
7.9	El conjunto de todos los Matchings Estables	146
	Listas Unicas + Listas Simples	146
	Listas Unicas + Listas por Pares	146
	Listas Simples + Listas por Pares	147
	Listas Unicas + Listas Simples + Listas por Pares	147
CAPÍTULO 8: CONCLUSIONES, RESULTADOS Y PROBLEMAS ABIERTOS		148
	Resultados y Conclusiones	148
	Problemas Abiertos	150
BIBLIOGRAFÍA		153

Capítulo 1: Introducción

Trataremos a continuación la problemática que motivó esta tesis, e introduciremos otros problemas que dieron origen a este trabajo, describiendo el camino seguido hasta desembocar en los temas tratados. Finalmente daremos un sumario de los temas desarrollados en cada capítulo.

Problemática y motivación

El problema de encontrar un Matching Estable o *Matrimonio Estable* entre dos grupos de individuos fue introducido por Gale y Shapley en el año 1962 [GS/62].

Pero ¿qué es un Matrimonio o Matching Estable?

Sea un conjunto de hombres y otro de mujeres, donde cada individuo de un conjunto tiene una lista ordenada de preferencias sobre todos los individuos del otro conjunto. Si asignamos cada hombre con cada mujer tal que no exista ningún par hombre-mujer que no están juntos, pero que ambos hubieran preferido estar el uno con el otro antes que con su pareja actual, entonces ese matching es estable. Al par o pares cuyos integrantes sin estar juntos hubieran preferido estarlo antes que con sus asignaciones actuales, se le conoce con el nombre de *par bloqueante*, por lo que también podemos definir *matching estable* como aquel matching en el cual no existen pares bloqueantes.

Gale y Shapley aplicaron este concepto para resolver el problema de asignación de estudiantes a colegios donde el objetivo era poder asociar estudiantes a colegios de modo tal que no haya un estudiante que prefiera ingresar en un colegio distinto del que le fue asignado, y a su vez que el colegio también lo prefiera por sobre alguno de los alumnos que le fueron asignados.

Pero aún antes de que Gale y Shapley plantearan el problema, existía en Estados Unidos un programa a nivel nacional: El National Resident Matching Program (NRMP), el cual es un servicio centralizado que permite organizar la asignación de residentes a los hospitales según sus preferencias. El programa fue implementado en el año 1951 y, 11 años después, Gale y Shapley desarrollaron un algoritmo equivalente en un contexto diferente. El éxito del NRMP se debió a que encuentra un matching estable [Rot/90] por lo que si un estudiante no es asignado a su hospital favorito, es porque el hospital tiene sus vacantes cubiertas con estudiantes a los que prefiere.

En el contexto del NRMP, una instancia del problema de matching estable, consiste de dos conjuntos disjuntos de participantes a ser asociados (un conjunto de estudiantes y un conjunto de vacantes en hospitales) y una lista de preferencias de algunos de los participantes sobre las vacantes y de los hospitales sobre los estudiantes. Notemos que en este caso, las listas de preferencias son incompletas, si un estudiante no tiene en su lista la vacante que le ofrece el hospital H , o si el hospital H no tiene en su lista al estudiante S , el par (S, H) no es un par aceptable. Un conjunto de asignaciones aceptables es un *matching estable*, si ningún estudiante y hospital se prefieren antes que sus asignaciones actuales. Además, este mecanismo garantiza que los estudiantes y las vacantes sin asignar serán los mismos para cualquier matching estable.

Gale y Shapley mostraron que siempre existe un matching estable para cualquier conjunto de listas de preferencias, siempre que la lista de preferencias de cada agente incluya a todos los agentes del conjunto opuesto. También está demostrado que existe un matching que es óptimo para uno de los dos conjuntos en el sentido que cada agente está en su mejor asignación posible, y que si hubiera otro matching estable, estaría igual o menos conforme según sus preferencias.

Pero ¿qué ocurriría si en lugar de contar con dos conjuntos de agentes tuviéramos tres conjuntos? ¿Qué pasaría si tuviéramos que organizar comisiones de tres personas pertenecientes a áreas distintas?

El problema original fue planteado por Donald Knuth como una extensión del problema de los matrimonios estables sobre tres conjuntos [Knu/76] en donde cita como ejemplo el problema de hallar un matching entre hombres, mujeres y perros. Es justamente en el ejemplo planteado por el autor donde podemos ver que las preferencias no son independientes, uno podría suponer que las personas priorizan su elección sobre personas a sus preferencias sobre los perros. Esto da lugar a una subclasificación del problema original en instancias cuyas listas de preferencias tienen características particulares. Nuestro trabajo define esta clasificación, amplía las definiciones a cada clase y hace un análisis de cada conjunto de instancias. Presentaremos la definición de estabilidad para ternas, analizaremos la existencia de una solución, definiremos y analizaremos el problema en todas sus variantes y daremos algoritmos para resolver dichas variantes. Para cada caso presentaremos una serie de propiedades y ejemplos que permitan una mejor comprensión y análisis del problema particular sobre el que estamos trabajando. Desde el punto de vista de la Programación Lineal Entera, daremos las restricciones que definen el poliedro correspondiente a cada una de las clases de instancias definidas, como herramienta de especificación del conjunto de matchings estables y punto de partida para futuros análisis sobre estos conjuntos.

Veamos a continuación algunos posibles casos de aplicación.

En una empresa se requieren formar comisiones de trabajo compuestas por empleados pertenecientes a tres áreas distintas. Para no favorecer ciertas comisiones en perjuicio de otras, proponemos buscar un matching estable entre los empleados de cada área. Para esto, cada empleado expresa sus preferencias a través de dos listas, una sobre los empleados de cada una de las otras dos áreas. Que el matching sea estable nos garantiza que no hay tres empleados de distintas áreas que prefieran pertenecer a una comisión a la que no están asignados.

Otra aplicación que puede modelarse como matching estable tridimensional sobre tres conjuntos la encontramos dentro del área de estudios de marketing, en el Market Basket Analysis. Este tema se ocupa de estudiar el comportamiento de las ventas de productos, relacionándolos entre sí. Su aplicación está dada en poder armar ofertas de productos, conociendo de antemano las ventajas que esa agrupación produciría. Por ejemplo puede ser útil el uso de una herramienta que permita armar paquetes u ofertas de tres productos, donde el rendimiento en las ventas sea equilibrado, modelando el problema como la búsqueda de un matching estable tridimensional sobre tres conjuntos, en donde se asocie a cada producto una lista de otras dos clases de productos, según las estadísticas históricas de sus ventas.

Otro caso de aplicación es una ampliación del NRMP donde queremos que cada par estudiante-hospital tenga asignado un supervisor de residencia. En este caso el problema se reduce a la formación de ternas estudiante, hospital y supervisor de forma tal que no haya una terna que prefiriendo trabajar en conjunto no lo haga.

Desarrollo de esta tesis

En este primer capítulo tratamos el problema en líneas generales indicando la problemática y los fundamentos de esta tesis. Dimos también una introducción a la necesidad de definir y estudiar el problema en todas sus variantes.

En el segundo capítulo de esta tesis presentamos una recopilación de trabajos previos junto con sus principales definiciones, algoritmos y teoremas. También presentamos un conjunto de referencias que constituyen el origen del presente trabajo.

Cabe destacar que sólo registramos antecedentes de una de las clases de problemas analizadas en esta tesis. Esta clase será analizada en el Capítulo 6 y fue tratada previamente en [NH/95], donde se demuestra la NP-Complejidad del problema de saber si una instancia tiene un matching estable y en [Alk/87] donde se da un formato de instancias que no tienen solución; no registramos otros trabajos previos sobre el problema objeto de esta tesis y sus variantes. Ninguno de estos trabajos aborda estos problemas mediante las restricciones de Programación Lineal Entera.

En el tercer capítulo definiremos el problema de hallar un matching estable tridimensional sobre tres conjuntos (MET3), daremos el conjunto de definiciones básicas y la notación común que servirá como fuente de consulta para el transcurso de los siguientes capítulos y presentaremos un conjunto de propiedades que cumplen todas las instancias del problema. Finalmente definiremos una clasificación de las instancias en función de los formatos de las listas de preferencias.

Cada objeto de la clasificación antes mencionada será analizada en los siguientes capítulos: MET3 con listas de preferencias únicas, con listas simples, con listas por pares y, finalmente, con listas combinadas. Para cada una definiremos y analizaremos el problema de hallar un matching estable, daremos algoritmos que la resuelvan, y definiremos el conjunto de soluciones desde la Programación Lineal Entera. Finalmente analizaremos la estructura del conjunto de los matchings estables.

En el cuarto capítulo trataremos el problema de encontrar un matching estable tridimensional sobre tres conjuntos con listas de preferencias únicas. Haremos una subclasificación según las preferencias de cada conjunto y daremos algoritmos que resuelvan cada clase. Daremos un algoritmo polinomial no completo para una de estas clasificaciones, *listas de preferencias circulares*, que se menciona en [NH/95] donde se plantea el cálculo de la complejidad como un problema abierto. Mostraremos que no existe un orden entre los matchings estables, sin embargo definiremos una relación de equivalencia entre los mismos bajo la cual mostraremos la existencia de un orden.

En el quinto capítulo estudiaremos el problema de encontrar un matching estable tridimensional sobre tres conjuntos con listas simples, para el cual presentaremos un algoritmo que resuelve el problema en tiempo polinomial. Finalmente mostraremos que existe un orden en el conjunto de los matchings estables para instancias de este tipo.

En el sexto capítulo analizaremos el problema de encontrar un matching estable tridimensional sobre tres conjuntos con listas por pares. Se trata del problema original planteado por Knuth en [Knu/76] y tratado por Ng y Hirschberg en [NH/95]. Dado que el problema es NP-Completo, daremos un algoritmo polinomial no completo que puede encontrar solución para ciertas instancias. Luego trataremos algunos subconjuntos de instancias con características particulares y, en algunos casos, daremos algoritmos exactos que lo resuelvan. Finalmente mostraremos que existe un orden en el conjunto de los matchings estables para instancias de este tipo.

En el séptimo capítulo presentaremos el problema de encontrar un matching estable tridimensional sobre tres conjuntos con listas combinadas. Este capítulo tiene por objetivo tratar aquellas instancias que combinan distintos tipos de listas de preferencias obedeciendo a los formatos definidos en los capítulos anteriores. Para la mayoría de los casos daremos algoritmos que los resuelven, mientras que para otros daremos propiedades que los llevan a problemas conocidos.

En el último capítulo se resumen los principales resultados, las conclusiones y planteamos algunos problemas abiertos que podrán ser estudiados a futuro.

Capítulo 2: Antecedentes

En este capítulo presentamos el problema de los matrimonios estables en particular y otros problemas de matching estable en general. Desde su introducción en 1962 por Gale y Shapley [GS/62], los problemas de matching estable han sido de sumo interés no sólo en el área de ciencias de la computación, sino también de la matemática, economía y teoría de juegos.

También hubo una serie de libros profundizando distintos aspectos, entre los cuales podemos citar como más destacados el de Knuth [Knu/76] conocido por los doce problemas abiertos que plantea y cuya primera edición es de 1976, el de Gusfield e Irving [GI/89] y el de Roth y Sotomayor [RS/92].

Presentaremos entonces los principales problemas de matching estable como el problema de los hospitales y residentes, el problema de los compañeros de cuarto, matching ‘muchos a muchos’ y por último, el problema de matching estable en tres dimensiones, cuyo análisis sienta las bases del presente trabajo.

2.1 El problema de los Matrimonios Estables

En términos generales, un *matching* es una correspondencia entre agentes pertenecientes a algún conjunto o conjuntos. En el problema de los matrimonios estables se busca encontrar un matching entre agentes de dos conjuntos disjuntos que cumpla con la propiedad de *estabilidad*. Se dice que un matching es estable si no existe ningún par de agentes que hubieran preferido estar juntos, antes que con su pareja actual. Si existiera dicho par, el mismo es denominado *par bloqueante*.

Este escenario general, provee un modelo para una variedad de problemas con interpretación práctica. En el problema clásico de los matrimonios estables, hay dos conjuntos de igual tamaño, uno formado por hombres y el otro por mujeres, donde cada agente tiene una lista de preferencias estrictamente ordenada con respecto a todos los integrantes del conjunto opuesto. Bajo estas condiciones se considera un matrimonio al conjunto de las relaciones ‘uno a uno’ de cada mujer con cada hombre y se considera que el matrimonio es estable si no existe ningún par bloqueante.

El principal resultado obtenido por Gale y Shapley [GS/62] fue que para cualquier instancia del problema de matrimonios estables, existe por lo menos un matching que cumple con la propiedad de estabilidad. Estos autores probaron este resultado a través de un algoritmo que siempre encuentra un matching para cualquier instancia dada. Por otro lado, demuestran que el algoritmo siempre le otorga a cada hombre (mujer, si los roles se invierten) la mejor pareja según su lista de preferencias, con respecto a las asignaciones que le hubieran tocado en cualquier otro matching estable.

A continuación mostramos un ejemplo del problema de matrimonios estables, con una instancia de tamaño $n=4$ y los conjuntos $H=\{h_0, h_1, h_2, h_3\}$ (hombres) y $M=\{m_0, m_1, m_2, m_3\}$ (mujeres), donde cada miembro tiene una lista de preferencias con respecto al conjunto opuesto. También damos un matching estable y uno que no lo es.

Ejemplo 2.1

h_0 :	$m_0 m_1 m_2 m_3$	m_0 :	$h_0 h_2 h_3 h_1$
h_1 :	$m_2 m_1 m_3 m_0$	m_1 :	$h_0 h_1 h_2 h_3$
h_2 :	$m_2 m_1 m_3 m_0$	m_2 :	$h_1 h_2 h_0 h_3$
h_3 :	$m_1 m_2 m_3 m_0$	m_3 :	$h_3 h_0 h_2 h_1$

En esta instancia, el matching $M_1=\{(h_0, m_0), (h_1, m_2), (h_2, m_1), (h_3, m_3)\}$ es estable. Para verificar la estabilidad se puede verificar que todos los posibles pares no bloqueen al matching. Otra forma más

Antecedentes

eficiente es chequear sólo los posibles pares bloqueantes. En este caso los candidatos son: (h_2, m_2) , (h_3, m_1) y (h_3, m_2) . Alcanza con ver que la mujer m_2 prefiera al hombre que le fue asignado en el matching, antes que al hombre h_2 y al h_3 , y que la mujer m_1 prefiera a su pareja actual antes que al hombre h_3 . Observando las listas de preferencias de m_1 y de m_2 puede apreciarse que ambas prefieren a sus parejas, por lo que no hay pares bloqueantes para M_1 . En cambio el matching $M_2 = \{(h_0, m_1), (h_1, m_2), (h_2, m_0), (h_3, m_3)\}$ no es estable ya que el par (h_0, m_0) bloquea al matching, pues el hombre h_0 prefiere estar con la mujer m_0 antes que con su pareja actual (m_1) , y m_0 prefiere estar con el hombre h_0 antes que con su pareja actual (h_2) .

2.1.1 El Algoritmo de Gale y Shapley

En esta sección daremos el algoritmo de Gale y Shapley que encuentra un matrimonio estable para dos conjuntos de igual cardinalidad y con listas de preferencias completas, es decir sobre todos los integrantes del conjunto opuesto. Los autores probaron que siempre existe al menos un matrimonio estable para cualquier instancia del problema [GS/62].

Informalmente el algoritmo puede describirse como una secuencia de "propuestas" de los hombres hacia las mujeres. Todos los hombres hacen su propuesta a la primera mujer de su lista tal que no los haya rechazado. Si una misma mujer recibe más de una propuesta, ella se queda con el hombre preferido, según su lista de preferencias. El algoritmo continúa hasta que todas las parejas se hayan completado. A medida que se va avanzando en el algoritmo, los hombres que van siendo rechazados son cada vez menos favorecidos, mientras que cada mujer que va recibiendo mayor cantidad de propuestas, termina mejor favorecida.

Cuando una mujer no asignada recibe una propuesta, la acepta, pero si más adelante recibe otra propuesta, ella comparará su pareja actual con su nuevo candidato, y se quedará con el que ella prefiera, pudiendo rechazar entonces a su pareja anterior. El algoritmo termina cuando cada hombre tiene asignada una mujer distinta, y el matching resultante es estable.

A continuación mostramos el algoritmo de Gale y Shapley:

```

Cada persona está libre
Mientras algún hombre  $h$  esté libre hacer
Comienzo
     $m$ :=primera mujer de la lista de preferencias de  $h$  que no ha sido propuesta
    Si  $m$  está libre entonces
        Asignar  $h$  a  $m$ 
    sino
        Si  $m$  prefiere a  $h$  antes que a su pareja actual  $h'$ 
            romper la relación  $m-h'$  y asociar  $m$  con  $h$ 
        sino
             $m$  rechaza a  $h$  y  $h$  permanece libre
Fin
El matching obtenido es estable.

```

Figura 2.1 Algoritmo de Gale y Shapley

El matching obtenido es *óptimo* para el conjunto que hace las propuestas, y es el peor para el conjunto que las recibe. En este caso, el matching obtenido es *óptimo-masculino*. Si invertimos los roles en el algoritmo, de forma tal que la mujer realice las propuestas, el matching resultante será *óptimo-femenino*. Observando el algoritmo puede verse que el mismo posee ciertos elementos no determinísticos en el sentido del orden en el cual cada hombre libre hace sus propuestas, pero esto no afecta al resultado final del algoritmo, que siempre es el mismo.

Antecedentes

Si observamos los pasos de este algoritmo, podemos ver que permanentemente se recorren las listas de preferencias de las mujeres para comparar las posiciones de dos hombres, este recorrido constante de las listas de preferencia hace que el algoritmo tenga una complejidad de $O(n^3)$. Para realizar una implementación más eficiente necesitamos poder determinar en tiempo constante si una mujer prefiere a un hombre o a otro, esto puede lograrse utilizando una *matriz de ranking* R_M de dimensiones $n \times n$ en la cual $R_M(m,h)=k$ si h está en la k -ésima posición en la lista de preferencias de m . Luego, la complejidad de este algoritmo es de $O(n^2)$ para el peor caso. Casi 30 años después de que Gale y Shapley publicaran el algoritmo, Ng y Hirschberg demuestran que el algoritmo es óptimo en el sentido de que el orden logrado es el mínimo posible para resolver este problema [NH/90].

Relacionado con los matchings óptimos existen una serie de propiedades interesantes. Por ejemplo Mc Vitie y Wilson [VW/71] observaron que el matching estable óptimo masculino resulta ser el pésimo matching estable femenino. Por otro lado si el matching estable óptimo-masculino y el óptimo-femenino coinciden, entonces existe un único matching estable.

2.1.2 El conjunto de todos los Matchings Estables

Dada una instancia cualquiera del problema de los matrimonios estables, el matching óptimo-masculino y el óptimo-femenino son los extremos de todos los matchings estables, pudiendo existir otros matchings intermedios. Como vemos, puede existir más de un matching estable para una instancia dada del problema, y es posible definir un orden entre tales matchings en función de las listas de preferencias de los individuos en ambos conjuntos. De esta manera, se considera que un matching es preferible a otro matching por los individuos en un conjunto, digamos A , si los individuos de A se encuentran relacionados en un matching con individuos de B a los que prefieren por sobre sus parejas en el segundo matching. Para ello se define una relación de *dominación* entre los matrimonios estables de acuerdo a lo siguiente: el matching M_1 *domina* al matching M_2 con respecto a un conjunto (hombres) si todos los individuos de ese conjunto (hombres) prefieren o igualan sus parejas en M_1 a las de M_2 . Esta relación de dominación define un orden parcial entre todos los matchings estables de una instancia dada, formando un reticulado distributivo en el cual el matching óptimo masculino (femenino) será el supremo, y el femenino (masculino) será el ínfimo. Esta estructura de reticulado fue observada por primera vez por John Conway [Knu/76].

Knuth también demostró que si M y M' son estables, entonces MM' y MM' son matchings estables.

Recordemos que un reticulado distributivo es un orden parcial \geq en el cual:

1. cada par de elementos (a, b) tiene ínfimo, denotado $a \wedge b$ tal que $a \geq a \wedge b$ y $b \geq a \wedge b$ y no existe elemento c tal que $a \geq c$, $b \geq c$ y $c > a \wedge b$.
2. cada par de elementos (a, b) tiene un supremo, denotado $a \vee b$ tal que $a \vee b \geq a$ y $a \vee b \geq b$ y no existe elemento c tal que $c \geq a$, $c \geq b$ y $a \vee b > c$.
3. se conservan las leyes distributivas, es decir,

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

Esta estructura de reticulado dio origen a un conjunto de propiedades que contribuyeron en gran medida en el estudio de la estructura de todos los matchings estables. También permitió desarrollar algoritmos eficientes que construyen todos los matchings partiendo de alguno de los óptimos, en $O(n^2 + n|S(I)|)$ donde $S(I)$ es el conjunto de todos los matchings estables de la instancia I y n es el tamaño de I [ILG/86].

Para un mejor entendimiento, daremos a continuación un ejemplo de una instancia de tamaño $n=4$ con diez matrimonios estables, que representaremos como reticulado.

Ejemplo 2.2

$h_0:$	$m_0 m_1 m_2 m_3$	$m_0:$	$h_3 h_2 h_1 h_0$
$h_1:$	$m_1 m_0 m_3 m_2$	$m_1:$	$h_2 h_3 h_0 h_1$
$h_2:$	$m_2 m_3 m_1 m_0$	$m_2:$	$h_1 h_0 h_3 h_2$

Antecedentes

 $h_3: \quad m_3 \ m_2 \ m_1 \ m_0$
 $m_3: \quad h_0 \ h_1 \ h_2 \ h_3$

Los siguientes matchings son estables:

 $M_1 = \{(h_0, m_0), (h_1, m_1), (h_2, m_2), (h_3, m_3)\}$
 $M_2 = \{(h_0, m_1), (h_1, m_0), (h_2, m_2), (h_3, m_3)\}$
 $M_3 = \{(h_0, m_0), (h_1, m_1), (h_2, m_3), (h_3, m_2)\}$
 $M_4 = \{(h_0, m_1), (h_1, m_0), (h_2, m_3), (h_3, m_2)\}$
 $M_5 = \{(h_0, m_1), (h_1, m_3), (h_2, m_0), (h_3, m_2)\}$
 $M_6 = \{(h_0, m_2), (h_1, m_0), (h_2, m_3), (h_3, m_1)\}$
 $M_7 = \{(h_0, m_3), (h_1, m_2), (h_2, m_0), (h_3, m_1)\}$
 $M_8 = \{(h_0, m_2), (h_1, m_3), (h_2, m_1), (h_3, m_0)\}$
 $M_9 = \{(h_0, m_2), (h_1, m_3), (h_2, m_1), (h_3, m_0)\}$
 $M_{10} = \{(h_0, m_3), (h_1, m_2), (h_2, m_1), (h_3, m_0)\}$

En la estructura de reticulado ilustrada en la Figura 2.2, se muestra un grafo dirigido, donde los nodos son los matrimonios estables y los arcos reflejan la relación de dominación. Los matchings M_1 y M_{10} representan el óptimo y el pésimo matching respectivamente, con respecto a las preferencias de los hombres.

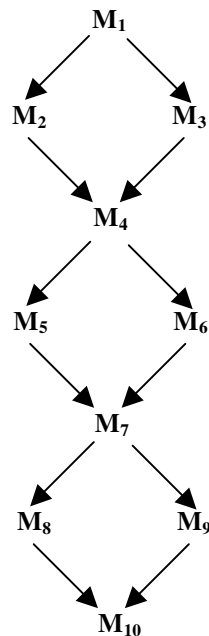


Figura 2.2 Estructura de reticulado para la instancia el Ejemplo 2.2

En una instancia del problema puede haber muchos matrimonios estables y está demostrado que la cantidad de soluciones o matrimonios estables tiene un crecimiento exponencial en función del tamaño de la instancia [Knu/76]. Es aún un problema abierto determinar para una instancia de tamaño n , cuál es la cantidad máxima de matrimonios estables y cuál es la estructura de las listas de preferencias para maximizar ese valor. Irving y Leather demostraron que determinar la cantidad de matchings estables en una instancia dada es un problema NP-Completo [IL/86].

Desde el punto de vista de la programación entera, podemos caracterizar el conjunto de todos los matchings estables, como los puntos factibles del correspondiente poliedro que satisface un conjunto de restricciones, donde las variables pueden tomar los valores 0 ó 1.

Antecedentes

Es posible caracterizar los vectores de incidencia de matchings estables de una instancia del problema de hallar un matrimonio estable como los puntos factibles del correspondiente poliedro definido por el siguiente sistema de inecuaciones lineales [Abe/93]:

Sea H el conjunto de hombres y M el conjunto de mujeres.

$$\text{Sea } x_{h,m} = \begin{cases} 1, & \text{si } (h,m) \text{ pertenece al matching} \\ 0, & \text{si } (h,m) \text{ no pertenece al matching} \end{cases} \quad \text{con } (h,m) \in H \times M$$

$$\bullet \quad x_{h,j} = 1, \quad h \in H, \quad j \in M \quad (1)$$

$$\bullet \quad x_{i,m} = 1, \quad m \in M, \quad i \in H \quad (2)$$

$$\bullet \quad x_{h,j} + x_{i,m} + x_{h,m} = 1, \quad (h,m) \in H \times M, \quad \begin{matrix} j > m \\ h > i \end{matrix} \quad (3)$$

$$x_{h,m} \in \{0,1\} \quad (h,m) \in H \times M \quad (4)$$

Puede verse que (1) garantiza que todos los hombres se asocian con una y sólo una mujer. Asimismo (2) garantiza que todas las mujeres se asocien con un solo hombre. Por otro lado las restricciones (1), (2), y (4) garantizan que se trata de un matching, mientras que (3) garantiza la condición de estabilidad.

2.1.3 Variantes

Como mencionamos anteriormente, existe una serie de trabajos que plantean variantes al problema original de los matrimonios estables. En esta sección citaremos los más conocidos.

Supongamos que los participantes de ambos conjuntos no están obligados a incluir en sus listas de preferencias a todos los miembros del otro conjunto. Es fácil ver que puede haber instancias en las cuales no exista un matching estable. Este tipo de listas es conocido con el nombre de *listas incompletas*, y fue planteado y analizado por primera vez por Knuth [Knu/76], y más tarde por Gale y Sotomayor entre otros. En la instancia del Ejemplo 2.3 podemos observar que el único matrimonio posible $M = \{(h_0, m_0), (h_1, m_3), (h_2, m_2), (h_3, m_1)\}$ es bloqueado por el par (h_1, m_2) .

Ejemplo 2.3

$h_0:$	m_0	$m_0:$	$h_2 \ h_0 \ h_1$
$h_1:$	$m_2 \ m_0 \ m_3$	$m_1:$	$h_1 \ h_0 \ h_2 \ h_3$
$h_2:$	$m_2 \ m_0 \ m_3$	$m_2:$	$h_0 \ h_1 \ h_2 \ h_3$
$h_3:$	$m_1 \ m_0 \ m_2$	$m_3:$	$h_0 \ h_1$

Sin embargo, una conversión apropiada de las listas incompletas en completas [Knu/76] permite determinar la existencia o no de solución.

Más tarde, Gale y Sotomayor muestran que en una instancia con listas incompletas, si un individuo queda sin pareja en un matrimonio estable, estará solo en cualquier otro matrimonio estable que posea esa instancia.

Otra de las variantes estudiadas consiste en trabajar con *conjuntos de distinta cardinalidad*. Como puede verse en el Ejemplo 2.4, hay dos hombres y tres mujeres, por lo cual habrá una mujer sin pareja en los

Antecedentes

matchings estables resultantes. Extendiendo la noción de estabilidad, diremos que un matching M es inestable si existe un par (m,w) no perteneciente al matching tal que m y w prefieren estar juntos, antes que estar solos o con sus parejas actuales.

Ejemplo 2.4

$h_0:$	$m_1 m_2 m_0$	$m_0:$	$h_1 h_0$
$h_1:$	$m_2 m_1 m_0$	$m_1:$	$h_0 h_1$
		$m_2:$	$h_0 h_1$

En esta instancia el matching $M = \{(h_0, m_1), (h_1, m_2)\}$ es estable.

McVitie y Wilson [VW/70] fueron los precursores en el estudio de este tipo de instancias. Su principal resultado muestra que el individuo que se queda sin pareja en un matching estable, estará solo en todos los matchings estables que tenga esa instancia.

Por último, si los individuos pueden tener una lista de preferencias en la cual la posición de algunos integrantes pueda ser equivalente, es decir, sin un orden de preferencia estricto, se presenta una nueva variante que se conoce como *listas con indiferencia*, y tres nuevas definiciones posibles de estabilidad. Una primer noción de estabilidad asume que un matching es inestable si existe un par hombre-mujer que no están juntos, pero prefieren estarlo o a ambos les resulta indistinto el cambio, por estar en posiciones equivalentes la pareja actual y la que provoca la inestabilidad. Esta definición es conocida como *superestabilidad*. Una segunda definición algo menos exigente, es que un matching es inestable si existe un par hombre-mujer que no están juntos, tal que uno de los dos prefiere estrictamente al otro antes que a su pareja actual, y el otro es al menos indiferente. A esta definición se la conoce con el nombre de *estabilidad fuerte*. Por último, la tercer definición y probablemente la más razonable en donde se permiten listas con indiferencia dice que un matching es estable si no existe un par hombre-mujer que no están juntos, pero que ambos prefieren estarlo en forma estricta con respecto a sus parejas actuales. Esta estabilidad se la conoce con el nombre de *estabilidad débil*. Notemos que utilizando esta definición, puede aplicarse el algoritmo de Gale y Shapley para hallar un matching estable.

2.2 El problema de Hospitales-Residentes

Desde 1952 el National Resident Matching Program, que permitía asociar residentes de medicina con hospitales de Estados Unidos, venía utilizando un algoritmo similar al que posteriormente plantearon Gale y Shapley. Una de las principales diferencias con este algoritmo es que posteriormente se lo modificó para que favorezca igualmente a los residentes y a los hospitales, pues en sus comienzos favorecía a los hospitales. Otra de las principales diferencias con el modelo estándar, es la asimetría que existe entre ambos conjuntos, es decir, un hospital tiene una cantidad de vacantes por cubrir, mientras que un residente puede asistir a un solo hospital. Por esta razón este es un modelo 'uno a muchos' pues un hospital busca tener una determinada cantidad de residentes, cumpliendo con un criterio de estabilidad. Como la cantidad de vacantes no necesariamente coincide con la cantidad de residentes, se permite que cualquiera de los dos conjuntos tenga listas de preferencias con elementos inaceptables.

En este contexto, un matching es una asignación parcial entre un conjunto de residentes y hospitales a los cuales se les asocia una cantidad de residentes que no supere la cantidad de vacantes de cada hospital. Un matching es estable si no existe ningún par $r-h$ (residente-hospital) tal que cumpla las siguientes condiciones:

- i) ambos son aceptables entre sí (es decir aparecen en sus listas de preferencias)
- ii) r quedó sin asignar, o prefiere el hospital h al que le hubiera tocado
- iii) h quedó con vacantes, o h prefiere el residente r a alguno de los residentes asignados.

Antecedentes

El sistema fue modificado entre 1995 y 1997 para orientarlo a favorecer a los aspirantes más que a los hospitales tal como era el sistema original. Sin embargo las mejoras alcanzaron un poco menos del 0.1% de los aspirantes debido a que las listas de preferencias de los hospitales son muy similares en sus primeros participantes (los mejores estudiantes) y, consecuentemente, la cantidad de matchings posibles es limitada [Rot/95] y [Rot/97].

Una extensión de este problema es la *inclusión de parejas*, donde dos estudiantes pueden pedir estar juntos en el mismo hospital o misma ciudad. En este caso cada pareja puede tener una lista de pares de hospitales. La NP-Complejidad de este problema fue probada por Ronn en 1990 [Ron/90]. Hay instancias de este problema que no tienen matching estable. Bajo instancias de este tipo, los residentes u hospitales sin asignar son los mismos para todos los matchings estables posibles [WV/70].

2.3 El problema de los Compañeros de Cuarto

El problema de los compañeros de cuarto plantea resolver la asignación de pares de estudiantes en cuartos de una universidad, donde cada estudiante tiene una lista de preferencias estrictamente ordenada de los compañeros restantes. Se trata de una generalización del problema de los matrimonios estables, en el cual una instancia consiste en un conjunto de n personas de cardinalidad par, cada una incluyendo en su lista de preferencias a las $n-1$ personas restantes en orden estricto. Un matching entre los miembros de este conjunto es una partición del mismo en pares disjuntos. Al igual que para el problema de los matrimonios estables, se dice que un matching es estable si no existe ningún par de personas tal que hubieran preferido estar juntas, antes que con sus compañeros actuales.

A diferencia del problema de los matrimonios estables, este problema puede no tener solución, lo cual fue demostrado por Gale y Shapley [GS/62] a través del siguiente contraejemplo:

Ejemplo 2.5

c_0 : $c_2 c_1 c_3$
 c_1 : $c_0 c_2 c_3$
 c_2 : $c_1 c_0 c_3$
 c_3 : \dots

En esta instancia ninguno de los pares (c_0, c_3) , (c_1, c_3) y (c_2, c_3) podrá pertenecer a un matching estable pues si el par (c_0, c_3) pertenece al matching, entonces el par (c_0, c_1) lo bloquea. Asimismo, si alguno de los pares (c_1, c_3) o (c_2, c_3) perteneciera al matching, éste sería bloqueado por los pares (c_1, c_2) y (c_2, c_3) respectivamente.

Irving mostró en 1985 que determinar si este problema tiene solución puede ser resuelto en $O(n^2)$, en cambio Ronn demostró en [Ron/90] que la variante del problema que acepta la inclusión de parejas es un problema NP-Completo.

Por último, en [Chu/98] se presentó una condición suficiente para que una instancia tenga solución.

2.4 Problemas de Matching Estable Tridimensional

Consideremos el problema de asignar $3n$ estudiantes a n grupos de trabajo distintos. Cada uno tiene una lista de preferencias formada por todos los pares de estudiantes restantes. En este contexto un matching es estable si no existe ninguna terna de estudiantes tal que prefieran estar juntos, antes que con las asignaciones del matching. Como puede verse este problema es una generalización del problema de los compañeros de cuarto, en donde se busca formar ternas en lugar de pares.

El problema de los matrimonios estables tiene una generalización similar, en donde tenemos tres conjuntos disjuntos, cuyos integrantes tienen listas de preferencias formadas por pares de ambos conjuntos restantes. Este problema fue uno de los doce problemas abiertos planteados por Knuth en su libro *Marriages Stables* [Knu/76].

Una instancia de este problema está formada por tres conjuntos, A, B y C de igual cardinalidad y las listas de preferencias de los agentes. Un "matrimonio" en esta instancia es un matching completo de tres conjuntos, digamos un subconjunto de $A \times B \times C$ donde cada elemento de A, B y C aparece exactamente una vez.

Veamos un ejemplo.

Ejemplo 2.6

Sean los conjuntos A, B y C, donde $A=\{a_0, a_1, a_2\}$, $B=\{b_0, b_1, b_2\}$ y $C=\{c_0, c_1, c_2\}$ y sean las siguientes listas de preferencias:

a_0 : $(b_0, c_0) (b_1, c_0) (b_2, c_0) (b_1, c_2) (b_2, c_1) (b_2, c_2) (b_1, c_1) (b_0, c_2) (b_0, c_1)$
 a_1 : $(b_1, c_2) (b_0, c_1) (b_1, c_0) (b_2, c_0) (b_0, c_0) (b_1, c_1) (b_2, c_2) (b_0, c_2) (b_2, c_1)$
 a_2 : $(b_0, c_2) (b_2, c_1) (b_0, c_1) (b_1, c_2) (b_0, c_0) (b_1, c_1) (b_2, c_2) (b_1, c_0) (b_2, c_0)$

b_0 : $(a_0, c_1) (a_1, c_1) (a_2, c_2) (a_1, c_0) (a_0, c_2) (a_2, c_0) (a_2, c_1) (a_0, c_0) (a_1, c_2)$
 b_1 : $(a_1, c_0) (a_0, c_1) (a_2, c_0) (a_0, c_2) (a_2, c_1) (a_1, c_2) (a_0, c_0) (a_1, c_1) (a_2, c_2)$
 b_2 : $(a_1, c_0) (a_2, c_0) (a_0, c_0) (a_1, c_1) (a_1, c_2) (a_2, c_1) (a_0, c_1) (a_2, c_2) (a_0, c_2)$

c_0 : $(a_0, b_1) (a_0, b_2) (a_2, b_2) (a_1, b_1) (a_0, b_0) (a_1, b_0) (a_1, b_2) (a_2, b_1) (a_2, b_2)$
 c_1 : $(a_0, b_1) (a_0, b_2) (a_1, b_1) (a_2, b_2) (a_1, b_0) (a_0, b_0) (a_2, b_1) (a_1, b_2) (a_2, b_0)$
 c_2 : $(a_2, b_0) (a_0, b_2) (a_2, b_1) (a_1, b_2) (a_0, b_1) (a_1, b_0) (a_1, b_1) (a_2, b_2) (a_0, b_0)$

Podemos encontrar el siguiente matching estable: $M=\{(a_0, b_1, c_2), (a_1, b_0, c_1), (a_2, b_2, c_0)\}$.

Veamos ahora una instancia donde no es posible encontrar matching estable [NH/95].

Ejemplo 2.7

Sean los conjuntos A, B y C, donde $A=\{a_0, a_1\}$, $B=\{b_0, b_1\}$ y $C=\{c_0, c_1\}$ y sean las siguientes listas de preferencias:

a_0 : $(b_0, c_1) (b_0, c_0) (b_1, c_1) (b_1, c_0)$
 a_1 : $(b_1, c_1) (b_0, c_0) (b_1, c_0) (b_0, c_1)$

b_0 : $(a_1, c_0) (a_0, c_1) (a_0, c_0) (a_1, c_1)$
 b_1 : $(a_1, c_0) (a_0, c_0) (a_1, c_1) (a_0, c_1)$

c_0 : $(a_0, b_1) (a_0, b_0) (a_1, b_0) (a_1, b_1)$
 c_1 : $(a_0, b_0) (a_1, b_1) (a_0, b_1) (a_1, b_0)$

Para cada matching posible veamos que no se trata de un matching estable:

$\{(a_0, b_0, c_0), (a_1, b_1, c_1)\}$ no es estable pues es bloqueado por (a_0, b_0, c_1)
 $\{(a_0, b_0, c_1), (a_1, b_1, c_0)\}$ no es estable pues es bloqueado por (a_1, b_0, c_0)
 $\{(a_0, b_1, c_0), (a_1, b_0, c_1)\}$ no es estable pues es bloqueado por (a_0, b_0, c_1)
 $\{(a_0, b_1, c_1), (a_1, b_0, c_0)\}$ no es estable pues es bloqueado por (a_1, b_1, c_1) .

A diferencia del problema de los matrimonios estables, hallar un matching estable en instancias de estos problemas puede no tener solución. Ng y Hirschberg llaman a este problema 3GSM (Three Gender Stable

Marriage), y demuestran que decidir si una instancia del mismo tiene solución estable es un problema NP-Completo [NH/95].

También presentan el 3PSA (Three Person Stable Assignment) en donde el objetivo es hallar un matching estable tridimensional, pero los agentes pertenecen a un mismo conjunto. Los autores demuestran que el problema de determinar la existencia de un matching estable en instancias de este tipo también es NP-Completo.

El mismo trabajo plantea dos problemas abiertos. El primero es analizar la complejidad del problema si las listas de preferencias son consistentes, es decir, que sea $(x,y) \succeq_a(x,z)$ para todos los x' s o para ninguno. Por ejemplo: $a_0: (b_0,c_1) (b_0,c_0) (b_1,c_1) (b_1,c_0)$ es una lista consistente, pues todos los b_i están primero con c_1 y luego con c_0 .

El segundo es analizar la complejidad del problema de encontrar un matching estable en tres dimensiones, pero en donde los agentes de cada conjunto tienen listas de preferencias con respecto a un único conjunto y en forma circular (3GSM circular). Los agentes pertenecientes al conjunto A tienen preferencias definidas sobre los agentes de B, los de B sobre los agentes del conjunto C, y los agentes de C sobre los de A. Estos problemas serán tratados en los Capítulos 6 y 4, respectivamente.

2.5 El problema de los Matchings ' muchos a muchos'

En este problema existen dos conjuntos de agentes en donde cada uno puede formar sociedades con miembros del conjunto opuesto. Cada agente tiene una cuota que representa el número de agentes con los que se asociará, y un orden de preferencias entre todos los conjuntos posibles de socios. En estos términos un matching es estable si ningún subconjunto S de agentes, puede formar grupos de agentes en donde todos los miembros de S prefieran estar. Existen tres formas diferentes de precisar la estabilidad:

- a) Un matching es *estable de a pares (pairwise)* si no existen duplas de agentes p y q tales que no son socios, pero preferirían serlo, y que a pesar de los intercambios, es posible mantener las cuotas de los agentes.
- b) Un matching es *centralmente estable (corewise)* si no existe ningún subconjunto de agentes que puedan formar grupos sólo entre ellos, tal que sean estrictamente preferidos a sus conjuntos originales.
- c) Un matching es *conjunto-estable (setwise)* si no existe ningún subconjunto de agentes que puedan formar grupos sólo entre ellos, y que a pesar de los intercambios, es posible mantener las cuotas de los agentes. [Sot/99]

Ejemplo 2.8

Sean P y Q dos conjuntos de agentes, y sean M y M' los siguientes matchings:



Listas de preferencias:

$p_1: \dots q_3 q_2 \dots$
 $q_3: \dots p_1 p_2 \dots$

En este ejemplo podemos ver que en M, p_1 se asocia a $\{q_1, q_2\}$, pero prefiere más a q_3 que a q_2 . Por otro lado q_3 se asocia a $\{p_2, p_3\}$, sin embargo prefiere más a p_1 que a p_2 . El matching M no es estable de a pares porque p_1 y q_3 prefieren estar juntos, como puede verse en M' .

Roth definió en *Stability and polarization of interest in job matching*, 1984 [Sot/99], el concepto de *estabilidad de grupos* para los problemas de matching ‘muchos a muchos’ en donde las preferencias son estrictas.

Capítulo 3: Matching Estable Tridimensional sobre tres Conjuntos (MET3)

En este capítulo presentaremos el problema de hallar un matching estable tridimensional sobre tres conjuntos. En la primera parte daremos algunas definiciones básicas para el resto de esta tesis, comenzaremos dando la definición del problema de los matrimonios estables planteado por Gale y Shapley, extenderemos el problema a tres dimensiones y daremos las definiciones correspondientes a nuestro trabajo. También introduciremos la notación utilizada tanto en éste como en los siguientes capítulos. Enunciaremos algunas propiedades que cumplen todas las instancias de este problema para terminar dando una clasificación que será fuente de los problemas tratados en los siguientes capítulos.

3.1 Definiciones Básicas y Notación

Damos a continuación las definiciones básicas del problema planteado por Gale y Shapley en [GS/62]. Más adelante en este capítulo, extenderemos estas definiciones al problema de hallar un matching estable tridimensional sobre tres conjuntos.

3.1.1 Matrimonios Estables

Como introdujimos en el Capítulo 2, el problema de los matrimonios estables consiste en encontrar una asociación entre individuos de dos conjuntos disjuntos, uno de hombres y otro de mujeres, tal que no haya una pareja donde ambos prefieran estar juntos pero no lo estén.

Sea $A = \{h_0, h_1, \dots, h_{n-1}\}$ el conjunto de los hombres y $B = \{m_0, m_1, \dots, m_{n-1}\}$ el conjunto de las mujeres. Asociado con cada persona x hay una lista estrictamente ordenada conteniendo a todos los individuos del conjunto opuesto, esta lista recibe el nombre de lista de preferencias y es notada $P(x)$. Se dice que un hombre h prefiere a una mujer m_i antes que a m_j y es notado $m_i >_h m_j$, si m_i precede a m_j en la lista de preferencias de h . Las preferencias de las mujeres son establecidas de forma análoga.

Un matching M es una correspondencia uno a uno entre hombres y mujeres, donde cada hombre y cada mujer aparecen a lo sumo una vez. Es decir que $M = \{(a_{i_0}, b_{j_0}), \dots, (a_{i_{n-1}}, b_{j_{n-1}})\}$ es un matching si $i_0 \neq i_1 \neq \dots \neq i_{n-1}$ y $j_0 \neq j_1 \neq \dots \neq j_{n-1}$.

Notaremos $M_A(m)$ al $h \in A$ tal que $(h, m) \in M$ y $M_B(h)$ al $m \in B$ tal que $(h, m) \in M$. Un matching es completo si $\forall h \in A, \exists m \in B$ tal que $(h, m) \in M$, y $\forall m \in B, \exists h \in A$ tal que $(h, m) \in M$.

Se dice que un par (h, m) bloquea a un matching M o que es un par bloqueante para M si $h >_m M_A(m)$ y $m >_h M_B(h)$. Un matching es estable si es completo y no tiene pares bloqueantes. Un matching que cumpla estas características también será referenciado como matrimonio estable.

Vimos en el Capítulo 2, que el algoritmo planteado por Gale y Shapley [GS/62] es óptimo en el sentido de que el orden logrado es el mínimo posible para resolver este problema [NH/90]. A partir de esto, cada vez que se necesite hallar un matching estable bidimensional asumiremos que se utiliza este algoritmo.

3.1.2 Matchings Estables Tridimensionales sobre tres

Conjuntos

Sean A, B y C tres conjuntos disjuntos de agentes o individuos de tamaño n , $A = \{a_0, a_1, \dots, a_{n-1}\}$, $B = \{b_0, b_1, \dots, b_{n-1}\}$, $C = \{c_0, c_1, \dots, c_{n-1}\}$. Los agentes tienen asociadas listas que reflejan sus preferencias sobre los agentes de los otros conjuntos.

Llamaremos lista de preferencias simple de un agente a y notaremos $P(a)$, a una lista estrictamente ordenada de los individuos de un conjunto B , entonces $P(a) = b_{i_0}, \dots, b_{i_{n-1}}$ con $b_{ij} \in B, \forall 0 \leq j \leq n-1$. Una lista de preferencias por pares es una lista estrictamente ordenada de pares pertenecientes al producto cartesiano de dos conjuntos $B \times C$, sin repetir elementos, entonces $P(a) = (b_{i_0}, c_{j_0}), \dots, (b_{i_{n-1}}, c_{j_{n-1}})$ con $b_{ik} \in B$ y $c_{jk} \in C, \forall k, 0 \leq k \leq n-1$.

Utilizaremos el término lista de preferencias para referirnos tanto a listas de preferencias simples como a listas de preferencias por pares indistintamente. Notaremos $P(x)$ a la/s lista/s de preferencias del agente x . Denotaremos P_A al conjunto de las listas de preferencias de todos los agentes $a \in A$ y P al conjunto de todas las listas de preferencias de una instancia. Luego, $P_A = \{P(a_0), \dots, P(a_{n-1})\}$ y $P = \{P(a_0), \dots, P(a_{n-1}), P(b_0), \dots, P(b_{n-1}), P(c_0), \dots, P(c_{n-1})\}$.

Una lista de preferencias completa es aquella que contiene a todos los individuos de un conjunto o de un producto cartesiano entre dos conjuntos según sea esta lista simple o por pares.

Consideremos el problema de hallar un matching estable entre agentes pertenecientes a tres conjuntos disjuntos A, B y C de cardinalidad n , donde cada agente tiene listas de preferencias completas sobre agentes de los otros conjuntos. Las listas de preferencias para cada conjunto pueden ser:

- i) cada agente tiene una lista por pares sobre el producto cartesiano de los conjuntos restantes,
- ii) cada agente tiene una lista simple sobre alguno de los conjuntos restantes, o
- iii) cada agente tiene dos listas simples, una sobre cada uno de los conjuntos restantes.

Dado que las listas de preferencias de un agente se expresan sobre los otros conjuntos, podemos ver que toda terna perteneciente a un matching es de la forma (a, b, c) , con $a \in A, b \in B$ y $c \in C$.

Al igual que para matrimonios estables, decimos que el agente a prefiere a x_i antes que x_j y notamos $x_i >_a x_j$ si el agente x_i precede a x_j en la lista de preferencias de a . Del mismo modo, decimos que a prefiere al par (b_{j_1}, c_{k_1}) antes que al par (b_{j_2}, c_{k_2}) y lo notamos $(b_{j_1}, c_{k_1}) >_a (b_{j_2}, c_{k_2})$, si (b_{j_1}, c_{k_1}) precede a (b_{j_2}, c_{k_2}) en la lista de preferencias de a , con $0 \leq j_1, j_2, k_1, k_2 \leq n-1$.

También se define la relación \geq_a , entre agentes de la siguiente manera: $x_i \geq_a x_j$ si el agente x_i precede o es igual a x_j . Del mismo modo, $(b_{j_1}, c_{k_1}) \geq_a (b_{j_2}, c_{k_2})$, si (b_{j_1}, c_{k_1}) precede o es igual a (b_{j_2}, c_{k_2}) .

Una instancia I del problema de hallar un matching estable tridimensional sobre tres conjuntos queda definida por los conjuntos de agentes (A, B y C) y el conjunto de las listas de preferencias (P). Notaremos a la instancia $I = (A, B, C; P)$.

Un matching tridimensional M para una instancia I es un subconjunto del producto cartesiano $A \times B \times C$ tal que cada agente $x \in A \cup B \cup C$ aparece a lo sumo una vez. Un matching M es completo si cada agente $x \in A \cup B \cup C$ aparece exactamente una vez en M . Notemos que un matching completo tiene cardinalidad n . Notaremos $M = \{(a_{i_1}, b_{j_1}, c_{k_1}), (a_{i_2}, b_{j_2}, c_{k_2}), \dots, (a_{i_n}, b_{j_n}, c_{k_n})\}$ al matching completo formado por las ternas $(a_{i_1}, b_{j_1}, c_{k_1}), (a_{i_2}, b_{j_2}, c_{k_2}), \dots, (a_{i_n}, b_{j_n}, c_{k_n})$, con $0 \leq i_l, j_l, k_l \leq n-1, \forall 1 \leq l \leq n$.

Para un matching M sobre una instancia dada, definimos la proyección sobre A de M y notamos M_A al conjunto de los $a \in A$ tal que $(a, b, c) \in M$, para algún $b \in B$ y algún $c \in C$. Definimos la proyección sobre AB de M y notamos M_{AB} al conjunto de los pares $(a, b) \in A \times B$ tal que $(a, b, c) \in M$ para algún $c \in C$. Denotamos

$M_b(a)$ al agente $b \in B$ tal que $(a,b,c) \in M$ para algún $c \in C$. Del mismo modo $M_{BC}(a)$ es el par $(b,c) \in B \times C$ tal que $(a,b,c) \in M$ (análogamente para $M_{AC}(b)$ y $M_{AB}(c)$).

Decimos que una terna (a,b,c) bloquea a un matching M o es una terna bloqueante para M en una instancia I , si $(a,b,c) \notin M$ y:

1. $(b, c) \succeq_a M_{BC}(a)$ y
2. $(a, c) \succeq_b M_{AC}(b)$ y
3. $(a, b) \succeq_c M_{AB}(c)$

Recordemos que $(a, M_B(a), M_C(a))$, $(M_A(b), b, M_C(b))$ y $(M_A(c), M_B(c), c)$ son ternas del matching M .

Decimos que M es un matching estable si M es completo y no tiene ternas bloqueantes. Referenciamos al problema de hallar un matching estable tridimensional sobre tres conjuntos como MET3.

3.1.2.1 Algunas definiciones adicionales

Sean los conjuntos A, B y C de tamaño n . Sean $t=(a,b,c)$ una terna $\in A \times B \times C$, definimos la posición de t en la lista de preferencias de a , y notamos $Pos(a, t)$, a:

- la cantidad de elementos que preceden a (b,c) en $P(a)$, si $P(a)$ es una lista de preferencias por pares, o
- la cantidad de elementos que preceden a $z \in \{b,c\}$ en $P(a)$, si $P(a)$ es una lista de preferencias simple sobre los agentes del conjunto al cual z pertenece.

Esta definición se extiende a los otros agentes que componen la terna $(b$ y $c)$.

Sea I una instancia del problema de hallar matching tridimensional sobre tres conjuntos, definimos *frame de matching* para una instancia I , a un conjunto de especificaciones del tipo $M_{AB}(c)=(a,b)$ o $M_A(b)=a$ que presenta correspondencias entre elementos de los conjuntos A, B y C , tal que la misma es suficiente para especificar un conjunto de matchings de I . Un frame establece qué elementos deben pertenecer a una misma terna y qué elementos deben pertenecer a distintas ternas del matching, dejando elementos sin especificar. Decimos que un frame es un *frame de matching estable* y lo notamos $FM(I)$, si todos los matchings que satisfacen las especificaciones del frame son estables. Por simplicidad especificaremos el frame de matchings como una matriz tal como se ve en el siguiente ejemplo:

$$FM(I) = \begin{matrix} a_0 & b_2 & - \\ a_1 & b_0 & c_2 \\ a_2 & - & - \end{matrix}$$

Sea I una instancia, definimos *familia de matchings estables* en función a un frame y notamos $fam(FM(I))$, al conjunto de todos los matchings estables que obedecen al frame $FM(I)$.

Decimos que un matching M es *canónico* en una instancia I si M cumple alguna de las siguientes condiciones:

- i) $\forall t=(a,b,c) \in M, Pos(a, t) = 0$, ó
- ii) $\forall t=(a,b,c) \in M, Pos(b, t) = 0$, ó
- iii) $\forall t=(a,b,c) \in M, Pos(c, t) = 0$.

Dada una instancia I , decimos que una terna $t=(a, b, c)$ es una *terna fija* si $\forall M$ matching estable de $I, t \in M$. Esta definición es una extensión de la definición de par fijo presentada en [IG/89].

Decimos que una terna $t=(a, b, c)$ es una *terna fuerte* en una instancia I si $Pos(a,t)=0, Pos(b,t)=0$ y $Pos(c,t)=0$

Una lista de preferencias es consistente si:

- i) $(a, b) >_c (a, b') \Rightarrow (a', b) >_c (a', b')$ $\forall a, a' \in A, \forall b, b' \in B, \forall c \in C$ y
- ii) $(a, b) >_c (a', b) \Rightarrow (a, b') >_c (a', b')$ $\forall a, a' \in A, \forall b, b' \in B, \forall c \in C$

La definición de *listas consistentes* sobre instancias con listas de preferencias por pares fue planteada en [NH/95], y la complejidad del problema de decidir si instancias con estas listas tienen matching estable queda como un problema abierto. En este trabajo veremos que esta propiedad no es exclusiva de listas de preferencias por pares y la ampliamos a todos los tipos de listas de preferencias. Cuando corresponda analizaremos el subconjunto de instancias que la cumplen.

Sean M y M' dos matrimonios estables de tamaño n tal que $M \subset A \times B$ y $M' \subset B \times C$, definimos la *junta* por el conjunto B de dos matchings como una función J_B que relaciona ambos matchings a través de los $b \in B$, obteniendo de esta forma un matching $M'' \subset A \times B \times C$. Formalmente, $M J_B M' = M''$ si $\forall (a_i, b_j, c_k) \in M'', (a_i, b_j) \in M$ y $(b_j, c_k) \in M'$.

Ejemplo 3.1

Sean M y M' dos matchings bidimensionales, tenemos que $M J_B M' = M''$ tal como se ve en el ejemplo:
 Sean $M = \{(a_0, b_1), (a_1, b_2), (a_2, b_0)\}$ y $M' = \{(b_0, c_0), (b_1, c_2), (b_2, c_1)\}$.
 Entonces, $M'' = \{(a_0, b_1, c_2), (a_1, b_2, c_1), (a_2, b_0, c_0)\}$. En el ejemplo podemos observar que la junta es la operación inversa de la proyección pues $M''_{AB} = M$ y $M''_{BC} = M'$.

Anteriormente, vimos que un *frame de matchings estables* da un conjunto de condiciones que determina una familia de matchings estables. Veamos un ejemplo.

Ejemplo 3.2

Sea $I = (A, B, C; P)$ una instancia donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ con las siguientes listas de preferencias:

P:

a_0 : $(b_1, c_2) (b_0, c_0) (b_1, c_0) (b_0, c_1) (b_2, c_2) (b_0, c_2) (b_2, c_0) (b_1, c_1) (b_2, c_1)$

a_1 : $(b_2, c_2) (b_0, c_0) (b_1, c_0) (b_0, c_1) (b_1, c_2) (b_0, c_2) (b_2, c_0) (b_1, c_1) (b_2, c_1)$

a_2 : $(b_2, c_1) (b_0, c_0) (b_1, c_0) (b_0, c_1) (b_1, c_2) (b_0, c_2) (b_2, c_0) (b_1, c_1) (b_2, c_2)$

b_0 : $(a_2, c_2) (a_0, c_0) (a_1, c_0) (a_0, c_1) (a_1, c_2) (a_0, c_2) (a_2, c_0) (a_1, c_1) (a_2, c_1)$

b_1 : $(a_0, c_2) (a_0, c_0) (a_1, c_0) (a_0, c_1) (a_1, c_2) (a_2, c_2) (a_2, c_0) (a_1, c_1) (a_2, c_1)$

b_2 : $(a_2, c_2) (a_0, c_0) (a_1, c_0) (a_0, c_1) (a_1, c_2) (a_0, c_2) (a_2, c_0) (a_1, c_1) (a_2, c_1)$

c_0 : $(a_2, b_2) (a_0, b_0) (a_1, b_0) (a_0, b_1), (a_1, b_2) (a_0, b_2) (a_2, b_0) (a_1, b_1) (a_2, b_1)$

c_1 : $(a_2, b_2) (a_0, b_0) (a_1, b_0) (a_0, b_1), (a_1, b_2) (a_0, b_2) (a_2, b_0) (a_1, b_1) (a_2, b_1)$

c_2 : $(a_0, b_1) (a_0, b_0) (a_1, b_0) (a_2, b_2), (a_1, b_2) (a_0, b_2) (a_2, b_0) (a_1, b_1) (a_2, b_1)$

Sea $FM(I)$ el siguiente frame de matchings estables:

$FM(I) =$

$a_0 \ b_1 \ c_2$

$a_1 \ b_0 \ -$

$a_2 \ b_2 \ -$

Sean M' y M'' los siguientes matchings

M' :

$a_0 b_1 c_2$

$a_1 b_0 c_1$

$a_2 b_2 c_0$

M'' :

$a_0 b_1 c_2$

$a_1 b_0 c_0$

$a_2 b_2 c_1$

Como $FM(I)$ es frame de matching estable y M' y M'' son matchings pertenecientes a $\text{fam}(FM(I))$ determinada por el frame $FM(I) \Rightarrow M'$ y M'' son matchings estables de I .

En este ejemplo particular, M' y M'' son los únicos matchings estables que pueden hallarse en la instancia I , luego, la terna (a_0, b_1, c_2) que pertenece a ambos matchings es una terna fija pues pertenece a todos los matchings estables de I .

Una vez especificado el problema daremos algunas propiedades que cumplen las instancias del mismo.

3.2 Propiedades

Observando las listas de preferencias de una instancia es posible determinar la existencia de ternas fijas y matchings canónicos, ambas características son de suma utilidad en instancias donde no sea posible determinar polinomialmente la existencia de solución. En particular, determinar que ciertas ternas son fijas permite reducir la instancia original recortando las listas para excluir a dichas ternas del algoritmo de búsqueda y, según el tamaño de la instancia y de la cantidad de ternas fijas halladas, incrementar las posibilidades de computabilidad del problema. Por otro lado, si bien los matchings canónicos no son frecuentes y la posibilidad de ocurrir decrece en función del tamaño de la instancia, hallar un matching canónico permite, en la mayoría de las clases de instancias, hallar también un matching estable.

Las siguientes propiedades nos ayudan a encontrar tanto ternas fijas como matchings canónicos en ciertas instancias del problema.

Propiedad 3.1

Sea $t=(a, b, c)$ la única terna fuerte que incluye a a, b ó c , en una instancia I dada, entonces t es una terna fija, es decir $\forall M$ matching estable de $I, t \in M$.

Demostración

Sea $t=(a, b, c)$ una terna fuerte.

Supongamos existe M tal que M es un matching estable y no contiene a t .

Sea $t' = (a, M_B(a), M_C(a))$. Sabemos que $t' \in M$ y $t \notin M$ luego $t' \neq t$.

Por definición de posición, $\text{Pos}(a, t') \geq 0$.

Por hipótesis, la única terna fuerte que incluye a a es t , entonces $\text{Pos}(a, t') \neq 0$. Luego $\text{Pos}(a, t') > 0$.

De la misma forma vemos que:

$\text{Pos}(b, (M_A(b), b, M_C(b))) > 0$

$\text{Pos}(c, (M_A(c), M_B(c), c)) > 0$

Pero t es una terna fuerte, es decir que $\text{Pos}(a, t)=0, \text{Pos}(b, t)=0$ y $\text{Pos}(c, t)=0$, luego

$\text{Pos}(a, t) < \text{Pos}(a, (a, M_B(a), M_C(a)))$ y

$\text{Pos}(b, t) < \text{Pos}(b, (M_A(b), b, M_C(b)))$ y

$\text{Pos}(c, t) < \text{Pos}(c, (M_A(c), M_B(c), c))$

Luego t bloquea M , entonces M no es estable. Absurdo.
Entonces si M es estable, $t \in M$. ♦

Propiedad 3.2

Sea I una instancia y sea M un matching de I , si todas las ternas del matching M son fuertes entonces el matching M es canónico para la instancia I .

Demostración

$\forall t=(a,b,c) \in M$ es una terna fuerte
 \Leftrightarrow
 $\forall t=(a,b,c) \in M$ $\text{Pos}(a, t)=0$, $\text{Pos}(b, t)=0$ y $\text{Pos}(c, t)=0$
 \Leftrightarrow (todas las ternas son ternas fuertes)
 $\forall a_i \in A$, $\text{Pos}(a_i, (a_i, M_B(a_i), M_C(a_i))) = 0$ y
 $\forall b_j \in B$, $\text{Pos}(b_j, (M_A(b_j), b_j, M_C(b_j))) = 0$ y
 $\forall c_k \in C$, $\text{Pos}(c_k, (M_A(c_k), M_C(c_k), c_k)) = 0$
 \Rightarrow
 M es canónico. ♦

Como mencionamos anteriormente, los agentes tienen listas de preferencias que pueden obedecer a distintos formatos. En función de los formatos de estas listas es posible dar una clasificación del problema de hallar un matching estable tridimensional sobre tres conjuntos en cuatro categorías básicas. Veamos a continuación dicha clasificación.

3.3 Clasificación

De acuerdo a la estructura de las listas de preferencias de los agentes en cada conjunto de la instancia analizada, proponemos una clasificación del problema tal como describimos a continuación. Estas listas pueden involucrar a uno solo de los otros dos conjuntos o pueden considerar a ambos. A su vez, estas últimas pueden considerar independientemente la preferencia sobre los agentes de cada conjunto o combinarla. Proponemos entonces la siguiente clasificación.

1. Con Listas Únicas.
2. Con Listas Simples.
3. Con Listas por Pares.
4. Con Listas Combinadas.

Veamos los formatos de las instancias de cada clase por separado.

3.3.1 Con Listas de Preferencias Únicas (LU)

En las instancias de esta clase, cada individuo tiene una lista de preferencias simple, ordenando en ella a los individuos de uno de los otros dos conjuntos. Notaremos con flechas (\rightarrow , \leftarrow o \leftrightarrow) las preferencias entre los conjuntos. Es decir que si los individuos del conjunto A definen sus preferencias sobre los individuos del conjunto B , lo notaremos como $A \rightarrow B$. Si los agentes de A eligen sobre B , los de B sobre C y los de C sobre A lo notaremos como $A \rightarrow B \rightarrow C \rightarrow A$ mientras que si los individuos del conjunto A

definen sus preferencias sobre los individuos del conjunto B, los de B sobre A y los de C sobre B lo notaremos como $A \leftrightarrow B \leftarrow C$.

Ejemplo 3.3

Dada la instancia $I=(A,B,C;P)$, determinada por $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:

$a_1:$ $b_1 b_2 b_3$
 $a_2:$ $b_2 b_1 b_3$
 $a_3:$ $b_3 b_1 b_2$

$b_1:$ $c_3 c_2 c_1$
 $b_2:$ $c_1 c_3 c_2$
 $b_3:$ $c_2 c_3 c_1$

$c_1:$ $a_2 a_3 a_1$
 $c_2:$ $a_1 a_3 a_2$
 $c_3:$ $a_2 a_1 a_3$

Vemos que las preferencias son $A \rightarrow B \rightarrow C \rightarrow A$ y podemos encontrar el siguiente matching estable: $M=\{(a_1,b_1,c_3),(a_2,b_2,c_1),(a_3,b_3,c_2)\}$. Mientras que $M'=\{(a_1,b_1,c_1),(a_2,b_3,c_2),(a_3,b_2,c_3)\}$ no es estable ya que es bloqueado por la terna (a_2,b_1,c_3) .

3.3.2 Con Listas de Preferencias Simples (LS)

En las instancias de esta clase, cada individuo tiene dos listas de preferencias simples, ordenando en cada una de ellas a los individuos de los otros dos conjuntos independientemente. Notemos que en este caso las preferencias que los agentes tengan sobre uno de los otros dos conjuntos no afectan sus preferencias sobre el tercer conjunto.

Ejemplo 3.4

Dada la instancia $I=(A,B,C;P)$, determinada por $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:

$a_1:$ $b_1 b_2$	$b_1:$ $a_1 a_2$	$c_1:$ $a_1 a_2$
$c_1 c_2$	$c_2 c_1$	$b_2 b_1$
$a_2:$ $b_2 b_1$	$b_2:$ $a_1 a_2$	$c_2:$ $a_1 a_2$
$c_1 c_2$	$c_2 c_1$	$b_1 b_2$

Podemos encontrar el siguiente matching estable: $M=\{(a_1,b_1,c_1), (a_2,b_2,c_2)\}$. Mientras que $M'=\{(a_1,b_2,c_2),(a_2,b_1,c_1)\}$ no es estable ya que es bloqueado por la terna (a_1,b_1,c_1) .

3.3.3 Con Listas de Preferencias por Pares (LP)

En las instancias de esta clase cada individuo tiene una lista de preferencias, ordenando en ella al producto cartesiano de los otros dos conjuntos. Es decir que las preferencias sobre un conjunto no son independientes de las preferencias sobre el otro conjunto. Se trata del problema tratado en [NH/95] y en [Alk/87].

Ejemplo 3.5

Dada la instancia $I=(A,B,C;P)$, determinada por $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:

a_1 : $(b_1, c_1) (b_1, c_2) (b_2, c_1) (b_2, c_2)$

a_2 : $(b_1, c_1) (b_2, c_1) (b_1, c_2) (b_2, c_2)$

b_1 : $(a_1, c_2) (a_2, c_1) (a_1, c_1) (a_2, c_2)$

b_2 : $(a_1, c_1) (a_1, c_2) (a_2, c_1) (a_2, c_2)$

c_1 : $(a_2, b_2) (a_1, b_1) (a_2, b_1) (a_1, b_2)$

c_2 : $(a_1, b_1) (a_2, b_1) (a_1, b_2) (a_2, b_2)$

Podemos encontrar el siguiente matching estable: $M = \{(a_1, b_1, c_2), (a_2, b_2, c_1)\}$

Mientras que $M' = \{(a_1, b_1, c_1), (a_2, b_2, c_2)\}$ no es estable ya que es bloqueado por la terna (a_2, b_2, c_1) .

3.3.4 Con Listas de Preferencias Combinadas (LC)

Los individuos en cada conjunto tienen listas de preferencias únicas, simples o por pares sobre los individuos de los otros conjuntos. Si bien tratamos las instancias de listas combinadas dentro de una misma clase, podemos ver que se trata de una clasificación más compleja, donde los individuos en cada conjunto tienen distintos tipos de listas de preferencias sobre los individuos en los otros conjuntos. En el capítulo que trata este clase de instancias daremos y analizaremos dicha clasificación.

3.4 El conjunto de todos los Matchings Estables

El problema de enumerar todos los matrimonios estables bidimensionales ha sido estudiado a partir de encontrar algunas propiedades sobre la estructura que forma el conjunto de los mismos.

Como hemos visto en el Capítulo 2, se define una relación de orden entre todos los matrimonios estables, los cuales forman un reticulado distributivo bajo esta relación.

Recordemos qué es un reticulado distributivo.

Consideremos un conjunto L con un orden parcial \cdot .

Una *cota superior* de un subconjunto X de L , es un elemento a en L tal que $a \cdot x$ para todo x en X .

Llamamos *supremo* de X (y denotamos $\sup(X)$) a la menor de las cotas superiores de X , si ésta existe. Por la propiedad de antisimetría que cumple el orden definido en L , si existe el supremo entonces es único.

Análogamente, una *cota inferior* de un subconjunto X de L , es un elemento a en L tal que $x \cdot a$ para todo x en X . Llamamos *ínfimo* de X (y denotamos $\inf(X)$) a la mayor de las cotas inferiores de X , si ésta existe.

Por la propiedad de antisimetría que cumple el orden definido en L , si existe el ínfimo entonces es único.

Un *reticulado* es un conjunto L parcialmente ordenado donde cualquier par de elementos (x, y) tienen un supremo $(x \vee y)$ y un ínfimo $(x \wedge y)$.

Un reticulado es *completo* cuando cada subconjunto X tiene un supremo y un ínfimo en L .

Un reticulado es *distributivo* si:

1. $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ y
2. $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ $\forall x, y, z \text{ en } L.$

La importancia de que el conjunto de todos los matrimonios estables sea un reticulado radica fundamentalmente, en que es la base para el desarrollo de un algoritmo que computa todos los matchings estables en orden $O(n^2 + n|S(I)|)$ donde $S(I)$ es el conjunto de todos los matchings estables de la instancia I y n es el tamaño de I [Gus/85]. La característica de ser un reticulado permitió además derivar algunas propiedades dentro del conjunto de todos los matchings estables, como por ejemplo la existencia de un matching óptimo-femenino que se corresponde con el pésimo-masculino.

La relación de orden parcial permite obtener un ordenamiento de los matchings estables desde el punto de vista de alguno de los conjuntos.

En los capítulos siguientes, analizaremos para cada tipo de instancias del problema de hallar un matching estable tridimensional sobre tres conjuntos, la existencia de un orden dentro del conjunto de soluciones y si forma un reticulado bajo esta relación. Además de las características propias de ser un orden parcial, este resultado podría servir, entre otras cosas, como punto de partida al estudio necesario para el desarrollo de un algoritmo óptimo que compute todos los matchings estables, considerando los resultados obtenidos para matchings bidimensionales.

3.4.1 El conjunto de todos los Matchings Estables

Tridimensionales sobre tres Conjuntos

Dada una instancia del problema de matchings estables tridimensionales sobre tres conjuntos, $I = (A, B, C; P)$, definimos una relación de *dominación* entre todos los matchings estables de acuerdo a lo siguiente: el matching M_1 *domina* al matching M_2 con respecto a un conjunto s y sólo si todos los individuos de ese conjunto prefieren o igualan sus parejas en M_1 a las de M_2 .

Definimos entonces una relación \geq_A , donde:

$$M \geq_A M' \text{ si } M_{BC}(a) \geq_a M'_{BC}(a) \forall a \in A$$

Decimos que M y M' son *no comparables* para un conjunto A si algunos agentes $a \in A$ prefieren M' a M y otros prefieren M a M' .

Es decir, M y M' son *no comparables* si $\exists a, a' \in A$ tales que $M \geq_a M'$ y $M' \geq_{a'} M$.

En los próximos capítulos veremos que esta relación no siempre es un orden y, en la mayoría de los casos no forma un reticulado. Este estudio nos permitirá saber en qué casos podemos heredar las propiedades encontradas para el caso de dos dimensiones en el problema de encontrar todos los matchings estables.

3.4.1.1 Representación del conjunto de todos los matchings estables

La relación de dominación \geq_A puede representarse mediante un grafo dirigido en donde:

- Cada vértice del grafo dirigido representa uno o varios matchings estables $M_{i1}.. M_{ik}$
- Dos matchings M, M' están representados por el mismo vértice v si $M_{BC}(a) = M'_{BC}(a)$.

Es decir, si para todo individuo $a \in A$, $\text{Pos}(a, (a, M_{BC}(a))) = \text{Pos}(a, (a, M'_{BC}(a)))$.

- Existe un arco desde un vértice v_i hacia otro vértice v_j ($v_i \rightarrow v_j$) si $M_i \geq_A M_j \forall M_i$ en v_i y $\forall M_j$ en v_j (es decir, M_i domina a M_j)

Decimos que M y M' están en la misma *clase de equivalencia* si ambos están representados por el mismo vértice. Podemos ver que la relación definida entre dos matchings pertenecientes a una misma clase de equivalencia es, justamente, una relación de equivalencia, es decir cumple con las propiedades reflexiva, simétrica y transitiva.

3.4.1.2 Formulación del problema MET3 como un problema de Programación Lineal Entera

En esta sección, presentaremos el problema de encontrar los matchings estables tridimensionales sobre tres conjuntos como un problema de programación lineal entera.

Sea la configuración de un matching M una matriz X de ceros y unos tal que:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching, es decir, si } M_{BC}(a)=(b,c) \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases}$$

con $(a,b,c) \in A \times B \times C$

De acuerdo a esto, el conjunto de todos los matchings (sin tener en cuenta aún estabilidad de los mismos) puede ser representado por las siguientes restricciones lineales:

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$x_{a,b,c} \in \{0,1\} \quad (a,b,c) \in A \times B \times C \quad (4)$$

Estas cuatro restricciones garantizan que X es un matching, esto es, sus elementos son 0's y 1's y cada agente en un conjunto es ligado a exactamente un agente en los otros dos conjuntos.

Para definir estabilidad, agregamos una nueva restricción:

$$\sum_{\substack{(i,j) \geq (b,c) \\ a}} x_{a,i,j} + \sum_{\substack{(i,k) \geq (a,c) \\ b}} x_{i,b,k} + \sum_{\substack{(i,j) \geq (a,b) \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (5)$$

Esta restricción aplica a todas las instancias del problema de MET3, siempre que todos los elementos de las listas de preferencias de los agentes para todos los conjuntos sean comparables entre sí. En los capítulos siguientes daremos, para aquellas instancias en las cuales alguno de los conjuntos tenga listas de preferencias simples, una nueva restricción para garantizar la estabilidad, ya que en esta clase de instancias tenemos elementos no comparables. En cambio, para aquellas instancias en las cuales ningún

conjunto tenga listas de preferencias simples, daremos una nueva restricción que será equivalente a la dada recientemente, dentro del contexto de la clase de instancias tratadas.

Para que se cumpla la desigualdad (5), por lo menos uno de los cuatro términos debe ser igual a 1, es decir, para cada terna (a,b,c) debe cumplirse al menos una de las siguientes igualdades:

$$\sum_{\substack{(i,j) \geq (b,c) \\ a}} x_{a,i,j} = 1, \quad (5.1)$$

$$\sum_{\substack{(i,k) \geq (a,c) \\ b}} x_{i,b,k} = 1, \quad (5.2)$$

$$\sum_{\substack{(i,j) \geq (a,b) \\ c}} x_{i,j,c} = 1, \quad (5.3)$$

$$x_{a,b,c} = 1, \quad (5.4)$$

Podemos ver que (5.1) se cumple si para la terna (a,b,c), el agente a prefiere a su pareja actual en el matching antes que al par (b,c).

De la misma manera, (5.2) se cumple si para la terna (a,b,c), el agente b prefiere a su pareja actual en el matching antes que al par (a,c). Y (5.3) es válido si para la terna (a,b,c), el agente c prefiere a su pareja actual en el matching antes que al par (a,b).

Luego, para garantizar estabilidad, deberá pasar que para toda terna (a,b,c), se cumpla alguna de las siguientes condiciones:

- (a,b,c) ∈ M, es decir, x_{a,b,c}=1. En este caso, se cumple 5.4.
- (a,b,c) ∉ M y (a,b,c) no debe bloquear a M. En este caso, no pueden cumplirse conjuntamente que a prefiera al par (b,c) antes que a su pareja actual, b prefiera al par (a,c) antes que a su pareja actual y c prefiera al par (a,b) antes que a su pareja actual. Esta condición queda asegurada si se cumple al menos una igualdad entre (5.1), (5.2) y (5.3).

El problema MET3 fue formulado como un problema de programación lineal entera con las siguientes características:

- El número de variables es n³
Esto puede verse fácilmente, ya que la dimensión de la matriz x es exactamente igual a la cantidad de ternas posibles para una instancia de tamaño n, la cual es igual a n³.
- La cantidad de restricciones es 3n + 2n³.
Veamos que para el caso de la desigualdad (1), existirán n restricciones, una por cada agente del conjunto A. De la misma manera, para (2) y (3) existirán n restricciones en cada caso.
Por otra parte, tanto la desigualdad (4) como la (5), se expresan sobre cada una de las ternas, siendo el total de las mismas igual a n³.
Luego, la cantidad total de restricciones es 3n + 2n³.

Capítulo 4: Matching Estable Tridimensional sobre Tres Conjuntos con Listas Únicas

En este capítulo presentaremos los resultados del análisis del problema de hallar un matching estable tridimensional sobre tres conjuntos con listas de preferencias únicas.

Definimos el problema general de hallar un matching estable con listas únicas (LU) como una generalización del problema de 3GSM circular que se deja abierto en [NH/95], ampliando el universo de instancias y clasificándolo en dos categorías de acuerdo a la preferencia entre los conjuntos: listas únicas mutuas (LUM) y listas únicas circulares (LUC). Para el primer caso, presentamos una condición necesaria y suficiente para determinar si una instancia tiene matching estable y definimos un nuevo tipo de estabilidad, a la que llamamos estabilidad débil, que tiene solución en todos los casos. Brindamos un algoritmo polinomial que encuentra siempre un matching estable débil. Acerca del problema en instancias de LUC, cuya complejidad es un tema abierto, conjeturamos que el mismo es NP-Completo y brindamos tanto la formulación entera que lo resuelve en forma exacta, como así también un algoritmo polinomial no completo que, para algunas instancias, puede encontrar solución en tiempos razonables para valores de n grandes.

Finalmente, analizamos la relación de dominación entre matchings estables de una instancia dada para cada subproblema, como así también la estructura que conforma el conjunto de todos los matchings estables.

4.1 Introducción

Considerando el problema de 3GSM circular que se plantea como tema abierto en [NH/95], en el cual los individuos del conjunto A tienen sus listas de preferencias sólo sobre individuos de B , los de B sólo sobre agentes de C y los de C únicamente sobre los individuos de A , surge la idea de que un agente pueda tener listas de preferencias sobre agentes de uno solo de los conjuntos restantes, como una variante del problema de matching estable tridimensional sobre tres conjuntos. Generalizamos esta idea, ampliando el universo de instancias a considerar para incluir las distintas variantes de acuerdo a la preferencia que se establece entre los conjuntos.

Una instancia del problema de MET3 con *listas de preferencias únicas*, estará formada por tres conjuntos disjuntos, A , B y C , de cardinalidad n , donde los agentes de cada uno de los conjuntos tiene una lista de preferencias formada por agentes de uno de los otros conjuntos. Dichas listas están estrictamente ordenadas.

Se dice que un agente a *prefiere* al par (b,c) antes que al (b',c') , o dicho de otro modo, que $(b,c) \succ_a (b',c')$ si $b \succ_a b'$, es decir, si $\text{Pos}(a,(a,b,c'')) < \text{Pos}(a,(a,b',c'')) \forall c'' \in C$. De igual manera se define para $b \in B$ y $c \in C$ una relación \succ_b y \succ_c sobre el producto cartesiano $A \times C$ y $A \times B$ respectivamente.

Como mencionamos anteriormente, este universo de instancias del problema de MET3 con listas de preferencias únicas puede ser particionado en función de la estructura que tengan las preferencias entre los individuos de los distintos conjuntos. De acuerdo a esto, distinguimos dos esquemas:

- *Esquema de preferencias mutuas (LUM)*: en este esquema se produce una elección mutua entre dos conjuntos, digamos A y B . Es decir, los individuos en A expresan sus preferencias sobre individuos en B , los de B expresan sus preferencias sobre individuos en A y los de C expresan sus preferencias sobre individuos en A o en B .

- *Esquema circular de preferencias (LUC)*: en este esquema (equivalente al 3GSM circular presentado en [NH/95]) se forma un ciclo entre los tres conjuntos, es decir: individuos en A expresan sus preferencias sobre individuos en B, los de B expresan sus preferencias sobre individuos en C y los de C expresan sus preferencias sobre individuos en A.

4.2 Con Listas de Preferencias Mutuas (LUM)

4.2.1 Matching y Estabilidad

En instancias de este problema, los individuos de dos de los conjuntos dan sus listas de preferencias mutuamente y el conjunto restante elige sobre agentes de alguno de estos. Sin pérdida de generalidad, diremos que los individuos del conjunto A definen sus preferencias sobre los individuos del conjunto B, los de B sobre A y los de C sobre A.

Decimos que una terna (a,b,c) es una *terna bloqueante* para un matching M si $(a,b,c) \notin M$ y:

1. $b \succeq_a M_B(a)$ y
2. $a \succeq_b M_A(b)$ y
3. $a \succeq_c M_A(c)$

A continuación se muestra un ejemplo de una instancia con listas de preferencias mutuas, para la cual se da un matching estable y otro no estable, en el cual se puede ver la existencia de por lo menos una terna bloqueante.

Ejemplo 4.1

Dada la instancia $I=(A,B,C;P)$ con $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:					
a_0	$b_0 b_2 b_1$	b_0	$a_0 a_1 a_2$	c_0	$a_1 a_2 a_0$
a_1	$b_1 b_2 b_0$	b_1	$a_0 a_2 a_1$	c_1	$a_2 a_1 a_0$
a_2	$b_2 b_0 b_1$	b_2	$a_0 a_2 a_1$	c_2	$a_0 a_1 a_2$

El matching $M_1=\{(a_0,b_0,c_2),(a_1,b_1,c_0),(a_2,b_2,c_1)\}$ es estable para la instancia I.

En cambio, el matching $M_2=\{(a_0,b_0,c_2),(a_1,b_2,c_1),(a_2,b_1,c_0)\}$ no es estable, ya que la terna (a_1,b_2,c_0) es bloqueante.

Si observamos las listas de preferencias en una instancia cualquiera del problema de LUM, vemos que las mismas mantienen el orden de los agentes de un conjunto independientemente del tercero. Esto es bastante trivial, ya que cada conjunto está interesado en agentes de un solo conjunto, sin considerar al restante. Esto nos lleva a recordar la propiedad de consistencia dada en el Capítulo 3, observando que la misma aplica a nuestro problema, lo cual queda demostrado con la propiedad que sigue:

Propiedad 4.1

Las listas de preferencias mutuas cumplen la propiedad de consistencia.

Demostración

Para probar que una lista de preferencias es consistente, debemos probar que:

- 1) $(a,b) >_c (a,b') \Rightarrow (a',b) >_c (a',b') \quad \forall a,a' \in A, \forall b,b' \in B, \forall c \in C$ y
 2) $(a,b) >_c (a',b) \Rightarrow (a,b') >_c (a',b') \quad \forall a,a' \in A, \forall b,b' \in B, \forall c \in C$

Para listas de preferencias mutuas:

- Si $C \rightarrow A$
 - 1) $(a,b) >_c (a,b')$ no se cumple
 - 2) $(a,b) >_c (a',b) \Rightarrow a >_c a' \Rightarrow (a,b') >_c (a',b')$
- Si $C \rightarrow B$
 - 1) $(a,b) >_c (a,b') \Rightarrow b >_c b' \Rightarrow (a',b) >_c (a',b')$
 - 2) $(a,b) >_c (a',b)$ no se cumple. ♦

En el Ejemplo 4.1 mostrábamos la existencia de un matching estable y uno no estable para una instancia de nuestro problema de LUM. Sin embargo, no es posible encontrar esta característica en todas las instancias. Veamos con otro ejemplo, que existen instancias del problema de LUM que no tienen matching estable.

Ejemplo 4.2

La instancia $I=(A,B,C;P)$ con $A=\{a_0,a_1\}$, $B=\{b_0,b_1\}$, $C=\{c_0,c_1\}$ y las siguientes listas de preferencias:

- P:
- | | | |
|---|---|---|
| a ₀ : b ₀ b ₁ | b ₀ : a ₀ a ₁ | c ₀ : a ₁ a ₀ |
| a ₁ : b ₁ b ₀ | b ₁ : a ₁ a ₀ | c ₁ : a ₁ a ₀ |

no tiene matching estable.

La siguiente tabla lista todos los matchings y muestra que para cada uno existe al menos una terna bloqueante, con lo cual no existe matching estable para la instancia I.

Matching	Terna bloqueante
M ₁ : { (a ₀ ,b ₀ ,c ₀), (a ₁ ,b ₁ ,c ₁) }	(a ₁ ,b ₁ ,c ₀)
M ₂ : { (a ₀ ,b ₀ ,c ₁), (a ₁ ,b ₁ ,c ₀) }	(a ₁ ,b ₁ ,c ₁)
M ₃ : { (a ₀ ,b ₁ ,c ₀), (a ₁ ,b ₀ ,c ₁) }	(a ₁ ,b ₀ ,c ₀)
M ₄ : { (a ₀ ,b ₁ ,c ₁), (a ₁ ,b ₀ ,c ₀) }	(a ₁ ,b ₀ ,c ₁)

4.2.2 Búsqueda de un Matching Estable

Como ya mencionamos, existen instancias de LUM que no tienen solución al problema de encontrar un matching estable. Veremos a continuación que el número de casos sin solución es bastante grande y aumenta exponencialmente con la cardinalidad de los conjuntos.

Sin embargo, es posible determinar en forma polinomial si una instancia tiene o no solución. El teorema siguiente da una condición necesaria y suficiente para que una instancia de este problema tenga matching estable.

Teorema 4.1

Sea I una instancia del problema de hallar un matching estable con listas únicas mutuas, con la siguiente relación de preferencia: $A \rightarrow B$, $B \rightarrow A$, $C \rightarrow A$. Entonces, I tiene matching estable \iff el primer elemento de las listas de los C son distintos.

Demostración

\Rightarrow

Supongamos que M es un matching estable de I y existen al menos dos $c \in C$ tal que el primer elemento de sus respectivas listas es el mismo a . Por ser M un matching, a se encuentra ligado con un solo elemento de C , sea c_k alguno de los $c \in C$ tal que el primer elemento de su lista de preferencias es a y además no pertenece a la terna de a en el matching, esto es, $(a, b, c_k) \notin M$ para ningún $b \in B$.

Pero entonces $(a, M_B(a), c_k)$ es terna bloqueante de M pues

$$a = M_A(M_B(a)) \text{ y } M_B(a) = M_B(a) \text{ y } a \succeq_{c_k} M_A(c)$$

Luego, M no es estable. Absurdo.

\Leftarrow

Baste para eso ver que el siguiente Algoritmo 4.1 es correcto y completo, es decir que siempre encuentra un matching estable para instancias con esta característica. \blacklozenge

Presentamos a continuación un algoritmo que siempre encuentra un matching estable para instancias de LUM que cumplan la condición del Teorema 4.1, es decir, tal que el primer elemento de las listas de los C sean distintos.

Algoritmo 4.1

El algoritmo puede comenzar desde cualquiera de los conjuntos A o B . Comenzaremos por A .

- 1) Se encuentra un matching estable entre los individuos del conjunto A y los de B por el algoritmo de Gale y Shapley.

Todos los c_k ($1 \leq k \leq n$) están sin asignar a los pares (a_i, b_j) formados en el punto anterior.

- 2) Para cada c_k ($1 \leq k \leq n$)

2.1) c_k es asignado a un par (a_i, b_j) tal que a_i es el primer elemento de la lista de c_k .

El matching así obtenido es estable

El siguiente teorema demuestra correctitud y completitud del algoritmo que hemos detallado.

Teorema 4.2

El Algoritmo 4.1 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.

Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un b_j asignado, para todo $1 \leq i \leq n$ (por correctitud del algoritmo de Gale y Shapley).

En el paso 2, cada c_k es asignado una y sólo una vez a pares distintos de (a_i, b_j) .

b) Probar que el matching encontrado es estable

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a, b, c) no perteneciente al matching M , tal que:

$$b \succ_a M_B(a) \text{ y}$$

$$a \succ_b M_A(b) \text{ y}$$

$$a \succ_c M_A(c)$$

Supongamos que existe tal terna.

De la definición de terna bloqueante surgen 2 alternativas:

- 1) $a \neq M_A(b)$ es decir que $b \succ_a M_B(a)$ y $a \succ_b M_A(b)$
Esto significa que existe un par (a, b) que bloquea al matching de pares obtenido entre los conjuntos A y B en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- 2) $a = M_A(b)$ es decir que $b = M_B(a)$, luego para que la terna sea bloqueante debe darse que $a \succ_c M_A(c)$ (descartamos $a = M_A(c)$ pues (a, b, c) sería una terna del matching), pero es absurdo ya que $M_A(c)$ es el primer elemento en la lista de c , luego no existe a que esté mejor ubicado.

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina y la asignación de los c es siempre posible. ♦

De acuerdo al Teorema 4.1, vemos que para que instancias de LUM tengan solución se requiere una fuerte restricción sobre el formato de las listas de C . Para dar una idea de la magnitud de esta restricción basta observar la cantidad de instancias que la cumplen con relación al total.

Siendo n la cardinalidad de los conjuntos, la cantidad total de instancias del problema será $(n!)^{3n}$. Si restringimos las listas de uno de los conjuntos de acuerdo a la condición del Teorema 4.1, la cantidad de instancias se reduce a $(n!)^{2n} \cdot ((n-1)!)^n n!$

Luego, el porcentaje de instancias de LUM que tienen matching estable es:

$$\frac{(n!)^{2n} \cdot ((n-1)!)^n n!}{(n!)^{3n}} = \frac{((n-1)!)^n n!}{(n!)^n} = \frac{((n-1)!)^n}{(n!)^{n-1}} = \frac{((n-1)!)^n}{(n(n-1)!)^{n-1}} = \frac{((n-1)!)^n}{n^{n-1} (n-1)!^{n-1}} = \frac{(n-1)!}{n^{n-1}}$$

Vemos que esta relación disminuye a medida que crece n .

A raíz de esto, redefinimos el problema en LUM para contemplar como matchings estables aquellos en los cuales no existe terna tal que los tres integrantes de la misma hubiesen preferido estar juntos antes que con su pareja actual. Notemos que esta nueva definición de terna bloqueante difiere de la dada previamente en que todos los agentes de la misma tienen que mejorar su condición con respecto a la que tienen actualmente en el matching, no siendo válido mantener la misma pareja. En otras palabras, podríamos decir que en este caso, todos los agentes de la terna bloqueante mejoran su condición mientras que antes, no la empeoraban.

Definimos entonces una nueva estabilidad, más débil que la anterior, que permitirá encontrar solución en todos los casos.

Decimos ahora, que una terna (a,b,c) es una *terna bloqueante* para un matching M si $(a,b,c) \notin M$ y:

1. $b \succ_a M_B(a)$ y
2. $a \succ_b M_A(b)$ y
3. $a \succ_c M_A(c)$

Llamaremos entonces *estabilidad fuerte* al tipo de estabilidad más restrictivo (según la definición original de terna bloqueante) y *estabilidad débil*, o simplemente *estabilidad*, a la que cumple con la no existencia de ternas bloqueante de acuerdo a esta última definición.

De la misma manera, diremos que un matching es *fuertemente estable* si cumple con la estabilidad fuerte y *débilmente estable* o simplemente, *estable*, si cumple con la estabilidad débil.

Excepto que se especifique de otra manera, a partir de este momento y a lo largo de todo el análisis de LUM, trabajaremos con la estabilidad débil.

Para clarificar la diferencia entre ambas estabilidades, veamos un ejemplo de una instancia de LUM que no tiene matching fuertemente estable, pero sí tiene matching débilmente estable.

Ejemplo 4.3

En el Ejemplo 4.2 vimos que la instancia $I=(A,B,C;P)$ donde $A=\{a_0,a_1\}$, $B=\{b_0,b_1\}$, $C=\{c_0,c_1\}$ y donde las listas de preferencias son:

P:			
a_0 :	$b_0 \ b_1$	b_0 :	$a_0 \ a_1$
a_1 :	$b_1 \ b_0$	b_1 :	$a_1 \ a_0$
		c_0 :	$a_1 \ a_0$
		c_1 :	$a_1 \ a_0$

no tiene matchings fuertemente estables.

Sin embargo, el matching $M_1=\{(a_0,b_0,c_0),(a_1,b_1,c_1)\}$ es débilmente estable. La terna (a_1,b_1,c_0) , si bien bloquea la estabilidad fuerte, no lo hace con la débil, ya que $a_1 \succ M_A(b_1)$. Podemos ver que, en todos los matchings del Ejemplo 4.3, las ternas bloqueantes estaban formadas por $(a, M_B(a), c)$ para algún $a \in A$ y algún $c \in C$.

Una de las características a destacar sobre este problema y que podrá observarse desde el algoritmo que daremos más adelante, es que las listas de preferencias de los individuos de C no influyen sobre la estabilidad del matching hallado entre los conjuntos A y B .

El problema de hallar un matching estable en instancias de LUM puede verse entonces, como una extensión del problema de matrimonios estables en dos dimensiones, con la incorporación de individuos de un tercer conjunto que no influyen en las elecciones que se efectúan entre los agentes de los conjuntos que se prefieren mutuamente.

Sin embargo, es posible que un matching sea estable en instancias de este problema, pero su proyección sobre los conjuntos que se prefieren mutuamente no lo sea en la correspondiente instancia de dos dimensiones.

Es decir, puede existir un matching M estable para $I=(A,B,C;P)$ con $A \rightarrow B$, $B \rightarrow A$ y $C \rightarrow A$ tal que M_{AB} no es estable en $I=(A,B;P_{AB})$ con $P_{AB}=\{P(a)/a \in A\} \cup \{P(b)/b \in B\}$.

Veamos esto con un ejemplo.

Ejemplo 4.4

Este ejemplo muestra que no todos los matchings estables para instancias de LUM se obtienen a partir de un matching estable M_{AB} .

Sea $I=(A,B,C;P)$ con $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ con las siguientes listas de preferencias:

P:

$a_0: b_1 b_2 b_0$

$a_1: b_0 b_2 b_1$

$a_2: b_1 b_0 b_2$

$b_0: a_0 a_1 a_2$

$b_1: a_0 a_2 a_1$

$b_2: a_0 a_2 a_1$

$c_0: a_1 a_2 a_0$

$c_1: a_2 a_1 a_0$

$c_2: a_0 a_1 a_2$

El matching $M = \{(a_0, b_0, c_2), (a_1, b_2, c_1), (a_2, b_1, c_0)\}$ es estable para I.

Sin embargo, el matching $M_{AB} = \{(a_0, b_0), (a_1, b_2), (a_2, b_1)\}$ resultante de la proyección AB del matching M, no es estable considerando la instancia correspondiente a los conjuntos A y B (el par (a_0, b_1) es bloqueante).

Si bien, como vimos en el Ejemplo 4.4, si M es estable no implica que M_{AB} sea estable, la implicación inversa es válida. Es decir, como dijimos previamente, los agentes del conjunto C no influyen en la estabilidad conseguida entre los conjuntos que se prefieren mutuamente, A y B. La siguiente propiedad muestra esta implicación.

Propiedad 4.2

Sean $I=(A,B;P)$ una instancia del problema de matrimonios estables de cardinalidad n y M un matching entre A y B.

Sea $I'=(A,B,C;P')$ una instancia de LUM, con C conjunto de agentes de cardinalidad n y

$P'=P \cup \{P(c)/c \in C\}$.

M es estable en $I \Rightarrow M' = \{(a,b,c)/(a,b) \in M \text{ y } c \in C\}$ es estable en I' .

Demostración

Supongamos que M es estable en I y que existe $M' = \{(a,b,c)/(\exists a)(\exists b)(a,b) \in M\}$ tal que no es estable en $I' \Rightarrow \exists (a,b,c) \notin M'$ que bloquea a $M' \Rightarrow b \succ_a M'_B(a)$ y $a \succ_b M'_A(b)$ y $a \succ_c M'_A(c) \Rightarrow b \succ_a M_B(a)$ y $a \succ_b M_A(b)$, ya que $P(a)=P'(a)$ y $P(b)=P'(b) \Rightarrow (a,b)$ bloquea a M en I. Absurdo, por hipótesis.

Luego, M' es estable en I' . ♦

A continuación daremos una propiedad que nos indica que en instancias LUM donde sea posible hallar un matching canónico, hallamos también un matching débilmente estable.

Propiedad 4.3

Todo matching canónico es débilmente estable.

Demostración

Debemos ver que si M es canónico, entonces no existe terna (a,b,c) no perteneciente a M tal que:

1. $b \succ_a M_B(a)$ y

2. $a \succ_b M_A(b)$ y

3. $a \succ_c M_A(c)$

Sabemos que, como M es canónico,

i) $\text{Pos}(a, (a, M_B(a), M_C(a))) = 0$, o

ii) $\text{Pos}(b, (M_A(b), b, M_C(b))) = 0$, o

iii) $\text{Pos}(c, (M_A(c), M_B(c), c)) = 0$

Si $\text{Pos}(a, (a, M_B(a), M_C(a))) = 0 \Rightarrow M_B(a)$ es el primero en la lista de preferencias de a \Rightarrow no existe b tal que $b \succ_a M_B(a)$

Si $\text{Pos}(b, (M_A(b), b, M_C(b))) = 0 \Rightarrow M_A(b)$ es el primero en la lista de preferencias de b \Rightarrow no existe a tal que $a \succ_b M_A(b)$

Si $\text{Pos}(c, (M_A(c), M_B(c), c)) = 0 \Rightarrow M_A(c)$ es el primero en la lista de preferencias de $c \Rightarrow$ no existe a tal que $a \succ_c M_A(c)$

Luego, no existe terna bloqueante para M canónico. ♦

Damos a continuación un algoritmo que busca un matching estable entre los individuos de A , B y C para una instancia dada I con listas de preferencias mutuas, donde las preferencias entre conjuntos son: $A \rightarrow B$, $B \rightarrow A$ y $C \rightarrow A$.

Este algoritmo se basa en la propiedad 4.2, dada anteriormente.

Algoritmo 4.2

El algoritmo puede comenzar desde cualquiera de los conjuntos A o B . Comenzaremos por A .

- 1) Se encuentra un matching estable entre los individuos del conjunto A y los de B por el algoritmo de Gale y Shapley

Al finalizar este paso, todos los c_k ($1 \leq k \leq n$) están sin asignar a los pares (a_i, b_j) formados en el punto anterior.

- 2) Para cada c_k ($1 \leq k \leq n$)

- 2.1) c_k es asignado a un par (a_i, b_j) que no tenga c asignado. Por ejemplo, c_k es asignado al par (a_k, b_j)

El matching así obtenido es estable, lo cual queda demostrado por el siguiente teorema.

Teorema 4.3

El Algoritmo 4.2 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.
Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un b_j asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

En el paso 2, cada c_k es asignado una y sólo una vez a pares distintos de (a_i, b_j) .

- b) Probar que el matching encontrado es estable.
Por correctitud del algoritmo de Gale y Shapley, M_{AB} es estable.
Por la propiedad 4.2, M_{AB} estable $\Rightarrow M$ es estable.

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de MET3 en instancias con listas únicas con preferencias mutuas.

Complejidad e Implementación

Sabemos que el algoritmo de Gale y Shapley tiene complejidad de $O(n^2)$, lo cual nos determina la complejidad del paso 1.

Para cumplir el paso 2, el orden dependerá del método elegido para efectuar la asignación. En el caso propuesto, vemos que para cada c_i la asignación se realiza en forma directa, con lo cual el orden es n . Luego, el algoritmo tiene complejidad del $O(n^2)$.

Para conseguir que el orden del algoritmo sea de n^2 , es decir, que el algoritmo de Gale y Shapley se resuelva en ese tiempo, se debe optimizar la elección que un agente realiza sobre dos o más agentes de otro conjunto sobre los cuales existe conflicto, evitando recorrer cada vez la lista de preferencias. Para eso, se utiliza lo que denominamos *Matriz de Ranking*.

En general, una *matriz de ranking* para un conjunto dado es una matriz que, para cada agente perteneciente al conjunto en cuestión, indica la posición en la lista de preferencias, de todos los agentes de el o los conjuntos sobre los cuales tiene elección.

Sea una instancia $I=(A,B,C;P)$ del problema de LUM, con $A \rightarrow B$, $B \rightarrow A$ y $C \rightarrow A$.

Una matriz de ranking para el conjunto A es una matriz R_A de dimensiones $n \times n$ en la cual $R_A(a, b)=k$ si b está en la k -ésima posición en la lista de preferencias de a (con $a \in A$ y $b \in B$).

De la misma forma, se puede definir R_B y R_C las correspondientes matrices de ranking para los conjuntos B y C .

Observemos que para el algoritmo propuesto, sólo es necesario implementar la matriz para el conjunto B , siendo el único que puede elegir entre varios agentes para resolver conflictos.

4.2.3 El conjunto de todos los Matchings Estables

Como se vio en el Capítulo 3, desde el punto de vista de la programación lineal entera, podemos definir la estructura del conjunto de todos los matchings estables, como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones. A continuación damos la formulación lineal entera para representar el conjunto de todos los matchings estables en instancias de LUM, reemplazando la restricción que garantiza estabilidad por una nueva que se ajusta a este tipo de instancias.

$$\text{Sea } x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{i > b \\ c' \in C}} x_{a,i,c} + \sum_{\substack{j > a \\ c'' \in C}} x_{j,b,c} + \sum_{\substack{k > a \\ b' \in B}} x_{k,b,c} + x_{a,b,c} \cdot 1, \quad \forall (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\} \quad (a,b,c) \in A \times B \times C \quad (5)$$

Para definir el conjunto de matchings fuertemente estable, la restricción 4 es reemplazada por:

$$\sum_{\substack{i \geq_a b \\ c' \in C}} x_{a,i,c} + \sum_{\substack{j \geq_b a \\ c'' \in C}} x_{j,b,c} + \sum_{\substack{k \geq_c a \\ b' \in B}} x_{k,b,c} + x_{a,b,c} \cdot 1, \quad \forall (a,b,c) \in A \times B \times C \quad (4')$$

Propiedad 4.4

La cantidad de matchings estables débiles de una instancia de LUM crece exponencialmente en función del tamaño de dicha instancia.

Demostración

Sea $I=(A,B,C;P)$ una instancia de LUM de tamaño n . De la demostración del Algoritmo 4.2 se desprende que si un matching M' es estable para la instancia I' sobre los conjuntos que se prefieren mutuamente (digamos A y B), luego, existe una correspondencia 'uno a muchos' entre los matchings estables de I' y los matchings fuertemente estables de I .

Vimos en el Capítulo 2 que la cantidad de matchings estables bidimensionales crece exponencialmente en función del tamaño de la instancia, entonces el conjunto de soluciones de I' crece exponencialmente en función de n y, por consiguiente, el conjunto de soluciones de I también. ♦

Veremos ahora que la relación de dominación entre los matchings estables (\geq_A), tal como fue definida en el Capítulo 3, no es una relación de orden, ya que no cumple con la propiedad antisimétrica.

Es decir, $M \geq_A M'$ y $M' \geq_A M$ no implica $M=M'$. Esto sucede porque la proyección sobre A y B de dos matchings estables M y M' puede ser la misma y, sin embargo, diferir la proyección sobre el conjunto C , con lo cual $M \geq_A M'$ y $M' \geq_A M$, pero $M \neq M'$.

A continuación veremos mediante dos ejemplos, cómo se comporta el conjunto de matchings estables en instancias con listas únicas mutuas.

En el Ejemplo 4.5, se muestra la no existencia de un orden entre los matchings estables y, a partir del análisis del mismo, se plantea una relación entre las clases de equivalencia, la cual define un orden parcial. El mismo ejemplo permite observar que el orden definido entre las clases de equivalencia no conforma un reticulado, pudiendo en algunos casos no existir supremo, ínfimo, e incluso formar un grafo no conexo.

Mediante el Ejemplo 4.6, se puede ver el comportamiento de los matchings estables de una instancia cuyas proyecciones sobre los conjuntos que se prefieren mutuamente también son estables en la correspondiente instancia determinada por estos conjuntos.

Ejemplo 4.5

Dada la instancia $I=(A,B,C;P)$ con $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:			
a_0 :	$b_1 b_2 b_0$	b_0 :	$a_0 a_1 a_2$
a_1 :	$b_0 b_2 b_1$	b_1 :	$a_0 a_2 a_1$
a_2 :	$b_1 b_0 b_2$	b_2 :	$a_0 a_2 a_1$
		c_0 :	$a_1 a_2 a_0$
		c_1 :	$a_2 a_1 a_0$
		c_2 :	$a_0 a_1 a_2$

veamos cómo queda definida la relación de dominación existente entre todos los matchings estables.

La instancia I tiene los siguientes matchings estables:

- $M_1 = \{(a_0,b_0,c_2), (a_1,b_1,c_0), (a_2,b_2,c_1)\}$
- $M_2 = \{(a_0,b_0,c_2), (a_1,b_2,c_1), (a_2,b_1,c_0)\}$
- $M_3 = \{(a_0,b_0,c_2), (a_1,b_2,c_0), (a_2,b_1,c_1)\}$
- $M_4 = \{(a_0,b_1,c_0), (a_1,b_0,c_1), (a_2,b_2,c_2)\}$
- $M_5 = \{(a_0,b_1,c_0), (a_1,b_0,c_2), (a_2,b_2,c_1)\}$
- $M_6 = \{(a_0,b_1,c_1), (a_1,b_0,c_0), (a_2,b_2,c_2)\}$
- $M_7 = \{(a_0,b_1,c_1), (a_1,b_0,c_2), (a_2,b_2,c_0)\}$
- $M_8 = \{(a_0,b_1,c_2), (a_1,b_0,c_1), (a_2,b_2,c_0)\}$
- $M_9 = \{(a_0,b_1,c_2), (a_1,b_0,c_0), (a_2,b_2,c_1)\}$
- $M_{10} = \{(a_0,b_1,c_2), (a_1,b_2,c_0), (a_2,b_0,c_1)\}$
- $M_{11} = \{(a_0,b_2,c_2), (a_1,b_1,c_0), (a_2,b_0,c_1)\}$
- $M_{12} = \{(a_0,b_2,c_2), (a_1,b_0,c_1), (a_2,b_1,c_0)\}$
- $M_{13} = \{(a_0,b_2,c_2), (a_1,b_0,c_0), (a_2,b_1,c_1)\}$

Como se mencionó en el capítulo anterior, podemos observar para un conjunto dado A , la existencia de clases de equivalencia, dentro de las cuales se agrupan todos los matchings estables cuyas proyecciones sobre A y el conjunto sobre el cual A elige, coinciden.

En este ejemplo, M_2 y M_3 son equivalentes para A , es decir las correspondientes proyecciones de M sobre A y B (el conjunto que A prefiere) son iguales:

$$M_{AB}(M_2) = \{(a_0,b_0), (a_1,b_2), (a_2,b_1)\} = M_{AB}(M_3)$$

Si calculamos las proyecciones sobre AB de los matchings anteriores, podemos observar que los M_{AB} correspondientes a los matchings $M_4, M_5, M_6, M_7, M_8, M_9$ son estables en la instancia definida por los conjuntos A y B , no así los restantes.

El conjunto de todos los matching estables forman las redes graficadas a continuación, de acuerdo a la relación de dominación definida entre ellos.

Desde el conjunto A :

Desde el punto de vista del conjunto A , los matchings estables de I se agrupan en las siguientes clases de equivalencia:

- $E_1 = \{M_1\}$
- $E_2 = \{M_2, M_3\}$
- $E_3 = \{M_4, M_5, M_6, M_7, M_8, M_9\}$
- $E_4 = \{M_{10}\}$
- $E_5 = \{M_{11}\}$
- $E_6 = \{M_{12}, M_{13}\}$

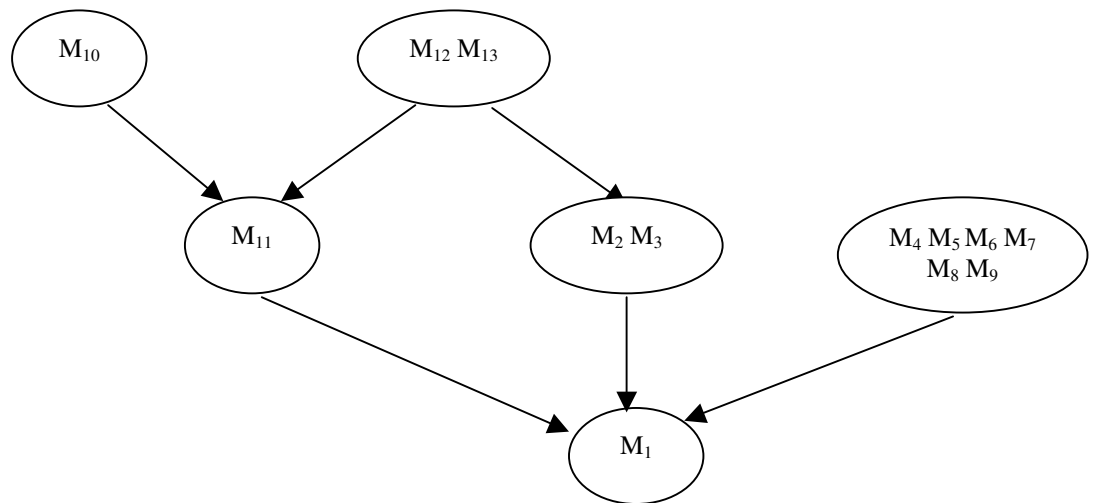


Figura 4.1 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.5 según el conjunto A

Desde el conjunto B:

Desde el punto de vista del conjunto B, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

- $E_1 = \{M_1\}$
- $E_2 = \{M_2, M_3\}$
- $E_3 = \{M_4, M_5, M_6, M_7, M_8, M_9\}$
- $E_4 = \{M_{10}\}$
- $E_5 = \{M_{11}\}$
- $E_6 = \{M_{12}, M_{13}\}$

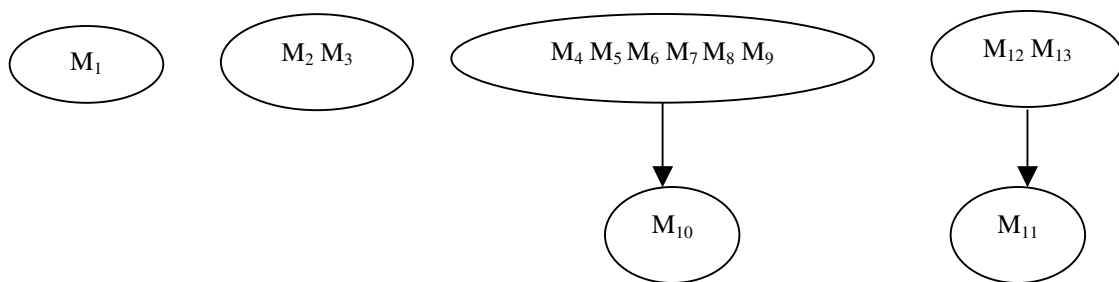


Figura 4.2 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.5 según el conjunto B

Desde el conjunto C:

Desde el punto de vista del conjunto C, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

- $E_1 = \{M_1, M_3, M_9, M_{10}, M_{11}, M_{13}\}$
- $E_2 = \{M_2, M_8, M_{12}\}$
- $E_3 = \{M_5\}$
- $E_4 = \{M_6\}$

$$E_5 = \{M_4\}$$

$$E_6 = \{M_7\}$$

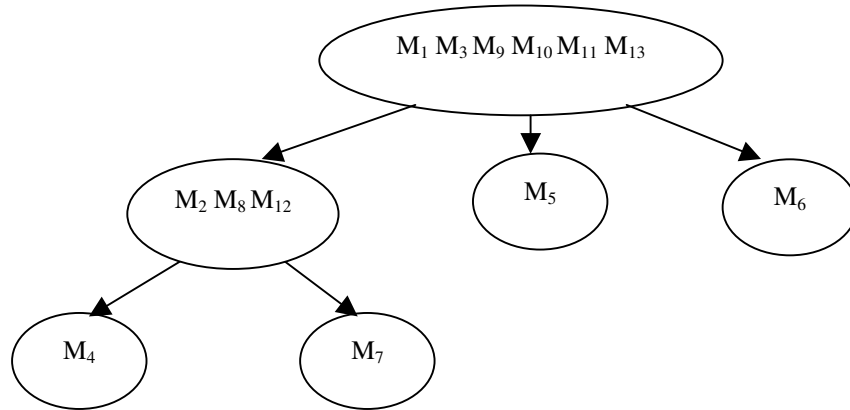


Figura 4.3 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.5 según el conjunto C

En el Ejemplo 4.5 vimos que la relación de dominación \geq_A entre los matchings estables tal como fue definida en el Capítulo 3, no es una relación de orden.

Si en cambio, definimos la relación de dominación entre clases de equivalencia, ahora sí tendremos una relación de orden, es decir cumple las propiedades reflexiva, antisimétrica y transitiva.

Definimos una relación de *dominación* entre todas las clases de equivalencia definidas por todos los matchings estables de acuerdo a lo siguiente: la clase de equivalencia E_1 *domina* a la clase de equivalencia E_2 con respecto a un conjunto si M_1 domina a $M_2 \forall M_1 \in E_1, \forall M_2 \in E_2$.

Es decir, $E \geq_A E'$ si $M \geq_A M' \forall M \in E, \forall M' \in E'$.

Veamos entonces que esta nueva relación es un orden:

- Reflexiva: $E \geq_A E$.
Trivial ya que $E=E$
- Antisimétrica: $E \geq_A E'$ y $E' \geq_A E \Rightarrow E = E'$
 $E \geq_A E'$ y $E' \geq_A E \Rightarrow M \geq_A M'$ y $M' \geq_A M \forall M \in E, \forall M' \in E' \Rightarrow M_{BC}(a) \geq_a M'_{BC}(a)$ y $M'_{BC}(a) \geq_a M_{BC}(a)$
 $\forall M \in E, \forall M' \in E', \forall a \in A \Rightarrow M_{BC}(a) = M'_{BC}(a) \forall M \in E, \forall M' \in E', \forall a \in A \Rightarrow M$ y M' están en la misma clase de equivalencia $\Rightarrow E=E'$
- Transitiva: $E \geq_A E'$ y $E' \geq_A E'' \Rightarrow E \geq_A E''$
 $E \geq_A E'$ y $E' \geq_A E'' \Rightarrow M \geq_A M'$ y $M' \geq_A M'' \forall M \in E, \forall M' \in E', \forall M'' \in E'' \Rightarrow M_{BC}(a) \geq_a M'_{BC}(a)$ y $M'_{BC}(a) \geq_a M''_{BC}(a) \forall M \in E, \forall M' \in E', \forall M'' \in E'', \forall a \in A \Rightarrow M_{BC}(a) \geq_a M''_{BC}(a) \forall M \in E, \forall M'' \in E'', \forall a \in A$. ♦

Los grafos dirigidos que representan esta nueva relación de dominación serán idénticos a los anteriores, en donde cada nodo representa indistintamente la clase de equivalencia o el conjunto de matchings pertenecientes a la misma.

Podemos observar que, si bien ahora existe una relación de orden, igualmente el conjunto de las clases de equivalencia no constituye un reticulado. Volviendo al Ejemplo 4.5, notemos que desde el conjunto A existe ínfimo pero no supremo (Figura 4.1), a diferencia del conjunto C, desde el cual existe supremo pero no ínfimo (Figura 4.3).

Notemos que, por la propiedad 4.2 ,dado que todo matching estable entre dos conjuntos A y B, es proyección de algún matching estable en cualquier instancia I de LUM conformada por los conjuntos A, B y un nuevo conjunto C junto con sus listas de preferencias, el conjunto de todos los matchings estables

entre A y B estará incluido en el conjunto de las proyecciones de todos los matchings estables para cualquier instancia I.

La propiedad 4.5 que veremos a continuación, nos permite dar una relación de precedencia entre matchings estables con listas únicas mutuas en función de la relación de precedencia que tienen los matrimonios estables en 2D que conforman la instancia.

Propiedad 4.5

$$M_{AB} >_A M'_{AB} \Rightarrow M >_A M'$$

Demostración

$$M_{AB} >_A M'_{AB} \Rightarrow \forall a_i \in A, M_B(a_i) >_{a_i} M'_B(a_i) \Rightarrow M >_A M'. \quad \blacklozenge$$

Esta propiedad nos muestra que, tomando un representante de cada clase de equivalencia, existe un orden en un subconjunto de los matchings estables para instancias con listas únicas mutuas y este orden estaría dado por el orden que los matchings tengan en los matrimonios estables que lo conforman. Este subconjunto podría no incluir a todas las clases de equivalencia de la instancia ya que, como vimos antes, no todos los matchings estables en instancias de LUM tienen sus correspondientes proyecciones estables en 2D.

El siguiente es un ejemplo que nos muestra cómo se comporta el subconjunto de los matchings estables de una instancia $I=(A,B,C;P)$ que también son estables en la instancia $I'=(A,B;P')$ con $P'=\{P(a)/a \in A\} \cup \{P(b)/b \in B\}$.

Ejemplo 4.6

Dada la instancia $I=(A,B,C;P)$ con $A=\{a_0, a_1, a_2\}$, $B=\{b_0, b_1, b_2\}$, $C=\{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:			
a_0 :	$b_0 \ b_1 \ b_2$	b_0 :	$a_2 \ a_1 \ a_0$
a_1 :	$b_1 \ b_0 \ b_2$	b_1 :	$a_0 \ a_2 \ a_1$
a_2 :	$b_2 \ b_0 \ b_1$	b_2 :	$a_1 \ a_2 \ a_0$
		c_0 :	$a_1 \ a_2 \ a_0$
		c_1 :	$a_0 \ a_2 \ a_1$
		c_2 :	$a_1 \ a_0 \ a_2$

veamos cómo queda definida la relación de dominación definida entre todos los matching estables.

La instancia I tiene los siguientes matchings estables:

- $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$
- $M_2 = \{(a_0, b_0, c_0), (a_1, b_1, c_2), (a_2, b_2, c_1)\}$
- $M_3 = \{(a_0, b_0, c_1), (a_1, b_1, c_0), (a_2, b_2, c_2)\}$
- $M_4 = \{(a_0, b_0, c_1), (a_1, b_1, c_2), (a_2, b_2, c_0)\}$
- $M_5 = \{(a_0, b_0, c_2), (a_1, b_1, c_1), (a_2, b_2, c_0)\}$
- $M_6 = \{(a_0, b_0, c_2), (a_1, b_1, c_0), (a_2, b_2, c_1)\}$
- $M_7 = \{(a_0, b_1, c_0), (a_1, b_0, c_1), (a_2, b_2, c_2)\}$
- $M_8 = \{(a_0, b_1, c_0), (a_1, b_0, c_2), (a_2, b_2, c_1)\}$
- $M_9 = \{(a_0, b_1, c_1), (a_1, b_0, c_0), (a_2, b_2, c_2)\}$
- $M_{10} = \{(a_0, b_1, c_1), (a_1, b_0, c_2), (a_2, b_2, c_0)\}$
- $M_{11} = \{(a_0, b_1, c_2), (a_1, b_0, c_1), (a_2, b_2, c_0)\}$
- $M_{12} = \{(a_0, b_1, c_2), (a_1, b_0, c_0), (a_2, b_2, c_1)\}$
- $M_{13} = \{(a_0, b_1, c_0), (a_1, b_2, c_1), (a_2, b_0, c_2)\}$
- $M_{14} = \{(a_0, b_1, c_0), (a_1, b_2, c_2), (a_2, b_0, c_1)\}$
- $M_{15} = \{(a_0, b_1, c_1), (a_1, b_2, c_0), (a_2, b_0, c_2)\}$
- $M_{16} = \{(a_0, b_1, c_1), (a_1, b_2, c_2), (a_2, b_0, c_0)\}$

$$M_{17} = \{(a_0, b_1, c_2), (a_1, b_2, c_1), (a_2, b_0, c_0)\}$$

$$M_{18} = \{(a_0, b_1, c_2), (a_1, b_2, c_0), (a_2, b_0, c_1)\}$$

$$M_{19} = \{(a_0, b_2, c_1), (a_1, b_1, c_2), (a_2, b_0, c_0)\}$$

$$M_{20} = \{(a_0, b_2, c_1), (a_1, b_0, c_2), (a_2, b_1, c_0)\}$$

Todos los matchings detallados anteriormente, excepto M_{19} y M_{20} , son estables en la instancia

$$I=(A,B;P') \text{ con } P'=\{P(a)/a \in A\} \cup \{P(b)/b \in B\}$$

El conjunto de todos los matching estables forman las redes graficadas a continuación, de acuerdo a la relación de dominación definida entre ellos.

Desde el conjunto A:

Desde el punto de vista del conjunto A, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

$$E_1 = \{M_1, M_2, M_3, M_4, M_5, M_6\}$$

$$E_2 = \{M_7, M_8, M_9, M_{10}, M_{11}, M_{12}\}$$

$$E_3 = \{M_{13}, M_{14}, M_{15}, M_{16}, M_{17}, M_{18}\}$$

$$E_4 = \{M_{19}\}$$

$$E_5 = \{M_{20}\}$$

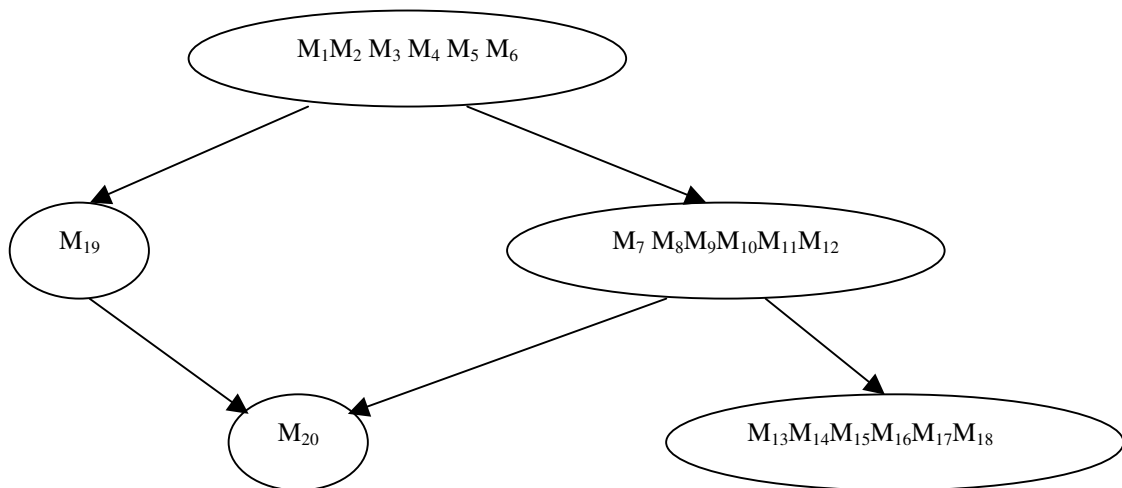


Figura 4.4 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.6 según el conjunto A

Desde el conjunto B:

Desde el punto de vista del conjunto B, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

$$E_1 = \{M_1, M_2, M_3, M_4, M_5, M_6\}$$

$$E_2 = \{M_7, M_8, M_9, M_{10}, M_{11}, M_{12}\}$$

$$E_3 = \{M_{13}, M_{14}, M_{15}, M_{16}, M_{17}, M_{18}\}$$

$$E_4 = \{M_{19}\}$$

$$E_5 = \{M_{20}\}$$

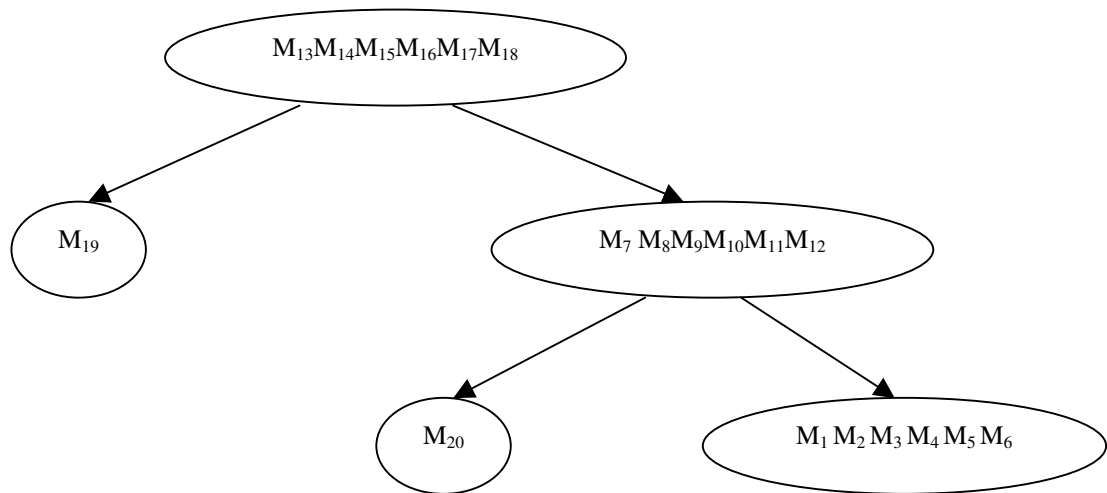


Figura 4.5 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.6 según el conjunto B

Desde el conjunto C:

Desde el punto de vista del conjunto C, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

- $E_1 = \{M_1, M_7, M_{13}\}$
- $E_2 = \{M_2, M_8, M_{14}\}$
- $E_3 = \{M_3, M_9, M_{15}\}$
- $E_4 = \{M_4, M_{10}, M_{16}, M_{19}, M_{20}\}$
- $E_5 = \{M_5, M_{11}, M_{17}\}$
- $E_6 = \{M_6, M_{12}, M_{18}\}$

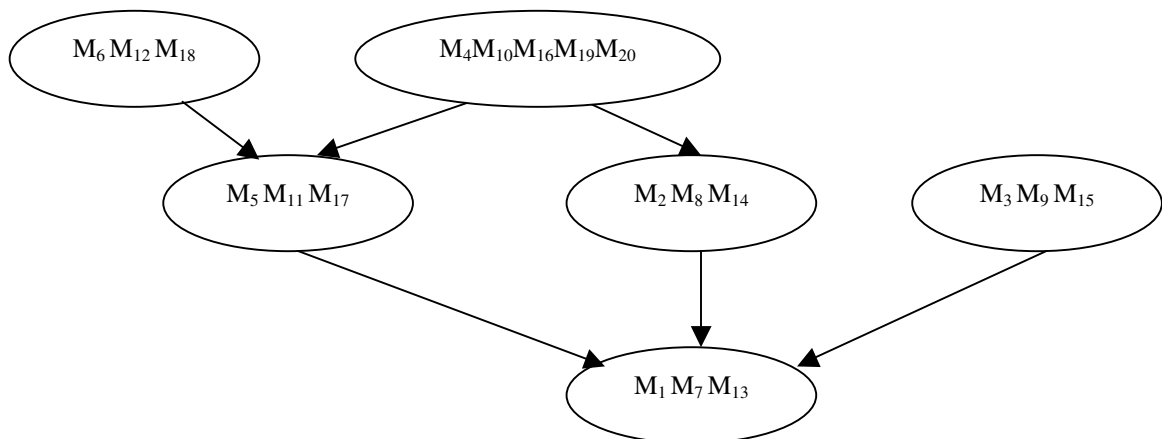


Figura 4.6 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.6 según el conjunto C

Podemos representar mediante un grafo dirigido las relaciones de dominación que existen entre los matchings estables de la instancia I que también lo son en I' .

Sea $G = (V,A)$ el grafo dirigido de todos los matchings estables de I definido anteriormente.

Sea $G' = (V', A')$ el grafo dirigido de todos los matchings estables de I' .

Sabemos que cada vértice de G representa una clase de equivalencia E del conjunto A , es decir $M_B(a) = M'_B(a) \forall a \in A, \forall M, M' \in E$. De acuerdo a esto, cada uno de los vértices de G en los cuales la proyección AB de sus integrantes sea estable en I' , es mapeado directamente a un vértice de G' .

Dado que todos los matchings estables en I' también lo son en I , G' representará el conjunto de todos los matchings estables en I' , cumpliendo todas las propiedades que han sido estudiadas para el caso de matchings estables de dos dimensiones.

Luego, para el caso de A , G' será:

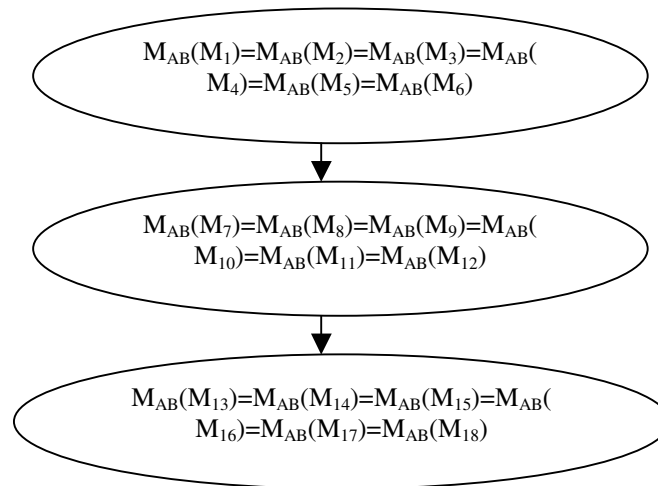


Figura 4.7 Relación de dominación según el conjunto A entre todas las proyecciones de los matchings estables para la instancia I del Ejemplo 4.5 que también lo son para la instancia I' .

Esta propiedad nos permitirá encontrar un subconjunto de los matchings estables en LUM, a partir del conjunto de todos los matchings estables para I' . Este subconjunto estará formado por un representante de cada clase de equivalencia de I que se corresponda con un matching estable en I' .

El algoritmo 4.2 encuentra un matching M estable en I , cuya proyección M_{AB} también lo es en I' . A partir de este matching M_{AB} , el cual representa el óptimo para el conjunto A , podemos encontrar el conjunto de todos los matchings estables entre los conjuntos A y B ($S(I)$), en $O(n^2 + nS(I))$, como vimos en el capítulo 3. Asignando los valores de los agentes c en todas las combinaciones posibles, obtenemos el subconjunto de todos los matchings estables en I cuyas proyecciones también lo son en I' .

Consideremos ahora el conjunto de todos los matchings fuertemente estables: en este caso el resultado que se obtiene para los conjuntos con preferencia mutua es un reticulado distributivo equivalente al reticulado formado por los matchings estables entre los conjuntos A y B . Esto queda demostrado con la Propiedad 4.6, que daremos a continuación.

Además, para el conjunto C , todos los matchings estables pertenecerán a la misma clase de equivalencia, ya que, de acuerdo a lo demostrado en el Teorema 4.1, la pareja a de cada agente c en todo matching fuertemente estable, será el primer elemento en su lista de preferencias.

Propiedad 4.6

Sea $I = (A, B; P)$ una instancia del problema de matrimonios estables de cardinalidad n y $S(I)$ el conjunto de todos los matchings estables entre A y B .

Sea $I' = (A, B, C; P')$ una instancia de LUM, con C conjunto de agentes de cardinalidad n , la siguiente relación de preferencia: $A \rightarrow B$, $B \rightarrow A$, $C \rightarrow A$, y $P' = P \cup \{P(c) | c \in C\}$. Entonces, si existe matching estable

en I' , el conjunto de matchings fuertemente estables de I' queda definido por el conjunto de soluciones $S(I)$ y la relación de dominación entre los mismos se mantiene. Esto es,

- i) $S(I') = \emptyset$ o $S(I') = \{M' / M' = M J_A \mid \{(a,c) / (\exists b) \text{Pos}(c,(a,b,c))=0\}, \text{ con } M \in S(I)\}$
- ii) $M_i >_A M_j \Rightarrow M'_i >_A M'_j \quad \forall M_i, M_j \in S(I), \forall M'_i, M'_j \in S(I')$
- iii) $M_i >_B M_j \Rightarrow M'_i >_B M'_j \quad \forall M_i, M_j \in S(I), \forall M'_i, M'_j \in S(I')$

Demostración

i) Supongamos $S(I') \neq \emptyset$.

Por el Teorema 4.1, el primer elemento de las listas de los C son distintos. Esto significa que $\{(a,c) / (\exists b) \text{Pos}(c,(a,b,c))=0\}$ existe y es único. Llamemos MC a este conjunto.

Por correctitud del Algoritmo 4.1, el matching obtenido por $M J_A MC$ es fuertemente estable para I' . Luego, $\{M J_A MC / M \in S(I)\} \subseteq S(I')$.

Falta demostrar que $S(I') \subseteq \{M J_A MC / M \in S(I)\}$

Supongamos que $\exists M' / M' \in S(I')$ y $M' \notin \{M J_A MC / M \in S(I)\} \Rightarrow \exists (a,b,c) \in M'$ tal que:

$(b \geq_a M'_B(a) \text{ y } a \geq_b M'_A(b) \text{ y } a \geq_c M'_A(c)) \text{ y } ((a,b) \notin M \forall M \in S(I) \text{ o } \text{Pos}(c,(M_A(c),b,c)) \neq 0)$

\Rightarrow

1) $b \geq_a M_B(a) \text{ y } a \geq_b M_A(b) \text{ y } a \geq_c M'_A(c) \text{ y } (a,b) \notin M \forall M \in S(I)$ o

2) $b \geq_a M_B(a) \text{ y } a \geq_b M_A(b) \text{ y } a \geq_c M'_A(c) \text{ y } \text{Pos}(c,(M_A(c),b,c)) \neq 0$

\Rightarrow

1) $b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a \geq_c M'_A(c)$ o

2) $b \geq_a M_B(a) \text{ y } a \geq_b M_A(b) \text{ y } a \geq_c M'_A(c) \text{ y } (\exists a')(a' >_c M'_A(c))$

Si se cumplen 1 y 2 $\Rightarrow b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a \geq_c M'_A(c) \text{ y } (\exists a')(a' >_c M'_A(c)) \Rightarrow (a,b,c)$ es terna bloqueante de M' ya que $b >_a M'_B(a) \text{ y } a >_b M'_A(b) \text{ y } a \geq_c M'_A(c)$

Si se cumplen 1 y no 2 $\Rightarrow b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a \geq_c M'_A(c) \text{ y no } (\exists a')(a' >_c M'_A(c)) \Rightarrow b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a = M'_A(c) \Rightarrow (a,b,c)$ es terna bloqueante de M' ya que $b >_a M'_B(a) \text{ y } a >_b M'_A(b) \text{ y } a = M'_A(c)$

Si se cumplen no 1 y 2 $\Rightarrow b = M_B(a) \text{ y } a \geq_c M'_A(c) \text{ y } (\exists a')(a' >_c M'_A(c)) \Rightarrow b = M_B(a) \text{ y } a >_c M'_A(c)$, ya que si no (a,b,c) pertenecería a $M' \Rightarrow (a,b,c)$ es terna bloqueante de M' ya que $b = M'_B(a) \text{ y } a = M'_A(b) \text{ y } a >_c M'_A(c)$.

ii) $M_i >_A M_j \Rightarrow M'_i >_A M'_j \quad \forall M_i, M_j \in S(I), \forall M'_i, M'_j \in S(I')$
 $M_i >_A M_j \Rightarrow M_{iB}(a) >_a M_{jB}(a) \forall a \in A \Rightarrow M'_{iB}(a) >_a M'_{jB}(a) \forall a \in A$, ya que $P(a) = P'(a) \Rightarrow M'_i >_A M'_j$.

iii) $M_i >_B M_j \Rightarrow M'_i >_B M'_j \quad \forall M_i, M_j \in S(I), \forall M'_i, M'_j \in S(I')$
 $M_i >_B M_j \Rightarrow M_{iA}(b) >_b M_{jA}(b) \forall b \in B \Rightarrow M'_{iA}(b) >_b M'_{jA}(b) \forall b \in B$, ya que $P(b) = P'(b) \Rightarrow M'_i >_B M'_j$. ♦

Esta propiedad nos permitirá ahora encontrar el conjunto de todos los matchings fuertemente estables en LUM, siempre que exista solución, a partir del conjunto de todos los matchings estables para la instancia 2D correspondiente, nuevamente en $O(n^2 + n|S(I)|)$.

Propiedad 4.7

La cantidad de matchings estables fuertes de una instancia de LUM que tienen solución fuertemente estable crece exponencialmente en función del tamaño de dicha instancia.

Demostración

Sea $I=(A,B,C;P)$ una instancia de LUM de tamaño n . De la propiedad 4.6 se desprende que si un matching M' es estable para la instancia I' sobre los conjuntos que se prefieren mutuamente (digamos A y B) y la instancia tiene matching fuertemente estable, luego, existe una correspondencia 'uno a uno' entre los matchings estables de I' y los matchings fuertemente estables de I .

Vimos en el Capítulo 2 que la cantidad de matchings estables bidimensionales crece exponencialmente en función del tamaño de la instancia, el conjunto de soluciones de I' crece exponencialmente en función de n y, por consiguiente, el conjunto de soluciones de I también. ♦

4.3 Con Listas de Preferencias Circulares (LUC)

4.3.1 Matching y Estabilidad

Este problema fue introducido por Ng y Hirschberg, quienes lo plantearon como problema abierto. [NH/95]

Cada individuo tiene una lista de preferencias, eligiendo sobre individuos de uno de los otros dos conjuntos, los individuos de este conjunto eligen sobre los del tercer conjunto y estos a los del primero. Sin pérdida de generalidad, diremos que los individuos del conjunto A definen sus preferencias sobre los individuos del conjunto B , los de B sobre C y los de C sobre A .

Decimos que la terna (a,b,c) es una *terna bloqueante* del matching M o que bloquea a M si $(a,b,c) \notin M$ y:

1. $b \succeq_a M_B(a)$ y
2. $c \succeq_b M_C(b)$ y
3. $a \succeq_c M_A(c)$

A continuación se muestra un ejemplo en el cual para una instancia particular con listas de preferencias circulares, se da un matching estable y otro no estable, en el cual se puede ver la existencia de por lo menos una terna bloqueante.

Ejemplo 4.7

Sea $I=(A,B,C;P)$ con $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:

$a_0:$	$b_1 b_2 b_0$	$b_0:$	$c_0 c_1 c_2$	$c_0:$	$a_1 a_2 a_0$
$a_1:$	$b_0 b_2 b_1$	$b_1:$	$c_0 c_2 c_1$	$c_1:$	$a_2 a_1 a_0$
$a_2:$	$b_1 b_0 b_2$	$b_2:$	$c_0 c_2 c_1$	$c_2:$	$a_0 a_1 a_2$

El matching $M_1=\{(a_0,b_1,c_2),(a_1,b_0,c_0),(a_2,b_2,c_1)\}$ es estable para la instancia I , mientras que el matching $M_2=\{(a_0,b_0,c_0),(a_1,b_2,c_1),(a_2,b_1,c_2)\}$ no es estable, ya que es bloqueado por la terna (a_0,b_2,c_2) .

Al igual que en LUM, vemos que la propiedad de consistencia aplica al problema de LUC, lo cual queda demostrado con la Propiedad 4.8, que daremos a continuación.

Propiedad 4.8

Las listas de preferencias únicas circulares cumplen la propiedad de consistencia.

Demostración

Para probar que una lista de preferencias es consistente, debemos probar que:

- 1) $(a,b) >_c (a,b') \Rightarrow (a',b) >_c (a',b') \quad \forall a,a' \in A, \forall b,b' \in B, \forall c \in C$ y
 2) $(a,b) >_c (a',b) \Rightarrow (a,b') >_c (a',b') \quad \forall a,a' \in A, \forall b,b' \in B, \forall c \in C$

Para listas de preferencias circulares, como $C \rightarrow A$:

- 1) $(a,b) >_c (a,b')$ no se cumple
 2) $(a,b) >_c (a',b) \Rightarrow a >_c a' \Rightarrow (a,b') >_c (a',b')$. ♦

Tal como habíamos definido para LUM, redefinimos el problema para contemplar como matchings estables aquellos en los cuales no existe terna tal que los tres integrantes de la misma hubiesen preferido estar juntos antes que con su pareja actual, no permitiéndose mantener la misma pareja. Definimos también para LUC una nueva estabilidad, más débil que la anterior.

Decimos ahora, que una terna (a,b,c) es una *terna bloqueante* para un matching M si $(a,b,c) \notin M$ y:

1. $b >_a M_B(a)$ y
2. $c >_b M_C(b)$ y
3. $a >_c M_A(c)$

Llamaremos entonces *estabilidad fuerte* al tipo de estabilidad más restrictivo (según la definición original de terna bloqueante) y *estabilidad débil*, o simplemente *estabilidad*, a la que cumple con la no existencia de ternas bloqueante de acuerdo a esta última definición.

De la misma manera, diremos que un matching es *fuertemente estable* si cumple con la estabilidad fuerte y *débilmente estable* o, simplemente, *estable*, si cumple con la estabilidad débil.

Excepto que se especifique de otra manera, a partir de este momento y a lo largo de todo el análisis de LUC, trabajaremos con la estabilidad débil.

Podemos ver, en base a las propiedades enunciadas a continuación, algunas características particulares en cuanto a la estabilidad de los matchings para instancias de tamaño $n=2$.

Propiedad 4.9

Sea I una instancia de LUC de tamaño $n=2$, luego todos los matchings tienen estabilidad débil.

Demostración

Supongamos que existe M de tamaño $n=2$ tal que no es estable \Rightarrow

existe una terna (a,b,c) no perteneciente a M tal que $a >_c M_A(c)$, $b >_a M_B(a)$ y $c >_b M_C(b)$.

Para que estas desigualdades estrictas se cumplan, los agentes a,b y c que forman la terna deberían estar dentro del matching en ternas distintas, por lo que el tamaño del matching debería ser mayor o igual que 3, lo que se contradice con la hipótesis. ♦

Propiedad 4.10

Sea I una instancia de LUC de tamaño $n=2$, siempre existe matching fuertemente estable.

Demostración

Sea $I=(A,B,C;P)$, con $A=\{a_0,a_1\}$, $B=\{b_0,b_1\}$, $C=\{c_0,c_1\}$

Los posibles matchings para I son:

$$M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1)\}$$

$$M_2 = \{(a_0, b_0, c_1), (a_1, b_1, c_0)\}$$

$$M_3 = \{(a_0, b_1, c_0), (a_1, b_0, c_1)\}$$

$$M_4 = \{(a_0, b_1, c_1), (a_1, b_0, c_0)\}$$

Demostraremos que no es posible construir P de tal forma que ninguno de estos matching sean estables.

Todas las ternas posibles entre A, B y C son:

$$t_1 = (a_0, b_0, c_0)$$

$$t_2 = (a_1, b_1, c_1)$$

$$t_3 = (a_0, b_0, c_1)$$

$$t_4 = (a_1, b_1, c_0)$$

$$t_5 = (a_0, b_1, c_0)$$

$$t_6 = (a_1, b_0, c_1)$$

$$t_7 = (a_0, b_1, c_1)$$

$$t_8 = (a_1, b_0, c_0)$$

Luego,

$M_1 = \{t_1, t_2\}$ tiene como posibles ternas bloqueantes $t_3, t_4, t_5, t_6, t_7, t_8$

$M_2 = \{t_3, t_4\}$ tiene como posibles ternas bloqueantes $t_1, t_2, t_5, t_6, t_7, t_8$

$M_3 = \{t_5, t_6\}$ tiene como posibles ternas bloqueantes $t_1, t_2, t_3, t_4, t_7, t_8$

$M_4 = \{t_7, t_8\}$ tiene como posibles ternas bloqueantes $t_1, t_2, t_3, t_4, t_5, t_6$

Quiero construir P de tal forma que los cuatro matching sean bloqueados por alguna terna.

Supongamos, sin pérdida de generalidad, que elijo t_3 para que bloquee a $M_1 \Rightarrow c_1 >_{b_0} c_0$ y $a_0 >_{c_1} a_1 \Rightarrow t_1, t_2,$

t_6 y t_8 no bloquean a M_2 y t_1 y t_8 no bloquean a M_3 y t_2 y t_6 no bloquean a $M_4 \Rightarrow$

$M_1 = \{t_1, t_2\}$ es bloqueado por t_3

$M_2 = \{t_3, t_4\}$ tiene como posibles ternas bloqueantes t_5, t_7

$M_3 = \{t_5, t_6\}$ tiene como posibles ternas bloqueantes t_2, t_3, t_4, t_7

$M_4 = \{t_7, t_8\}$ tiene como posibles ternas bloqueantes t_1, t_3, t_4, t_5

Con las preferencias definidas hasta el momento ($c_1 >_{b_0} c_0$ y $a_0 >_{c_1} a_1$), intento bloquear ahora al matching M_2 , analizando las dos posibles ternas bloqueantes (t_5 y t_7).

1) Supongamos que t_5 bloquea a $M_2 \Rightarrow b_1 >_{a_0} b_0$ y $a_0 >_{c_0} a_1 \Rightarrow t_1, t_3, t_4$ y t_8 no bloquean a M_3 y t_1 y t_3 no bloquean a $M_4 \Rightarrow$

$M_1 = \{t_1, t_2\}$ es bloqueado por t_3

$M_2 = \{t_3, t_4\}$ es bloqueado por t_5

$M_3 = \{t_5, t_6\}$ tiene como posibles ternas bloqueantes t_2, t_7

$M_4 = \{t_7, t_8\}$ tiene como posibles ternas bloqueantes t_4, t_5

Nuevamente, y considerando ahora las nuevas preferencias definidas ($b_1 >_{a_0} b_0$ y $a_0 >_{c_0} a_1$), intento bloquear al matching M_3 , analizando las dos posibles ternas bloqueantes (t_2 y t_7).

1.1) Supongamos que t_2 bloquea a $M_3 \Rightarrow b_1 >_{a_1} b_0$ y $c_1 >_{b_1} c_0 \Rightarrow t_4$ y t_8 no bloquean a $M_4 \Rightarrow M_4$ es estable.

1.2) Si, en cambio, t_7 bloquea a $M_3 \Rightarrow c_1 >_{b_1} c_0$ y $a_0 >_{c_1} a_1 \Rightarrow t_4$ y t_8 no bloquean a $M_4 \Rightarrow M_4$ es estable.

2) Si, en cambio, es t_7 quien bloquea a $M_2 \Rightarrow b_1 >_{a_0} b_0$ y $c_1 >_{b_1} c_0 \Rightarrow t_1$ y t_3 no bloquean a M_3 y t_1, t_3, t_4 y t_5 no bloquean a $M_4 \Rightarrow M_4$ es estable.

Como vimos, no es posible construir listas de preferencias para una instancia de tamaño $n=2$ tal que todos los matchings posibles sean bloqueados por alguna terna.

Luego, no existe I instancia de LUC de tamaño $n=2$ que no tenga matching fuertemente estable. ♦

A continuación daremos una propiedad que nos indica que en instancias LUC donde sea posible hallar un matching canónico, hallamos también un matching débilmente estable.

Todo matching canónico tiene estabilidad débil.

Demostración

Debemos ver que si M es canónico, entonces no existe terna (a,b,c) no perteneciente a M tal que:

1. $b >_a M_B(a)$ y
2. $c >_b M_C(b)$ y
3. $a >_c M_A(c)$

Sabemos que, como M es canónico,

- i) $\text{Pos}(a, (a, M_B(a), M_C(a))) = 0$, o
- ii) $\text{Pos}(b, (M_A(b), b, M_C(b))) = 0$, o
- iii) $\text{Pos}(c, (M_A(c), M_B(c), c)) = 0$

Si $\text{Pos}(a, (a, M_B(a), M_C(a))) = 0 \Rightarrow M_B(a)$ es el primero en la lista de preferencias de $a \Rightarrow$ no existe b tal que $b >_a M_B(a)$

Si $\text{Pos}(b, (M_A(b), b, M_C(b))) = 0 \Rightarrow M_C(b)$ es el primero en la lista de preferencias de $b \Rightarrow$ no existe c tal que $c >_b M_C(b)$

Si $\text{Pos}(c, (M_A(c), M_B(c), c)) = 0 \Rightarrow M_A(c)$ es el primero en la lista de preferencias de $c \Rightarrow$ no existe a tal que $a >_c M_A(c)$

Luego, no existe terna bloqueante para M canónico. ♦

4.3.2 Búsqueda de un Matching Estable

La complejidad del problema de encontrar un matching estable en instancias de LUC es un problema abierto [NH/95]. Conjeturamos que el mismo es NP-Completo para ambas definiciones de estabilidad y, en base a eso, brindamos un algoritmo que, para algunas instancias, encuentra en tiempo polinomial un matching estable. Es decir, el algoritmo es correcto (si devuelve un resultado, el mismo es solución) pero no completo (puede no encontrar matching estable aún si el mismo existe).

El algoritmo que detallaremos encuentra un frame de matchings estables, es decir, a partir del resultado obtenido, se puede generar una familia de matchings estables.

La idea básica del algoritmo consiste en asignar los agentes de los conjuntos evitando la generación de ternas bloqueantes. Para ello, se realizan las asignaciones tratando de minimizar la cantidad de agentes generadores de ternas bloqueantes y luego buscando asignar a los mismos sus mayores preferencias, en la medida de lo posible.

Como veremos, este algoritmo no beneficia a un conjunto en particular, sino a algunos agentes “privilegiados” de cada uno de los conjuntos: a los agentes más solicitados por el conjunto que los elige y, por tanto, más factibles de participar en una terna bloqueante.

Antes de dar el algoritmo, veamos algunos ejemplos que faciliten posteriormente el seguimiento del mismo, permitiendo de esta forma un mejor entendimiento.

Ejemplo 4.8

Sea la instancia $I=(A,B,C;P)$ con $A=\{a_0, a_1, a_2, a_3, a_4\}$, $B=\{b_0, b_1, b_2, b_3, b_4\}$, $C=\{c_0, c_1, c_2, c_3, c_4\}$ y las siguientes listas de preferencias:

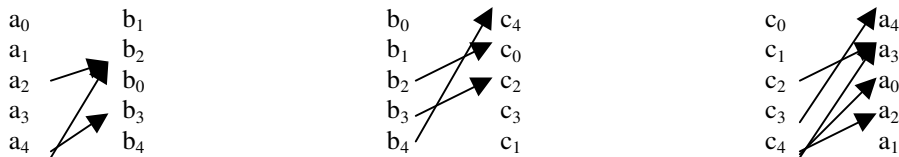
<p>P: A $a_0: b_1 b_2 b_0 b_3 b_4$ $a_1: b_2 b_1 b_0 b_3 b_4$</p>	<p>B $b_0: c_4 c_1 c_3 c_2 c_0$ $b_1: c_0 c_1 c_2 c_3 c_4$</p>	<p>C $c_0: a_4 a_3 a_0 a_1 a_2$ $c_1: a_3 a_2 a_0 a_1 a_4$</p>
--	--	--

$a_2: b_2 b_0 b_1 b_3 b_4$ $b_2: c_0 c_2 c_3 c_1 c_4$ $c_2: a_3 a_0 a_2 a_4 a_1$
 $a_3: b_3 b_1 b_4 b_2 b_0$ $b_3: c_2 c_3 c_0 c_1 c_4$ $c_3: a_4 a_2 a_0 a_1 a_3$
 $a_4: b_3 b_2 b_4 b_1 b_0$ $b_4: c_4 c_1 c_0 c_2 c_3$ $c_4: a_2 a_0 a_3 a_1 a_4$

Si asignamos arbitrariamente agentes distintos de B a los de A, las únicas posibles ternas bloqueantes estarían formadas por los pares (a_i, b_j) tales que b_j esté antes que la pareja b que le fue asignada a a_i . Buscamos, para cada conjunto, el primer agente en su lista de preferencias tal que no haya sido asignado, resultando:

a_0	b_1	b_0	c_4	c_0	a_4
a_1	b_2	b_1	c_0	c_1	a_3
a_2	b_0	b_2	c_2	c_2	a_0
a_3	b_3	b_3	c_3	c_3	a_2
a_4	b_4	b_4	c_1	c_4	a_1

y marcamos los pares posibles generadores de ternas bloqueantes en cada caso:



Las pares unidos por las flechas en el gráfico representan los pares de posibles ternas bloqueantes. Por ejemplo, la flecha que va desde a_2 hacia b_2 indica que el par (a_2, b_2) podría ser parte de una terna bloqueante (esto sucede porque $b_2 >_{a_2} b_0$). Además, todas las posibles ternas bloqueantes estarán formadas por estos pares, exclusivamente.

Buscamos entonces, minimizar la cantidad de agentes que han sido “apuntados”, es decir, la cantidad de agentes que se deberán privilegiar para romper los posibles bloqueos.

Vemos que, en el primer caso, sólo dos b distintos deberán ser privilegiados en su elección. Por lo tanto, elijamos AB como partida.

Ahora, debemos asignar los c a b_2 y b_3 de forma tal de romper las posibles ternas bloqueantes. Como las primeras preferencias de las listas de b_2 y b_3 son distintas, podemos realizar tal asignación asegurando que ni b_2 ni b_3 generarán ternas bloqueantes.

Luego, construimos el siguiente frame de matchings estables:

FM(I):

a_0	b_1	-
a_1	b_2	c_0
a_2	b_0	-
a_3	b_3	c_2
a_4	b_4	-

Cualquier matching perteneciente a la familia generada por FM será matching estable.

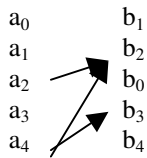
Ejemplo 4.9

Veamos ahora qué pasaría si cuando realizamos las elecciones de b_2 y b_3 del ejemplo anterior, los primeros c_k de sus listas de preferencias coinciden.

Para eso modificaremos las listas de preferencias de b_3 y c_2 de la instancia del Ejemplo 4.8, resultando P:

A	B	C
$a_0: b_1 b_2 b_0 b_3 b_4$	$b_0: c_4 c_1 c_3 c_2 c_0$	$c_0: a_4 a_3 a_0 a_1 a_2$
$a_1: b_2 b_1 b_0 b_3 b_4$	$b_1: c_0 c_1 c_2 c_3 c_4$	$c_1: a_3 a_2 a_0 a_1 a_4$
$a_2: b_2 b_0 b_1 b_3 b_4$	$b_2: c_0 c_2 c_3 c_1 c_4$	$c_2: a_3 a_0 a_1 a_4 a_2$
$a_3: b_3 b_1 b_4 b_2 b_0$	$b_3: c_0 c_3 c_2 c_1 c_4$	$c_3: a_4 a_2 a_0 a_1 a_3$
$a_4: b_3 b_2 b_4 b_1 b_0$	$b_4: c_4 c_1 c_0 c_2 c_3$	$c_4: a_2 a_0 a_3 a_1 a_4$

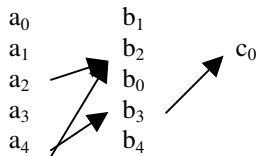
Nuevamente, se parte del par de conjuntos AB y se tienen las siguientes asignaciones y posibles bloqueos:



Ahora, al intentar asignar los primeros c de las listas de preferencias de b_2 y b_3 , nos encontramos con la existencia de conflicto, ya que ambos prefieren a c_0 .

Analicemos la asignación para el par (a_1, b_2) .

Primero b_2 elige a c_0 , pero como c_0 podría formar terna bloqueante junto con (a_4, b_3) , es descartado ya que podría generarse lo siguiente:



Para romper esa posible terna bloqueante (a_4, b_3, c_0) , debemos ver que no se cierre el ciclo $a_4 \rightarrow b_3 \rightarrow c_0 \rightarrow a_4$, es decir c_0 deberá preferir a su pareja más que a a_4 . Como $a_4 >_{c_0} a_1$, c_0 no puede ser asignado a b_2 .

Siguiendo con este criterio, buscamos el primer c_k de la lista de preferencias de b_2 tal que $a_1 >_{c_k} a_i$ para todo a_i participante de una posible terna bloqueante que no contenga a b_2 (ya que b_2 no puede formar terna bloqueante junto con la pareja que le es asignada), en nuestro caso el único a_i posible es a_4 .

c_2 es el candidato, ya que $a_1 >_{c_2} a_4$.

Además, (a_2, b_2, c_0) y (a_4, b_2, c_0) son posibles ternas bloqueantes, por lo cual c_0 deberá asignarse al más preferido por él entre a_2 y a_4 o algún a_i más preferido que ambos.

Veamos ahora qué pasa con b_3 .

c_0 no puede ser asignado ya que $a_4 >_{c_0} a_3$.

Si c_3 fuera asignado a b_3 , podría formar terna bloqueante con (a_2, b_2) o (a_4, b_2) , pero b_2 ya tiene un c_k asignado más preferido que c_3 ($c_2 >_{b_2} c_3$). Asignamos entonces c_3 a b_3 .

Los c que fueron descartados forman al menos una posible terna bloqueante. En nuestro caso sólo c_0 forma la posible terna bloqueante (a_4, b_3, c_0) .

Por lo tanto, ahora que a todos los b que estaban marcados les fueron asignados c, debemos desbloquear las posibles ternas bloqueantes, las cuales son (a_2, b_2, c_0) , (a_4, b_2, c_0) y (a_4, b_3, c_0) . Es decir, debemos asignar c_0 al a más preferido entre a_2 y a_4 o a algún a más preferido que ambos. Ya que $a_4 >_{c_0} a_2$ y a_4 está aún sin c asignado asignamos c_0 a a_4 .

Luego, construimos el siguiente frame de matching estables:

FM(I):	a_0	b_1	-
	a_1	b_2	c_2
	a_2	b_0	-
	a_3	b_3	c_3
	a_4	b_4	-

Cualquier matching perteneciente a la familia generada por FM será matching estable.

Veamos ahora el algoritmo que construye un frame de matchings estables siguiendo las ideas mostradas en los ejemplos anteriores.

Algoritmo 4.3

Sea $I=(A,B,C;P)$ una instancia de LUC con las siguientes relaciones de preferencia:
 $A \rightarrow B$, $B \rightarrow C$ y $C \rightarrow A$.

- 1) Para cada par de conjuntos XY (con $XY \in \{AB, BC, CA\}$)
 - 1.1) Buscar un matching de dos dimensiones entre los conjuntos X e Y tal que, mediante alguna heurística, minimice la sumatoria de las posiciones de los agentes elegidos.
Proponemos las siguientes heurísticas:
 - i) Para cada $x \in X$ asignar el y_j más preferido tal que no haya sido asignado.
 - ii) Recorrer las listas de preferencias de X por columna e ir asignando los y que estén sin asignar. Es decir,
Para $0 \leq i \leq n$ (es decir, para cada posición de las listas de preferencias)
Para cada $x \in X$ sin asignar
Si y_j tal que $\text{Pos}(x, (x, y_j, z)) = i$ está sin ser asignado
Asignar y_j a x , es decir $M_Y(x) = y_j$
 - 1.2) Para cada $x \in X$, y cada $y \in Y$ tal que $\text{Pos}(x, (x, y, z)) < \text{Pos}(x, (x, M_Y(x), z))$
 - 1.2.1) Marcar y .
 - 1.2.2) Guardar el par (x, y) como posible participante de posibles ternas bloqueantes.
 $\text{PosParesTB} = \text{PosParesTB} \cup \{(x, y)\}$
- 2) Sea XY el par de conjuntos del paso 1 tal que Y tiene la menor cantidad de agentes marcados. Elegir al par XY y al conjunto X como conjunto de partida.
- 3) Sea XY el par de conjuntos elegido en el paso 2. Sea Z el conjunto restante.
Para cada $y_j \in Y$ tal que esté marcado, asignar el primer z_k de su lista si éste está sin asignar.
Es decir, $M_Z(y_j) = z_k$ si z_k está sin asignar y $\text{Pos}(y_j, (x, y_j, z_k)) = 0$
- 4) Si para cada $y_j \in Y$ tal que esté marcado, existe z_k asignado, obtener el frame resultante de la siguiente manera:
 $\{(x, M_Y(x), -) / M_Y(x) \text{ está sin marcar}\} \cup \{(x, M_Y(x), M_Z(M_Y(x))) / M_Y(x) \text{ está marcado}\}$
- 5) Si algún y_j marcado no resultó con z_k asignado
 - 5.1) Desasignar las asignaciones del paso 3
 - 5.2) Para cada par $(x, M_Y(x))$ obtenido en el paso 1.1 tal que $M_Y(x)$ esté marcado
 - 5.2.1) Buscar el primer $z_k \in Z$ según el orden de la lista de preferencias de $M_Y(x)$ tal que no esté asignado y que no genere posibles ternas bloqueantes.
 $x \succ_{z_k} x' \quad \forall x' \in \{x / (\exists y) (y \neq M_Y(x) \text{ y } (x, y) \in \text{PosParesTB} \text{ y } (M_Z(y) \text{ está sin asignar o } z_k \succ_y M_Z(y))\}$
 - 5.2.2) Si existe z_k ,
 - 5.2.2.1) Asignar z_k a x , es decir, $M_Z(x) = z_k$

5.2.2.2) Eliminar como posibles ternas bloqueantes todas las ternas que contengan a z_k
 $PosTB = PosTB - \{(x_i, y_j, z_k)\}$

5.2.2.3) Guardar como posibles ternas bloqueantes todas las ternas de la forma $(x_i, M_Y(x), z)$ con
 i) $(x_i, M_Y(x)) \in PosParesTB$
 ii) $z >_{M_Y(x)} z_k$
 $PosTB = PosTB \cup \{(x_i, M_Y(x), z) / (x_i, M_Y(x)) \in PosParesTB \text{ y } z >_{M_Y(x)} z_k\}$

5.2.3) Si no existe z_k , entonces no pudo encontrarse frame de matchings estables.

6) Para cada $z_k \in \{z / (\exists x)(\exists y) (x, y, z) \in PosTB\}$

6.1) Buscar $x_i \in X$ tal que no genere ternas bloqueantes, es decir
 $Pos(z_k, (x_i, y, z_k)) \leq Pos(z_k, (x', y, z_k)) \forall x' \in \{x / (\exists y) (x', y, z_k) \in PosTB\}$

6.2) Si se encontró x_i en el paso anterior:

6.2.1) Asignar z_k a x_i , es decir, $M_Z(x_i) = z_k$

6.2.2) Eliminar todas las posibles ternas bloqueantes que contengan a z_k
 $PosTB = PosTB - \{(x, y, z_k)\}$

7) Si $PosTB$ no es vacío, no se pudo obtener frame de matching estables

8) Si $PosTB$ es vacío, obtener el frame resultante de la siguiente manera:

$$FM = \{(x, M_Y(x), M_Z(x)) / M_Y(x) \text{ está sin marcar y } M_Z(x) \text{ está asignado}\} \cup \\ \{(x, M_Y(x), -) / M_Y(x) \text{ está sin marcar y } M_Z(x) \text{ no está asignado}\} \cup \\ \{(x, M_Y(x), M_Z(M_Y(x))) / M_Y(x) \text{ está marcado}\}$$

El frame así obtenido es un frame de matching estables.

Observación: Una pequeña variante de este algoritmo que aumentaría las posibilidades de encontrar un matching estable, sería la de repetir los pasos 3 al 8 para los otros dos pares de conjuntos posibles. Otra variante consiste en plantear una nueva heurística para resolver el paso 1.1, en donde se buscaba minimizar la sumatoria de las posiciones de los agentes elegidos para construir un matching inicial entre dos conjuntos.

Teorema 4.4

El Algoritmo 4.3 es correcto.

Demostración

Demostración de correctitud

Debemos probar que la solución que el algoritmo genera es un frame de matchings estables. Para probar que esto, debemos ver:

- a) El frame obtenido no tiene elementos repetidos, es decir, $\forall (x, y, z) \in FM$:
- i) $x \neq x' \forall x' \in \{x'' / (\exists y)(\exists z) (x'', y, z) \in FM \text{ y } x' \neq x''\}$
 - ii) $y \neq y' \forall y' \in \{y'' / (\exists x)(\exists z) (x, y'', z) \in FM \text{ y } y' \neq y''\}$

iii) z está sin asignar (-) o $z \neq z' \forall z' \in \{z'' / (\exists x)(\exists y) (x,y,z'') \in FM \text{ y } z' \neq z''\}$

b) El frame obtenido es un frame de matchings estables, es decir, para cualquier $z \in Z$ con los que se completan los valores que estén sin asignar, no deben existir ternas bloqueantes.

Sabemos que las secuencias de pasos posibles que arriban a solución son :

- Caso 1: Pasos 1, 2, 3, 4.
- Caso 2: Pasos 1, 2, 3, 5.2.1, 5.2.2, 6, 8.

Probar que el frame obtenido no tiene elementos repetidos.

i) $x \neq x' \forall x' \in \{x'' / (\exists y)(\exists z) (x'',y,z) \in FM \text{ y } x' \neq x''\}$

Los x son todos diferentes ya que se parte de todos los agentes de ese conjunto, los cuales no son reasignados en todo el algoritmo.

ii) $y \neq y' \forall y' \in \{y'' / (\exists x)(\exists z) (x,y'',z) \in FM \text{ y } y' \neq y''\}$

En el paso 1 del algoritmo se asignan los agentes y de Y considerando que los mismos no hayan sido ya asignados, con lo cual como resultado de dicho paso resultarán todos los y diferentes, es decir, resulta un matching de dos dimensiones entre los conjuntos X e Y . Además, los mismos no son reasignados en todo el algoritmo.

iii) z está sin asignar (-) o $z \neq z' \forall z' \in \{z'' / (\exists x)(\exists y) (x,y,z'') \in FM \text{ y } z' \neq z''\}$

Los z pueden ser asignados en los pasos 3, 5.2.2.1 y 6.2.1.

- En los pasos 3 y 5.2.2.1 se asignan los z verificando previamente que no hayan sido asignados.
- En el paso 6.2.1 se asignan los z que pertenecen al conjunto de posibles ternas bloqueantes. Los mismos aún no han sido asignados, ya que al asignarse se eliminan todas sus posibles ternas bloqueantes (pasos 5.2.2.2 y 6.2.2).

a) Demostrar que el frame obtenido es un frame de matching estables.

Sea M un matching perteneciente a la familia de matchings estables determinada por el frame FM resultante del algoritmo. Supongamos, sin pérdida de generalidad, que el frame fue obtenido seleccionando el par de conjuntos AB como partida (paso 2)

Supongamos que existe una terna (a,b,c) no perteneciente a M tal que es bloqueante.

Entonces:

- i) $b >_a M_B(a)$ y
- ii) $c >_b M_C(b)$ y
- iii) $a >_c M_A(c)$

Como resultado de los pasos 1 y 2, tenemos un matching M_{AB} , en el cual existe un subconjunto $A' \subset A$ cuyos agentes no están con su mejor asignación. Por otro lado existe un subconjunto $B' \subset B$ cuyos miembros son para algún $a_i \in A'$, más preferidos que su pareja que le ha sido asignada.

Con tales asignaciones, los agentes $a_i \in (A-A')$ no pueden formar parte de ninguna terna bloqueante, ya que tienen asignado el primer b_j de su lista de preferencias.

Asimismo, los agentes $b_j \in (B-B')$ no pueden formar parte de ninguna terna bloqueante, ya que ningún $a_i \in A$ lo prefiere más que a su pareja actual.

Luego, los únicos candidatos a formar ternas bloqueantes son los pares (a_i, b_j) con $a_i \in A'$ y $b_j \in B'$, donde $Pos(a_i, (a_i, b_j, c)) < Pos(a_i, (a_i, M_B(a_i), c))$ (paso 1.2)

- Caso 1: Pasos 1, 2, 3, 4

En el paso 3 al elegir para cada $b_j \in B'$ un $c_k \in C$ tal que sea el más preferido por b_j eliminamos la posibilidad que b_j pertenezca a una terna bloqueante (es decir, no existe $c \in C$ tal que $c >_{b_j} M_C(b_j)$). Esto contradice el punto ii de nuestra hipótesis.

Luego, no existe terna bloqueante.

- Caso 2: Pasos 1, 2, 3, 5.2.1, 5.2.2, 6, 8

Como $b >_a M_B(a)$, significa que b fue marcado en el paso 1.2.2 y (a,b) se guardó como posible par de una terna bloqueante.

Como b fue marcado y $c >_b M_C(b)$, quiere decir que $M_C(b)$ no fue asignado a b en el paso 3. Es decir, el par $(M_A(b), b)$ fue analizado en el paso 5.2.

En 5.2.1 se encontró un c_k (caso contrario, el algoritmo no hubiese encontrado solución). Posteriormente, en 5.2.3, se guardó la terna (a,b,c) como posible terna bloqueante, ya que (a,b) estaba como posible par de terna bloqueante y $c >_b M_C(b)$.

Si (a,b,c) no se elimina como posible terna bloqueante, el algoritmo terminaría sin solución.

(a,b,c) puede ser eliminado como posible terna bloqueante en los pasos 5.2.2.2 ó 6.2.2.

- Para ser eliminada en 5.2.2.2, c tuvo que ser asignado a un a_i tal que $a_i >_c a'$ (con algún b' tal que (a',b',c) estuviera guardada como posible terna bloqueante). En particular, $a_i >_c a$. Esto significa que $M_A(c) >_c a$. Absurdo, contradice hipótesis.
- Para ser eliminada en 6.2.2, c tuvo que ser asignado a un a_i tal que $a_i >_c a'$ o $a_i = a'$ (con algún b' tal que (a',b',c) estuviera guardada como posible terna bloqueante). En particular, $a_i >_c a$ o $a_i = a$. Esto significa que $M_A(c) >_c a$ o $M_A(c) = a$. Absurdo, contradice hipótesis

Luego, no existe terna (a,b,c) tal que bloquee al matching M .

Luego, el frame obtenido es un frame de matchings estables. ♦

A continuación mostraremos, mediante un ejemplo, la no completitud del Algoritmo 4.3.

Ejemplo 4.10

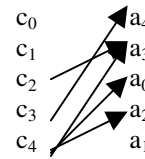
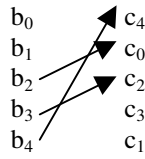
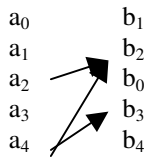
Sea $I=(A,B,C;P)$ con $A=\{a_0, a_1, a_2\}$, $B=\{b_0, b_1, b_2\}$, $C=\{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:	A	B	C
a_0 :	$b_1 b_2 b_0 b_3 b_4$	$b_0: c_4 c_1 c_3 c_2 c_0$	$c_0: a_4 a_3 a_0 a_1 a_2$
a_1 :	$b_2 b_1 b_0 b_3 b_4$	$b_1: c_0 c_1 c_2 c_3 c_4$	$c_1: a_3 a_2 a_0 a_1 a_4$
a_2 :	$b_2 b_0 b_1 b_3 b_4$	$b_2: c_0 c_2 c_3 c_1 c_4$	$c_2: a_3 a_0 a_4 a_1 a_2$
a_3 :	$b_3 b_1 b_4 b_2 b_0$	$b_3: c_0 c_3 c_2 c_1 c_4$	$c_3: a_4 a_2 a_0 a_1 a_3$
a_4 :	$b_3 b_2 b_4 b_1 b_0$	$b_4: c_4 c_1 c_0 c_2 c_3$	$c_4: a_2 a_0 a_3 a_4 a_1$

- 1) Buscamos, para cada par de conjuntos un matching de dos dimensiones usando la primer heurística:

a_0	b_1	b_0	c_4	c_0	a_4
a_1	b_2	b_1	c_0	c_1	a_3
a_2	b_0	b_2	c_2	c_2	a_0
a_3	b_3	b_3	c_3	c_3	a_2
a_4	b_4	b_4	c_1	c_4	a_1

y marcamos los pares posibles generadores de ternas bloqueantes en cada caso:



- 2) Seleccionamos el par de conjuntos AB como partida.
- 3) $M_C(b_2) = c_0$
 $M_C(b_3)$ no puede ser asignado, ya que c_0 está asignado a b_2 .
- 4) b_3 está sin c asignado \Rightarrow no se puede determinar frame.

5) Para el par (a_1, b_2) intentamos asignar c .

5.2.1) Recorremos la lista de preferencias de b_2 .

Analizamos c_0

$a_4 >_{c_0} a_1$ donde $(a_4, b_3) \in \text{PosParesTB}$ y $M_C(b_3)$ está sin asignar \Rightarrow se descarta c_0

Analizamos c_2

$a_4 >_{c_2} a_1$ donde $(a_4, b_3) \in \text{PosParesTB}$ y $M_C(b_3)$ está sin asignar \Rightarrow se descarta c_0

Analizamos c_3

$a_4 >_{c_3} a_1$ donde $(a_4, b_3) \in \text{PosParesTB}$ y $M_C(b_3)$ está sin asignar \Rightarrow se descarta c_0

Analizamos c_1

$a_4 >_{c_1} a_1$ donde $(a_4, b_3) \in \text{PosParesTB}$ y $M_C(b_3)$ está sin asignar \Rightarrow se descarta c_0

Analizamos c_4

$a_4 >_{c_4} a_1$ donde $(a_4, b_3) \in \text{PosParesTB}$ y $M_C(b_3)$ está sin asignar \Rightarrow se descarta c_0

5.2.3) No encontramos c para $b_2 \Rightarrow$ no pudo encontrarse matching estable

El algoritmo termina sin encontrar solución. Sin embargo el matching

$M = \{(a_0, b_2, c_2), (a_1, b_4, c_3), (a_2, b_0, c_4), (a_3, b_1, c_1), (a_4, b_3, c_0)\}$ es estable.

Complejidad e Implementación

Para el análisis de la complejidad del algoritmo anterior, suponemos la utilización de las matrices de ranking en los casos necesarios, tal como fue explicado para el análisis de complejidad del algoritmo de LUM.

Sea una instancia $I=(A,B,C;P)$ del problema de LUC, con $A \rightarrow B$, $B \rightarrow C$ y $C \rightarrow A$.

Una matriz de ranking para el conjunto A es una matriz R_A de dimensiones $n \times n$ en la cual $R_A(a, b) = k$ si b está en la k -ésima posición en la lista de preferencias de a (con $a \in A$ y $b \in B$).

De la misma forma, se puede definir R_B y R_C las correspondientes matrices de ranking para los conjuntos B y C .

Observemos que para el algoritmo propuesto, es necesario implementar las matrices para los tres conjuntos A , B y C .

Veamos la complejidad para cada paso del algoritmo:

1) Realiza un ciclo 3 veces, dentro del cual:

1.1) Analizando la primera heurística, vemos que para cada $x_i \in X$ (es decir, n veces) deberá recorrer, en el peor caso, $i-1$ elementos de la lista de preferencias, ya que $i-1$ es justamente la cantidad de agentes y que han sido asignados hasta el momento. Esto nos da $\sum_{1 \leq i \leq n} (i-1) = n(n-1)/2 \equiv O(n^2)$

1.2) Si analizamos la segunda heurística, vemos que para cada elemento de la lista de preferencias (es decir, n veces), deberemos recorrer cada agente x para verificar si tiene o no un y asignado. $\equiv O(n^2)$

Orden del paso 1: $3 \cdot n^2 \equiv O(n^2)$

- 2) Este paso en el peor caso tendría complejidad del orden de n , si tuviera que recorrer cada $y \in Y$ para verificar si está marcado. Obviamente, se puede implementar con un acumulador en el paso 1 que evite este recorrido.
- 3) Simplemente recorre cada $y \in Y$ que esté marcado (a lo sumo n), y para cada uno, analiza el primer elemento de su lista de preferencias. $\equiv O(n)$
- 4) Recorre los $y \in Y$ que estén marcados (a lo sumo n) $\equiv O(n)$
- 5) 5.1) Recorre cada $x \in X$ para desasignarlos si tuvieran y asignados. $O(n)$
 5.2) Recorre cada par asignado ($O(n)$) y para cada uno
 - 5.2.1) En el peor caso, recorrerá toda la lista de preferencias. $O(n)$
 - 5.2.2) En 5.2.2.2) recorre todas las ternas bloqueantes generadas hasta el momento. Sabiendo que el total de ternas posibles es n^3 , este paso tiene una complejidad del orden de n^3 .
 En 5.2.2.3) se recorren todos los pares de posibles ternas bloqueantes generados hasta el momento ($O(n^2)$) y, para cada uno se verifican las preferencias del correspondiente y (utilizando la matriz de ranking, esta comparación es directa)
 Orden del paso 5: $n + n*(n+n^3+n^2) \equiv O(n^4)$
- 6) En lugar de recorrer todas las posibles ternas bloqueantes para detectar los z pertenecientes a alguna de ellas, se podrían guardar dichos z_s en los pasos anteriores, bajando la complejidad de $O(n^3)$ a $O(n)$.
 Para cada z encontrado:
 En los pasos 6.1) y 6.2) recorre las posibles ternas bloqueantes. $\equiv O(n^3)$.
 Orden del paso 6: $n*(n^3+n^3) \equiv O(n^4)$

Los pasos 7) y 8) son directos.

Complejidad del algoritmo: $O(n^2 + n + n + n^4 + n^4) \equiv O(n^4)$

Observemos que el Algoritmo 4.3 puede modificarse para encontrar un matching fuertemente estable, aunque en este caso las posibles ternas bloqueantes que pueden generarse en cada paso aumenta considerablemente, ya que un agente podría ser parte de una terna bloqueante junto con su propia pareja. Veamos cómo debería modificarse el algoritmo para obtener un matching fuertemente estable.

Conservaremos la idea de guardar como posibles pares de ternas bloqueantes sólo aquellos casos en los que se cumple la relación $>$ estricta.

- Los pasos 1 y 2 del algoritmo no varían.
- Los pasos 3 y 4 se eliminan, y consecuentemente el paso 5.1, ya que no son válidos.

Veamos en el caso del Ejemplo 4.8, que el frame obtenido podría generar matchings que no sean fuertemente estables.

A	B	C
$a_0: b_1 b_2 b_0 b_3 b_4$	$b_0: c_4 c_1 c_3 c_2 c_0$	$c_0: a_4 a_3 a_0 a_1 a_2$
$b_1: c_0 c_1 c_2 c_3 c_4$	$c_1: a_3 a_2 a_0 a_1 a_4$	$a_1: b_2 b_1 b_0 b_3 b_4$
$c_2: a_3 a_0 a_2 a_4 a_1$	$a_2: b_2 b_0 b_1 b_3 b_4$	$b_2: c_0 c_2 c_3 c_1 c_4$
$a_3: b_3 b_1 b_4 b_2 b_0$	$b_3: c_2 c_3 c_0 c_1 c_4$	$c_3: a_4 a_2 a_0 a_1 a_3$
$a_4: b_3 b_2 b_4 b_1 b_0$	$b_4: c_4 c_1 c_0 c_2 c_3$	$c_4: a_2 a_0 a_3 a_1 a_4$
FM(I): a_0	b_1	-
a_1	b_2	c_0
a_2	b_0	-
a_3	b_3	c_2
a_4	b_4	-

Sea $M = \{(a_0, b_1, c_1), (a_1, b_2, c_0), (a_2, b_0, c_3), (a_3, b_3, c_2), (a_4, b_4, c_4)\}$ perteneciente a la familia generada por FM(I). La terna (a_0, b_1, c_4) es bloqueante para M , ya que $a_0 = M_A(b_1)$, $c_4 >_{b_1} M_C(b_1)$ y $a_0 >_{c_4} M_A(c_4)$

- El paso 5.2.1 deberá modificarse para que al buscar las posibles ternas bloqueantes, contemple también el par que está analizando. Es decir, se elimina la condición $y \neq M_Y(x)$.
- El paso 5.2.2 no cubre todas las posibles ternas bloqueantes que se generan. Para cada par $(x, M_Y(x))$ analizado se deberán agregar, además, las ternas $(x, M_Y(x), z)$ para cada z descartado.
- Se agrega a continuación un paso 5.3, en el cual para cada par $(x, M_Y(x))$ con $M_Y(x)$ sin marcar (es decir, los pares no analizados en el paso anterior), se intentará asignar un z_k tal que no esté asignado y no genere posibles ternas bloqueantes ni confirme las ya generadas. Es decir, para cada $(x, M_Y(x))$ con $M_Y(x)$ sin marcar, se recorrerá la lista de preferencias de $M_Y(x)$ hasta encontrar un z_k no asignado tal que $x \succ_{z_k} x' \forall x' \neq x$ tal que $M_Z(x')$ es nulo o $z_k \succ_{M_Y(x)} M_Z(x')$ y $x \succ_{z_k} x' \forall x' \in \{x / (\exists y) (x, y, z_k) \in \text{PosTB}\}$
Si se pueden encontrar tales z_k , se asignan los mismos a x , y se guardan como posibles ternas bloqueantes los $(x, M_Y(x), z)$ para cada z descartado.
- El resto del algoritmo no varía.

Para demostrar la correctitud del algoritmo modificado, volvamos al Teorema 4.4:

- Notemos que ahora el único caso posible de arribar a la solución es siguiendo los pasos 1, 2, 5.2.1, 5.2.2, 5.3, 6, 8
- a) La demostración de que el frame obtenido no tiene elementos repetidos, sigue siendo válida por el hecho de que las nuevas asignaciones realizadas son siempre considerando que el agente a ser asignado no lo esté hasta el momento.
- b) Demostrar que el frame obtenido es un frame de matchings fuertemente estables.
- Supongamos que existe una terna (a, b, c) no perteneciente a M tal que es bloqueante $\Rightarrow b \succeq_a M_B(a)$ y $c \succeq_b M_C(b)$ y $a \succeq_c M_A(c) \Rightarrow$
- $b = M_B(a)$ y $c \succ_b M_C(b)$ y $a \succ_c M_A(c)$ o
 - $b \succ_a M_B(a)$ y $c = M_C(b)$ y $a \succ_c M_A(c)$ o
 - $b \succ_a M_B(a)$ y $c \succ_b M_C(b)$ y $a = M_A(c)$ o
 - $b \succ_a M_B(a)$ y $c \succ_b M_C(b)$ y $a \succ_c M_A(c)$

Veamos cada caso, suponiendo que el algoritmo comienza por el conjunto A :

- $b = M_B(a)$ y $c \succ_b M_C(b)$ y $a \succ_c M_A(c) \Rightarrow c \succ_{M_B(a)} M_C(a)$ y $a \succ_c M_A(c)$.
Como $b = M_B(a)$, significa que b no fue marcado en el paso 1, con lo cual el par $(a, M_B(a))$ fue analizado en 5.3. En este paso se busca un c tal que no cumpla $c \succ_{M_B(a)} M_C(a)$ y $a \succ_c M_A(c)$
- $b \succ_a M_B(a)$ y $c = M_C(b)$ y $a \succ_c M_A(c)$
El par (a, b) fue analizado en 5.2. El paso 5.2.1 se modificó para que contemple el caso $c = M_C(b)$, generando la correspondiente posible terna bloqueante, la que será eliminada, de ser posible, en el paso 6. En caso contrario, el algoritmo no generará solución.
- $b \succ_a M_B(a)$ y $c \succ_b M_C(b)$ y $a = M_A(c)$
 - Si $M_B(a)$ estaba marcada, el par $(a, M_B(a))$ fue analizado en el paso 5.2. Si hasta ese momento b no tuviera c_k asignado, entonces el c no podría haber sido asignado a a . Si lo tuviera, como $c \succ_b M_C(b)$, tampoco hubiera sido asignado.
 - Si $M_B(a)$ estaba sin marcar, el par $(a, M_B(a))$ fue analizado en el paso 5.3, en cuyo caso se validan las mismas condiciones anteriores.
- $b \succ_a M_B(a)$ y $c \succ_b M_C(b)$ y $a \succ_c M_A(c)$. Aplica la demostración del teorema para matching débilmente estable. ♦

4.3.3 El conjunto de todos los Matchings Estables

Como se vio en el Capítulo 3, desde el punto de vista de la programación lineal entera, podemos definir la estructura del conjunto de todos los matchings estables, como los puntos enteros factibles en el poliedro

definido por un sistema de inecuaciones lineales o restricciones. A continuación damos la formulación lineal entera para representar el conjunto de todos los matchings estables en instancias de LUC, reemplazando la restricción que garantiza estabilidad por una nueva que representa específicamente a este tipo de instancias.

$$\text{Sea } x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{i > a \\ c' \in C}} x_{a,i,c'} + \sum_{\substack{j > b \\ a' \in A}} x_{a',b,j} + \sum_{\substack{k > c \\ b' \in B}} x_{k,b',c} + x_{a,b,c} \cdot 1, \quad \forall (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\} \quad (a,b,c) \in A \times B \times C \quad (5)$$

Para definir el conjunto de matchings fuertemente estable, la restricción 4 es reemplazada por:

$$\sum_{\substack{i \geq a \\ c' \in C}} x_{a,i,c'} + \sum_{\substack{j \geq b \\ a' \in A}} x_{a',b,j} + \sum_{\substack{k \geq c \\ b' \in B}} x_{k,b',c} + x_{a,b,c} \cdot 1, \quad \forall (a,b,c) \in A \times B \times C \quad (4')$$

De la misma forma que para LUM, la relación de dominación entre los matchings estables (\geq_A), tal como fue definida en el Capítulo 3, no es una relación de orden, ya que no cumple con la propiedad antisimétrica.

Es decir, $M \geq_A M'$ y $M' \geq_A M$ no implican $M=M'$. Esto sucede porque la proyección sobre A y B de dos matchings estables M y M' puede ser la misma y, sin embargo, diferir la proyección sobre el conjunto C, con lo cual $M \geq_A M'$ y $M' \geq_A M$, pero $M \neq M'$.

Veamos esto con un ejemplo.

Ejemplo 4.11

Dada la instancia $I=(A,B,C;P)$ con $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:					
a_0 :	$b_1 \ b_2 \ b_0$	b_0 :	$c_0 \ c_1 \ c_2$	c_0 :	$a_1 \ a_2 \ a_0$
a_1 :	$b_0 \ b_2 \ b_1$	b_1 :	$c_0 \ c_2 \ c_1$	c_1 :	$a_2 \ a_1 \ a_0$
a_2 :	$b_1 \ b_0 \ b_2$	b_2 :	$c_0 \ c_2 \ c_1$	c_2 :	$a_0 \ a_1 \ a_2$

veamos cómo queda definida la relación de dominación definida entre todos los matching estables.

La instancia I tiene los siguientes matchings estables:

$$M_1 = \{(a_0,b_0,c_1), (a_1,b_1,c_0), (a_2,b_2,c_2)\}$$

- $M_2 = \{(a_0, b_0, c_2), (a_1, b_1, c_0), (a_2, b_2, c_1)\}$
- $M_3 = \{(a_0, b_1, c_1), (a_1, b_0, c_0), (a_2, b_2, c_2)\}$
- $M_4 = \{(a_0, b_1, c_2), (a_1, b_0, c_1), (a_2, b_2, c_0)\}$
- $M_5 = \{(a_0, b_1, c_2), (a_1, b_0, c_0), (a_2, b_2, c_1)\}$
- $M_6 = \{(a_0, b_2, c_2), (a_1, b_1, c_0), (a_2, b_0, c_1)\}$
- $M_7 = \{(a_0, b_0, c_1), (a_1, b_2, c_0), (a_2, b_1, c_2)\}$
- $M_8 = \{(a_0, b_0, c_2), (a_1, b_2, c_0), (a_2, b_1, c_1)\}$
- $M_9 = \{(a_0, b_1, c_2), (a_1, b_2, c_1), (a_2, b_0, c_0)\}$
- $M_{10} = \{(a_0, b_1, c_1), (a_1, b_2, c_2), (a_2, b_0, c_0)\}$
- $M_{11} = \{(a_0, b_1, c_2), (a_1, b_2, c_0), (a_2, b_0, c_1)\}$
- $M_{12} = \{(a_0, b_2, c_0), (a_1, b_0, c_0), (a_2, b_1, c_2)\}$
- $M_{14} = \{(a_0, b_2, c_2), (a_1, b_0, c_1), (a_2, b_1, c_0)\}$
- $M_{15} = \{(a_0, b_2, c_1), (a_1, b_0, c_2), (a_2, b_1, c_0)\}$
- $M_{16} = \{(a_0, b_2, c_2), (a_1, b_0, c_0), (a_2, b_1, c_1)\}$

El conjunto de todos los matchings estables forman las siguientes redes, de acuerdo al orden parcial definido entre ellos y agrupándose en clases de equivalencia.

Desde el conjunto A:

Desde el punto de vista del conjunto A, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

- $E_1 = \{M_1, M_2\}$
- $E_2 = \{M_3, M_4, M_5\}$
- $E_3 = \{M_6\}$
- $E_4 = \{M_7, M_8\}$
- $E_5 = \{M_9, M_{10}, M_{11}\}$
- $E_6 = \{M_{12}, M_{13}, M_{14}, M_{15}, M_{16}\}$

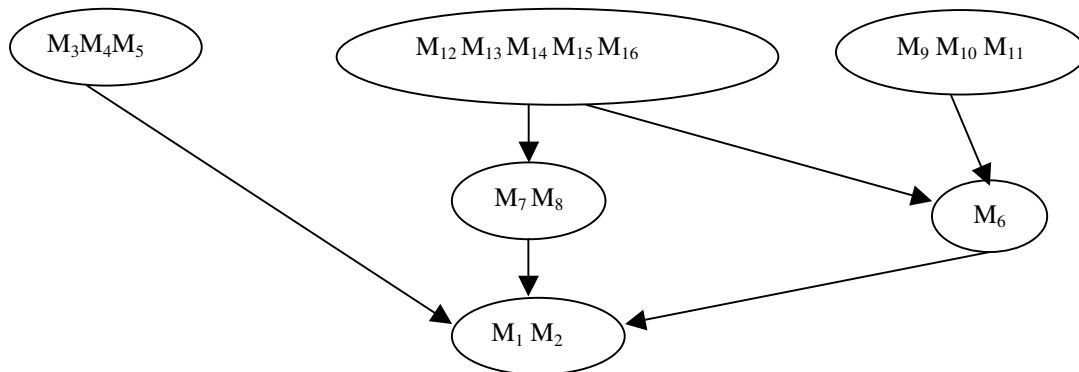


Figura 4.8 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.11 según el conjunto A

Desde el conjunto B:

Desde el punto de vista del conjunto B, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

- $E_1 = \{M_1, M_6, M_{14}\}$
- $E_2 = \{M_2, M_{15}\}$
- $E_3 = \{M_3, M_{10}, M_{16}\}$

$$E_4 = \{M_4, M_7, M_{11}, M_{12}\}$$

$$E_5 = \{M_5, M_9, M_{13}\}$$

$$E_6 = \{M_8\}$$

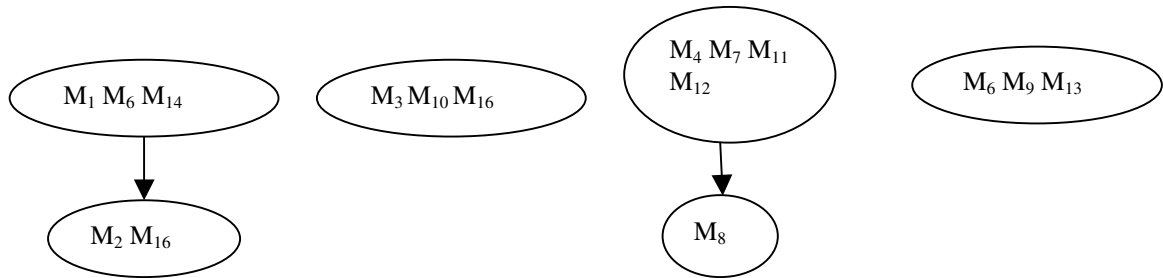


Figura 4.9 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.11 según el conjunto B

Desde el conjunto C:

Desde el punto de vista del conjunto C, los matchings estables de I se agrupan en las siguientes clases de equivalencia:

$$E_1 = \{M_1, M_3, M_7, M_{13}\}$$

$$E_2 = \{M_2, M_5, M_6, M_8, M_{11}, M_{16}\}$$

$$E_3 = \{M_4, M_9, M_{14}\}$$

$$E_4 = \{M_{10}, M_{15}\}$$

$$E_5 = \{M_{12}\}$$

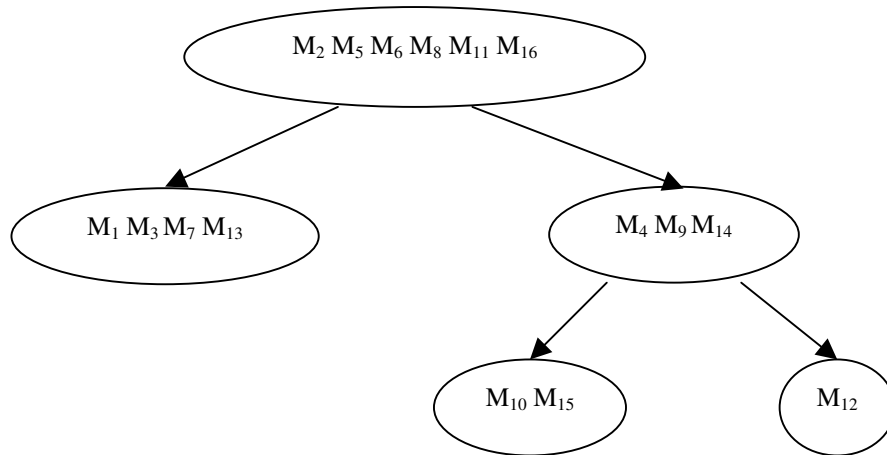


Figura 4.10 Relación de dominación entre todos los matchings estables para la instancia del Ejemplo 4.11 según el conjunto C.

Si consideramos la relación de dominación entre clases de equivalencia, tal como fue definida para LUM, ahora sí tendremos una relación de orden, es decir cumplirá las propiedades reflexiva, antisimétrica y transitiva.

- Reflexiva: $E \geq_A E$.
Trivial ya que $E=E$

- Antisimétrica: $E \geq_A E'$ y $E' \geq_A E \Rightarrow E = E'$
 $E \geq_A E'$ y $E' \geq_A E \Rightarrow M \geq_A M'$ y $M' \geq_A M \forall M \in E, \forall M' \in E' \Rightarrow M_{BC(a)} \geq_a M'_{BC(a)}$ y
 $M'_{BC(a)} \geq_a M_{BC(a)} \forall M \in E, \forall M' \in E', \forall a \in A \Rightarrow M_{BC(a)} = M'_{BC(a)} \forall M \in E, \forall M' \in E', \forall a \in A \Rightarrow M$ y
 M' están en la misma clase de equivalencia $\Rightarrow E = E'$
- Transitiva: $E \geq_A E'$ y $E' \geq_A E'' \Rightarrow E \geq_A E''$
 $E \geq_A E'$ y $E' \geq_A E'' \Rightarrow M \geq_A M'$ y $M' \geq_A M'' \forall M \in E, \forall M' \in E', \forall M'' \in E'' \Rightarrow M_{BC(a)} \geq_a M'_{BC(a)}$ y
 $M'_{BC(a)} \geq_a M''_{BC(a)} \forall M \in E, \forall M' \in E', \forall M'' \in E'', \forall a \in A \Rightarrow M_{BC(a)} \geq_a M''_{BC(a)} \forall M \in E, \forall M'' \in E'',$
 $\forall a \in A. \blacklozenge$

Los grafos representando esta nueva relación de dominación serán idénticos a los anteriores, en donde cada nodo representa indistintamente la clase de equivalencia o el conjunto de matchings pertenecientes a la misma.

Podemos observar que, si bien ahora existe una relación de orden, igualmente el conjunto de las clases de equivalencia no constituye un reticulado. Vimos en el Ejemplo 4.11 que, desde el conjunto A existe ínfimo pero no supremo (Figura 4.8), a diferencia del conjunto C, desde el cual existe supremo pero no ínfimo (Figura 4.10).

Capítulo 5: Matching Estable Tridimensional

sobre tres Conjuntos con Listas Simples

En este capítulo presentamos los resultados del análisis del problema de hallar un matching estable tridimensional sobre tres conjuntos con Listas de Preferencias Simples.

Comenzamos definiendo este nuevo problema, en donde se busca hallar un matching estable tridimensional sobre tres conjuntos disjuntos y donde cada agente tiene un nuevo tipo de listas de preferencias al que hemos denominado Listas Simples (LS).

Luego damos el algoritmo para encontrar un matching estable tridimensional en instancias de listas simples, y estudiamos la complejidad e implementación correspondientes, que a diferencia del problema estudiado en [NH/95], y cuyo estudio se hará en el Capítulo 6, veremos que aquí todas las instancias poseen matching estable, y dicha búsqueda tiene complejidad polinomial. También se demuestra que las listas de preferencias simples cumplen con la propiedad de consistencia.

Por último se analiza la estructura del conjunto de todos los matchings estables y la representación ordenada de los mismos.

5.1 Matching y Estabilidad

Como hemos visto en el Capítulo 3, en una instancia del problema de matching estable tridimensional con listas por pares, las listas de preferencias están formadas por pares de agentes de los dos conjuntos restantes, ordenados en forma estricta, donde el primer elemento del par corresponde a un conjunto y el segundo miembro al otro de los conjuntos. Para poder modelar un problema en donde las listas de preferencias de cada agente se expresen en forma independiente, definimos un nuevo tipo de lista de preferencias a las cuales denominamos Listas Simples (LS). Por ejemplo las listas de preferencias de $a_0 \in A$ sobre el conjunto B pueden ser $a_0: b_0, b_1$ y sobre el conjunto C $a_0: c_1, c_0$.

Como las dos listas de preferencias simples correspondientes a un agente son completamente independientes, los pares de agentes resultantes del producto cartesiano de ambos conjuntos no siempre pueden compararse. Por ejemplo, si las listas de preferencias de a_x para los agentes en B están dadas por la siguiente lista simple $a_x: b_0, b_1, b_2$ y las listas de preferencias de a_x para los agentes de C están dadas por la lista simple $a_x: c_0, c_1, c_2$, podemos ver que el par (b_0, c_1) es más preferido para a_x que el par (b_0, c_2) , pero no puede compararse con el par (b_1, c_0) , pues a_x prefiere b_0 a b_1 pero c_1 es menos preferido que c_0 .

Definimos entonces la relación de preferencia para una instancia de listas simples.

Se dice que un agente a *prefiere* al par (b_x, c_i) antes que al (b_y, c_h) , o dicho de otro modo, que $(b_x, c_i) >_a (b_y, c_h)$ si $(b_x \geq_a b_y \text{ y } c_i >_a c_h)$ o $(b_x >_a b_y \text{ y } c_i \geq_a c_h)$. De igual manera se define para $b \in B$ y $c \in C$ una relación $>_b$ y $>_c$ sobre el producto cartesiano $A \times C$ y $A \times B$ respectivamente.

Decimos que una terna (a, b, c) no perteneciente al matching M es una *terna bloqueante* de M si se cumplen las siguientes condiciones:

1. $(b \geq_a M_B(a) \text{ y } c >_a M_C(a))$ o $(b >_a M_B(a) \text{ y } c \geq_a M_C(a))$ y
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b))$ o $(a >_b M_A(b) \text{ y } c \geq_b M_C(b))$ y
3. $(a \geq_c M_A(c) \text{ y } b >_c M_B(c))$ o $(a >_c M_A(c) \text{ y } b \geq_c M_B(c))$

Ejemplo 5.1

Sea $I=(A, B, C; P)$ con $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ la instancia determinada por las siguientes listas de preferencias:

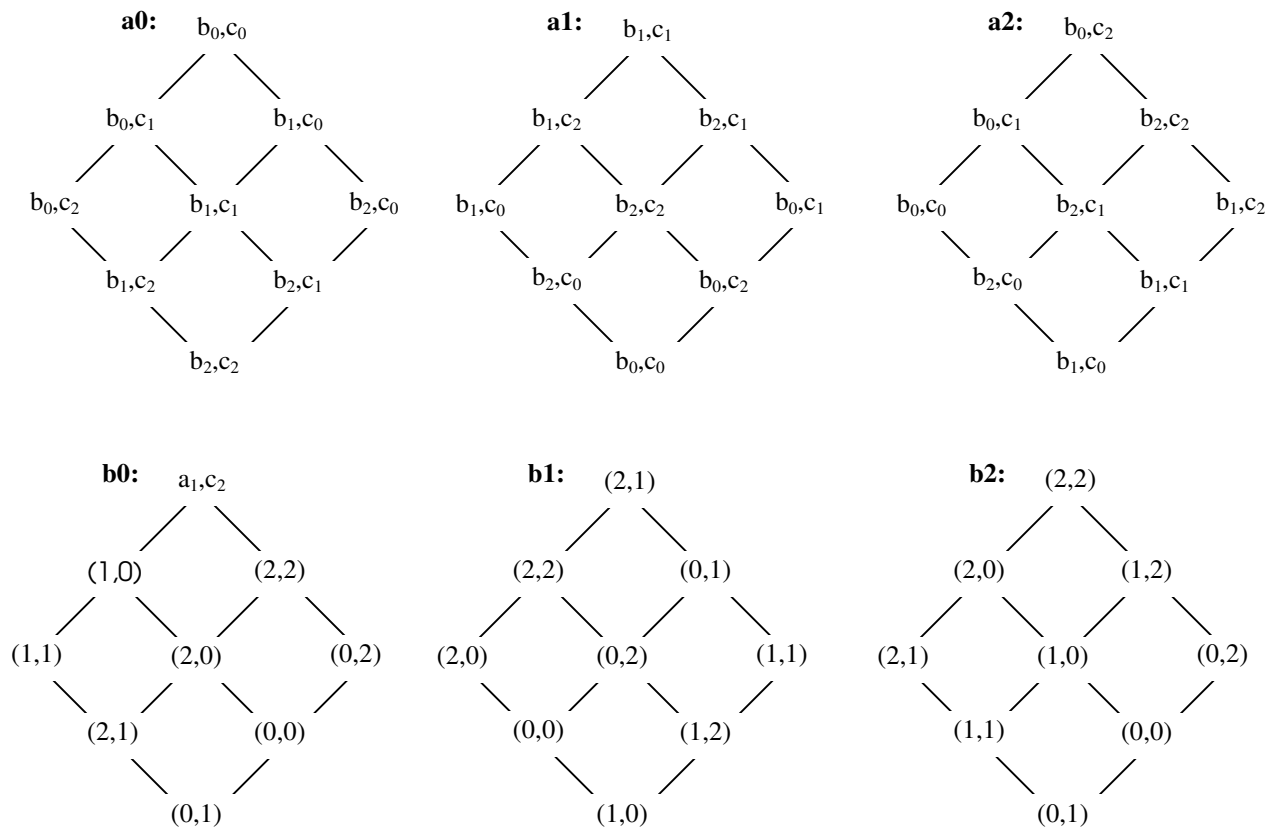
P:			
a ₀ :	b ₀ b ₁ b ₂ c ₀ c ₁ c ₂	b ₀ :	a ₁ a ₂ a ₀ c ₂ c ₀ c ₁
a ₁ :	b ₁ b ₂ b ₀ c ₁ c ₂ c ₀	b ₁ :	a ₂ a ₀ a ₁ c ₁ c ₂ c ₀
a ₂ :	b ₀ b ₂ b ₁ c ₂ c ₁ c ₀	b ₂ :	a ₂ a ₁ a ₀ c ₂ c ₀ c ₁
		c ₀ :	a ₁ a ₂ a ₀ b ₂ b ₀ b ₁
		c ₁ :	a ₁ a ₀ a ₂ b ₂ b ₀ b ₁
		c ₂ :	a ₁ a ₀ a ₂ b ₀ b ₂ b ₁

Para la instancia I, el matching $M = \{(a_0, b_1, c_1), (a_1, b_2, c_0), (a_2, b_0, c_2)\}$ es un matching estable. Mientras que $M' = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ no es un matching estable ya que es bloqueado por la terna (a_2, b_0, c_2) .

Como comentamos anteriormente, en una lista de preferencias simples no todos los pares de agentes son comparables entre sí, por lo que no existe un orden lineal entre los mismos. A continuación mostramos la representación gráfica del orden que tienen las listas de preferencias, en donde el par que se encuentra al tope del grafo corresponde al par más preferido, y el que se encuentra en la parte inferior, corresponde al menos preferido.

Ejemplo 5.2

A continuación damos la representación gráfica de las listas de cada uno de los elementos del Ejemplo 5.1.



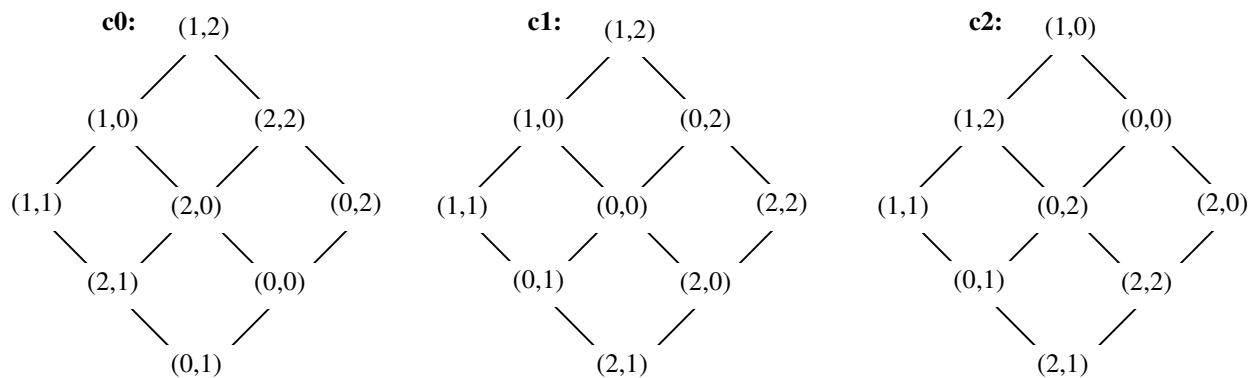


Figura 5.1 Gráfico correspondiente al orden de precedencia de las listas para cada agente del Ejemplo 5.1

Mencionamos anteriormente que uno de los problemas que quedan abiertos en [NH/95] es el de hallar un matching estable en instancias con listas consistentes. Si observamos las listas de preferencias en una instancia cualquiera del problema con listas simples, vemos que las mismas mantienen el orden de los agentes de un conjunto independientemente de las preferencias sobre el otro conjunto. Si bien el caso de listas simples no fue definido previamente, podemos ver con la siguiente propiedad que LS cubre una parte del problema de hallar un matching estable con listas consistentes. Veamos con la siguiente propiedad que las listas simples son consistentes.

Propiedad 5.1

Veamos que las listas de preferencias simples cumplen con la propiedad de consistencia, es decir que para una instancia con listas simples se cumple que:

1. $(a, b) >_c (a, b') \Rightarrow (a', b) >_c (a', b') \quad \forall a, a' \in A, \forall b, b' \in B, \forall c \in C$ y
2. $(a, b) >_c (a', b) \Rightarrow (a, b') >_c (a', b') \quad \forall a, a' \in A, \forall b, b' \in B, \forall c \in C$

Demostración

Vimos que en una instancia de listas simples $(a_1, b_1) >_c (a_2, b_2)$ si $(a_1 \geq_c a_2$ y $b_1 >_a b_2)$ o $(a_1 >_c a_2$ y $b_1 \geq_c b_2)$. Luego $(a, b) >_c (a, b') \Rightarrow (a \geq_c a$ y $b >_c b')$ o $(a >_c a$ y $b \geq_c b') \Rightarrow (b >_c b'$ y $a' \geq_c a')$ $\Rightarrow (b \geq_c b'$ y $a' >_c a')$ o $(b >_c b'$ y $a' \geq_c a')$ $\Rightarrow (a', b) >_c (a', b')$.

De la misma manera podemos demostrar que vale ii) $(a, b) >_c (a', b) \Rightarrow (a, b') >_c (a', b') \quad \forall a, a' \in A, \forall b, b' \in B, \forall c \in C$.

Luego las listas simples cumplen con la propiedad de consistencia. ♦

A continuación daremos una propiedad que nos indica que en instancias LS donde sea posible hallar un matching canónico, hallamos también un matching estable.

Propiedad 5.2

Todo matching canónico de MET3 con listas simples es estable.

Demostración

Supongamos que existe $t=(a,b,c)$ que bloquea al matching M canónico, es decir que

1. $(a, b, c) \notin M$ y
2. $(b \succeq_a M_B(a) \text{ y } c >_a M_C(a)) \text{ ó } (b >_a M_B(a) \text{ y } c \succeq_a M_C(a))$ y
3. $(a \succeq_b M_A(b) \text{ y } c >_b M_C(b)) \text{ ó } (a >_b M_A(b) \text{ y } c \succeq_b M_C(b))$ y
4. $(a \succeq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ ó } (a >_c M_A(c) \text{ y } b \succeq_c M_B(c))$

Como M es canónico vale que:

- i) $\forall t \in M, \text{Pos}(a, t) = 0$, ó
- ii) $\forall t \in M, \text{Pos}(b, t) = 0$, ó
- iii) $\forall t \in M, \text{Pos}(c, t) = 0$.

Supongamos, sin pérdida de generalidad que vale i).

Sea $t'=(a, M_B(a), M_C(a)) \in M$, entonces t' cumple que $\text{Pos}(a, t') = 0 \Leftrightarrow M_B(a) >_a b \forall b \in B, b \neq M_B(a)$ y $M_C(a) \forall c \in C, c \neq M_C(a)$. Absurdo pues contradice 2. \blacklozenge

5.2 Búsqueda de un Matching Estable

Damos a continuación un algoritmo que nos permite hallar un matching estable para instancias con listas simples. Este algoritmo resuelve el problema en tiempo polinomial sobre el tamaño de la instancia.

Algoritmo 5.1

- 1) Buscar un matching estable M' entre A y B a través del algoritmo de Gale y Shapley
- 2) Buscar un matching estable M'' entre A y C a través del algoritmo de Gale y Shapley
- 3) $M = M' \cup M''$

El matching M obtenido es estable.

Teorema

El algoritmo 5.1 es correcto y completo.

Demostración

Para que un matching sea estable, el mismo no debe tener ternas bloqueantes. Supongamos que el matching M no es estable en una instancia I de LS, entonces $\exists(a,b,c) \notin M$ bloqueante de M tal que:

1. $(b \succeq_a M_B(a) \text{ y } c >_a M_C(a)) \text{ o } (b >_a M_B(a) \text{ y } c \succeq_a M_C(a))$ y
2. $(a \succeq_b M_A(b) \text{ y } c >_b M_C(b)) \text{ o } (a >_b M_A(b) \text{ y } c \succeq_b M_C(b))$ y
3. $(a \succeq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } (a >_c M_A(c) \text{ y } b \succeq_c M_B(c))$

(aplicando propiedad distributiva \Leftrightarrow)

- 1.1- $(b \succeq_a M_B(a) \text{ o } b >_a M_B(a))$ y
- 1.2- $(b \succeq_a M_B(a) \text{ o } c \succeq_a M_C(a))$ y

- 1.3- $(c >_a M_C(a) \text{ o } b >_a M_B(a)) \text{ y}$
- 1.4- $(c >_a M_C(a) \text{ o } c \geq_a M_C(a)) \text{ y}$
- 2.1- $(a \geq_b M_A(b) \text{ o } a >_b M_A(b)) \text{ y}$
- 2.2- $(a \geq_b M_A(b) \text{ o } c \geq_b M_C(b)) \text{ y}$
- 2.3- $(c >_b M_C(b) \text{ o } a >_b M_A(b)) \text{ y}$
- 2.4- $(c >_b M_C(b) \text{ o } c \geq_b M_C(b)) \text{ y}$
- 3.1- $(a \geq_c M_A(a) \text{ o } a >_c M_A(c)) \text{ y}$
- 3.2- $(a \geq_c M_A(a) \text{ o } b \geq_c M_B(c)) \text{ y}$
- 3.3- $(b >_c M_B(c) \text{ o } a >_c M_A(c)) \text{ y}$
- 3.4- $(b >_c M_B(c) \text{ o } b \geq_c M_B(c))$

$\Leftrightarrow (1.1-(b \geq_a M_B(a) \text{ o } b >_a M_B(a)) \Leftrightarrow b \geq_a M_B(a)) \quad (\text{ídem para 1.4, 2.1, 2.4, 3.1 y 3.4})$

- 1.1- $b \geq_a M_B(a) \text{ y}$
- 1.2- $(b \geq_a M_B(a) \text{ o } c \geq_a M_C(a)) \text{ y}$
- 1.3- $(c >_a M_C(a) \text{ o } b >_a M_B(a)) \text{ y}$
- 1.4- $c \geq_a M_C(a) \text{ y}$
- 2.1- $a \geq_b M_A(b) \text{ y}$
- 2.2- $(a \geq_b M_A(b) \text{ o } c \geq_b M_C(b)) \text{ y}$
- 2.3- $(c >_b M_C(b) \text{ o } a >_b M_A(b)) \text{ y}$
- 2.4- $c \geq_b M_C(b) \text{ y}$
- 3.1- $a \geq_c M_A(a) \text{ y}$
- 3.2- $(a \geq_c M_A(a) \text{ o } b \geq_c M_B(c)) \text{ y}$
- 3.3- $(b >_c M_B(c) \text{ o } a >_c M_A(c)) \text{ y}$
- 3.4- $b \geq_c M_B(c)$

$\Leftrightarrow (1.2 \text{ y } 1.4 \Leftrightarrow 1.4 \text{ es decir que } ((b \geq_a M_B(a) \text{ o } c \geq_a M_C(a)) \text{ y } c \geq_a M_C(a)) \Leftrightarrow c \geq_a M_C(a)$
 $(\text{ídem para 2.2 y } 2.4 \Leftrightarrow 2.4, \text{ y para } 3.2 \text{ y } 3.4 \Leftrightarrow 3.4))$

$$\begin{aligned}
 & b \geq_a M_B(a) \text{ y } (c >_a M_C(a) \text{ o } b >_a M_B(a)) \text{ y } c \geq_a M_C(a) \text{ y} \\
 & a \geq_b M_A(b) \text{ y } (c >_b M_C(b) \text{ o } a >_b M_A(b)) \text{ y } c \geq_b M_C(b) \text{ y} \\
 & a \geq_c M_A(a) \text{ y } (b >_c M_B(c) \text{ o } a >_c M_A(c)) \text{ y } b \geq_c M_B(c) \quad (1)
 \end{aligned}$$

Caso 1: supongamos que la terna bloqueante (a,b,c) está en 3 filas distintas del matching M. Es decir que:

$$a \neq M_A(b) \neq M_A(c) \text{ y } b \neq M_B(a) \neq M_B(c) \text{ y } c \neq M_C(a) \neq M_C(b)$$

(De (1)) \Leftrightarrow

$$b >_a M_B(a) \text{ y } (c >_a M_C(a) \text{ o } b >_a M_B(a)) \text{ y } c >_a M_C(a) \text{ y}$$

$$a >_b M_A(b) \text{ y } (c >_b M_C(b) \text{ o } a >_b M_A(b)) \text{ y } c >_b M_C(b) \text{ y}$$

$$a >_c M_A(a) \text{ y } (b >_c M_B(c) \text{ o } a >_c M_A(c)) \text{ y } b >_c M_B(c)$$

\Leftrightarrow

$$(b >_a M_B(a) \text{ y } (c >_a M_C(a) \text{ o } b >_a M_B(a)) \Leftrightarrow b >_a M_B(a)) \quad \text{ídem para b y c.}$$

\Leftrightarrow

$$(b >_a M_B(a) \text{ y } c >_a M_C(a)) \text{ y } (a >_b M_A(b) \text{ y } c >_b M_C(b)) \text{ y } (a >_c M_A(a) \text{ y } b >_c M_B(c))$$

\Leftrightarrow

$$(b >_a M_B(a) \text{ y } a >_b M_A(b)) \text{ y } (c >_a M_C(a) \text{ y } a >_c M_A(a)) \text{ y } (c >_b M_C(b) \text{ y } b >_c M_B(c))$$

$$\Leftrightarrow (a,b) \text{ bloquea } M_{AB}, (b,c) \text{ bloquea } M_{BC} \text{ y } (a, c) \text{ bloquea } M_{AC}$$

Caso 2: Supongamos que la terna (a,b,c) se encuentra en sólo 2 filas de M. Analicemos el caso a y b se encuentran en la misma fila del matching mientras c se encuentra en otra. Es decir que:

$$a = M_A(b), b = M_B(a), c \neq M_C(a) \quad (\Rightarrow c \neq M_C(b), M_C(a) = M_C(b))$$

Reescribamos (1)

(1) \Leftrightarrow

$b =_a M_B(a)$ y $(c >_a M_C(a) \text{ o } b >_a M_B(a))$ y $c >_a M_C(a)$ y

$a =_b M_A(b)$ y $(c >_b M_C(b) \text{ o } a >_b M_A(b))$ y $c >_b M_C(b)$ y

$a >_c M_A(a)$ y $(b >_c M_B(c) \text{ o } a >_c M_A(c))$ y $b >_c M_B(c)$

\Leftrightarrow $(a =_b M_A(b) \text{ y } b =_a M_B(a) \text{ son parte de la hipótesis})$
 $(c >_a M_C(a) \text{ o } b >_a M_B(a)) \text{ y } c >_a M_C(a) \Rightarrow c >_a M_C(a)$ ídem para b y c.

$c >_a M_C(a)$ y

$c >_b M_C(b)$ y

$a >_c M_A(a)$ y $b >_c M_B(c)$

\Leftrightarrow

$c >_a M_C(a)$ y $a >_c M_A(a)$ y $c >_b M_C(b)$ y $b >_c M_B(c)$

\Leftrightarrow (a,c) bloquea M_{AC} y (b,c) bloquea M_{BC}

Notemos que para analizar el caso 2 asumimos que a y b están en la misma fila mientras c está en otra. Debemos también proyectar el caso 2 para b y c en la misma fila y para a y c en la misma fila.

Nótese también que si a, b y c están en la misma fila, serían una terna del matching y una terna de M nunca bloquea a M.

Uniendo todas las posibilidades (caso 1 y caso 2) llegamos a:

(a,b,c) es terna bloqueante \Leftrightarrow

$((a,b) \text{ bloquea } M_{AB} \text{ y } (b,c) \text{ bloquea } M_{BC} \text{ y } (a, c) \text{ bloquea } M_{AC})$ o (por caso 1)

$((a,c) \text{ bloquea } M_{AC} \text{ y } (b,c) \text{ bloquea } M_{BC})$ o (por caso 2)

$((b,c) \text{ bloquea } M_{BC} \text{ y } (a,b) \text{ bloquea } M_{AB})$ o (por caso 2)

$((a,b) \text{ bloquea } M_{AB} \text{ y } (a,c) \text{ bloquea } M_{AC})$ (por caso 2)

Como sabemos, un matching es estable si no tiene ternas bloqueantes. Es decir que no queremos que se cumpla la condición anterior, baste para ello que no existan (a,b,c) tal que (a,b) bloquee M_{AB} y (b,c) que bloquee M_{BC} . Sabemos que para que no existan tales pares alcanza con que el matching M_{AB} y el matching M_{BC} son estables. ♦

Corolario

Siempre existe matching estable en instancias de este tipo.

Si un matching M es estable en una instancia I de LS, no implica que el mismo sea junta de dos matrimonios estables, es decir que puede darse que M_{AB} , M_{AC} , M_{BC} no sean estables en el problema planteado en matrimonios estables. Ilustremos con un ejemplo.

Ejemplo 5.3

Dadas las listas de preferencias del Ejemplo 3.1, el matching estable $M = \{(a_0, b_2, c_0), (a_1, b_0, c_2), (a_2, b_1, c_1)\}$ no contiene un matrimonio estable, esto es M_{AB} , M_{AC} , M_{BC} no son estables.

Complejidad e Implementación

Para implementar el algoritmo en forma eficiente proponemos utilizar una matriz de ranking para cada conjunto, definida de la siguiente forma:

Para cada agente contamos una matriz indicando la posición de cada elemento en la lista del agente: el vector la matriz tiene $R_A[a,b,c] = (r, s) \Leftrightarrow$ el agente b está en la posición r en las listas de a y el agente c están en la posición s en la lista de preferencias del agente a .

De la misma forma definimos las matrices para B y C ($R_B[b,a,c]$ y $R_C[c,a,b]$).

La matriz de ranking nos permite acceder en forma directa a la posición que tiene un par en la lista de preferencias de un agente, sin necesidad de recorrer las listas de preferencias para encontrar la posición de los pares que se están comparando.

Analicemos el orden del algoritmo asumiendo que contamos con la matriz de ranking:

Paso 1: Encontrar un matrimonio estable entre los agentes de A y los de B , tiene $O(n^2)$

Paso 2: De la misma forma, hallar un matrimonio estable entre B y C tiene $O(n^2)$

Paso 3: la junta por un conjunto tiene $O(n)$.

Entonces, podemos decir que el algoritmo anterior tiene una complejidad de $2O(n^2)+O(n)$, es decir $O(n^2)$.

5.3 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir la estructura del conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones. En este caso, la formulación entera que hemos dado en forma genérica en el Capítulo 3 para todas las instancias de MET3, no aplica a instancias con listas simples, ya que en las mismas existen elementos no comparables. La formulación entera para definir la estructura en instancias de LS queda definida de la siguiente forma:

$$\text{Sea } x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{i > b \\ j > c \\ a}} x_{a,i,j} + \sum_{\substack{k > a \\ l > c \\ b}} x_{k,b,l} + \sum_{\substack{m > a \\ n > b \\ c}} x_{m,n,c} + x_{a,b,c} = 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C$$

Veamos, con la siguiente propiedad que el problema de hallar todos los matchings estables de una instancia en LS es un problema no polinomial.

Propiedad 5.3

El número promedio de soluciones a una instancia de LS crece exponencialmente en función del tamaño de la instancia.

Demostración

Sea $I=(A,B,C;P)$ una instancia de LS de tamaño n . De la demostración de correctitud del Algoritmo 5.1, se desprende que si un matching M' es estable para la instancia I los conjuntos A y B , y un matching M'' es estable para los conjuntos A y C , luego $M' \Join_A M''$ es estable para I .

Sea S' el conjunto de todos los matchings estables entre A y B ($M' \in S'$), sea S'' el conjunto de todos los matchings entre A y C ($M'' \in S''$).

Vimos en el Capítulo 2 que la cantidad de matchings estables bidimensionales crece exponencialmente en función del tamaño de la instancia, luego el tamaño esperado de S' y S'' crece de la misma manera y, por consiguiente $M' \Join_A M''$ con $M' \in S'$ y $M'' \in S''$ crece exponencialmente en función de n . ♦

A partir de esta propiedad, vemos que el conjunto de soluciones a un problema S se torna más complejo cuando más grande es la instancia analizada.

En cuanto a la representación de todos los matchings estables, veremos que la relación de dominación entre los matchings estables (\geq_A), tal como fue definida en el Capítulo 3, es una relación de orden parcial, pues cumple con la propiedades reflexiva, antisimétrica y transitiva.

- Reflexiva: $M \geq_A M$.
Trivial, dado que $M=M$.
- Antisimétrica: $M \geq_A M'$ y $M' \geq_A M \Rightarrow M=M'$.
 $M \geq_A M'$ y $M' \geq_A M \Rightarrow (M_B(a) \geq_a M'_B(a) \text{ y } M_C(a) \geq_a M'_C(a)) \text{ y } (M'_B(a) \geq_a M_B(a) \text{ y } M'_C(a) \geq_a M_C(a)) \forall a \in A \Rightarrow M_B(a)=M'_B(a) \text{ y } M_C(a)=M'_C(a) \forall a \in A \Rightarrow M=M'$.
- Transitiva: $M \geq_A M'$ y $M' \geq_A M'' \Rightarrow M \geq_A M''$.
 $(M_B(a) \geq_a M'_B(a) \text{ y } M'_C(a) \geq_a M''_C(a)) \text{ y } (M'_B(a) \geq_a M''_B(a) \text{ y } M'_C(a) \geq_a M''_C(a)) \forall a \in A \Rightarrow M_B(a) \geq_a M''_B(a) \text{ y } M_C(a) \geq_a M''_C(a) \forall a \in A \Rightarrow M(a) \geq_a M''(a) \forall a \in A$. ♦

Si bien la relación entre todos los matchings estables de este problema es un orden parcial, al igual que lo es la relación entre las soluciones del problema de los matrimonios estables, dicho orden no forma un reticulado distributivo. El ejemplo que damos a continuación muestra que la representación de todos los matchings estables de instancias con listas simples no forma un reticulado distributivo, y que puede haber grafos no conexos, sin supremo y sin ínfimo.

Ejemplo 5.4

Sea $I=(A,B,C;P)$ una instancia de LS donde $A=\{a_0, a_1, a_2\}$, $B=\{b_0, b_1, b_2\}$ y $C=\{c_0, c_1, c_2\}$ con las siguientes listas de preferencias:

P:			
a_0 :	b_0, b_1, b_2	b_0 :	a_0, a_1, a_2
	c_1, c_2, c_0		c_1, c_2, c_0
a_1 :	b_1, b_2, b_0	b_1 :	a_1, a_2, a_0
	c_1, c_2, c_0		c_1, c_2, c_0
		c_0 :	a_0, a_1, a_2
			b_1, b_2, b_0
		c_1 :	a_1, a_2, a_0
			b_0, b_1, b_2

$a_2:$ $b_2 b_1 b_0$ $b_2:$ $a_2 a_1 a_0$ $c_2:$ $a_0 a_1 a_2$
 $c_1 c_2 c_0$ $c_1 c_2 c_0$ $b_1 b_2 b_0$

El conjunto de todos los matchings estables $S(I)$ tiene los siguientes 10 matchings estables:

- $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$
- $M_2 = \{(a_0, b_0, c_1), (a_1, b_1, c_2), (a_2, b_2, c_0)\}$
- $M_3 = \{(a_0, b_0, c_2), (a_1, b_1, c_1), (a_2, b_2, c_0)\}$
- $M_4 = \{(a_0, b_1, c_2), (a_1, b_0, c_1), (a_2, b_2, c_0)\}$
- $M_5 = \{(a_0, b_1, c_2), (a_1, b_2, c_0), (a_2, b_0, c_1)\}$
- $M_6 = \{(a_0, b_2, c_0), (a_1, b_1, c_2), (a_2, b_0, c_1)\}$
- $M_7 = \{(a_0, b_2, c_2), (a_1, b_1, c_1), (a_2, b_0, c_0)\}$
- $M_8 = \{(a_0, b_2, c_2), (a_1, b_1, c_0), (a_2, b_0, c_1)\}$
- $M_9 = \{(a_0, b_2, c_0), (a_1, b_0, c_1), (a_2, b_1, c_2)\}$
- $M_{10} = \{(a_0, b_2, c_2), (a_1, b_0, c_1), (a_2, b_1, c_0)\}$

Notemos que M_2 es $M_{2AB} \Join_B M_{2BC}$, M_3 es $M_{3AB} \Join_A M_{3AC}$ y M_4 es $M_{4AC} \Join_C M_{4BC}$, con M_{2AB} , M_{2BC} , M_{3AB} , M_{3AC} , M_{4AC} y M_{4BC} estables en las correspondientes instancias 2D.

Veamos a continuación la relación de dominación entre los matching estables desde cada uno de los conjuntos.

Desde el conjunto A:

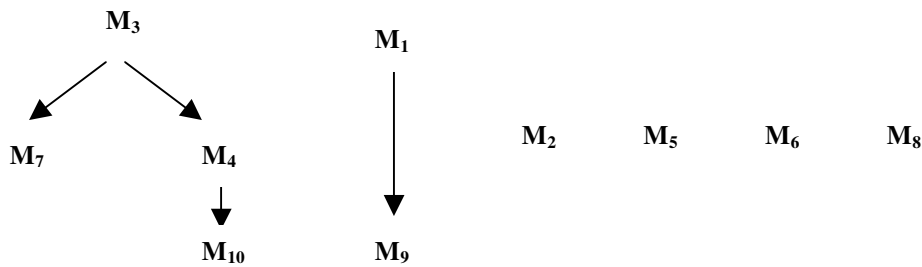


Figura 5.1 Grafo dirigido de todos los matchings estables según el conjunto C

Desde el conjunto B:

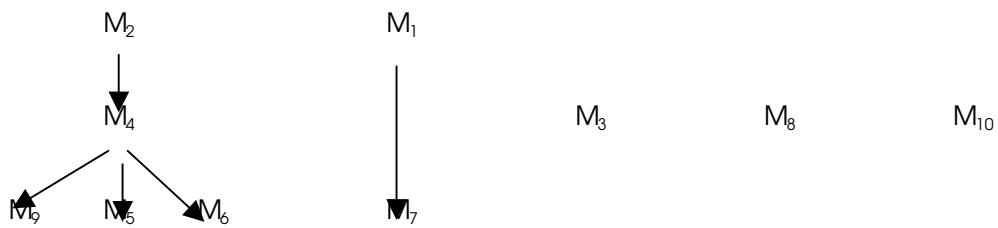


Figura 5.2 Grafo dirigido de todos los matchings estables según el conjunto C

Desde el conjunto C:

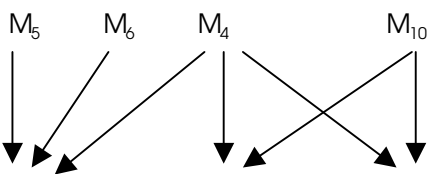




Figura 5.3 Grafo dirigido de todos los matchings estables según el conjunto C

Para las tres perspectivas, vemos que la relación de dominación puede no tener supremo, ínfimo e incluso puede ser un grafo no conexo.

Más allá de que el orden en todos los matchings estables no constituya un reticulado, podemos dar una relación de precedencia entre matchings estables con listas simples en función de la relación de precedencia que tienen los matchings estables en dos dimensiones que conforman la instancia, tal como sigue.

Propiedad 5.2

Si $(M_{AB} \geq_A M'_{AB} \text{ y } M_{AC} >_A M'_{AC})$ o $(M_{AB} >_A M'_{AB} \text{ y } M_{AC} \geq_A M'_{AC}) \Rightarrow M >_A M'$

Demostración

$(M_{AB} \geq_A M'_{AB} \text{ y } M_{AC} >_A M'_{AC})$ o $(M_{AB} >_A M'_{AB} \text{ y } M_{AC} \geq_A M'_{AC})$
 \Rightarrow
 $\forall a_i \in A, (M_B(a_i) \geq_{ai} M'_B(a_i) \text{ y } M_C(a_i) >_{ai} M'_C(a_i))$ o $(M_B(a_i) >_{ai} M'_B(a_i) \text{ y } M_C(a_i) \geq_{ai} M'_C(a_i))$
 \Rightarrow
 $\forall a_i \in A, (M_B(a_i) \geq_{ai} M'_B(a_i)) \text{ y } (M_B(a_i) \geq_{ai} M'_B(a_i) \text{ o } M_C(a_i) \geq_{ai} M'_C(a_i)) \text{ y}$
 $(M_C(a_i) >_{ai} M'_C(a_i) \text{ o } M_B(a_i) >_{ai} M'_B(a_i)) \text{ y } (M_C(a_i) \geq_{ai} M'_C(a_i))$
 \Rightarrow
 $\forall a_i \in A, (M_B(a_i) \geq_{ai} M'_B(a_i)) \text{ y } (M_C(a_i) \geq_{ai} M'_C(a_i))$
 \Rightarrow
 $M >_A M'. \blacklozenge$

Esta propiedad nos permite asumir cierto orden en el conjunto de todos los matchings estables para instancias con listas simples y este orden estaría dado por el orden que tengan en los matchings estables de dos dimensiones que lo conforman, pero no incluiría a todos los matchings estables de la instancia ya que, como vimos antes, no todos los matchings estables en instancias con listas simples son producto de la junta de matchings estables de dos dimensiones.

Vimos que el algoritmo 5.1 encuentra un matching M estable en I , cuyas proyecciones M_{AB} y M_{AC} también lo son en las correspondientes instancias de 2D. A partir de estos matchings M_{AB} y M_{AC} , los cuales representan los correspondientes óptimos para el conjunto A (con relación a B y a C , respectivamente), podemos encontrar el conjunto de todos los matchings estables entre los conjuntos A y B ($S(I')$), en $O(n^2 + n|S(I)|)$ y el conjunto de todos los matchings estables entre los conjuntos A y C ($S(I'')$), también en $O(n^2 + n|S(I'')|)$, tal como vimos en el capítulo 3. Por la propiedad 5.2, realizando las juntas entre todos los pares de matchings M_{AB} y M_{AC} tales que ambos conserven las mismas relaciones de dominación con respecto a los otros matchings, podemos obtener un subconjunto del conjunto de todos los matchings estables en I .

Capítulo 6: Matching Estable Tridimensional sobre tres Conjuntos con Listas por Pares

En este capítulo presentamos los resultados del análisis del problema de matching estable tridimensional sobre tres conjuntos con listas de preferencias por pares. Comenzamos definiendo el problema general, cuya NP-completitud fue estudiada en [NH/95], y damos la formulación lineal entera correspondiente que resuelve el problema en forma exacta. También presentamos un algoritmo polinomial no completo que dependiendo de la instancia puede encontrar solución en tiempos razonables para valores de n más grandes. Luego presentamos los tres subconjuntos de instancias con los que hemos trabajado y para las cuales hemos encontrado caracterizaciones que tienen solución, en donde se utilizaron mapeos con instancias de las clases de listas únicas (LU) y de listas simples (LS), estudiadas en los Capítulos 4 y 5 respectivamente. Por último se analiza para cada tipo de instancia la estructura del conjunto de todos los matchings estables y la representación ordenada de los mismos.

6.1 Matching y Estabilidad

Como ya hemos mencionado anteriormente, el problema de hallar un matching estable tridimensional sobre tres conjuntos donde los agentes de cada conjunto tienen *listas de preferencias por pares* (LP) fue estudiado por Cheng Ng y Hirschberg quienes demuestran que decidir si una instancia dada tiene un matching estable es un problema NP-Completo [NH/95]. Este problema fue uno de los doce problemas abiertos planteados por Knuth en el libro *Marriages Stables* [Knu/76].

Una instancia del problema de MET3 con *listas de preferencias por pares*, está formada por 3 conjuntos disjuntos, A, B y C, de cardinalidad n , donde n es el tamaño de la instancia. Cada uno de estos conjuntos está formado por agentes, quienes tienen una lista de preferencias completa formada por pares de agentes (producto cartesiano) de los conjuntos opuestos. Dichas listas están estrictamente ordenadas.

Se dice que un agente a *prefiere* al par (b_i, c_x) antes que al (b_j, c_y) , o dicho de otro modo, que $(b_i, c_x) >_a (b_j, c_y)$ si $\text{Pos}(a_i, (a_i, b_i, c_x)) < \text{Pos}(a_k, (a_k, b_j, c_y))$. De igual manera se define para $b \in B$ y $c \in C$ una relación $>_b$ y $>_c$ sobre el producto cartesiano $A \times C$ y $A \times B$ respectivamente.

Decimos que una terna (a, b, c) no perteneciente al matching M es una *terna bloqueante* de M si se cumplen las siguientes condiciones [NH/95]:

1. $(b, c) \geq_a M_{BC}(a)$ y
2. $(a, c) \geq_b M_{AC}(b)$ y
3. $(a, b) \geq_c M_{AB}(c)$

Como puede verse, esta definición utiliza la relación (\geq_x) para comparar cada uno de los pares de la terna bloqueante con cada par del matching. Como trabajaremos con listas ordenadas en forma estricta, sin casos de indiferencia entre pares, nunca se cumple la igualdad, pues la terna bloqueante no puede pertenecer al matching. Redefinimos entonces terna bloqueante utilizando la relación $(>_x)$ como sigue:

1. $(b, c) >_a M_{BC}(a)$ y
2. $(a, c) >_b M_{AC}(b)$ y
3. $(a, b) >_c M_{AB}(c)$

Un matching es *estable* en instancias de listas por pares si no tiene ninguna terna bloqueante.

Los siguientes ejemplos muestran instancias del problema de hallar un matching estable con listas por pares.

Ejemplo 6.1

Sea $I = (A, B, C; P)$ donde $A = \{a_0, a_1\}$, $B = \{b_0, b_1\}$ y $C = \{c_0, c_1\}$, con las siguientes listas de preferencias:

P:

a_0 : (b_0, c_0) (b_1, c_0) (b_1, c_1) (b_0, c_1)

a_1 : (b_1, c_0) (b_0, c_1) (b_0, c_0) (b_1, c_1)

b_0 : (a_0, c_1) (a_0, c_0) (a_1, c_0) (a_1, c_1)

b_1 : (a_1, c_0) (a_0, c_0) (a_0, c_1) (a_1, c_1)

c_0 : (a_0, b_1) (a_1, b_1) (a_1, b_0) (a_0, b_0)

c_1 : (a_1, b_1) (a_0, b_1) (a_0, b_0) (a_1, b_0)

En la instancia del Ejemplo 6.1 el matching $M_1 = \{(a_0, b_0, c_1), (a_1, b_1, c_0)\}$ es estable. Para verificarlo alcanza con ver que las posibles ternas bloqueantes, no bloqueen al matching. Observando las listas de preferencias de a_0 , puede verse que las candidatas a bloquear son aquellas ternas a las cuales el correspondiente agente a_i prefiere más que a su pareja actual. En este caso, las posibles ternas bloqueantes son (a_0, b_0, c_0) , (a_0, b_1, c_0) y (a_0, b_1, c_1) , pues a_0 prefiere a los pares de estas ternas, antes que a (b_0, c_1) , sin embargo los agentes b_0 , c_0 y c_1 prefieren el par que les fue asignado en el matching, antes que a sus respectivos pares en las posibles ternas bloqueantes, por lo cual dichas ternas no bloquean.

Por otro lado, el matching $M_2 = \{(a_0, b_0, c_0), (a_1, b_1, c_1)\}$ no es estable por ser bloqueado por la terna (a_1, b_1, c_0) pues $(a_1, b_1) >_{c_0} (a_0, b_0)$, $(b_1, c_0) >_{a_1} (b_1, c_1)$ y $(a_1, c_0) >_{b_1} (a_1, c_1)$.

Ejemplo 6.2

Sea $I = (A, B, C; P)$ donde $A = \{a_0, a_1\}$, $B = \{b_0, b_1\}$ y $C = \{c_0, c_1\}$, con las siguientes listas de preferencias:

P:

a_0 : (b_0, c_1) (b_0, c_0) (b_1, c_1) (b_1, c_0)

a_1 : (b_1, c_1) (b_0, c_0) (b_1, c_0) (b_0, c_1)

b_0 : (a_1, c_0) (a_0, c_1) (a_0, c_0) (a_1, c_1)

b_1 : (a_1, c_0) (a_0, c_0) (a_1, c_1) (a_0, c_1)

c_0 : (a_0, b_1) (a_0, b_0) (a_1, b_0) (a_1, b_1)

c_1 : (a_0, b_0) (a_1, b_1) (a_0, b_1) (a_1, b_0)

Como se mencionó en capítulos anteriores, el problema de hallar un matching en instancias de esta clase, puede no tener solución estable. El Ejemplo 6.2 muestra una instancia en donde todos matchings posibles son bloqueados por alguna terna. En la siguiente tabla figuran todos los posibles matchings y la terna que lo bloquea:

Matching	Terna bloqueante
$M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1)\}$	(a_0, b_0, c_1)
$M_2 = \{(a_0, b_0, c_1), (a_1, b_1, c_0)\}$	(a_1, b_0, c_0)
$M_3 = \{(a_0, b_1, c_0), (a_1, b_0, c_1)\}$	(a_0, b_0, c_1)
$M_4 = \{(a_0, b_1, c_1), (a_1, b_0, c_0)\}$	(a_1, b_1, c_1)

A continuación veremos una propiedad que tienen los matchings canónicos en instancias de listas por pares.

Propiedad 6.1

Todo matching canónico en instancias con listas por pares es estable.

Demostración

Supongamos que existe una terna (a,b,c) que bloquea al matching M canónico, es decir que $(a,b,c) \notin M$ y $(b,c) \succ_a M_{BC}(a)$ y $(a,c) \succ_b M_{AC}(b)$ y $(a,b) \succ_c M_{AB}(c)$.

Como M es un matching canónico vale lo siguiente:

- 1) $\forall t \in M, \text{Pos}(a, t) = 0$, o
- 2) $\forall t \in M, \text{Pos}(b, t) = 0$, o
- 3) $\forall t \in M, \text{Pos}(c, t) = 0$.

Supongamos que vale (1).

Sea $t=(a, b', c') \in M$ donde $(b', c') \in M(a)$, entonces t cumple que $\text{Pos}(a, t)=0 \Leftrightarrow M_{BC}(a) \succ_a (b,c)$

$\forall (b,c) \in BXC, (b,c) \notin M_{BC}(a)$. Absurdo pues por hipótesis $(b,c) \succ_a M_{BC}(a)$. Al mismo resultado se llega si valen (2) o (3). ♦

6.2 Búsqueda de un Matching Estable

Como el problema de decidir si una instancia dada tiene solución estable es NP-Completo, lo es también hallar la solución. Por esta razón presentamos un algoritmo polinomial que si encuentra un matching, el mismo es estable. No obstante si no lo encuentra, no es suficiente para decir que dicho matching no existe.

Básicamente el algoritmo busca un matching estable recorriendo la instancia comenzando por las preferencias del conjunto A , luego continúa por B y por último por C . Esta búsqueda dividida, aumenta las probabilidades de encontrar un matching estable. Debido a que el algoritmo no es exhaustivo, puede no encontrar solución por alguno o ningún camino. En cuanto a la forma de recorrer la instancia y resolver conflictos entre elementos coincidentes, nos hemos basado en el algoritmo de Gale y Shapley para hallar un matrimonio estable [GS/62].

Algoritmo 6.1

El algoritmo intenta todos los caminos : AB, AC, BA, BC, CA, CB , en el orden mencionado hasta encontrar solución o arrojando resultado vacío.

- Camino AB

- 1) Para cada agente $a_i \in A$ sin par asignado, asignar el siguiente par (b_j, c_k) según su lista de preferencias. Llamaremos a dichos pares, *pares actuales*.
- 2) Mientras existan dos o más *pares actuales* del conjunto A en conflicto $(b_j, c_k) \dots (b_j, c_h)$ tal que los agentes b_j coincidan:
 - 2.1) Elegir entre los *pares actuales* en conflicto que tienen al agente b_j en común aquél cuyo valor $\text{Pos}(b_j, (a_i, b_j, c_k))$ sea el menor.

- 2.2) Desasignar de los a_i todos los pares descartados por b_j , dejando únicamente el elegido en el paso anterior.
- 3) Si dentro de los pares actuales del conjunto A , no hay conflicto entre los agentes c_k (con $0 \leq k \leq n-1$), entonces el resultado obtenido es un matching estable.
<Fin del algoritmo>
- Camino AC
- 1) Para cada agente $a_i \in A$ sin par asignado, asignar el siguiente par (b_j, c_k) según su lista de preferencias.
 - 2) Mientras existan dos o más *pares actuales* del conjunto A en conflicto $(b_i, c_j) \dots (b_k, c_j)$ tal que los agentes c_j coincidan:
 - 2.1) Elegir entre los pares actuales en conflicto que tienen al agente c_j en común aquél cuyo valor $\text{Pos}(c_j, (a_i, b_k, c_j))$ sea el menor.
 - 2.2) Desasignar de los a_i todos los pares descartados por c_j , dejando únicamente el elegido en el paso anterior.
 - 3) Si dentro de los *pares actuales* del conjunto A , no hay conflicto entre los agentes b_k (con $0 \leq k \leq n-1$), entonces el resultado obtenido es un matching estable.
<Fin del algoritmo>
- Seguir los mismos pasos para los 4 caminos restantes: BA, BC, CA y CB, haciendo los reemplazos según corresponda.

Teorema

El Algoritmo 6.1 es correcto.

Demostración

a) La solución que el algoritmo genera es un matching.

Supongamos, sin pérdida de generalidad, que el matching fue generado por el camino AB.

Debemos ver que cada agente a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Cada agente a_i tiene inicialmente un par asignado, para todo $0 \leq i \leq n-1$.

Si existe b_j (con $0 \leq j \leq n-1$) tal que ha sido asignado a a_{i_1} y a a_{i_2} (con $0 \leq i_1, i_2 \leq n-1$), se resolverá dicho conflicto en el Paso 2. Al finalizar el ciclo del paso 2 cada agente a_i tiene uno y sólo un b_j asociado y todos los b_j son distintos entre sí. En el paso 3 se verifica que todos los c_k sean distintos entre sí.

b) El matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a, b, c) no perteneciente al matching M , tal que:

$(b, c) \succ_a (M_B(a), M_C(a))$ y

$(a, c) \succ_b (M_A(b), M_C(b))$ y

$(a, b) \succ_c (M_A(c), M_B(c))$

Supongamos que existe tal terna.

Por el Paso 2.1 y 3 del algoritmo, si $(b,c) >_a (M_B(a), M_C(a))$, significa que en algún momento el agente a eligió a (b,c) antes de elegir a $(M_B(a), M_C(a))$. Si la terna (a,b,c) no pertenece al matching quiere decir que en algún momento el agente b prefirió otro par, digamos (a_i, c_k) más que a (a,c) .

Si $(a_i, c_k) = (M_A(b), M_C(b)) \Rightarrow (M_A(b), M_C(b)) = (a_i, c_k) >_b (a,c)$. Esto es absurdo pues contradice el punto 2 de la hipótesis.

Si $(a_i, c_k) \bullet (M_A(b), M_C(b)) \Rightarrow$ como el conjunto que elige es B , quiere decir que en algún momento b eligió a (a_j, c_h) sobre (a_i, c_k) . Siguiendo el mismo razonamiento, se llega a $(a_p, c_q) >_b \dots >_b (a_j, c_h) >_b (a_i, c_k) >_b (a,c)$. Esto es absurdo, pues contradice el punto 2 de la hipótesis. ♦

Si bien la complejidad del algoritmo es analizada en el siguiente punto de este capítulo, a simple vista puede verse que el algoritmo es de orden polinomial. Veamos que el algoritmo no es completo con el siguiente ejemplo, donde se muestra una instancia para la cual el algoritmo no encuentra solución y sin embargo la misma existe.

Ejemplo 6.3

Sea $I=(A,B,C;P)$ donde $A=\{a_0, a_1, a_2\}$, $B=\{b_0, b_1, b_2\}$ y $C=\{c_0, c_1, c_2\}$, con las siguientes listas de preferencias

P:

$a_0 : (b_0, c_1) (b_0, c_2) (b_2, c_2) (b_1, c_2) (b_1, c_1) (b_0, c_0) (b_2, c_1) (b_2, c_0) (b_1, c_0)$

$a_1 : (b_0, c_2) (b_0, c_1) (b_0, c_0) (b_2, c_1) (b_1, c_2) (b_2, c_0) (b_1, c_0) (b_1, c_1) (b_2, c_2)$

$a_2 : (b_2, c_1) (b_1, c_0) (b_0, c_1) (b_1, c_2) (b_0, c_0) (b_1, c_1) (b_2, c_2) (b_2, c_0) (b_0, c_2)$

$b_0 : (a_2, c_0) (a_2, c_1) (a_0, c_1) (a_0, c_2) (a_1, c_1) (a_2, c_2) (a_0, c_0) (a_1, c_0) (a_1, c_2)$

$b_1 : (a_1, c_2) (a_2, c_1) (a_1, c_0) (a_1, c_1) (a_2, c_2) (a_0, c_0) (a_0, c_2) (a_0, c_1) (a_2, c_0)$

$b_2 : (a_2, c_1) (a_1, c_0) (a_1, c_1) (a_1, c_2) (a_2, c_2) (a_0, c_2) (a_0, c_1) (a_2, c_0) (a_0, c_0)$

$c_0 : (a_1, b_2) (a_1, b_1) (a_1, b_0) (a_0, b_0) (a_0, b_2) (a_2, b_1) (a_0, b_1) (a_2, b_0) (a_2, b_2)$

$c_1 : (a_0, b_1) (a_2, b_1) (a_0, b_0) (a_2, b_2) (a_0, b_2) (a_1, b_2) (a_1, b_0) (a_1, b_1) (a_2, b_0)$

$c_2 : (a_1, b_2) (a_0, b_1) (a_0, b_2) (a_1, b_0) (a_2, b_1) (a_1, b_1) (a_2, b_2) (a_0, b_0) (a_2, b_0)$

Camino AB

$a_0 \mid (b_0, c_1)$

$a_1 \mid \cancel{(b_0, c_2)} \cancel{(b_0, c_1)} \cancel{(b_0, c_0)} \cancel{(b_2, c_1)} (b_1, c_2)$

$a_2 \mid (b_2, c_1)$

Existe coincidencia en c_1 , por lo tanto el algoritmo no encontró matching estable por el camino AB.

Camino AC

$a_0 \mid (b_0, c_1)$

$a_1 \mid (b_0, c_2)$

$a_2 \mid \cancel{(b_2, c_1)} (b_1, c_0)$

Existe coincidencia en b_0 , por lo tanto el algoritmo no encontró matching estable por el camino AC.

Camino BA

$b_0 \mid \cancel{(a_2, c_0)} \cancel{(a_2, c_1)} (a_0, c_1)$

$b_1 \mid (a_1, c_2)$

$b_2 \mid (a_2, c_1)$

Existe coincidencia en c_1 , por lo tanto el algoritmo no encontró matching estable por el camino BA.

Camino BC

$b_0 \mid (a_2, c_0)$
 $b_1 \mid (a_1, c_2)$
 $b_2 \mid (a_2, c_1)$

Existe coincidencia en a_2 , por lo tanto el algoritmo no encontró matching estable por el camino BC.

Camino CA

$c_0 \mid (a_1, b_2)$
 $c_1 \mid (a_0, b_1) \mid (a_2, b_1)$
 $c_2 \mid (a_1, b_2) \mid (a_0, b_1)$

Existe coincidencia en b_1 , por lo tanto el algoritmo no encontró matching estable por el camino CA.

Camino CB

$c_0 \mid (a_1, b_2)$
 $c_1 \mid (a_0, b_1) \mid (a_2, b_1)$
 $c_2 \mid (a_1, b_2) \mid (a_0, b_1) \mid (a_0, b_2) \mid (a_1, b_0)$

Existe coincidencia en a_1 , por lo tanto el algoritmo no encontró matching estable por el camino CB.

Como pudimos ver, el algoritmo no encontró matching estable, y sin embargo el siguiente matching M es estable: $M = \{ (a_0, b_2, c_2), (a_1, b_0, c_0), (a_2, b_1, c_1) \}$.

A continuación damos un algoritmo que si bien es muy simple, contribuye a ratificar lo expresado por los autores en [NH/95], referente a la existencia de verificación polinomial de la estabilidad de un matching en una instancia dada.

Algoritmo 6.2

- 1) Para cada terna (a, b, c) no perteneciente al matching
Si $[(a, b) >_c M_{AB}(c)]$ y $[(b, c) >_a M_{BC}(a)]$ y $[(a, c) >_b M_{AC}(b)]$,
el algoritmo termina indicando que el matching dado No es estable.
- 2) Llegado este punto el algoritmo termina indicando que el matching dado Es estable.

La correctitud y completitud de este algoritmo se deduce directamente de la definición de matching estable en instancias con listas de preferencias por pares.

Complejidad e Implementación

Analizando el orden del Algoritmo 6.2, que verifica la estabilidad de un matching en una instancia dada, puede observarse que para cada una de las n^3 ternas, compara los pares (a_i, b_j) , (b_j, c_k) y (c_k, a_i) con sus respectivas asignaciones en el matching. Como esta comparación se realiza a través de la matriz de ranking, no aumenta la complejidad del algoritmo. Luego la complejidad del algoritmo es de $O(n^3)$.

Haciendo un análisis de complejidad del Algoritmo 6.1, puede observarse que dado que las listas de preferencias tienen n^2 pares de elementos, en el peor de los casos deberá recorrer n veces (una por cada agente) los n^2 elementos, para elegir el mejor par. Esto tiene complejidad de $O(n^3)$. Adicionalmente, para cada agente que elige, debe comparar los pares en conflicto. Como esto se hace a través de la matriz de ranking, no hay un incremento en la complejidad. Si hasta aquí no encontró matching, repite el procedimiento, pero es el otro conjunto quien realiza su elección, por lo que también es de $O(n^3)$. Notemos que aunque tenga que realizar las tres partes del algoritmo, no aumenta la complejidad, pues: $3 * 2 * O(n^3) \cong O(6n^3) \cong O(n^3)$.

Al igual que en los capítulos anteriores, para implementar ambos algoritmos en forma eficiente utilizamos una matriz de ranking para cada conjunto, definida de la siguiente forma:

$R_A[a,b,c]=r \Leftrightarrow$ el par (b,c) están en la posición r en la lista de preferencias del agente a .

$R_B[a,b,c]=r \Leftrightarrow$ el par (a,c) están en la posición r en la lista de preferencias del agente b .

$R_C[a,b,c]=r \Leftrightarrow$ el par (a,b) están en la posición r en la lista de preferencias del agente c .

La matriz de ranking permite acceder en forma directa a la lista de preferencias, sin necesidad de recorrerla para encontrar la posición de los pares que se están comparando.

6.3 El conjunto de todos los Matching Estables

Como se vio en el Capítulo 3, desde el punto de vista de la programación lineal entera, podemos definir la estructura del conjunto de todos los matchings, como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones. A continuación damos la formulación lineal entera para representar el conjunto de todos los matchings estables en instancias de LP, siendo la penúltima de las restricciones la que define la estabilidad del matching.

$$\text{Sea } x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C$$

$$\sum_{\substack{(i,j) > \\ a}} x_{a,i,j} + \sum_{\substack{(k,l) > \\ b}} x_{k,b,l} + \sum_{\substack{(m,n) > \\ c}} x_{m,n,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C$$

$$x_{a,b,c} \in \{0,1\} \quad (a,b,c) \in A \times B \times C$$

En cuanto a la representación de todos los matchings estables, veremos que la relación de dominación entre los matchings estables (\geq_A), tal como fue definida en el Capítulo 3, es una relación de orden parcial, pues cumple con la propiedades reflexiva, antisimétrica y transitiva.

- Reflexiva: $M \geq_A M$.
Trivial ya que $M=M$
- Antisimétrica: $M \geq_A M'$ y $M' \geq_A M \Rightarrow M=M'$.
 $M \geq_A M'$ y $M' \geq_A M \Rightarrow M_{BC}(a) \geq_a M'_{BC}(a)$ y $M'_{BC}(a) \geq_a M_{BC}(a) \forall a \in A \Rightarrow M_{BC}(a) = M'_{BC}(a) \forall a \in A \Rightarrow M=M'$.

- Transitiva: $M \succeq_A M'$ y $M' \succeq_A M'' \Rightarrow M \succeq_A M''$.
 $M \succeq_A M'$ y $M' \succeq_A M'' \Rightarrow M_{BC}(a) \succeq_a M'_{BC}(a)$ y $M'_{BC}(a) \succeq_a M''_{BC}(a) \forall a \in A \Rightarrow M_{BC}(a) \succeq_a M''_{BC}(a) \forall a \in A$. ♦

Si bien la relación entre todos los matchings estables de este problema es un orden parcial, al igual que lo es la relación entre las soluciones del problema de los matrimonios estables, dicho orden no forma un reticulado distributivo. El ejemplo que damos a continuación muestra que la representación de todos los matchings estables de instancias con listas por pares no forma un reticulado distributivo, y que los grafos dirigidos representando estas relaciones pueden ser grafos no conexos, sin supremo y sin ínfimo.

Ejemplo 6.4

Sea $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$ y $C = \{c_0, c_1, c_2\}$, con las siguientes listas de preferencias

P:

a_0 : (b_0, c_0) (b_0, c_1) (b_0, c_2) (b_1, c_0) (b_1, c_1) (b_1, c_2) (b_2, c_0) (b_2, c_1) (b_2, c_2)
 a_1 : (b_1, c_2) (b_1, c_0) (b_1, c_1) (b_0, c_2) (b_0, c_0) (b_0, c_1) (b_2, c_2) (b_2, c_0) (b_2, c_1)
 a_2 : (b_2, c_2) (b_2, c_0) (b_2, c_1) (b_0, c_2) (b_0, c_0) (b_0, c_1) (b_1, c_2) (b_1, c_0) (b_1, c_1)

b_0 : (a_1, c_2) (a_0, c_2) (a_2, c_2) (a_1, c_1) (a_0, c_1) (a_2, c_1) (a_1, c_0) (a_0, c_0) (a_2, c_0)
 b_1 : (a_0, c_0) (a_2, c_0) (a_1, c_0) (a_0, c_2) (a_2, c_2) (a_1, c_2) (a_0, c_1) (a_2, c_1) (a_1, c_1)
 b_2 : (a_2, c_1) (a_0, c_1) (a_1, c_1) (a_2, c_2) (a_0, c_2) (a_1, c_2) (a_2, c_0) (a_0, c_0) (a_1, c_0)

c_0 : (a_1, b_2) (a_1, b_1) (a_1, b_0) (a_2, b_2) (a_2, b_1) (a_2, b_0) (a_0, b_2) (a_0, b_1) (a_0, b_0)
 c_1 : (a_0, b_2) (a_0, b_1) (a_0, b_0) (a_2, b_2) (a_2, b_1) (a_2, b_0) (a_1, b_2) (a_1, b_1) (a_1, b_0)
 c_2 : (a_1, b_1) (a_1, b_0) (a_1, b_2) (a_0, b_1) (a_0, b_0) (a_0, b_2) (a_2, b_1) (a_2, b_0) (a_2, b_2)

El conjunto de matchings estables $S(I)$:

$M_1 = \{ (a_0, b_0, c_0), (a_1, b_1, c_2), (a_2, b_2, c_1) \}$
 $M_2 = \{ (a_0, b_0, c_1), (a_1, b_1, c_0), (a_2, b_2, c_2) \}$
 $M_3 = \{ (a_0, b_0, c_1), (a_1, b_1, c_2), (a_2, b_2, c_0) \}$
 $M_4 = \{ (a_0, b_0, c_2), (a_1, b_1, c_0), (a_2, b_2, c_1) \}$
 $M_5 = \{ (a_0, b_1, c_0), (a_1, b_0, c_2), (a_2, b_2, c_1) \}$
 $M_6 = \{ (a_0, b_2, c_1), (a_1, b_1, c_2), (a_2, b_0, c_0) \}$
 $M_7 = \{ (a_0, b_2, c_1), (a_1, b_0, c_2), (a_2, b_1, c_0) \}$

Las siguientes figuras muestran la relación de orden entre los matchings estables según las preferencias de los conjunto A, B y C respectivamente.

Desde el conjunto A:

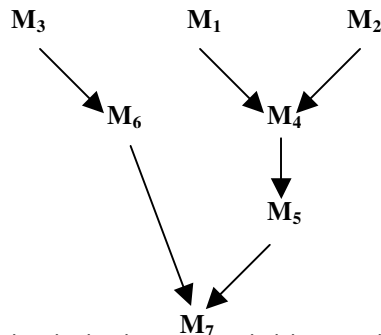


Figura 6.1 Grafo dirigido de todos los matchings estables según el conjunto A

Desde el conjunto B:

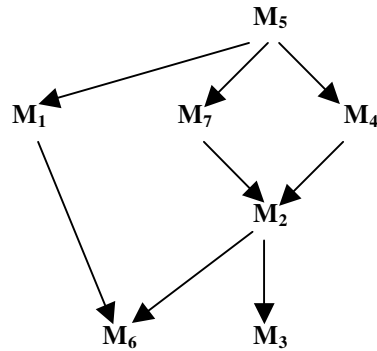
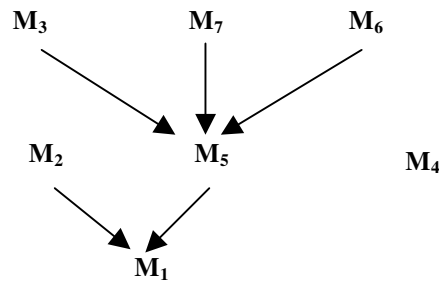


Figura 6.2 Grafo dirigido de todos los matchings estables según el conjunto B



Desde el conjunto C:

Figura 6.3 Grafo dirigido de todos los matchings estables según el conjunto C

La representación de todos los matchings estables según el conjunto A (Figura 6.1) muestra que el matching M_7 es el nodo ínfimo del grafo, mientras que no existe un supremo para M_3 y M_1 , M_1 y M_2 ni para M_2 y M_3 . En cambio en el grafo de la Figura 6.2 el matching M_5 es el supremo, pero los matchings M_6 y M_3 no tienen un ínfimo. Por último la representación según el conjunto C, muestra un grafo no conexo.

6.4 Subconjuntos de instancias

Hasta aquí hemos presentado el problema para instancias genéricas, en donde las listas de preferencias no cumplen con ningún tipo de patrón o criterio con el cual fueron formadas. Resulta luego interesante analizar si caracterizando las instancias de acuerdo a determinadas propiedades, el problema de decidir si existe un matching estable, continúa siendo un problema NP-Completo.

Las caracterizaciones que definimos y sobre las cuales trabajamos responden a cierto criterio por parte de los agentes que realizan su elección. Por ejemplo, si un agente tiene mayor interés en los agentes de uno de los conjuntos por sobre los del otro, y pudiera armar una lista de preferencias separada para el conjunto que más quiere, y una lista sobre el otro conjunto que puede ser diferente según el agente que le corresponda en el par, su lista de preferencias podría verse de la siguiente forma:

$c_x: (a_{i1}, b_{j1}), \dots, (a_{i1}, b_{jn}), (a_{i2}, b_{k1}), \dots, (a_{i2}, b_{kn}), \dots, (a_{in}, b_{m1}), \dots, (a_{in}, b_{mn})$;
 donde $a_x \in A$ y $b_y \in B$

Esta lista estaría indicando que el agente c_x le da prioridad o *prioriza* a los agentes del conjunto A : (a_1, a_2, \dots, a_m) y que para cada agente de A tiene una lista de preferencias ordenada sobre los agentes de B en forma diferente.

Otro criterio menos restrictivo, puede ser que las listas de preferencias sean *consistentes*, es decir, que si para un agente c_i el par (x,y) fuera más preferido que el par (x,z) , no puede pasar que el par (r,z) sea más preferido que el par (r,y) . De igual manera debe ser consistente para el otro miembro del par. Cabe aclarar que la NP-Complejidad bajo instancias que cumplan con la propiedad de consistencia es un problema abierto [NH/95].

Si bien el problema para listas consistentes continúa abierto, hemos encontrado algunas propiedades que presentamos en el siguiente párrafo. Posteriormente definimos y estudiamos las listas con prioridad para las cuales hemos hallado un conjunto de propiedades que caracterizan instancias con solución.

6.4.1 Listas por Pares Consistentes (LPC)

Una instancia I de LP es *consistente* si para toda lista de preferencias, se cumplen las siguientes reglas:

1. $(a, b) >_c (a, b') \Rightarrow (a', b) >_c (a', b') \quad \forall a, a' \in A, \forall b, b' \in B, \forall c \in C$ y
2. $(a, b) >_c (a', b) \Rightarrow (a, b') >_c (a', b') \quad \forall a, a' \in A, \forall b, b' \in B, \forall c \in C$

La propiedad de consistencia no garantiza existencia de matching estable. El siguiente ejemplo [Alk/87] muestra la plantilla de una instancia que no tiene solución y sus listas cumplen con la propiedad de consistencia. Todas las instancias consistentes que se formen basándose en esta plantilla no poseerán matching estable.

Ejemplo 6.5

a_0 : $(b_0, c_2) (b_1, c_2) (b_0, c_0) \dots$
 a_1 : $(b_1, c_2) (b_1, c_1) (b_2, c_2) \dots$
 a_2 : $(b_2, c_2) \dots$

b_0 : $(a_0, c_0) \dots$
 b_1 : $(a_1, c_2) (a_0, c_2) (a_1, c_1) \dots$
 b_2 : $(a_1, c_2) (a_2, c_2) \dots$

c_0 : $(a_0, b_0) \dots$
 c_1 : $(a_1, b_1) \dots$
 c_2 : $(a_0, b_2) (a_1, b_2) (a_0, b_1) (a_2, b_2) \dots$

El siguiente ejemplo muestra una instancia completa con listas por pares consistentes.

Ejemplo 6.6

Sea $I = (A, B, C; P)$ con listas consistentes, donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$ y $C = \{c_0, c_1, c_2\}$ con las siguientes listas de preferencias:

P :

a_0 : $(b_0, c_0) (b_0, c_1) (b_1, c_0) (b_2, c_0) (b_0, c_2) (b_1, c_1) (b_2, c_1) (b_1, c_2) (b_2, c_2)$
 a_1 : $(b_1, c_0) (b_0, c_0) (b_1, c_1) (b_2, c_0) (b_0, c_1) (b_1, c_2) (b_2, c_1) (b_0, c_2) (b_2, c_2)$
 a_2 : $(b_0, c_0) (b_0, c_1) (b_1, c_0) (b_2, c_0) (b_0, c_2) (b_1, c_1) (b_2, c_1) (b_1, c_2) (b_2, c_2)$

b_0 : $(a_1, c_0) (a_1, c_1) (a_0, c_0) (a_2, c_0) (a_1, c_2) (a_0, c_1) (a_2, c_1) (a_0, c_2) (a_2, c_2)$

$b_1: (a_0, c_0) (a_0, c_1) (a_1, c_0) (a_2, c_0) (a_0, c_2) (a_1, c_1) (a_2, c_1) (a_1, c_2) (a_2, c_2)$
 $b_2: (a_0, c_0) (a_0, c_1) (a_1, c_0) (a_2, c_0) (a_0, c_2) (a_1, c_1) (a_2, c_1) (a_1, c_2) (a_2, c_2)$
 $c_0: (a_0, b_1) (a_0, b_0) (a_1, b_1) (a_2, b_1) (a_0, b_2) (a_1, b_0) (a_2, b_0) (a_1, b_2) (a_2, b_2)$
 $c_1: (a_1, b_0) (a_1, b_1) (a_0, b_0) (a_2, b_0) (a_1, b_2) (a_0, b_1) (a_2, b_1) (a_0, b_2) (a_2, b_2)$
 $c_2: (a_0, b_0) (a_0, b_1) (a_1, b_0) (a_2, b_0) (a_0, b_2) (a_1, b_1) (a_2, b_1) (a_1, b_2) (a_2, b_2)$

En esta instancia, el matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es un matching estable mientras que $M_2 = \{(a_0, b_0, c_1), (a_1, b_1, c_0), (a_2, b_2, c_2)\}$ no es estable pues es bloqueado por terna (a_0, b_0, c_0) .

6.4.1.1 Relación con Instancias de Listas Simples

Notemos que una característica de estas listas de preferencias es que al fijar uno de los agentes de un par, para una misma lista, se mantiene el orden de preferencia de los otros agentes. Veamos por ejemplo las listas de a_1 : para todos los b_i , el orden de preferencia de los c_j es el mismo: c_0, c_1, c_2 , mientras que para todos los c_j el orden de los b_i también coincide: b_1, b_0, b_2 .

Podríamos escribir las sublistas individuales para el agente a_1 de la siguiente forma:

$b_i: b_1 b_0 b_2$
 $c_j: c_0 c_1 c_2$

Resulta interesante analizar si hay alguna relación entre las listas consistentes y las sublistas individuales "embebidas" tratadas como listas simples dado que estas últimas cumplen con la propiedad de consistencia. [Ver Capítulo 5]

$Pos(a_1, (b_i, c_j))$	0	1	2	3	4	5	6	7	8
Lista de a_1 :	(b_1, c_0)	(b_0, c_0)	(b_1, c_1)	(b_2, c_0)	(b_0, c_1)	(b_1, c_2)	(b_2, c_1)	(b_0, c_2)	(b_2, c_2)

Tabla 6.4 Lista de preferencias del agente a_1 con los números de posición de cada par

El gráfico correspondiente a las listas simples embebidas en la lista de preferencias de a_1 , se muestra en la Figura 6.4:

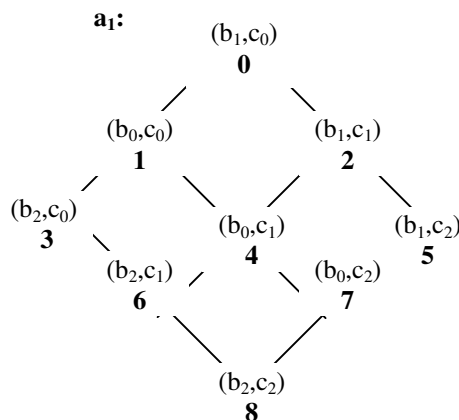


Figura 6.4 Gráfico correspondiente a la lista de preferencias del agente a_1 representada como listas simples.

Esto nos permite ver que hay una instancia del problema de hallar un matching estable en listas simples embebido en el problema de hallar un matching estable en instancias de listas por pares que satisfagan la característica de consistencia. La diferencia con las listas simples es que en las listas por pares consistentes todos los pares de agentes son comparables entre sí, pues la relación de preferencia define un orden lineal. Llamaremos *Instancia Embebida* a la instancia de LS correspondiente con una (o varias) instancias de LPC.

Para construir la instancia I' embebida de LS correspondiente a una instancia I de LPC damos el siguiente procedimiento:

Para cada agente $a \in A$, las listas de I' estarán formadas por la lista de a relativa a b_0 y la de a relativa a c_0 . De la misma manera obtenemos las listas de los $b \in B$ y los $c \in C$. Notemos que se toma arbitrariamente b_0 y c_0 pues como las listas son consistentes, todas las listas relativas a $b_i \in B$ y $c_i \in C$ son iguales.

Veamos en la siguiente propiedad que el orden de las listas en la instancia embebida se conserva en la instancia de LPC.

Propiedad 6.2

Sea I una instancia del problema de hallar un matching estable con listas por pares consistentes, sea I' la instancia embebida en LS. Si $(b,c) >_a (b',c')$ en $I' \Rightarrow (b,c) >_a (b',c')$ en I .

Demostración

Si $(b,c) >_a (b',c')$ en $I' \Rightarrow (b \geq_a b' \text{ y } c >_a c')$ o $(b >_a b' \text{ y } c \geq_a c')$ por definición de $>_a$ en listas simples. Podemos dividirlo en tres casos:

Caso 1: Si $b = b' \text{ y } c > c' \Rightarrow (b, c) >_a (b', c')$ en I por construcción de la instancia embebida

Caso 2: Si $c = c' \text{ y } b > b' \Rightarrow (b, c) >_a (b', c')$ en I por construcción de la instancia embebida

Caso 3: Si $b \neq b' \text{ y } c \neq c' \Rightarrow (b >_a b' \text{ y } c >_a c')$ por definición de $>_a$ en listas simples \Leftrightarrow por construcción de la instancia embebida

$$(b, c) >_a (b, c') \text{ y } (1)$$

$$(b', c) >_a (b', c') \text{ y } (2)$$

$$(b, c) >_a (b', c) \text{ y } (3)$$

$$(b, c') >_a (b', c') \text{ y } (4)$$

en I .

\Rightarrow (por 1 y 4)

$$(b, c) >_a (b, c') >_a (b', c') \text{ en } I. \blacklozenge$$

Propiedad 6.3

Sea I una instancia del problema de hallar un matching estable con listas por pares consistentes, sea I' la instancia embebida en LS. Si M es un matching estable en I , entonces M es matching estable en I' .

Demostración

Para demostrar esto vamos a probar que si una terna $t=(a,b,c) \notin M$ bloquea a M en I' (LS), entonces la terna (a,b,c) bloquea a M en I (LPC).

Por la definición general de terna bloqueante que es válida para instancias de LS, dado un matching M , una terna $(a,b,c) \notin M$ bloquea a M en I' si $(b,c) >_a M_{BC}(a)$ y $(a,c) >_b M_{AC}(b)$ y $(a,b) >_c M_{AB}(c)$. Por la Propiedad 6.2, el orden en las listas de preferencias de I se conservan en I' , lo cual implica que $(b,c) >_a M_{BC}(a)$ y $(a,c) >_b M_{AC}(b)$ y $(a,b) >_c M_{AB}(c)$ en I . Luego (a,b,c) bloquea al matching M en I . \blacklozenge

En el siguiente ejemplo mostramos la instancia embebida de LS del Ejemplo 6.5. También damos un matching estable de la instancia de LPC, que es estable en la instancia embebida de LS, y uno que no lo es, con cual quedaría demostrado que la recíproca de la propiedad anterior no es válida.

Ejemplo 6.7

Sea $I = (A, B, C; P)$ con listas consistentes, donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$ y $C = \{c_0, c_1, c_2\}$ con las siguientes listas de preferencias:

P:

a_0 : $(b_0, c_0) (b_0, c_1) (b_1, c_0) (b_2, c_0) (b_0, c_2) (b_1, c_1) (b_2, c_1) (b_1, c_2) (b_2, c_2)$

a_1 : $(b_1, c_0) (b_0, c_0) (b_1, c_1) (b_2, c_0) (b_0, c_1) (b_1, c_2) (b_2, c_1) (b_0, c_2) (b_2, c_2)$

a_2 : $(b_0, c_0) (b_0, c_1) (b_1, c_0) (b_2, c_0) (b_0, c_2) (b_1, c_1) (b_2, c_1) (b_1, c_2) (b_2, c_2)$

b_0 : $(a_1, c_0) (a_1, c_1) (a_0, c_0) (a_2, c_0) (a_1, c_2) (a_0, c_1) (a_2, c_1) (a_0, c_2) (a_2, c_2)$

b_1 : $(a_0, c_0) (a_0, c_1) (a_1, c_0) (a_2, c_0) (a_0, c_2) (a_1, c_1) (a_2, c_1) (a_1, c_2) (a_2, c_2)$

b_2 : $(a_0, c_0) (a_0, c_1) (a_1, c_0) (a_2, c_0) (a_0, c_2) (a_1, c_1) (a_2, c_1) (a_1, c_2) (a_2, c_2)$

c_0 : $(a_0, b_1) (a_0, b_0) (a_1, b_1) (a_2, b_1) (a_0, b_2) (a_1, b_0) (a_2, b_0) (a_1, b_2) (a_2, b_2)$

c_1 : $(a_1, b_0) (a_1, b_1) (a_0, b_0) (a_2, b_0) (a_1, b_2) (a_0, b_1) (a_2, b_1) (a_0, b_2) (a_2, b_2)$

c_2 : $(a_0, b_0) (a_0, b_1) (a_1, b_0) (a_2, b_0) (a_0, b_2) (a_1, b_1) (a_2, b_1) (a_1, b_2) (a_2, b_2)$

Listas simples embebidas:

a_0 : $B : b_0, b_1, b_2$
 $C : c_0, c_1, c_2$

a_1 : $B : b_1, b_0, b_2$
 $C : c_0, c_1, c_2$

a_2 : $B : b_0, b_1, b_2$
 $C : c_0, c_1, c_2$

b_0 : $A : a_1, a_0, a_2$
 $C : c_0, c_1, c_2$

b_1 : $A : a_0, a_1, a_2$
 $C : c_0, c_1, c_2$

b_2 : $A : a_0, a_1, a_2$
 $C : c_0, c_1, c_2$

c_0 : $A : a_0, a_1, a_2$
 $B : b_1, b_0, b_2$

c_1 : $A : a_1, a_0, a_2$
 $B : b_0, b_1, b_2$

c_2 : $A : a_0, a_1, a_2$
 $B : b_0, b_1, b_2$

Los matchings $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ y $M_2 = \{(a_0, b_1, c_0), (a_1, b_0, c_1), (a_2, b_2, c_2)\}$ son estables en ambas instancias. En cambio el matching $M_3 = \{(a_0, b_0, c_1), (a_1, b_1, c_0), (a_2, b_2, c_2)\}$ es estable en la instancia embebida de LS pero no es estable en la instancia de LPC pues es bloqueado por la terna (a_0, b_0, c_0) .

6.4.2 Listas con Prioridad

En este punto, analizamos las listas de preferencias en las cuales existe un orden sobre los agentes de uno de los conjuntos, y un orden dependiente del par, para los agentes del otro conjunto.

Definimos que una instancia I con listas por pares, tiene *listas de preferencias con prioridad* (LPP) de A sobre B (o que A prioriza a B) y notamos $A \succ B$ si para cada agente $a \in A$ se cumple que $\forall b, b' \in B$ si $(b, c) \succ_a (b', c') \Rightarrow (b, c'') \succ_a (b', c' \forall c, c', c'' \in C$.

Ejemplo 6.8

Sea $I = (A, B, C; P)$ con listas consistentes, donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$ y $C = \{c_0, c_1, c_2\}$ con las siguientes listas de preferencias:

P:

$a_0: (b_0, c_0) (b_0, c_1) (b_0, c_2) (b_2, c_2) (b_2, c_1) (b_2, c_0) (b_1, c_1) (b_1, c_2) (b_1, c_0)$
 $a_1: (b_1, c_0) (b_1, c_1) (b_1, c_2) (b_0, c_2) (b_0, c_1) (b_0, c_0) (b_2, c_1) (b_2, c_2) (b_2, c_0)$
 $a_2: (b_0, c_1) (b_0, c_0) (b_0, c_2) (b_1, c_2) (b_1, c_0) (b_1, c_1) (b_2, c_1) (b_2, c_2) (b_2, c_0)$

Nótese que las listas de preferencias de un conjunto, al tener prioridad sobre otro, agrupan a los pares de los agentes sobre el cual tienen prioridad, lo que permite una representación abreviada de las mismas. En el Ejemplo 6.8, el conjunto A prioriza al conjunto B, y aplicando la propiedad sobre las listas de preferencias por pares, se pueden definir las siguientes listas únicas de A hacia B:

$a_0: b_0, b_2, b_1$
 $a_1: b_1, b_0, b_2$
 $a_2: b_0, b_1, b_2$

y las listas únicas de A con respecto al conjunto C para cada b_i perteneciente a B:

$a_0: b_0 (c_0, c_1, c_2) ; b_2 (c_2, c_1, c_0) ; b_1 (c_1, c_2, c_0)$
 $a_1: b_1 (c_0, c_1, c_2) ; b_0 (c_2, c_1, c_0) ; b_2 (c_1, c_2, c_0)$
 $a_2: b_0 (c_1, c_0, c_2) ; b_1 (c_2, c_0, c_1) ; b_2 (c_1, c_2, c_0)$

La siguiente propiedad formaliza la equivalencia entre ambas representaciones.

Propiedad 6.3

Sea I una instancia del problema de matching estable tridimensional con listas por pares en la cual al menos uno de los tres conjuntos participantes (digamos A) tiene su lista de preferencias por pares con prioridad sobre alguno de los otros dos conjuntos (B).

Entonces:

- Para cada a_i , ($0 \leq i \leq n-1$), a_i puede definir una lista única sobre los elementos de B, respetando las preferencias de a_i sobre B establecidas en la lista por pares correspondiente.
- Para cada a_i ($0 \leq i \leq n-1$) y para un determinado b_j , a_i puede definir una lista única sobre los elementos de C, respetando las preferencias de a_i establecidas en la lista por pares correspondiente.

Demostración

a) Partiendo de la definición podemos observar que si $\forall b, b' \in B$ si $(b, c) >_a (b', c) \Rightarrow (b, c) >_a (b', c)$ $\forall c, c', c'' \in C$, existe un orden estricto sobre los $b \in B$: $b_i > b_j \dots > b_k$ por lo que podemos formar una lista única de agentes b_i, b_j, \dots, b_k

b) Al no haber restricciones sobre el orden de los $c \in C$, el agente $a \in A$ puede tener un orden para los c , según el par que le haya asignado a la lista de preferencias. ♦

De ahora en más llamaremos a las sublistas entre paréntesis, *listas de preferencias de a sobre C relativas a b*. Podemos definir entonces una relación de preferencia para estas sublistas, desde los a_i hacia los c_k relativa a un b_j . Decimos entonces que *a prefiere a c antes que a c' relativo a b* y notamos $c >_{a/b} c'$, si $(b, c) >_a (b, c')$

Analizando este subconjunto de instancias cuyas listas de preferencias tienen prioridad, podemos hacer una subclasificación según las prioridades entre los conjuntos. Existen dos posibilidades, la primera es que dos conjuntos se prioricen mutuamente y el conjunto restante tenga prioridad sobre alguno de los otros dos, y la segunda que la prioridad entre los conjuntos sea circular, es decir, que A priorice sobre B, B sobre C y C sobre A. A continuación veremos con más detalle cada uno de estos esquemas.

6.4.2.1 Esquema de Prioridad Mutua (LPPM)

En un Esquema de Prioridad Mutua existen dos conjuntos que se priorizan mutuamente, y el tercero prioriza a alguno de los otros dos, por ejemplo A prioriza a B, B prioriza a A y C prioriza a A. (Notaremos $C \succ A \succ B$). Dada una instancia I de LPPM, con prioridad mutua entre A y B, y C con prioridad sobre A, y un matching M de esa instancia, definimos que una terna (a,b,c) *bloquea* al matching M si cada uno de los elementos de la terna, o prefieren al agente de la terna que pertenece al conjunto que priorizan, o si son iguales, prefieren el par de la terna, antes que al par que le corresponde en el matching.

Luego, la terna (a,b,c) $\notin M$ bloquea a M o es una *terna bloqueante* para M si se cumplen las siguientes condiciones:

1. $b \succ_a M_B(a)$ o $(b = M_B(a) \text{ y } (b, c) \succ_a M_{BC}(a))$ y
2. $a \succ_b M_A(b)$ o $(a = M_A(b) \text{ y } (a, c) \succ_b M_{AC}(b))$ y
3. $a \succ_c M_A(c)$ o $(a = M_A(c) \text{ y } (a, b) \succ_c M_{AB}(c))$

Para el mismo esquema de prioridades podemos redefinir terna bloqueante utilizando el concepto de listas relativas, por lo que diremos que la terna (a,b,c) bloquea a M si se cumplen las siguientes condiciones:

1. $b \succ_a M_B(a)$ o $(b = M_B(a) \text{ y } c \succ_{a/b} M_C(a))$ y
2. $a \succ_b M_A(b)$ o $(a = M_A(b) \text{ y } c \succ_{b/a} M_C(b))$ y
3. $a \succ_c M_A(c)$ o $(a = M_A(c) \text{ y } b \succ_{c/a} M_B(c))$

El siguiente ejemplo muestra una instancia donde los conjuntos A y B se priorizan mutuamente.

Ejemplo 6.9

Sea $I = (A, B, C; P)$ con esquema de prioridad mutua, donde $A = \{a_0, a_1\}$, $B = \{b_0, b_1\}$ y $C = \{c_0, c_1\}$ con las siguientes listas de preferencias:

P:

a_0 : $b_0 (c_0, c_1)$; $b_1 (c_1, c_0)$

a_1 : $b_1 (c_0, c_1)$; $b_0 (c_1, c_0)$

b_0 : $a_0 (c_1, c_0)$; $a_1 (c_0, c_1)$

b_1 : $a_1 (c_0, c_1)$; $a_0 (c_0, c_1)$

c_0 : $a_1 (b_1, b_0)$; $a_0 (b_0, b_1)$

c_1 : $a_1 (b_0, b_1)$; $a_0 (b_0, b_1)$

En esta instancia el matching $M_1 = \{(a_0, b_0, c_1), (a_1, b_1, c_0)\}$ es estable, mientras que $M_2 = \{(a_1, b_0, c_1), (a_0, b_1, c_0)\}$ no es estable pues la terna (a_1, b_1, c_0) lo bloquea.

6.4.2.1.1 Búsqueda de un Matching Estable en LPPM

En este punto analizamos la forma de encontrar un matching estable en instancias LPPM, para lo cual comenzaremos enunciando la siguiente propiedad.

Propiedad 6.4

Sea I una instancia de LP con Esquema de Prioridad Mutua: $C \succ A \prec B$, y sean M_{AB} matching entre los conjuntos A y B y M_{AC} matching entre los conjuntos A y C utilizando las listas de a sobre c relativas a b

tal que $(a,b) \in M_{AB}$. Si los matchings M_{AB} y M_{AC} son estables, entonces el matching M resultante de la junta por A de ambos matchings es estable.

Demostración

Probaremos que si M_{AB} y M_{AC} , donde M_{AC} se obtuvo a partir de las listas de a sobre c relativas a b tal que $(a,b) \in M_{AB}$, son estables $\Rightarrow M = M_{AB} \cup M_{AC}$ es estable. Para ello vamos a suponer que M no es estable. Luego $\exists (a,b,c) \notin M$ tal que (a,b,c) bloquea a M . Veremos que alguno de los dos matchings, M_{AB} o M_{AC} no es estable.

Sea (a,b,c) terna bloqueante de M

\Rightarrow

$b >_a M_B(a)$ o $(b = M_B(a) \text{ y } c >_{a/b} M_C(a))$ y

$a >_b M_A(b)$ o $(a = M_A(b) \text{ y } c >_{b/a} M_C(b))$ y

$a >_c M_A(c)$ o $(a = M_A(c) \text{ y } b >_{c/a} M_B(c))$

\Rightarrow

$b >_a M_B(a)$ y $a >_b M_A(b)$ y $a >_c M_A(c)$ o

$b >_a M_B(a)$ y $a >_b M_A(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c)$ o

$b >_a M_B(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a >_c M_A(c)$ o

$b >_a M_B(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c)$ o

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a >_b M_A(b)$ y $a >_c M_A(c)$ o

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a >_b M_A(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c)$ o

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a >_c M_A(c)$ o

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c)$

Analizaremos cada uno de los 8 casos por separado:

$b >_a M_B(a)$ y $a >_b M_A(b)$ y $a >_c M_A(c) \Rightarrow b >_a M_B(a)$ y $a >_b M_A(b) \Rightarrow (a,b)$ bloquea a M_{AB} .

$b >_a M_B(a)$ y $a >_b M_A(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow b >_a M_B(a)$ y $a >_b M_A(b) \Rightarrow (a,b)$ bloquea a M_{AB} .

$b >_a M_B(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a >_c M_A(c) \Rightarrow b >_a M_B(a)$ y $a = M_A(b) \Rightarrow b >_a M_B(a)$ y $b = M_B(a)$ Absurdo.

$b >_a M_B(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow a = M_A(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow b = M_B(c)$ y $b >_{c/a} M_B(c)$ Absurdo.

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a >_b M_A(b)$ y $a >_c M_A(c) \Rightarrow b = M_B(a)$ y $a >_b M_A(b) \Rightarrow a = M_A(b)$ y $a >_b M_A(b)$ Absurdo.

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a >_b M_A(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow b = M_B(a)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow b = M_B(c)$ y $b >_{c/a} M_B(c)$ Absurdo.

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a >_c M_A(c) \Rightarrow c >_{a/b} M_C(a)$ y $a >_c M_A(c) \Rightarrow (a,c)$ bloquea a M_{AC} .

$b = M_B(a)$ y $c >_{a/b} M_C(a)$ y $a = M_A(b)$ y $c >_{b/a} M_C(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow a = M_A(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow b = M_B(c)$ y $b >_{c/a} M_B(c)$ Absurdo.

Vemos que cinco casos derivan en absurdo y los otros tres muestran que M_{AB} no es estable o M_{AC} no es estable. ♦

A continuación presentamos un algoritmo que encuentra un matching estable en LPPM, basado en esta propiedad. La implicación inversa de esta propiedad no es válida, lo cual demostraremos a través de un ejemplo al finalizar el algoritmo.

Algoritmo 6.3

El algoritmo está basado en la Propiedad 6.4 y utiliza el algoritmo de Gale y Shapley para encontrar un matching entre dos conjuntos. El mismo encuentra un matching estable para instancias LPPM.

Sean A , B y C los conjuntos de agentes con igual cardinalidad, y sea el esquema de prioridades: $C \succ A \succ B$. El algoritmo parte de las preferencias del conjunto que es priorizado por los dos conjuntos restantes, en este caso es el conjunto A .

- 1) Obtener un matching M_{AB} entre los agentes de A y los de B a través del algoritmo de Gale y Shapley para grafos bipartitos y utilizando las listas de preferencias únicas de A hacia B y de B hacia A .
- 2) Armar las listas de preferencias de a_i hacia c_k relativas a b_j para cada par (a_i, b_j) del matching resultante de 2).
- 3) Obtener un matching M_{AC} entre los agentes de A y los de C a través del algoritmo de Gale y Shapley para grafos bipartitos y utilizando las listas de preferencias de A hacia C obtenidas en el paso anterior, y las listas de preferencias únicas de C hacia A .
- 4) $M = M_{AB} \cup M_{AC}$ es estable.

Teorema

El Algoritmo 6.3 es correcto y completo.

Demostración

Demostración de correctitud

El Algoritmo 6.3 encuentra un matching estable pues utiliza la Propiedad 6.4.

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que exista solución el algoritmo la encuentra.

La utilización del algoritmo de Gale y Shapley para hallar un matching estable en instancias de listas por pares con prioridades mutuas, garantiza la existencia del mismo, por lo que el algoritmo siempre termina y encuentra solución. ♦

Corolario

Siempre existe un matching estable bajo el esquema de prioridades mutuas.

Complejidad e Implementación

En una primera fase, el Algoritmo 6.3 busca a través del algoritmo de Gale y Shapley, un matching entre los dos conjuntos que se prefieren mutuamente, digamos A y B , y como sabemos, la complejidad es de $O(n^2)$.

En una segunda fase, busca a través del algoritmo de Gale y Shapley un matching entre el tercer conjunto, digamos C y el conjunto por el que éste tiene prioridad, por ejemplo A . Nuevamente de $O(n^2)$.

Como la junta y el armado de las listas relativas no aumentan la complejidad, la complejidad del Algoritmo 6.3 es de $O(n^2)$.

Para implementar de forma eficiente el algoritmo de Gale y Shapley, también se utilizan las matrices de ranking de B y C que quedarían definidas de la siguiente forma:

$R_B [b,a] = r \Leftrightarrow$ el agente a ocupa la posición r en la lista de preferencias del agente b.

$R_C [c,a] = r \Leftrightarrow$ el agente a ocupa la posición r en la lista de preferencias del agente c.

6.4.2.1.2 El conjunto de todos los Matchings Estables

La formulación lineal entera para definir el conjunto de todos los matchings estables en instancias LPPM queda definida de la siguiente forma:

$$\text{Sea } x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j \in C \\ a > b \\ j > a}} x_{a,i,j} + \sum_{\substack{k > c \\ a}} x_{a,b,k} + \sum_{\substack{l > a \\ b \\ m \in C}} x_{l,b,m} + \sum_{\substack{n > a \\ b}} x_{a,b,n} + \sum_{\substack{p > a \\ c \\ q \in B}} x_{p,q,c} + \sum_{\substack{r > b \\ c}} x_{a,r,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

La diferencia entre la formulación lineal entera para listas por pares dada en 6.3 y la definida para instancias LPPM en el párrafo anterior, está dada por la penúltima restricción, la cual garantiza la estabilidad. Si bien la restricción de estabilidad dada para LP en general aplica a instancias LPPM, esta nueva restricción es más específica para este tipo de instancias.

En cuanto a la representación ordenada de los matchings estables, existe una relación entre el orden de los matchings proyectados de dos dimensiones y los pertenecientes a instancias LPPM, que explica la siguiente propiedad.

Propiedad 6.5

Sean M y M' matchings estables en LPPM con el esquema $G \rightarrow A \leftarrow B$ y sean M_{AB} y M'_{AB} proyecciones de M y de M' respectivamente, que son matchings estables sobre las listas únicas de A hacia B y M_{AC} y M'_{AC} proyecciones de M y de M' respectivamente, que son matchings estables sobre las listas de a_i sobre c_k , relativas a b_j con $(a_i, b_j) \in M_{AB}$. Si $M_{AB} = M'_{AB}$ y $M_{AC} \geq_A M'_{AC} \Rightarrow M \geq_A M'$

Demostración

Si $M_{AB} = M'_{AB}$ y $M_{AC} \geq_A M'_{AC} \Rightarrow \forall a \in A M_B(a) = M'_B(a)$ y $M_C(a) \geq_{a/MB(a)} M'_C(a) \Rightarrow \forall a \in A M_{BC}(a) \geq_a M'_{BC}(a)$ pues A prioriza sobre B, y dentro de la lista relativa de cada B rige el orden de los $c \in C \Rightarrow M \geq_A M'$. ♦

Veamos mediante un ejemplo, la representación ordenada del conjunto de todos los matchings estables para una instancia dada

Ejemplo 6.10

Sea $I = (A, B, C; P)$ una instancia con listas LPPM, donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$ y $C = \{c_0, c_1, c_2\}$ y P como sigue:

P:

a_0 : $b_0(c_0, c_1, c_2); b_2(c_2, c_1, c_0); b_1(c_1, c_2, c_0)$
 a_1 : $b_1(c_0, c_2, c_1); b_0(c_2, c_1, c_0); b_2(c_1, c_2, c_0)$
 a_2 : $b_0(c_0, c_1, c_2); b_1(c_2, c_0, c_1); b_2(c_1, c_2, c_0)$

b_0 : $a_1(c_0, c_2, c_1); a_2(c_1, c_2, c_0); a_0(c_1, c_2, c_0)$
 b_1 : $a_2(c_1, c_0, c_2); a_0(c_2, c_1, c_0); a_1(c_1, c_2, c_0)$
 b_2 : $a_0(c_2, c_0, c_1); a_2(c_1, c_2, c_0); a_1(c_1, c_2, c_0)$

c_0 : $a_1(b_2, b_1, b_0); a_2(b_0, b_1, b_2); a_0(b_1, b_2, b_0)$
 c_1 : $a_0(b_1, b_0, b_2); a_2(b_2, b_1, b_0); a_1(b_0, b_2, b_1)$
 c_2 : $a_1(b_2, b_0, b_1); a_2(b_1, b_2, b_0); a_0(b_0, b_2, b_1)$

Sea $S(I)$ el conjunto de los matchings estables:

$M_1 = \{ (a_0, b_2, c_0), (a_1, b_1, c_2), (a_2, b_0, c_1) \}$
 $M_2 = \{ (a_0, b_2, c_1), (a_1, b_1, c_0), (a_2, b_0, c_2) \}$
 $M_3 = \{ (a_0, b_2, c_1), (a_1, b_1, c_2), (a_2, b_0, c_0) \}$
 $M_4 = \{ (a_0, b_2, c_2), (a_1, b_1, c_1), (a_2, b_0, c_0) \}$
 $M_5 = \{ (a_0, b_2, c_2), (a_1, b_1, c_0), (a_2, b_0, c_1) \}$
 $M_6 = \{ (a_0, b_2, c_1), (a_1, b_0, c_0), (a_2, b_1, c_2) \}$
 $M_7 = \{ (a_0, b_2, c_1), (a_1, b_0, c_2), (a_2, b_1, c_0) \}$
 $M_8 = \{ (a_0, b_2, c_2), (a_1, b_0, c_0), (a_2, b_1, c_1) \}$

Desde el punto de vista del conjunto A, la relación de dominación entre los matchings estables es como se ve en la siguiente figura:

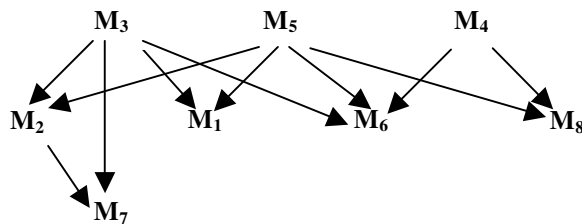


Figura 6.5 Grafo dirigido de todos los matchings estables según el conjunto A

Cabe aclarar que los matchings M_2 y M_5 son los únicos que tienen proyecciones que son matchings estables en dos dimensiones para la instancia del Ejemplo 6.10. Además dichos matchings cumplen que $M_{5AB} = M_{2AB}$ y $M_{5AC} \geq_a M_{2AC} \forall a \in A$, por lo cual según la Propiedad 6.5, $M_5 \geq M_2$.

Propiedad 6.6

La cantidad de matchings estables de una instancia de LPPM crece exponencialmente en función del tamaño de dicha instancia.

Demostración

Sea $I=(A,B,C;P)$ una instancia de LPPM de tamaño n , con esquema de prioridad mutua donde $A \succ B$, $B \succ A$ y $C \succ A$, y sean M' matching entre los conjuntos A y B y M'' matching entre los conjuntos A y C utilizando las listas de a sobre c relativas a b tal que $(a,b) \in M_{AB}$. Según la propiedad 6.4, si los matchings M_{AB} y M_{AC} son matchings estables bidimensionales, entonces el matching M resultante de la junta por A de ambos matchings es estable.

Vimos en el Capítulo 2 que la cantidad de matchings estables bidimensionales crece exponencialmente en función del tamaño de la instancia, entonces el conjunto de soluciones de I' y de I'' crece exponencialmente en función de n y, por consiguiente, también el conjunto de soluciones de I . ♦

6.4.2.2 Esquema de Prioridad Circular (LPPC)

En un Esquema de Prioridad Circular se produce un "ciclo" en las prioridades de las listas, es decir que A prioriza a B , B prioriza a C y C prioriza a A . (Notaremos: $A \succ B \succ C \succ A$). Dada una instancia I de LPPC, con prioridad circular entre A , B , y C , y un matching M de esa instancia, definimos que una terna (a,b,c) *bloquea* al matching M si cada uno de los elementos de la terna, o prefieren al agente de la terna que pertenece al conjunto que priorizan, o si son iguales, prefieren el par de la terna, antes que al par que le corresponde en el matching.

Luego la terna $(a,b,c) \notin M$ bloquea M o es una *terna bloqueante* para M si se cumplen las siguientes condiciones:

1. $b \succ_a M_B(a)$ o $(b = M_B(a) \text{ y } (b, c) \succ_a M_{BC}(a))$ y
2. $c \succ_b M_C(b)$ o $(c = M_C(b) \text{ y } (a, c) \succ_b M_{AC}(b))$ y
3. $a \succ_c M_A(c)$ o $(a = M_A(c) \text{ y } (a, b) \succ_c M_{AB}(c))$

Para el mismo esquema de prioridades podemos redefinir terna bloqueante utilizando el concepto de listas relativas, por lo que diremos que (a,b,c) bloquea al matching M si se cumplen las siguientes condiciones:

1. $b \succ_a M_B(a)$ o $(b = M_B(a) \text{ y } c \succ_{a/b} M_C(a))$ y
2. $c \succ_b M_C(b)$ o $(c = M_C(b) \text{ y } a \succ_{b/c} M_A(b))$ y
3. $a \succ_c M_A(c)$ o $(a = M_A(c) \text{ y } b \succ_{c/a} M_B(c))$

El siguiente ejemplo muestra una instancia con esquema de prioridad circular.

Ejemplo 6.11

Sea $I = (A,B,C;P)$ con esquema de prioridad circular, donde $A = \{a_0, a_1\}$, $B = \{b_0, b_1\}$ y $C = \{c_0, c_1\}$ con las siguientes listas de preferencias:

P:

a_0 : $b_0 (c_0, c_1)$; $b_1 (c_1, c_0)$

a_1 : $b_1 (c_0, c_1)$; $b_0 (c_1, c_0)$

b_0 : $c_0 (a_1, a_0)$; $c_1 (a_0, a_1)$

b_1 : $c_1 (a_1, a_0)$; $c_0 (a_0, a_1)$

c_0 : $a_1 (b_1, b_0)$; $a_0 (b_1, b_0)$

$c_1: a_1(b_0, b_1); a_0(b_0, b_1)$

En este ejemplo, el matching $M_1 = \{(a_0, b_0, c_1), (a_1, b_1, c_0)\}$ es estable, mientras que $M_2 = \{(a_0, b_1, c_1), (a_1, b_0, c_0)\}$ no es estable pues es bloqueado por la terna (a_1, b_1, c_1) .

Conjeturamos que el problema de hallar un matching estable en instancias de listas por pares y prioridad circular es NP-Completo.

6.4.2.2.1 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir la estructura del conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones. La formulación entera para definir la estructura en instancias LPPC queda definida de la siguiente forma:

$$\text{Sea } x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j \in C \\ a > b}} x_{a,i,j} + \sum_{\substack{k > c \\ a}} x_{a,b,k} + \sum_{\substack{l > c \\ b \\ m \in A}} x_{m,b,l} + \sum_{\substack{n > a \\ b}} x_{n,b,c} + \sum_{\substack{p > a \\ c \\ q \in B}} x_{p,q,c} + \sum_{\substack{r > b \\ c}} x_{a,r,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

En cuanto a la representación del conjunto de todos los matchings estables, valen las consideraciones generales dadas en 6.3 para LP.

6.4.2.3 Relación con Instancias de Listas Únicas

En el punto anterior se trataron las instancias de LP con prioridad, de las cuales pudimos obtener mayores resultados para las que tienen esquema de prioridad mutua (LPPM). En este punto analizamos la relación entre las instancias de LU y LP con prioridad desde el punto de vista de las soluciones estables en ambas clases de instancias. Esto es útil principalmente para poder hallar un matching estable en instancias con esquema de prioridad circular (LPPC). Comenzaremos definiendo cómo mapear una instancia a partir de la otra y luego daremos propiedades y un algoritmo no completo para hallar un matching estable en instancias LPPC.

Dada una instancia I del problema de hallar matching estable en casos con listas de preferencias con prioridades, definiremos *instancia abarcativa* a la instancia del problema de listas únicas conformado por

las listas de preferencias del conjunto priorizado por cada individuo en listas por pares. Si la instancia de listas por pares tiene esquema de prioridad mutua (LPPM) la instancia abarcativa correspondiente, será una instancia de listas únicas mutuas (LUM), en cambio si la instancia de listas por pares tiene esquema de prioridad circular (LPPC), la instancia abarcativa correspondiente, será una instancia de listas únicas circular (LUC).

Los siguientes ejemplos muestran las instancias abarcativas correspondientes a instancias LPPM y LPPC.

Ejemplo 6.12

Instancia abarcativa LUM correspondiente a la instancia LPPM del Ejemplo 6.10.

$a_0: (b_0, b_2, b_1)$
 $a_1: (b_1, b_0, b_2)$
 $a_2: (b_0, b_1, b_2)$

$b_0: (a_1, a_2, a_0)$
 $b_1: (a_2, a_0, a_1)$
 $b_2: (a_0, a_2, a_1)$

$c_0: (a_1, a_2, a_0)$
 $c_1: (a_0, a_2, a_1)$
 $c_2: (a_1, a_2, a_0)$

Ejemplo 6.13

Instancia abarcativa LUC correspondiente a la instancia LCPC del Ejemplo 6.11.

$a_0: (b_0, b_1)$ $b_0: (c_0, c_1)$ $c_0: (a_1, a_0)$
 $a_1: (b_1, b_0)$ $b_1: (c_1, c_0)$ $c_1: (a_1, a_0)$

Nótese que una misma instancia abarcativa de LUM (LUC), le puede corresponder a muchas instancias de LPPM (LPPC).

Las propiedades enunciadas a continuación, permiten relacionar el espacio de soluciones de ambas clases de instancias.

Propiedad 6.7

M es un matching estable de una instancia I de LPPM $\Rightarrow M$ es estable en la instancia abarcativa I' de LUM.

Demostración

Demostrar que si M es un matching estable de una instancia de LPPM $\Rightarrow M$ es estable en la instancia abarcativa I' de LUM, es equivalente a demostrar que si M no es estable en la instancia abarcativa I' de LUM $\Rightarrow M$ no es un matching estable en la instancia I de LPPM. Supongamos que la instancia I de LPPM tiene el siguiente esquema de prioridad $C \succ A \prec B$.

Como M no es matching estable de la instancia abarcativa I' , que es una instancia de LUM entonces $\exists (a,b,c)$ que bloquea al matching M tal que:

$$b \succ_a M_B(a) \text{ y } a \succ_b M_A(b) \text{ y } a \succ_c M_A(c) \quad (1)$$

Por otro lado, la definición de terna bloqueante en LPPM dice que la terna $(a,b,c) \notin M$ bloquea a M si se cumplen las siguientes condiciones:

1. $b \succ_a M_B(a)$ o $(b = M_B(a) \text{ y } (b, c) \succ_a M_{BC}(a))$ y
2. $a \succ_b M_A(b)$ o $(a = M_A(b) \text{ y } (a, c) \succ_b M_{AC}(b))$ y
3. $a \succ_c M_A(c)$ o $(a = M_A(c) \text{ y } (a, b) \succ_c M_{AB}(c))$

(2)

Luego, podemos ver que (1) implica (2). ♦

Propiedad 6.8

Si M es un matching estable de una instancia I de LPPC $\Rightarrow M$ es estable en la instancia abarcativa I' de LUC

Demostración

Demostrar que si M es un matching estable de una instancia de LPPC $\Rightarrow M$ es estable en la instancia abarcativa I' de LUC, es equivalente a demostrar que si M no es estable en la instancia abarcativa I' de LUC $\Rightarrow M$ no es un matching estable en la instancia I de LPPC. Como M no es matching estable de la instancia abarcativa I' , que es una instancia de LUC, entonces $\exists (a, b, c)$ que bloquea al matching M tal que:

$$b \succ_a M_B(a) \text{ y } c \succ_b M_C(b) \text{ y } a \succ_c M_A(c) \quad (3)$$

Por otro lado, la definición de terna bloqueante en LPPC dice que la terna $(a,b,c) \notin M$ bloquea a M si

1. $b \succ_a M_B(a)$ o $(b = M_B(a) \text{ y } (b, c) \succ_a M_{BC}(a))$ y
2. $c \succ_b M_C(b)$ o $(c = M_C(b) \text{ y } (a, c) \succ_b M_{AC}(b))$ y
3. $a \succ_c M_A(c)$ o $(a = M_A(c) \text{ y } (a, b) \succ_c M_{AB}(c))$

(4)

Luego, podemos ver que (3) implica (4). ♦

Con el siguiente contraejemplo mostramos que un matching M puede ser estable en la instancia abarcativa, y no ser estable en su instancia de listas por pares con prioridad correspondiente.

Ejemplo 6.14

Sea I instancia de LPPC:

$a_0: (b_1 c_0) (b_1 c_1) (b_0 c_0) (b_0 c_1)$
 $a_1: (b_1 c_1) (b_1 c_0) (b_0 c_1) (b_0 c_0)$

$b_0: (a_1 c_1) (a_0 c_1) (a_1 c_0) (a_0 c_0)$
 $b_1: (a_0 c_1) (a_1 c_1) (a_0 c_0) (a_1 c_0)$

$c_0: (a_1 b_0) (a_1 b_1) (a_0 b_0) (a_0 b_1)$
 $c_1: (a_1 b_0) (a_1 b_1) (a_0 b_0) (a_0 b_1)$

La instancia abarcativa I' de LUC correspondiente:

$a_0: b_1 b_0 b_0: c_1 c_0 c_0: a_1 a_0$
 $a_1: b_1 b_0 b_1: c_1 c_0 c_1: a_1 a_0$

Observemos que $M = \{(a_0, b_1, c_1), (a_1, b_0, c_0)\}$ es matching estable de I' pero no de I . En cambio $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1)\}$ es matching estable de ambas instancias.

6.4.2.3.1 Búsqueda de un Matching Estable

En este punto daremos una propiedad que será la base del algoritmo para encontrar un matching estable en instancias de LPPC.

Propiedad 6.9

Para una instancia dada del problema LPPC, si un matching M cumple los cuatro siguientes puntos, entonces M es estable en LPPC.

- 1) M es matching estable de la instancia abarcativa correspondiente de LUC.
- 2) El matching M_{AB} (proyección AB de M) es estable, donde las listas de preferencias de A hacia B derivan directamente de la instancia LPPC y las listas de B hacia A son las correspondientes listas de cada b_j ($0 \leq j \leq n-1$) relativas a $M_C(b_j)$.
- 3) El matching M_{BC} (proyección BC de M) es estable, donde las listas de preferencias de B hacia C derivan directamente de la instancia LPPC y las listas de C hacia B son las correspondientes listas de cada c_k ($0 \leq k \leq n-1$) relativas a $M_A(c_k)$.
- 4) El matching M_{AC} (proyección AC de M) es estable, donde las listas de preferencias de C hacia A derivan directamente de la instancia LPPC y las listas de A hacia C son las correspondientes listas de cada a_i ($0 \leq i \leq n-1$) relativas a $M_B(a_i)$.

Demostración

Demostrar que M estable en la instancia abarcativa LUC y M_{AB} estable en AB y M_{BC} estable en BC y M_{AC} estable en AC $\Rightarrow M$ estable en LPPC, es equivalente a probar lo siguiente:

Si $\exists(a,b,c) \notin M$ tal que bloquea a M en LPPC $\Rightarrow \exists(a',b',c')$ que bloquea a M en LUC o $\exists(a'',b'')$ que bloquea a M_{AB} en la instancia AB o $\exists(b''',c''')$ que bloquea a M_{BC} en la instancia BC o $\exists(a''',c''')$ que bloquea a M_{AC} en la instancia AC. En particular, probaremos que $a=a'=a''=a'''$ y $b=b'=b''=b'''$ y $c=c'=c''=c'''$.

(a,b,c) bloquea a M en LPPC

\Leftrightarrow

$[b >_a M_B(a) \text{ o } (b = M_B(a) \text{ y } c >_{a/b} M_C(a))]$ y

$[c >_b M_C(b) \text{ o } (c = M_C(b) \text{ y } a >_{b/c} M_A(b))]$ y

$[a >_c M_A(c) \text{ o } (a = M_A(c) \text{ y } b >_{c/a} M_B(c))]$

\Leftrightarrow

1) $[b >_a M_B(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c)]$ o

2) $[b >_a M_B(a) \text{ y } c >_b M_C(b) \text{ y } a = M_A(c) \text{ y } b >_{c/a} M_B(c)]$ o

3) $[b >_a M_B(a) \text{ y } c = M_C(b) \text{ y } a >_{b/c} M_A(b) \text{ y } a >_c M_A(c)]$ o

4) $[b >_a M_B(a) \text{ y } c = M_C(b) \text{ y } a >_{b/c} M_A(b) \text{ y } a = M_A(c) \text{ y } b >_{c/a} M_B(c)]$ o

5) $[b = M_B(a) \text{ y } c >_{a/b} M_C(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c)]$ o

6) $[b = M_B(a) \text{ y } c >_{a/b} M_C(a) \text{ y } c >_b M_C(b) \text{ y } a = M_A(c) \text{ y } b >_{c/a} M_B(c)]$ o

7) $[b = M_B(a) \text{ y } c >_{a/b} M_C(a) \text{ y } c = M_C(b) \text{ y } a >_{b/c} M_A(b) \text{ y } a >_c M_A(c)]$ o

8) $[b = M_B(a) \text{ y } c >_{a/b} M_C(a) \text{ y } c = M_C(b) \text{ y } a >_{b/c} M_A(b) \text{ y } a = M_A(c) \text{ y } b >_{c/a} M_B(c)]$

\Leftrightarrow

1) $[b >_a M_B(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c)]$ o

2) $[b >_{M_A(c)} M_B(M_A(c)) \text{ y } c >_b M_C(b) \text{ y } b >_{c/a} M_B(c)]$ o

3) $[b >_a M_B(a) \text{ y } a >_{b/c} M_A(b) \text{ y } a >_{M_C(b)} M_A(M_C(b))]$ o

4) $[b >_{M_A(M_C(b))} M_B(M_A(M_C(b))) \text{ y } M_A(c) >_{b/c} M_A(b) \text{ y } b >_{c/a} M_B(c)]$ o

5) $[c >_{a/b} M_C(a) \text{ y } c >_{M_B(a)} M_C(M_B(a)) \text{ y } a >_c M_A(c)]$ o

6) $[c >_{M_A(c)/b} M_C(M_A(c)) \text{ y } c >_{M_B(M_A(c))} M_C(M_B(M_A(c))) \text{ y } M_B(M_A(c)) >_{c/a} M_B(c)]$ o

- 7) [$M_C(b) >_{a/b} M_C(a)$ y $a >_{M_B(a)/c} M_A(M_B(a))$ y $a >_{M_C(b)} M_A(M_C(b))$] o
 8) [$M_C(b) >_{M_A(M_C(b))/b} M_C(M_A(M_C(b)))$ y $M_A(M_C(b)) >_{M_B(M_A(M_C(b)))/c} M_A(M_B(M_A(M_C(b))))$ y $M_B(M_A(M_C(b))) >_{M_C(b)/a} M_B(M_C(b))$]

\Leftrightarrow

- 1) [$b >_a M_B(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$] o
 2) [$b >_{M_A(c)} M_B(c)$ y $c >_b M_C(b)$ y $b >_{c/a} M_B(c)$] o
 3) [$b >_a M_B(a)$ y $a >_{b/c} M_A(b)$ y $a >_{M_C(b)} M_A(b)$] o
 4) [$b >_{M_A(b)} b$ y $M_A(c) >_{b/c} M_A(b)$ y $b >_{c/a} M_B(c)$] o
 5) [$c >_{a/b} M_C(a)$ y $c >_{M_B(a)} M_C(a)$ y $a >_c M_A(c)$] o
 6) [$c >_{M_A(c)} c$ y $c >_{M_A(c)} c$ y $M_B(c) >_{c/a} M_B(c)$] o
 7) [$M_C(b) >_{a/b} M_C(a)$ y $a >_{M_B(a)/c} a$ y $a >_{M_C(b)} M_A(M_C(b))$] o
 8) [$M_C(b) >_{M_A(b)/b} M_C(b)$ y $M_A(b) >_{b/c} M_A(b)$ y $b >_{M_C(b)/a} b$]

Como 4, 6, 7 y 8 son absurdos, tenemos

- 1) [$b >_a M_B(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$] o
 2) [$b >_{M_A(c)} M_B(c)$ y $c >_b M_C(b)$ y $b >_{c/a} M_B(c)$] o
 3) [$b >_a M_B(a)$ y $a >_{b/c} M_A(b)$ y $a >_{M_C(b)} M_A(b)$] o
 4) [$c >_{a/b} M_C(a)$ y $c >_{M_B(a)} M_C(a)$ y $a >_c M_A(c)$]

Si se cumple 1) [$b >_a M_B(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$] \Rightarrow (a,b,c) bloquea a M en LUC.

Si se cumple 2) [$b >_{M_A(c)} M_B(c)$ y $c >_b M_C(b)$ y $b >_{c/a} M_B(c)$] $\Rightarrow c >_b M_C(b)$ y $b >_{c/a} M_B(c)$ \Rightarrow (b,c) bloquea a M en M_{BC} con las listas de C sobre B relativas al correspondiente a.

Si se cumple 3) [$b >_a M_B(a)$ y $a >_{b/c} M_A(b)$ y $a >_{M_C(b)} M_A(b)$] $\Rightarrow b >_a M_B(a)$ y $a >_{b/c} M_A(b)$ \Rightarrow (a,b) bloquea a M en M_{AB} con las listas de B sobre A relativas al correspondiente c.

Si se cumple 5) [$c >_{a/b} M_C(a)$ y $c >_{M_B(a)} M_C(a)$ y $a >_c M_A(c)$] $\Rightarrow c >_{a/b} M_C(a)$ y $a >_c M_A(c)$ \Rightarrow (a,c) bloquea a M en M_{AC} con las listas de A sobre C relativas al correspondiente b.

Luego, (a,b,c) no perteneciente a M bloquea a M en LPPC \Rightarrow (a,b,c) bloquea a M en LUC o (a,b) bloquea a M_{AB} en AB o (b,c) bloquea a M_{BC} en BC o (a,c) bloquea a M_{AC} en AC. \blacklozenge

Esta propiedad es condición suficiente pero no necesaria para encontrar un matching estable en LPPC a partir de los matchings en dos dimensiones o en LUC. El siguiente contraejemplo muestra una instancia I de LPPC con un matching estable M que es estable en su instancia abarcativa LUC de I, pero no es estable en alguna de las instancias de dos conjuntos.

Ejemplo 6.15

Sea $I=(A,B,C;P)$ instancia de LPPC con $A=\{a_0, a_1\}$, $B=\{b_0, b_1\}$, $C=\{c_0, c_1\}$ con las siguientes listas de preferencias:

P:

a_0 : $b_1 (c_0, c_1)$; $b_0 (c_0, c_1)$

a_1 : $b_1 (c_1, c_0)$; $b_0 (c_1, c_0)$

b_0 : $c_1 (a_1, a_0)$; $c_0 (a_1, a_0)$

b_1 : $c_1 (a_0, a_1)$; $c_0 (a_0, a_1)$

c_0 : $a_1 (b_0, b_1)$; $a_0 (b_0, b_1)$

c_1 : $a_1 (b_1, b_0)$; $a_0 (b_1, b_0)$

El único matching estable es $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1)\}$, pues $M_2 = \{(a_0, b_1, c_1), (a_1, b_0, c_0)\}$ es bloqueado por la terna (a_1, b_0, c_1) y los matchings $M_3 = \{(a_0, b_1, c_0), (a_1, b_0, c_1)\}$ y $M_4 = \{(a_1, b_1, c_0), (a_0, b_0, c_1)\}$ son bloqueados por la terna (a_1, b_1, c_1) .

Mapeando la instancia I a la instancia abarcativa LUC I' :

P' :

$a_0: b_1, b_0; c_1, c_0; a_1, a_0$

$a_1: b_1, b_0; c_1, c_0; a_1, a_0$

El matching M_1 es solución de esta instancia de LUC pues: ni b_1 prefiere el agente c_0 al agente c_1 , ni el agente c_1 prefiere el agente a_0 por sobre a_1 , ni el agente a_1 prefiere a_0 antes que a_1 .

Si para algún par $(a, b) \in M_{AB}$, $(b, c) \in M_{BC}$ o $(c, a) \in M_{CA}$ encontramos que la proyección de la solución en dos dimensiones no es estable, queda demostrado que aunque exista solución, no siempre se cumple la propiedad. Proyectando las columnas correspondientes a los conjuntos A y B de la solución M_1 en LPPC se obtiene $M_1' = \{(a, b_0), (a_1, b_1)\}$. Por otro lado, las listas correspondientes en dos dimensiones están dadas por:

P' :

$a_0: b_1, b_0$

$b_0: a_1, a_0$

$a_1: b_1, b_0$

$b_1: a_0, a_1$

Luego el matching M_1' no es estable pues el par (a, b_1) bloquea al matching.

Con este ejemplo pudimos ver que el matching M_1 es estable en LPPC y en LUC, pero su proyección a dos dimensiones M_{AB} no lo es.

A continuación presentamos un algoritmo que utiliza la Propiedad 6.9 para encontrar un matching estable en instancias LPPC.

Algoritmo 6.4

Sean los conjuntos A, B y C con listas de preferencias por pares, con esquema de prioridades circular (LPPC).

Dados como entrada todas las soluciones de la instancia abarcativa en LUC, cuyos conjuntos denominamos A, B y C.

1) Para cada solución M de la instancia abarcativa en LUC.

1.1) Para cada par de conjuntos: AB, BC y CA .

1.1.1) Obtener M_{AB} (proyección AB de M), donde las listas de preferencias de A hacia B derivan directamente de la instancia LPPC y las listas de B hacia A son las correspondientes listas de cada b_j ($0 \leq j \leq n-1$) relativas a $M_C(b_j)$.

1.1.2) Obtener M_{BC} (proyección BC de M), donde las listas de preferencias de B hacia C derivan directamente de la instancia LPPC y las listas de C hacia B son las correspondientes listas de cada c_k ($0 \leq k \leq n-1$) relativas a $M_A(c_k)$.

1.1.3) Obtener M_{AC} (proyección AC de M), donde las listas de preferencias de C hacia A derivan directamente de la instancia LPPC y las listas de A hacia C son las correspondientes listas de cada a_i ($0 \leq i \leq n-1$) relativas a $M_B(a_i)$.

1.2) Si M_{AB} , M_{BC} , y M_{AC} son estables en sus respectivas instancias bidimensionales, el matching M es estable en LPPC.

Teorema

El Algoritmo 6.4 es correcto.

Demostración

Como el algoritmo especifica la Propiedad 6.9, anteriormente demostrada, puede verse fácilmente que la solución es estable en LPPC. ♦

Complejidad e Implementación

El orden del Algoritmo 6.4 es equivalente al orden del algoritmo que verifica la estabilidad de un matching en una instancia de tamaño n . Tal como se vio al comienzo del capítulo, la complejidad de verificar la estabilidad, es de $O(n^3)$.

Sea m la cantidad de soluciones de la instancia abarcativa en LUC del problema original de LPPC. Como para cada par de conjuntos AB , BC , y CA debo verificar estabilidad $O(n^3)$, el orden del algoritmo es $m \cdot 3 \cdot O(n^3) \cong m \cdot O(n^3)$.

Capítulo 7: Matching Estable Tridimensional sobre tres Conjuntos con Listas Combinadas

En este capítulo presentamos los resultados del análisis del problema de hallar un matching estable tridimensional sobre tres conjuntos con listas de preferencias combinadas. Estas listas de preferencias combinan los tipos de listas vistos hasta el momento. Nuestro objetivo es definir y analizar qué ocurre con instancias donde los individuos en los distintos conjuntos tienen listas de distintas estructuras, es decir listas únicas, simples o por pares.

Definiremos una clasificación de estas instancias, y analizaremos todas las variantes definidas dando algoritmos y propiedades.

7.1 Introducción

Las clases de instancias que vimos hasta el momento tienen la particularidad de tener la misma estructura de listas de preferencias para los tres conjuntos. Pero estas clases no permiten modelar problemas donde las estructuras para los tres conjuntos difiera en el tipo de lista, por ejemplo, si uno de los tres conjuntos tuviera listas de preferencias por pares, y los otros dos tuvieran listas de preferencias únicas sobre alguno de los dos conjuntos restantes. También podría presentarse el caso que los tres conjuntos tengan distintas estructuras. Como paso inicial de nuestro análisis, debemos clasificar las instancias según el formato de las listas para los agentes en cada conjunto. Esta clasificación forma una partición considerando todo el espectro de combinación de listas, es decir que cualquier problema que combine los tipos de listas de preferencias definidos podrá ser asignado a alguna de las siguientes categorías:

- USP: Los individuos en uno de los conjuntos tienen listas únicas, en el segundo conjunto tienen listas simples mientras que en el tercero tienen listas por pares.
- UUS: Los individuos en dos de los conjuntos tienen listas únicas, mientras que en el tercer conjunto tienen listas simples.
- UUP: Los individuos en dos de los conjuntos tienen listas únicas, mientras que los del tercer conjunto tienen listas por pares.
- SSU: Los individuos en dos de los conjuntos tienen listas simples, mientras que los individuos en el tercer conjunto tienen listas únicas.
- SSP: Los individuos en dos de los conjuntos tienen listas simples, mientras que los individuos en el tercer conjunto tienen listas por pares.
- PPU: Los individuos en dos de los conjuntos tienen listas por pares, mientras que los individuos en el tercer conjunto tienen listas únicas.
- PPS: Los individuos en dos de los conjuntos tienen listas por pares, mientras que los individuos en el tercer conjunto tienen listas simples.

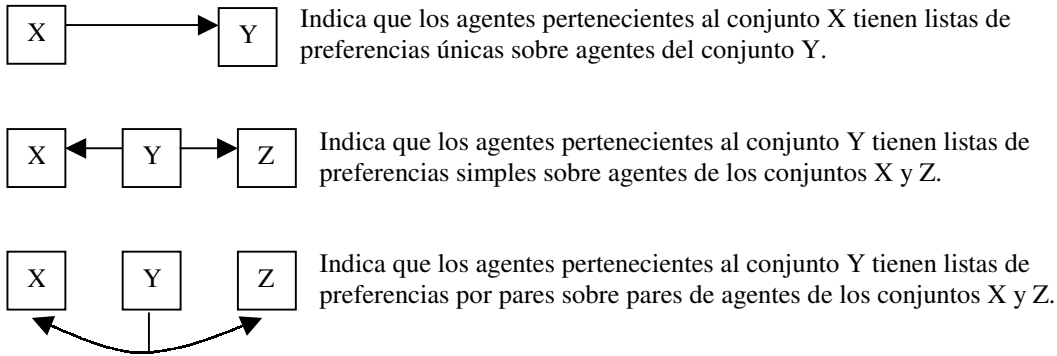
Con fines exclusivamente mnemotécnicos, damos nombre a la clase en función del formato de las listas. Por ejemplo: USP indica que uno de los conjuntos tiene listas **U**nicas, otro tiene **S**imples y el tercero tiene listas por **P**ares.

Para los casos con listas por pares, vimos en el capítulo anterior una característica que permite tratar particularmente un subconjunto de las instancias posibles. Esta característica es la de priorizar en las listas de preferencias a los individuos de uno de los conjuntos. Para los casos en que tratemos con listas por pares incluiremos el estudio de este tipo de instancias.

Esta división del problema en distintas familias tiene por único objetivo simplificar el análisis. Durante el estudio de estos casos se harán otras subdivisiones para dar características propias de cada problema tratado.

Para cada uno de los casos que trataremos a continuación daremos la definición de *terna bloqueante*, respetando la definición de matching estable como aquel matching que no tiene ternas bloqueantes y ejemplificaremos cada caso.

También mostraremos un gráfico que describe la relación entre los tres conjuntos, utilizando los siguientes trazos:



7.2 USP

Una instancia del problema de USP está formada por tres conjuntos A, B y C y un conjunto P de listas de preferencias, con las siguientes estructuras:

- $\forall a \in A$, a tiene listas de preferencias únicas,
- $\forall b \in B$, b tiene listas de preferencias simples y
- $\forall c \in C$, c tiene listas de preferencias por pares.

En función del conjunto sobre el cual los agentes del conjunto A expresen sus preferencias, podemos subdividir esta familia en 2 subfamilias:

- $U_{S}SP$: Las listas únicas se expresan sobre el conjunto B, el cual define sus preferencias como listas simples.
- $U_{P}SP$: Las listas únicas se expresan sobre el conjunto C, el cual define sus preferencias como listas por pares.

Definamos y analicemos cada caso por separado:

7.2.1 $U_{S}SP$

Recordemos que, en instancias de este tipo, los conjuntos A, B y C tienen las siguientes preferencias: A elige sobre B, B elige sobre ambos conjuntos independientemente y C elige sobre pares de A y B.

En la Figura 7.1 podemos ver representada esta relación:

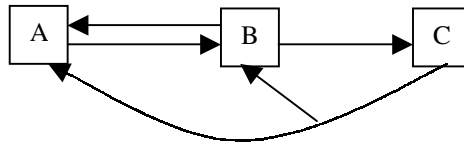


Figura 7.1 Preferencias entre los conjuntos en una instancia U_SSP .

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b >_a M_B(a)$ y
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b))$ o $(a >_b M_A(b) \text{ y } c \geq_b M_C(b))$ y
3. $(a, b) >_c M_{AB}(c)$

Veamos un ejemplo de una instancia de U_SSP , para la cual se muestra un matching estable y uno no estable, con una terna bloqueante del mismo.

Ejemplo 7.1

Dada la instancia $I=(A,B,C;P)$ con $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $b_0 \ b_1 \ b_2$
 a_1 : $b_2 \ b_1 \ b_0$
 a_2 : $b_2 \ b_0 \ b_1$

b_0 : $a_0 \ a_1 \ a_2$
 $c_0 \ c_1 \ c_2$

b_1 : $a_1 \ a_0 \ a_2$
 $c_0 \ c_1 \ c_2$

b_2 : $a_0 \ a_2 \ a_1$
 $c_2 \ c_0 \ c_1$

c_0 : $(a_0,b_1) \ (a_1,b_0) \ (a_1,b_2) \ (a_2,b_0) \ (a_2,b_1) \ (a_0,b_0) \ (a_1,b_1) \ (a_0,b_2) \ (a_2,b_2)$
 c_1 : $(a_0,b_0) \ (a_1,b_1) \ (a_2,b_2) \ (a_0,b_1) \ (a_1,b_2) \ (a_2,b_1) \ (a_0,b_2) \ (a_1,b_0) \ (a_2,b_0)$
 c_2 : $(a_2,b_1) \ (a_2,b_0) \ (a_1,b_2) \ (a_2,b_2) \ (a_1,b_1) \ (a_0,b_0) \ (a_0,b_1) \ (a_1,b_0) \ (a_0,b_2)$

Podemos ver que el matching $M_1 = \{(a_0,b_0,c_0), (a_1,b_1,c_1), (a_2,b_2,c_2)\}$ es estable para la instancia I , mientras que el matching $M_2 = \{(a_0,b_1,c_0), (a_1,b_0,c_1), (a_2,b_2,c_2)\}$ no es estable, ya que la terna (a_0,b_0,c_1) es bloqueante.

7.2.1.1 Búsqueda de un Matching Estable

Damos a continuación un algoritmo que encuentra siempre un matching estable en instancias U_SSP .

Algoritmo 7.1

El algoritmo puede comenzar desde alguno de los conjuntos A o B , comenzaremos por A .

- 1) Se encuentra un matching estable entre los individuos del conjunto A y los de B mediante el algoritmo de Gale y Shapley, utilizando para el conjunto B las listas de preferencias que tiene sobre A.

Todos los c_k ($1 \leq k \leq n$) están sin asignar a los pares (a_i, b_j) formados en el punto anterior.

- 2) Para cada c_k ($1 \leq k \leq n$)
 c_k se asigna a alguno de los pares formados en el paso anterior.
Por ejemplo: elige el primer a_i de su lista de preferencias tal que no esté asignado a algún c .

El matching así obtenido es estable.

Teorema

El Algoritmo 7.1 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.
Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un b_j asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley)

En el paso 2, cada c_k es asignado una y sólo una vez a pares distintos de (a_i, b_j)

- b) Probar que el matching encontrado es estable.
Sea M el matching encontrado por el algoritmo.
Debemos ver que no existe terna (a,b,c) no perteneciente al matching M , tal que:
 1. $b \succ_a M_B(a)$ y
 2. $(a \succeq_b M_A(b))$ y $c \succ_b M_C(b)$ o $(a \succ_b M_A(b))$ y $c \succeq_b M_C(b)$ y
 3. $(a, b) \succ_c M_{AB}(c)$

Supongamos que existe tal terna.

Por (2) $a \succ_b M_A(b)$ o $a = M_A(b)$

- Si $a \succ_b M_A(b)$, como $b \succ_a M_B(a)$ (1), significa que existe un par (a, b) que bloquea al matching de pares A-B obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- Si $a = M_A(b) \Rightarrow b = M_B(a)$. Absurdo (por 1)

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas U_SSP .

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$, asumiendo el uso de matriz de ranking.

Paso 1: El algoritmo de Gale y Shapley tiene orden n^2

Paso 2: Depende del método elegido para la asignación, pero podemos tomar como mejor caso la asignación secuencial. $O(n)$

7.2.1.2 Subconjuntos de instancias: U_SSP con prioridad

Para el conjunto de listas por pares, podría analizarse separadamente al subconjunto de las instancias de esta familia que cumplen con la característica de priorizar uno de los conjuntos sobre el otro. Veamos un ejemplo de instancias de esta clase.

Ejemplo 7.2

Dada la instancia $I = (A, B, C; P)$ de U_SSP con prioridad de C sobre A , donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $b_0 b_1 b_2$

a_1 : $b_2 b_1 b_0$

a_2 : $b_2 b_0 b_1$

b_0 : $a_0 a_1 a_2$

$c_0 c_1 c_2$

b_1 : $a_1 a_0 a_2$

$c_0 c_1 c_2$

b_2 : $a_0 a_2 a_1$

$c_2 c_0 c_1$

c_0 : $(a_0, b_1) (a_0, b_0) (a_0, b_2) (a_1, b_0) (a_1, b_2) (a_1, b_1) (a_2, b_0) (a_2, b_1) (a_2, b_2)$

c_1 : $(a_0, b_0) (a_0, b_1) (a_0, b_2) (a_1, b_1) (a_1, b_2) (a_1, b_0) (a_2, b_1) (a_2, b_2) (a_2, b_0)$

c_2 : $(a_2, b_1) (a_2, b_0) (a_2, b_2) (a_1, b_2) (a_1, b_0) (a_1, b_1) (a_0, b_0) (a_0, b_1) (a_0, b_2)$

En esta instancia, el matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable, mientras que el matching $M_2 = \{(a_0, b_1, c_0), (a_1, b_0, c_1), (a_2, b_2, c_2)\}$ es bloqueado por la terna (a_0, b_0, c_1) , por lo que no es estable.

Dado que ya dimos un algoritmo que resuelve instancias de U_SSP en general en tiempo polinomial, no ahondaremos en el estudio de estos subcasos.

7.2.2 U_PSP

Recordemos que, en instancias de este tipo, los conjuntos A , B y C tienen las siguientes preferencias: A elige sobre B , B elige sobre ambos conjuntos independientemente y C elige sobre pares de A y B .

La Figura 7.2 representa gráficamente esta relación:

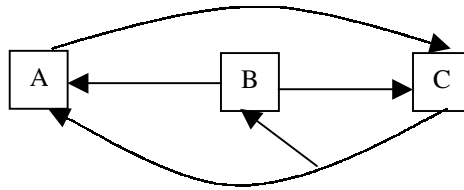


Figura 7.2 Preferencias entre los conjuntos en una instancia U_pSP .

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y

1. $c >_a M_C(a)$ y
2. $((a \geq_b M_A(b) \text{ y } c >_b M_C(b)) \text{ o } ((a >_b M_A(b) \text{ y } c \geq_b M_C(b)))$ y
3. $(a, b) >_c M_{AB}(c)$

Veamos el siguiente ejemplo que muestra para una instancia dada de U_pSP , un matching estable y uno no estable.

Ejemplo 7.3

Dada la instancia $I=(A,B,C;P)$ con $A=\{a_0,a_1,a_2\}$, $B=\{b_0,b_1,b_2\}$, $C=\{c_0,c_1,c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $c_0 \ c_1 \ c_2$

a_1 : $c_2 \ c_1 \ c_0$

a_2 : $c_2 \ c_0 \ c_1$

b_0 : $a_0 \ a_1 \ a_2$

$c_0 \ c_1 \ c_2$

b_1 : $a_1 \ a_0 \ a_2$

$c_0 \ c_1 \ c_2$

b_2 : $a_0 \ a_2 \ a_1$

$c_2 \ c_0 \ c_1$

c_0 : $(a_0,b_1) (a_1,b_0) (a_1,b_2) (a_2,b_0) (a_2,b_1) (a_0,b_0) (a_1,b_1) (a_0,b_2) (a_2,b_2)$

c_1 : $(a_0,b_0) (a_1,b_1) (a_2,b_2) (a_0,b_1) (a_1,b_2) (a_2,b_1) (a_0,b_2) (a_1,b_0) (a_2,b_0)$

c_2 : $(a_2,b_1) (a_2,b_0) (a_1,b_2) (a_2,b_2) (a_1,b_1) (a_0,b_0) (a_0,b_1) (a_1,b_0) (a_0,b_2)$

Podemos ver que el matching $M_1 = \{(a_0,b_0,c_0), (a_1,b_2,c_1), (a_2,b_1,c_2)\}$ es estable para la instancia I . En cambio, el matching $M_2 = \{(a_0,b_1,c_1), (a_1,b_0,c_2), (a_2,b_2,c_0)\}$ es bloqueado por la terna (a_0,b_1,c_0) , resultando no estable.

7.2.2.1 Subconjuntos de instancias: U_pSP con prioridad

Al no haber encontrado un algoritmo que nos permita hallar una solución para el conjunto de instancias de U_pSP , trataremos separadamente al subconjunto de las instancias de esta familia en los cuales el conjunto C (aquel en el cual sus listas de preferencias son por pares) cumple con la característica de priorizar uno de los conjuntos sobre el otro. En principio, no distinguiremos sobre qué conjunto se establece esta prioridad.

Veamos con un ejemplo una instancia de esta categoría, para la cual se presenta un matching estable y uno no estable.

Ejemplo 7.4

Dada la instancia $I = (A, B, C; P)$ con el conjunto C priorizando a A , donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P :

a_0 : $c_0 c_1 c_2$

a_1 : $c_2 c_1 c_0$

a_2 : $c_2 c_0 c_1$

b_0 : $a_0 a_1 a_2$

$c_0 c_1 c_2$

b_1 : $a_1 a_0 a_2$

$c_0 c_1 c_2$

b_2 : $a_0 a_2 a_1$

$c_0 c_2 c_1$

c_0 : $(a_0, b_1) (a_0, b_0) (a_0, b_2) (a_1, b_0) (a_1, b_2) (a_1, b_1) (a_2, b_0) (a_2, b_1) (a_2, b_2)$

c_1 : $(a_0, b_0) (a_0, b_1) (a_0, b_2) (a_1, b_1) (a_1, b_2) (a_1, b_0) (a_2, b_1) (a_2, b_2) (a_2, b_0)$

c_2 : $(a_2, b_1) (a_2, b_0) (a_2, b_2) (a_1, b_2) (a_1, b_0) (a_1, b_1) (a_0, b_0) (a_0, b_1) (a_0, b_2)$

El matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable para I . En cambio, el matching $M_2 = \{(a_0, b_0, c_1), (a_1, b_1, c_0), (a_2, b_2, c_2)\}$ no es estable, siendo bloqueado por la terna (a_0, b_2, c_0) .

A continuación daremos un algoritmo que encuentra un matching estable en instancias U_pSP con prioridad, en las cuales los individuos del conjunto A listan sus preferencias sobre el conjunto C , los de B sobre ambos y los agentes c de C lo hacen sobre pares de agentes de ambos conjuntos, priorizando uno de ellos.

Algoritmo 7.2

- 1) Se encuentra un matching estable entre los individuos del conjunto C y los de su conjunto priorizado por el algoritmo de Gale y Shapley.

Todos los agentes del tercer conjunto (es decir, aquel que no intervino en el paso 1) están sin asignar a los pares formados en el punto anterior.

- 2) Se encuentra un matching estable entre los individuos del conjunto C y los del conjunto no priorizado, mediante el algoritmo de Gale y Shapley, utilizando las listas de preferencias de los agentes de C restringidas a las parejas que les fueron asignados en el paso anterior.

El matching así obtenido es estable.

Teorema

El Algoritmo 7.2 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.

Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada c_k resulta con un agente de uno de los conjuntos asignado, para todo $1 \leq k \leq n$. (por correctitud del algoritmo de Gale y Shapley)

Del paso 2 cada c_k junto con su pareja asignada en el paso 1 resulta con un elemento del tercer conjunto asignado, para todo $1 \leq k \leq n$. (por correctitud del algoritmo de Gale y Shapley)

b) Probar que el matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M , tal que:

1. $c >_a M_C(a)$ y
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b) \text{ o } (a >_b M_A(b) \text{ y } c \geq_b M_C(b)))$ y
3. $(a, b) >_c M_{AB}(c)$

Supongamos que existe tal terna. Distribuyendo, nos queda:

- i) $c >_a M_C(a)$ y $a \geq_b M_A(b)$ y $c >_b M_C(b)$ y $(a, b) >_c M_{AB}(c)$ o
- ii) $c >_a M_C(a)$ y $a >_b M_A(b)$ y $c \geq_b M_C(b)$ y $(a, b) >_c M_{AB}(c)$

Veamos cada uno de los términos de la disyunción:

- i) $c >_a M_C(a)$ y $a \geq_b M_A(b)$ y $c >_b M_C(b)$ y $(a, b) >_c M_{AB}(c)$
 - Si C tiene prioridad sobre A , $(a, b) >_c M_{AB}(c) \Rightarrow a >_c M_A(c)$ o $(a=M_A(c) \text{ y } b >_{c/a} M_B(c))$
 $c >_a M_C(a)$ y $a \geq_b M_A(b)$ y $c >_b M_C(b)$ y $(a, b) >_c M_{AB}(c) \Rightarrow$
 $c >_a M_C(a)$ y $a \geq_b M_A(b)$ y $c >_b M_C(b)$ y $(a >_c M_A(c) \text{ o } (a=M_A(c) \text{ y } b >_{c/a} M_B(c))) \Rightarrow$
 $c >_a M_C(a)$ y $a >_c M_A(c) \Rightarrow M_{AC}$ no es estable. Absurdo (por correctitud del algoritmo de Gale y Shapley)
 - Si C , en cambio, tiene prioridad sobre B , $(a, b) >_c M_{AB}(c) \Rightarrow b >_c M_B(c)$ o $(b=M_B(c) \text{ y } a >_{c/b} M_A(c))$
 $c >_a M_C(a)$ y $a \geq_b M_A(b)$ y $c >_b M_C(b)$ y $(a, b) >_c M_{AB}(c) \Rightarrow$
 $c >_a M_C(a)$ y $a \geq_b M_A(b)$ y $c >_b M_C(b)$ y $(b >_c M_B(c) \text{ o } (b=M_B(c) \text{ y } a >_{c/b} M_A(c))) \Rightarrow$
 $c >_b M_C(b)$ y $b >_c M_B(c) \Rightarrow M_{BC}$ no es estable. Absurdo (por correctitud del algoritmo de Gale y Shapley)
- ii) $c >_a M_C(a)$ y $a >_b M_A(b)$ y $c \geq_b M_C(b)$ y $(a, b) >_c M_{AB}(c)$
 - Si C tiene prioridad sobre A ,
 $c >_a M_C(a)$ y $a >_b M_A(b)$ y $c \geq_b M_C(b)$ y $(a, b) >_c M_{AB}(c) \Rightarrow$
 $c >_a M_C(a)$ y $a >_b M_A(b)$ y $c \geq_b M_C(b)$ y $(a >_c M_A(c) \text{ o } (a=M_A(c) \text{ y } b >_{c/a} M_B(c))) \Rightarrow$
 $c >_a M_C(a)$ y $a >_c M_A(c) \Rightarrow M_{AC}$ no es estable. Absurdo (por correctitud del algoritmo de Gale y Shapley)
 - Si C , en cambio, tiene prioridad sobre B ,
 $c >_a M_C(a)$ y $a >_b M_A(b)$ y $c \geq_b M_C(b)$ y $(a, b) >_c M_{AB}(c) \Rightarrow$
 $c >_a M_C(a)$ y $a >_b M_A(b)$ y $c \geq_b M_C(b)$ y $(b >_c M_B(c) \text{ o } (b=M_B(c) \text{ y } a >_{c/b} M_A(c)))$
 $\Rightarrow (c >_b M_C(b) \text{ y } b >_c M_B(c)) \quad \text{o} \quad (c >_a M_C(a) \text{ y } a >_{c/b} M_A(c))$
 $\Rightarrow M_{BC}$ no es estable o M_{AC} con las listas de C restringidas a $M_B(c)$ no es estable. Absurdo (por correctitud del algoritmo de Gale y Shapley)

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas U_pSP con prioridad.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$, ya que en ambos pasos se ejecuta el algoritmo de Gale y Shapley, el cual tiene complejidad de $O(n^2)$, asumiendo el uso de matriz de ranking.

7.2.3 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir el conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones.

Sea:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

Daremos a continuación la formulación entera que define el conjunto de todos los matchings estables en instancias USP.

Las restricciones 1, 2, 3 y 5 garantizan que la solución es un matching, por lo tanto son comunes a todas las familias de instancias dentro de USP. La cuarta restricción, la cual garantiza la estabilidad del matching, varía dependiendo de la subfamilia que estemos analizando.

La formulación entera para definir el conjunto de matchings estables en instancias U_SSP queda definida de la siguiente forma:

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{i \geq a \\ b \\ k > c}} x_{i,b,k} + \sum_{\substack{i > a \\ b \\ k \geq c}} x_{i,b,k} + \sum_{\substack{(i,j) > (a,b) \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

En instancias U_pSP la restricción (4) es reemplazada por:

$$\sum_{\substack{k > c \\ j \in B}} x_{a,j,k} + \sum_{\substack{i \geq a \\ b > c}} x_{i,b,k} + \sum_{\substack{i > a \\ b \geq c}} x_{i,b,k} + \sum_{\substack{i,j > (a,b) \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, (a,b,c) \in A \times B \times C \quad (4)$$

7.3 UUS

Una instancia del problema de UUS está formada por tres conjuntos A, B y C donde las listas de preferencias tienen las siguientes estructuras:

- $\forall a \in A$, a tiene listas de preferencias únicas,
- $\forall b \in B$, b tiene listas de preferencias únicas y
- $\forall c \in C$, c tiene listas de preferencias simples.

En función del conjunto sobre el cual los agentes de los conjuntos A y B expresen sus preferencias, podemos subdividir esta familia en 3 subfamilias:

- $U_U U_U S$: Los conjuntos que expresan sus preferencias como listas únicas lo hacen mutuamente sobre los individuos del otro conjunto de listas únicas. Es decir, los individuos de A expresan sus preferencias como listas únicas sobre individuos de B, los de B como listas únicas sobre individuos de A y los de C como listas simples sobre A y B.
- $U_U U_S S$: Uno de los conjuntos que expresan sus preferencias como listas únicas lo hace sobre individuos del otro conjunto de listas únicas, el otro conjunto de listas únicas lista sobre los individuos del conjunto de listas simples. Sin pérdida de generalidad, los individuos de A expresan sus preferencias como listas únicas sobre individuos de B, los de B como listas únicas sobre individuos de C y los de C como listas simples sobre A y B.
- $U_S U_S S$: Los conjuntos que expresan sus preferencias como listas únicas lo hacen sobre individuos del conjunto de listas simples. Es decir, los individuos de A y los de B expresan sus preferencias como listas únicas sobre individuos de C y los de C como listas simples sobre A y B.

La definición de terna bloqueante para instancias de UUS depende de la subfamilia en la cual se ubique, de acuerdo a la categorización que dimos previamente.

En instancias $U_U U_U S$, decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b >_a M_B(a)$ y
2. $a >_b M_A(b)$ y
3. $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$

En la Figura 7.3 podemos observar la representación gráfica de las relaciones entre los conjuntos.

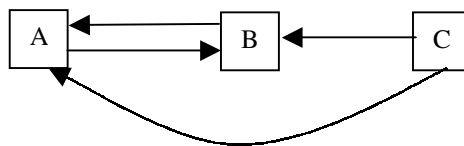


Figura 7.3 Preferencias entre los conjuntos en una instancia $U_U U_U S$.

En instancias $U_U U_S S$, decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b >_a M_B(a)$ y
2. $c >_b M_C(b)$ y
3. $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$

En la Figura 7.4 se representa gráficamente las preferencias entre los conjuntos.

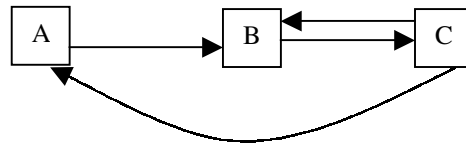


Figura 7.4 Preferencias entre los conjuntos en una instancia $U_U U_S S$.

Por último, en instancias $U_S U_S S$, decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $c >_a M_C(a)$ y
2. $c >_b M_C(b)$ y
3. $((a >_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$

En la Figura 7.5 podemos observar gráficamente la relación entre los conjuntos según sus listas de preferencias.

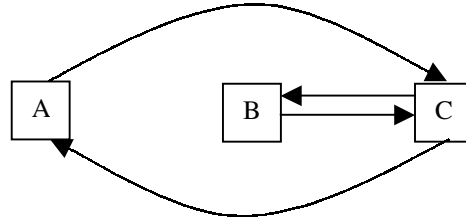


Figura 7.5 Preferencias entre los conjuntos en una instancia $U_S U_S S$.

A continuación se muestra con un ejemplo una instancia de una familia de UUS, junto con un matching estable y uno no estable para la misma.

Ejemplo 7.5

Dada la instancia de la familia $U_U U_U S$ $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

- P:
- a_0 : $b_0 \ b_1 \ b_2$
 - a_1 : $b_2 \ b_1 \ b_0$
 - a_2 : $b_2 \ b_0 \ b_1$

 - b_0 : $a_0 \ a_1 \ a_2$
 - b_1 : $a_1 \ a_0 \ a_2$
 - b_2 : $a_0 \ a_2 \ a_1$

 - c_0 : $a_0 \ a_1 \ a_2$
 $b_0 \ b_1 \ b_2$
 - c_1 : $a_0 \ a_1 \ a_2$
 $b_0 \ b_1 \ b_2$
 - c_2 : $a_2 \ a_1 \ a_0$
 $b_2 \ b_0 \ b_1$

Podemos ver que el matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable para la instancia I, mientras que el matching $M_2 = \{(a_0, b_1, c_0), (a_1, b_0, c_1), (a_2, b_2, c_2)\}$ no es estable, ya que es bloqueado por la terna (a_0, b_0, c_0) .

7.3.1 Búsqueda de un Matching Estable

Damos a continuación un algoritmo que encuentra siempre un matching estable en instancias de UUS. Este algoritmo aplica a cualquiera de las subfamilias en las que hemos dividido a este conjunto de instancias, por lo que sólo se harán distinciones entre las mismas cuando sea necesario.

Algoritmo 7.3

- 1) Se encuentra un matching estable entre los individuos de dos conjuntos que se elijan mutuamente por el algoritmo de Gale y Shapley.

Todos los agentes del conjunto restante están sin asignar a los pares formados en el punto anterior.

- 2) Para cada agente x del conjunto sin asignar
 - x se asigna a alguno de los pares formados en el paso anterior. Por ejemplo: x elige el primer agente de su lista de preferencias tal que no esté asignado.

El matching así obtenido es estable.

Teorema

El Algoritmo 7.3 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.
Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 resultan todos los agentes de dos conjuntos asignados 1 a 1 (por correctitud del algoritmo de Gale y Shapley).

En el paso 2, cada agente del tercer conjunto es asignado una y sólo una vez a pares distintos de individuos de los otros dos conjuntos.

- b) Probar que el matching encontrado es estable.
Sea M el matching encontrado por el algoritmo.
Debemos ver que no existe terna bloqueante para el matching M , para lo cual estudiaremos los diferentes casos de acuerdo a la subfamilia a la cual pertenezca la instancia:

- Caso 1: la instancia es $U_U U_U S$
Debemos ver que no existe terna (a, b, c) no perteneciente al matching M , tal que:
 1. $b \succ_a M_B(a)$ y
 2. $a \succ_b M_A(b)$ y
 3. $((a \succeq_c M_A(c) \text{ y } b \succ_c M_B(c)) \text{ o } ((a \succ_c M_A(c) \text{ y } b \succeq_c M_B(c)))$
 Supongamos que existe tal terna.

Sabemos que en el paso 1 del algoritmo se obtuvo un matching estable entre los conjuntos A y B, ya que ambos son los únicos conjuntos que se prefieren mutuamente.

Como $a \succ_b M_A(b)$ y $b \succ_a M_B(a)$, significa que existe un par (a, b) que bloquea al matching de pares A-B obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

- Caso 2: la instancia es $U_U U_S S$

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M, tal que:

1. $b \succ_a M_B(a)$ y
2. $c \succ_b M_C(b)$ y
3. $((a \succeq_c M_A(c) \text{ y } b \succ_c M_B(c)) \text{ o } ((a \succ_c M_A(c) \text{ y } b \succeq_c M_B(c)))$

Supongamos que existe tal terna.

Sabemos que en el paso 1 del algoritmo se obtuvo un matching estable entre los conjuntos B y C, ya que ambos son los únicos conjuntos que se prefieren mutuamente.

Por (3) $b \succ_c M_B(c)$ o $b = M_B(c)$

- Si $b \succ_c M_B(c)$, como $c \succ_b M_C(b)$ (2), significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- Si $b = M_B(c) \Rightarrow c = M_C(b)$. Absurdo (por 2)

- Caso 3: la instancia es $U_S U_S S$

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M, tal que:

1. $c \succ_a M_C(a)$ y
2. $c \succ_b M_C(b)$ y
3. $((a \succeq_c M_A(c) \text{ y } b \succ_c M_B(c)) \text{ o } ((a \succ_c M_A(c) \text{ y } b \succeq_c M_B(c)))$

Supongamos que existe tal terna.

En el paso 1 del algoritmo se obtuvo un matching estable entre alguno de los conjuntos A o B y C, ya que en ambos casos los pares de conjuntos se prefieren mutuamente.

- i) Supongamos que se obtuvo un matching AC

Por (3) $a \succ_c M_A(c)$ o $a = M_A(c)$

- Si $a \succ_c M_A(c)$, como $c \succ_a M_C(a)$ (1), significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- Si $a = M_A(c) \Rightarrow c = M_C(a)$. Absurdo (por 1)

- ii) Supongamos que se obtuvo un matching BC

Por (3) $b \succ_c M_B(c)$ o $b = M_B(c)$

- Si $b \succ_c M_B(c)$, como $c \succ_b M_C(b)$ (2), significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- Si $b = M_B(c) \Rightarrow c = M_C(b)$. Absurdo (por 2)

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas UUS.

Complejidad e Implementación

La complejidad del algoritmo es de $O(n^2)$, suponiendo como siempre, la implementación de la matriz de ranking en los casos necesarios.

Paso 1: El algoritmo de Gale y Shapley tiene complejidad de $O(n^2)$

Paso 2: Depende del método elegido para la asignación, pero podemos tomar como mejor caso la asignación secuencial. $O(n)$

7.3.2 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir el conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones.

Sea:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

Daremos a continuación la formulación entera que define el conjunto de todos los matchings estables en instancias UUS.

Las restricciones 1, 2, 3 y 5 garantizan que la solución es un matching, por lo tanto son comunes a todas las familias de instancias dentro de UUS. La cuarta restricción, la cual garantiza la estabilidad del matching, varía dependiendo de la subfamilia que estemos analizando.

La formulación entera para definir el conjunto de matchings estables en instancias $U_U U_U S$ queda definida de la siguiente forma:

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{i > a \\ b \\ k \in C}} x_{i,,b,k} + \sum_{\substack{i \geq a \\ c \\ j > b}} x_{i,j,c} + \sum_{\substack{i > a \\ c \\ j \geq b}} x_{i,j,,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

En instancias $U_U U_S S$ la restricción (4) es reemplazada por:

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{k > c \\ b \\ i \in A}} x_{i,b,k} + \sum_{\substack{i \geq a \\ c \\ j > b}} x_{i,j,c} + \sum_{\substack{i > a \\ c \\ j \geq b}} x_{i,j,c} + x_{a,b,c} \geq 1, (a,b,c) \in A \times B \times C \quad (4)$$

En instancias $U_S U_S S$ la restricción (4) es reemplazada por:

$$\sum_{\substack{k > c \\ a \\ i \in A}} x_{a,j,k} + \sum_{\substack{k > c \\ b \\ i \in A}} x_{i,b,k} + \sum_{\substack{i \geq a \\ c \\ j > b}} x_{i,j,c} + \sum_{\substack{i > a \\ c \\ j \geq b}} x_{i,j,c} + x_{a,b,c} \geq 1, (a,b,c) \in A \times B \times C \quad (4)$$

7.4 UUP

Una instancia del problema de UUS está formada por tres conjuntos A, B y C donde las listas de preferencias tienen las siguientes estructuras:

- $\forall a \in A$, a tiene listas de preferencias únicas,
- $\forall b \in B$, b tiene listas de preferencias únicas y
- $\forall c \in C$, c tiene listas de preferencias por pares.

En función del conjunto sobre el cual los agentes de los conjuntos A y B expresen sus preferencias, podemos subdividir esta familia en 3 subfamilias:

- $U_U U_U P$: Los conjuntos que expresan sus preferencias como listas únicas lo hacen sobre individuos del otro conjunto de listas únicas. Es decir, los individuos de A expresan sus preferencias como listas únicas sobre individuos de B, los de B como listas únicas sobre individuos de A y los de C como listas por pares sobre A y B.
- $U_U U_P P$: Uno de los conjuntos en el cual los agentes expresan sus preferencias como listas únicas lo hace sobre individuos del otro conjunto de listas únicas, el otro conjunto de listas únicas lista sobre los individuos del conjunto de listas simples. Sin pérdida de generalidad, los individuos de A expresan sus preferencias como listas únicas sobre individuos de B, los de B como listas únicas sobre individuos de C y los de C como listas por pares sobre A y B.
- $U_P U_P P$: Los conjuntos que expresan sus preferencias como listas únicas lo hacen sobre individuos del conjunto de listas simples. Es decir, los individuos de A expresan sus preferencias como listas únicas sobre individuos de C, los de B como listas únicas sobre individuos de C y los de C como listas por pares sobre A y B.

Definamos y analicemos cada caso por separado:

7.4.1 $U_U U_U P$

Recordemos que, en instancias $U_U U_U P$, los individuos del conjunto A expresan sus preferencias como listas únicas sobre individuos de B, los de B como listas únicas sobre individuos de A y los de C como listas por pares sobre A y B.

En la Figura 7.6 se puede ver gráficamente las preferencias entre los conjuntos.

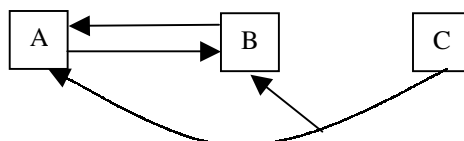


Figura 7.6 Preferencias entre los conjuntos en una instancia $U_U U_U P$.

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b \succ_a M_B(a)$ y
2. $a \succ_b M_A(b)$ y
3. $(a, b) \succ_c M_{AB}(c)$

Veamos un ejemplo de una instancia de $U_U U_U P$, que muestra un matching estable y uno no estable.

Ejemplo 7.6

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $b_0 b_1 b_2$
 a_1 : $b_2 b_1 b_0$
 a_2 : $b_2 b_0 b_1$

b_0 : $a_0 a_1 a_2$
 b_1 : $a_1 a_0 a_2$
 b_2 : $a_0 a_2 a_1$

c_0 : $(a_0, b_1) (a_0, b_0) (a_1, b_2) (a_1, b_0) (a_2, b_1) (a_1, b_1) (a_2, b_0) (a_0, b_2) (a_2, b_2)$
 c_1 : $(a_2, b_0) (a_0, b_1) (a_0, b_2) (a_1, b_1) (a_1, b_2) (a_1, b_0) (a_2, b_1) (a_2, b_2) (a_0, b_0)$
 c_2 : $(a_1, b_1) (a_2, b_0) (a_0, b_2) (a_1, b_2) (a_1, b_0) (a_2, b_1) (a_0, b_1) (a_0, b_0) (a_2, b_2)$

El matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable para la instancia I .

En cambio, el matching $M_2 = \{(a_0, b_2, c_0), (a_1, b_1, c_1), (a_2, b_0, c_2)\}$ no es estable. La terna (a_0, b_0, c_0) es bloqueante para el mismo.

7.4.1.1 Búsqueda de un Matching Estable

A continuación daremos un algoritmo que encuentra siempre un matching estable en instancias $U_U U_U P$.

Algoritmo 7.4

El algoritmo puede comenzar desde alguno de los conjuntos A o B , comenzaremos por A .

- 1) Se encuentra un matching estable entre los individuos del conjunto A y los de B por el algoritmo de Gale y Shapley.

Todos los c_k ($1 \leq k \leq n$) están sin asignar a los pares (a_i, b_j) formados en el punto anterior.

- 2) Para cada c_k ($1 \leq k \leq n$)
 c_k se asigna a alguno de los pares formados en el paso anterior.

El matching así obtenido es estable.

Teorema

El Algoritmo 7.4 es correcto y completo.

Demostración

Demostración de correctitud

a) Probar que la solución que el algoritmo genera es un matching.

Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 resultan todos los agentes de dos conjuntos asignados 1 a 1 (por correctitud del algoritmo de Gale y Shapley).

En el paso 2, cada agente del tercer conjunto es asignado una y sólo una vez a pares distintos de individuos de los otros dos conjuntos.

b) Probar que el matching encontrado es estable.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M , tal que:

1. $b \succ_a M_B(a)$ y
2. $a \succ_b M_A(b)$ y
3. $(a, b) \succ_c M_{AB}(c)$

Supongamos que existe tal terna.

Como $a \succ_b M_A(b)$ y $b \succ_a M_B(a)$, significa que existe un par (a, b) que bloquea al matching de pares A-B obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas $U \cup U \cup P$.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$

Paso 1: El algoritmo de Gale y Shapley tiene un complejidad de $O(n^2)$

Paso 2: Depende del método elegido para la asignación, pero podemos tomar como mejor caso la asignación secuencial. $O(n)$.

7.4.1.2 Subconjuntos de instancias: $U \cup U \cup P$ con prioridad

Para el conjunto de listas por pares, podría analizarse separadamente al subconjunto de las instancias de esta familia que cumplen con la característica de priorizar uno de los conjuntos sobre el otro.

Un ejemplo de instancias de esta clase es el siguiente:

Ejemplo 7.7

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $b_0 b_1 b_2$
 a_1 : $b_2 b_1 b_0$
 a_2 : $b_2 b_0 b_1$

b_0 : $a_0 a_1 a_2$
 b_1 : $a_1 a_0 a_2$
 b_2 : $a_0 a_2 a_1$

c_0 : $(a_0, b_1) (a_1, b_1) (a_2, b_1) (a_1, b_0) (a_0, b_0) (a_2, b_0) (a_1, b_2) (a_2, b_2) (a_0, b_2)$
 c_1 : $(a_0, b_0) (a_2, b_0) (a_1, b_0) (a_1, b_1) (a_0, b_1) (a_2, b_1) (a_0, b_2) (a_1, b_2) (a_2, b_2)$
 c_2 : $(a_2, b_1) (a_0, b_1) (a_1, b_1) (a_2, b_2) (a_1, b_2) (a_0, b_2) (a_0, b_0) (a_1, b_0) (a_2, b_0)$

Podemos ver que el matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable para la instancia I . En cambio, el matching $M_2 = \{(a_0, b_1, c_0), (a_1, b_0, c_1), (a_2, b_2, c_2)\}$ no es estable para I , ya que la terna (a_0, b_0, c_1) es bloqueante.

Dado que ya dimos un algoritmo que resuelve instancias de U_SSP en general en tiempo polinomial, no ahondaremos en el estudio de estos subcasos.

7.4.2 $U_U U_P P$

Recordemos que, en instancias $U_U U_P P$, los individuos de A expresan sus preferencias como listas únicas sobre agentes de B , los de B como listas únicas sobre individuos de C y los de C como listas por pares sobre A y B .

Decimos que (a, b, c) es una *terna bloqueante* del matching M si $(a, b, c) \notin M$ y:

1. $b \succ_a M_B(a)$ y
2. $c \succ_b M_C(b)$ y
3. $(a, b) \succ_c M_{AB}(c)$

Podemos ver gráficamente las preferencias entre los conjuntos en la Figura 7.7:

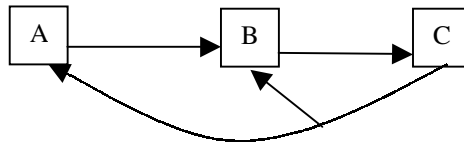


Figura 7.7 Preferencias entre los conjuntos en una instancia $U_U U_P P$.

Veamos un ejemplo de una instancia $U_U U_P P$, en el cual se muestra un matching estable y uno no estable para la misma.

Ejemplo 7.8

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:
 a_0 : $b_0 b_1 b_2$
 a_1 : $b_2 b_1 b_0$
 a_2 : $b_2 b_0 b_1$

b_0 : $c_0 c_1 c_2$
 b_1 : $c_1 c_0 c_2$
 b_2 : $c_0 c_2 c_1$

c_0 : $(a_0, b_1) (a_1, b_0) (a_1, b_2) (a_2, b_0) (a_2, b_1) (a_0, b_0) (a_1, b_1) (a_0, b_2) (a_2, b_2)$
 c_1 : $(a_0, b_0) (a_1, b_1) (a_0, b_2) (a_0, b_1) (a_1, b_2) (a_2, b_1) (a_2, b_2) (a_1, b_0) (a_2, b_0)$
 c_2 : $(a_2, b_1) (a_2, b_0) (a_1, b_2) (a_2, b_2) (a_1, b_1) (a_0, b_0) (a_0, b_1) (a_1, b_0) (a_0, b_2)$

El matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable para I, mientras que el matching $M_2 = \{(a_0, b_1, c_0), (a_1, b_0, c_1), (a_2, b_2, c_2)\}$ no es estable, ya que la terna (a_0, b_0, c_1) es bloqueante.

7.4.2.1 Subconjuntos de instancias: $U_U U_P P$ con prioridad

Para facilitar la búsqueda de una solución al problema de matching estable en instancias $U_U U_P P$, podemos tratar separadamente para el conjunto de listas por pares, al subconjunto de las instancias de esta familia que cumplen con la característica de priorizar uno de los conjuntos sobre el otro. De esta forma subclasificaremos las instancias según cuál sea el conjunto priorizado tal como sigue:

- $U_U U_P P_{Uu}$: los individuos del conjunto con listas de preferencias por pares priorizan a los individuos del conjunto que expresa sus preferencias como listas únicas sobre el otro conjunto de listas únicas.
- $U_U U_P P_{Up}$: los individuos del conjunto con listas de preferencias por pares priorizan a los individuos del conjunto que expresa sus preferencias como listas únicas sobre el de pares.

Veamos a continuación un ejemplo de una instancia de $U_U U_P P$ con prioridad

Ejemplo 7.9

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:
 a_0 : $b_0 b_1 b_2$
 a_1 : $b_2 b_1 b_0$
 a_2 : $b_2 b_0 b_1$

b_0 : $c_0 c_1 c_2$
 b_1 : $c_1 c_0 c_2$
 b_2 : $c_0 c_2 c_1$

c_0 : $(a_0, b_0) (a_0, b_1) (a_0, b_2) (a_2, b_0) (a_2, b_1) (a_2, b_2) (a_1, b_1) (a_1, b_2) (a_1, b_0)$
 c_1 : $(a_0, b_0) (a_0, b_1) (a_0, b_2) (a_1, b_1) (a_1, b_2) (a_1, b_0) (a_2, b_1) (a_2, b_0) (a_2, b_2)$
 c_2 : $(a_2, b_2) (a_2, b_1) (a_2, b_2) (a_1, b_2) (a_1, b_1) (a_1, b_0) (a_0, b_1) (a_0, b_0) (a_0, b_2)$

Observemos que el matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable para la instancia I, mientras que el matching $M_2 = \{(a_0, b_2, c_1), (a_1, b_0, c_2), (a_2, b_1, c_0)\}$ no es estable, al ser bloqueado por la terna (a_0, b_0, c_0) .

7.4.2.1.1 $U_U U_P P_{U_U}$

En instancias $U_U U_P P_{U_U}$, los individuos de A expresan sus preferencias como listas únicas sobre agentes de B, los de B como listas únicas sobre individuos de C y los de C como listas por pares sobre A y B, con prioridad sobre A.

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b >_a M_B(a)$ y
2. $c >_b M_C(b)$ y
3. $a >_c M_A(c)$ o $(a = M_A(c) \text{ y } b >_{c/a} M_B(c))$

Podemos ver que el problema de encontrar un matching estable en instancias $U_U U_P P_{U_U}$ es una extensión del mismo problema en LUC. Como mencionamos en el Capítulo 4 en el cual se estudiaron las instancias de LUC, conjeturamos que tal problema es NP-Completo. Observemos que las condiciones que definen una terna bloqueante incluyen las mismas condiciones que para LUC, incorporando además nuevas posibles ternas bloqueantes.

Esto nos lleva a basar nuestro estudio de instancias $U_U U_P P_{U_U}$ en lo estudiado oportunamente para LUC, incorporando las nuevas restricciones.

Lo enunciado previamente queda reflejado por la siguiente propiedad, la cual nos da una condición suficiente para encontrar un matching estable en instancias $U_U U_P P_{U_U}$.

Propiedad 7.1

Sea M un matching tridimensional y sean las siguientes instancias:

I = (A, B, C; P) una instancia de $U_U U_P P_{U_U}$

I' = (A, B, C; P') una instancia de LUC tal que $P' = \{P(a)/a \in A\} \cup \{P(b)/b \in B\} \cup \{P_A(c)/c \in C\}$.

I'' = (B, C; P'') una instancia del problema de matrimonios estables tal que $P'' = \{P(b)/b \in B\} \cup \{P_{B/M_A(c)}(c) / c \in C\}$, donde $P_{B/M_A(c)}(c)$ es la lista de preferencias del agente c sobre los $b \in B$, relativas a su agente a asignado, es decir, $M_A(c)$.

Entonces, M estable en I' y M_{BC} estable en I'' \Rightarrow M estable en I.

Demostración

Supongamos M estable en I' y M_{BC} estable en I'', pero M no estable en I, entonces $\exists (a,b,c)$ terna bloqueante de M en I, es decir $\exists (a,b,c)$ tal que: $b >_a M_B(a)$ y $c >_b M_C(b)$ y $(a >_c M_A(c)$ o $(a = M_A(c)$ y $b >_{c/a} M_B(c))$)

\Rightarrow

$b >_a M_B(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ o

$b >_a M_B(a)$ y $c >_b M_C(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c)$

Si $b >_a M_B(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c) \Rightarrow (a,b,c)$ bloquea a M en I'. Absurdo, por hipótesis.

Si $b >_a M_B(a)$ y $c >_b M_C(b)$ y $a = M_A(c)$ y $b >_{c/a} M_B(c) \Rightarrow c >_b M_C(b)$ y $b >_{c/M_A(c)} M_B(c) \Rightarrow (b,c)$ bloquea a M en I''. Absurdo, por hipótesis.

Luego, M estable en I' y M_{BC} estable en I'' \Rightarrow M estable en I. \blacklozenge

7.4.2.1.2 $U_U U_P P_{U_P}$

En instancias $U_U U_P P_{U_P}$, los individuos de A expresan sus preferencias como listas únicas sobre agentes de B, los de B como listas únicas sobre individuos de C y los de C como listas por pares sobre A y B, con prioridad sobre B.

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b \succ_a M_B(a)$ y
2. $c \succ_b M_C(b)$ y
3. $b \succ_c M_B(c)$ o $(b = M_B(c) \text{ y } a \succ_{c/b} M_A(c))$

Damos a continuación un algoritmo que encuentra un matching estable en instancias de $U_U U_P P_U P_P$.

Algoritmo 7.5

- 1) Se encuentra un matching estable entre los individuos del conjunto B y los de C por el algoritmo de Gale y Shapley, utilizando para C las listas elementales sobre B .

Todos los a_i ($1 \leq i \leq n$) están sin asignar a los pares (b_j, c_k) formados en el punto anterior.

- 2) Se encuentra un matching estable entre los individuos del conjunto A y los de B por el algoritmo de Gale y Shapley, utilizando para cada b_j ($1 \leq j \leq n$) la lista del c_k que le haya sido asignado en el paso anterior relativa a b_j .

El matching así obtenido es estable.

Teorema

El Algoritmo 7.5 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.

Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también

Del paso 1 cada b_j resulta con un c_k asignado, para todo $1 \leq j \leq n$. (por correctitud del algoritmo de Gale y Shapley).

Del paso 2 cada a_i resulta con un b_j asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

- b) Probar que el matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M , tal que:

1. $b \succ_a M_B(a)$ y
2. $c \succ_b M_C(b)$ y
3. $b \succ_c M_B(c)$ o $(b = M_B(c) \text{ y } a \succ_{c/b} M_A(c))$

Supongamos que existe tal terna.

Por (3) $b \succ_c M_B(c)$ o $b = M_B(c)$

- Si $b \succ_c M_B(c)$, como $c \succ_b M_C(b)$ (2), significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

- Si $b = M_B(c) \Rightarrow c = M_C(b)$.

$a \succ_{c/b} M_A(c) \Rightarrow a \succ_b M_A(M_C(b)) \Rightarrow a \succ_b M_A(b)$.

Como $b \succ_a M_B(a)$ (1), significa que existe un par (a, b) que bloquea al matching de pares A-B obtenido en el paso 2. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas $U_U U_P P_{U_P}$.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$.

7.4.3 $U_P U_P P$

Recordemos que, en instancias $U_P U_P P$, los individuos de A y de B expresan sus preferencias como listas únicas sobre agentes de C y los de C como listas por pares sobre A y B.

En la Figura 7.8 podemos ver gráficamente las preferencias en una instancia $U_P U_P P$.

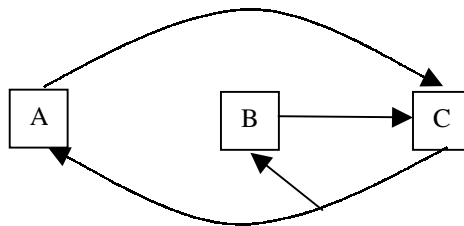


Figura 7.8 Preferencias entre los conjuntos en una instancia $U_P U_P P$.

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $c >_a M_C(a)$ y
2. $c >_b M_C(b)$ y
3. $(a, b) >_c M_{AB}(c)$

Damos a continuación un ejemplo de una instancia de $U_P U_P P$, en el cual se muestra un matching estable y uno no estable para la misma.

Ejemplo 7.10

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $c_1 c_0 c_2$

a_1 : $c_2 c_1 c_0$

a_2 : $c_2 c_0 c_1$

b_0 : $c_2 c_1 c_0$

b_1 : $c_1 c_0 c_2$

b_2 : $c_0 c_2 c_1$

c_0 : $(a_0, b_1) (a_1, b_0) (a_1, b_2) (a_2, b_0) (a_2, b_1) (a_0, b_0) (a_1, b_1) (a_0, b_2) (a_2, b_2)$

c_1 : $(a_0, b_1) (a_1, b_0) (a_2, b_2) (a_0, b_0) (a_1, b_2) (a_2, b_1) (a_0, b_2) (a_1, b_1) (a_2, b_0)$

c_2 : $(a_2, b_2) (a_2, b_0) (a_1, b_2) (a_2, b_1) (a_1, b_1) (a_0, b_0) (a_0, b_1) (a_1, b_0) (a_0, b_2)$

Podemos ver que, para la instancia I , el matching $M_1 = \{(a_0, b_1, c_1), (a_1, b_2, c_0), (a_2, b_0, c_2)\}$ es estable, mientras que el matching $M_2 = \{(a_0, b_2, c_2), (a_1, b_0, c_0), (a_2, b_1, c_1)\}$ no es estable, ya que es bloqueado por la terna (a_1, b_0, c_1) .

7.4.3.1 Subconjuntos de instancias: $U_P U_P P$ con prioridad

Para facilitar la búsqueda de una solución al problema de matching estable en instancias $U_P U_P P$, podemos tratar separadamente para el conjunto de listas por pares, al subconjunto de las instancias de esta familia que cumplen con la característica de priorizar uno de los conjuntos sobre el otro. Suponemos, sin pérdida de generalidad, que C tiene prioridad sobre A .

Decimos entonces que (a, b, c) es una *terna bloqueante* del matching M si $(a, b, c) \notin M$ y:

1. $c >_a M_C(a)$ y
2. $c >_b M_C(b)$ y
3. $a >_c M_A(c)$ o $(a = M_A(c) \text{ y } b >_{c/a} M_B(c))$

Daremos a continuación un algoritmo que siempre encuentra un matching estable en instancias $U_P U_P P$ con prioridad.

Algoritmo 7.6

- 1) Se encuentra un matching estable entre los individuos del conjunto A y los de C por el algoritmo de Gale y Shapley.

Todos los b_j ($1 \leq j \leq n$) están sin asignar a los pares (a_i, c_k) formados en el punto anterior.

- 2) Para cada b_j ($1 \leq j \leq n$)
 b_j se asigna a alguno de los pares formados en el paso anterior. Por ejemplo: elige el primer c_k de su lista de preferencias tal que no esté asignado a algún b .

El matching así obtenido es estable.

Teorema

El Algoritmo 7.6 es correcto y completo.

Demostración

Demostración de correctitud

a) Probar que la solución que el algoritmo genera es un matching.

Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un c_k asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

En el paso 2, cada b_j es asignado una y sólo una vez a pares distintos de (a_i, c_k)

b) Probar que el matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M , tal que:

1. $c >_a M_C(a)$ y
2. $c >_b M_C(b)$ y
3. $a >_c M_A(c)$ o $(a=M_A(c) \text{ y } b >_{c/a} M_B(c))$

o, lo que es lo mismo:

- $(c >_a M_C(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c))$ o
 $(c >_a M_C(a) \text{ y } c >_b M_C(b) \text{ y } a=M_A(c) \text{ y } b >_{c/a} M_B(c))$

Supongamos que existe tal terna.

- Si $(c >_a M_C(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c))$ significa que existe un par (a, c) que bloquea al matching de pares M_{AC} obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- Si $(c >_a M_C(a) \text{ y } c >_b M_C(b) \text{ y } a=M_A(c) \text{ y } b >_{c/a} M_B(c)) \Rightarrow c >_a M_C(a) \text{ y } c=M_C(a)$. Absurdo

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas $U_p U_p P$ con prioridad.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: Depende del método elegido para la asignación, pero podemos tomar como mejor caso la asignación secuencial. $O(n)$

7.4.4 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir el conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones.

Sea:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

Daremos a continuación la formulación entera que define el conjunto de todos los matchings estables en instancias UUP.

Las restricciones 1, 2, 3 y 5 garantizan que la solución es un matching, por lo tanto son comunes a todas las familias de instancias dentro de UUP. La cuarta restricción, la cual garantiza la estabilidad del matching, varía dependiendo de la subfamilia que estemos analizando.

La formulación entera para definir el conjunto de matchings estables en instancias $U_U U_P$ queda definida de la siguiente forma:

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{i > a \\ b \\ k \in C}} x_{i,b,k} + \sum_{\substack{(i,j) > (a,b) \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

En instancias $U_U U_P$ la restricción (4) es reemplazada por:

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{k > c \\ b \\ i \in A}} x_{i,b,k} + \sum_{\substack{(i,j) > (a,b) \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

En instancias $U_P U_P$ la restricción (4) es reemplazada por:

$$\sum_{\substack{k > c \\ a \\ i \in A}} x_{a,j,k} + \sum_{\substack{k > c \\ b \\ i \in A}} x_{i,b,k} + \sum_{\substack{(i,j) > (a,b) \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

7.5 SSU

Una instancia del problema de SSU está formada por tres conjuntos A, B y C donde las listas de preferencias tienen las siguientes estructuras:

- $\forall a \in A$, a tiene listas de preferencias simples,
- $\forall b \in B$, b tiene listas de preferencias simples y
- $\forall c \in C$, c tiene listas de preferencias únicas.

En la Figura 7.9, podemos ver gráficamente las preferencias entre los conjuntos en instancias SSU.

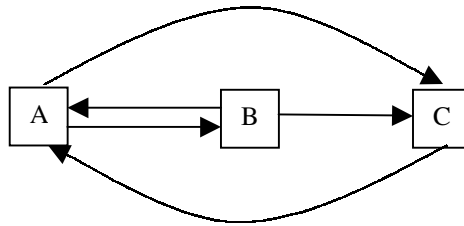


Figura 7.9 Preferencias entre los conjuntos en una instancia SSU.

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $(b \geq_a M_B(a) \text{ y } c >_a M_B(a))$ o $(b >_a M_B(a) \text{ y } c \geq_a M_C(a))$ y
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b))$ o $(a >_b M_A(b) \text{ y } c \geq_b M_C(b))$ y
3. $a >_c M_A(c)$

Veamos a continuación el siguiente ejemplo, el cual muestra una instancia de SSU, junto con un matching estable y uno no estable para la misma.

Ejemplo 7.11

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $b_0 \ b_1 \ b_2$

$c_0 \ c_2 \ c_1$

a_1 : $b_2 \ b_1 \ b_0$

$c_0 \ c_2 \ c_1$

a_2 : $b_2 \ b_0 \ b_1$

$c_0 \ c_2 \ c_1$

b_0 : $a_0 \ a_1 \ a_2$

$c_0 \ c_1 \ c_2$

b_1 : $a_1 \ a_0 \ a_2$

$c_0 \ c_1 \ c_2$

b_2 : $a_0 \ a_2 \ a_1$

$c_0 \ c_2 \ c_1$

c_0 : $a_0 \ a_1 \ a_2$

c_1 : $a_0 \ a_1 \ a_2$

c_2 : $a_2 \ a_1 \ a_0$

Podemos observar que el matching $M_1 = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable para I. En cambio, el matching $M_2 = \{(a_0, b_2, c_1), (a_1, b_0, c_2), (a_2, b_1, c_0)\}$ no es estable. La terna (a_0, b_0, c_0) es bloqueante.

7.5.1 Búsqueda de un Matching Estable

Daremos a continuación un algoritmo que encuentra siempre un matching estable en instancias SSU.

Algoritmo 7.7

El algoritmo puede comenzar desde cualquiera de los tres conjuntos, comenzaremos por A.

- 1) Se encuentra un matching estable entre los individuos del conjunto A y los de C por el algoritmo de Gale y Shapley, utilizando para los agentes del conjunto A las listas de preferencias sobre los c

Todos los b_j ($1 \leq j \leq n$) están sin asignar a los pares (a_i, c_k) formados en el punto anterior.

- 2) Para cada b_j ($1 \leq j \leq n$)
 b_j se asigna a alguno de los pares formados en el paso anterior. Por ejemplo: elige el primer a_i de su lista de preferencias tal que no esté asignado a algún b.

El matching así obtenido es estable.

Teorema

El Algoritmo 7.7 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.
 Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un c_k asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

En el paso 2, cada b_j es asignado una y sólo una vez a pares distintos de (a_i, c_k) .

- b) Probar que el matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a, b, c) no perteneciente al matching M, tal que:

1. $(b \geq_a M_B(a) \text{ y } c >_a M_B(a)) \text{ o } (b >_a M_B(a) \text{ y } c \geq_a M_C(a)) \text{ y}$
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b)) \text{ o } (a >_b M_A(b) \text{ y } c \geq_b M_C(b)) \text{ y}$
3. $a >_c M_A(c)$

Supongamos que existe tal terna.

Por (1) $c >_a M_C(a)$ o $c = M_C(a)$

- Si $c >_a M_C(a)$, como $a >_c M_A(c)$, significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- Si $c = M_C(a) \Rightarrow a = M_A(c)$. Absurdo (por 3)

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas SSU.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: Depende del método elegido para la asignación, pero podemos tomar como mejor caso la asignación secuencial. $O(n)$

7.5.2 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir el conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones. La formulación entera para definir el conjunto de matchings estables en instancias SSU queda definida de la siguiente forma:

Sea:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

Supongamos, sin pérdida de generalidad, que los agentes del conjunto C tienen sus listas de preferencias sobre los individuos de A.

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j \geq a \\ k > c \\ a}} x_{a,j,k} + \sum_{\substack{j > b \\ k \geq c \\ a}} x_{a,j,k} + \sum_{\substack{i \geq a \\ b \\ k > c}} x_{i,b,k} + \sum_{\substack{i > a \\ b \\ k \geq c}} x_{i,b,k} + \sum_{\substack{i > a \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

7.6 SSP

Una instancia del problema de SSP está formada por tres conjuntos A, B y C donde las listas de preferencias tienen las siguientes estructuras:

- $\forall a \in A$, a tiene listas de preferencias simples,
- $\forall b \in B$, b tiene listas de preferencias simples y
- $\forall c \in C$, c tiene listas de preferencias por pares.

En la Figura 7.10, podemos ver las preferencias entre los conjuntos en una instancia SSP.

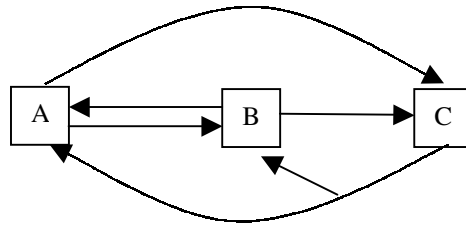


Figura 7.10 Preferencias entre los conjuntos en una instancia SSP.

Decimos que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $(b \geq_a M_B(a) \text{ y } c >_a M_B(a))$ o $(b >_a M_B(a) \text{ y } c \geq_a M_C(a))$ y
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b))$ o $(a >_b M_A(b) \text{ y } c \geq_b M_C(b))$ y
3. $(a, b) >_c M_{AB}(c)$

Veamos el siguiente ejemplo de una instancia SSP.

Ejemplo 7.12

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $b_0 \ b_1 \ b_2$

$c_0 \ c_2 \ c_1$

a_1 : $b_2 \ b_1 \ b_0$

$c_0 \ c_2 \ c_1$

a_2 : $b_2 \ b_0 \ b_1$

$c_1 \ c_2 \ c_0$

b_0 : $a_0 \ a_1 \ a_2$

$c_0 \ c_1 \ c_2$

b_1 : $a_1 \ a_0 \ a_2$

$c_0 \ c_1 \ c_2$

b_2 : $a_0 \ a_2 \ a_1$

$c_0 \ c_2 \ c_1$

c_0 : $(a_2, b_1) (a_2, b_2) (a_1, b_2) (a_2, b_0) (a_0, b_1) (a_0, b_0) (a_1, b_1) (a_0, b_2) (a_1, b_0)$
 c_1 : $(a_0, b_1) (a_1, b_0) (a_1, b_1) (a_0, b_0) (a_1, b_2) (a_2, b_1) (a_0, b_2) (a_2, b_2) (a_2, b_0)$
 c_2 : $(a_2, b_2) (a_2, b_0) (a_1, b_2) (a_0, b_0) (a_1, b_1) (a_2, b_1) (a_0, b_1) (a_1, b_0) (a_0, b_2)$

Vemos que para la instancia I, el matching $M_1 = \{(a_0, b_0, c_2), (a_1, b_1, c_1), (a_2, b_2, c_0)\}$ es estable, mientras que el matching $M_2 = \{(a_0, b_2, c_1), (a_1, b_1, c_0), (a_2, b_0, c_2)\}$ no es estable, ya que es bloqueado por la terna (a_0, b_0, c_0) .

7.6.1 Subconjuntos de instancias: SSP con prioridad

Para facilitar la búsqueda de una solución al problema de matching estable en instancias SSP, podemos tratar separadamente para el conjunto de listas por pares, al subconjunto de las instancias de esta familia que cumplen con la característica de priorizar uno de los conjuntos sobre el otro. Supongamos, sin pérdida de generalidad, que el conjunto C prioriza a A.

Decimos entonces que (a, b, c) es una *terna bloqueante* del matching M si $(a, b, c) \notin M$ y:

1. $(b \geq_a M_B(a) \text{ y } c >_a M_B(a))$ o $(b >_a M_B(a) \text{ y } c \geq_a M_C(a))$ y
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b))$ o $(a >_b M_A(b) \text{ y } c \geq_b M_C(b))$ y
3. $a >_c M_A(c)$ o $(a = M_A(c) \text{ y } b >_{c/a} M_B(c))$

Veamos el siguiente ejemplo de una instancia SSP con prioridad.

Ejemplo 7.13

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $b_0 \ b_1 \ b_2$
 $c_0 \ c_2 \ c_1$
 a_1 : $b_2 \ b_1 \ b_0$
 $c_0 \ c_2 \ c_1$
 a_2 : $b_2 \ b_0 \ b_1$
 $c_1 \ c_2 \ c_0$

b_0 : $a_0 \ a_1 \ a_2$
 $c_0 \ c_1 \ c_2$
 b_1 : $a_1 \ a_0 \ a_2$
 $c_0 \ c_1 \ c_2$
 b_2 : $a_0 \ a_2 \ a_1$
 $c_0 \ c_2 \ c_1$

c_0 : $(a_2, b_1) (a_2, b_2) (a_2, b_0) (a_0, b_0) (a_1, b_1) (a_1, b_2) (a_0, b_1) (a_0, b_2) (a_0, b_0)$
 c_1 : $(a_0, b_2) (a_0, b_0) (a_0, b_1) (a_1, b_0) (a_1, b_2) (a_1, b_1) (a_2, b_1) (a_2, b_2) (a_2, b_0)$
 c_2 : $(a_2, b_2) (a_2, b_0) (a_2, b_1) (a_0, b_0) (a_0, b_1) (a_0, b_2) (a_1, b_1) (a_1, b_0) (a_1, b_2)$

El matching $M_1 = \{(a_0, b_0, c_2), (a_1, b_1, c_0), (a_2, b_2, c_1)\}$ es estable para la instancia I.

En cambio, el matching $M_2 = \{(a_0, b_2, c_1), (a_1, b_0, c_2), (a_2, b_1, c_0)\}$ no es estable. La terna (a_0, b_0, c_2) es bloqueante.

Daremos a continuación un algoritmo que siempre encuentra un matching estable en instancias de SSP con prioridad.

Algoritmo 7.8

- 1) Hallar un matching estable M_{AB} a través del algoritmo de Gale y Shapley, entre los agentes del conjunto A y los de B utilizando las correspondientes listas elementales entre A y B.
- 2) Hallar un matching estable M_{AC} a través del algoritmo de Gale y Shapley, entre los agentes del conjunto A y los de C utilizando las correspondientes listas elementales entre A y C.
- 3) $M = M_{AB} J_A M_{AC}$

Teorema

El Algoritmo 7.8 es correcto y completo.

Demostración

Demostración de correctitud

a) Probar que la solución que el algoritmo genera es un matching.

Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un b_j asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

Del paso 1 cada par (a_i, b_j) resulta con un c_k asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

b) Probar que el matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a, b, c) no perteneciente al matching M , tal que:

1. $(b \geq_a M_B(a) \text{ y } c >_a M_B(a))$ o $(b >_a M_B(a) \text{ y } c \geq_a M_C(a))$ y
2. $(a \geq_b M_A(b) \text{ y } c >_b M_C(b))$ o $(a >_b M_A(b) \text{ y } c \geq_b M_C(b))$ y
3. $a >_c M_A(c)$ o $(a = M_A(c) \text{ y } b >_{c/a} M_B(c))$

Supongamos que existe tal terna.

Por (1) $c >_a M_C(a)$ o $c = M_C(a)$

- Si $c >_a M_C(a)$, como $a >_c M_A(c)$ (por 3, ya que no puede ser $a = M_A(c)$), significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 2. Absurdo (por correctitud del algoritmo de Gale y Shapley).
- Si $c = M_C(a) \Rightarrow b >_a M_B(a)$ (por 1). Además, $a \geq_b M_A(b)$ (por 2)
 $b >_a M_B(a)$ y $a \geq_b M_A(b) \Rightarrow b >_a M_B(a)$ y $a >_b M_A(b)$, lo cual significa que existe un par (a, b) que bloquea al matching de pares A-B obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud del algoritmo de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas SSP.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

7.6.2 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir el conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones. La formulación entera para definir el conjunto de matchings estables en instancias SSP queda definida de la siguiente forma:

Sea:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j \geq a \\ k > c}} x_{a,j,k} + \sum_{\substack{j > b \\ k \geq c}} x_{a,j,k} + \sum_{\substack{i \geq a \\ b > c}} x_{i,b,k} + \sum_{\substack{i > a \\ b \geq c}} x_{i,b,k} + \sum_{\substack{(i,j) > (a,b) \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

7.7 PPU

Una instancia del problema de PPU está formada por tres conjuntos A, B y C y un conjunto P de listas de preferencias, con las siguientes estructuras:

$\forall a \in A$, a tiene listas de preferencias por pares,

$\forall b \in B$, b tiene listas de preferencias por pares y

$\forall c \in C$, c tiene listas de preferencias únicas.

Esta familia no será subclasificada ya que las listas por pares incluyen los otros 2 conjuntos de individuos y es indistinto si las listas únicas afectan al primer o al segundo conjunto.

Decimos entonces que (a,b,c) es una *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $(b, c) \succ_a M_{BC}(a)$ y
2. $(a, c) \succ_b M_{AC}(b)$ y
3. $a \succ_c M_A(c)$

Nótese que dado que tanto A como B expresan sus preferencias de la misma manera (como listas por pares) podemos asumir sin pérdida de generalidad que los $c \in C$ eligen sobre los agentes de uno de los dos conjuntos, en este caso asumimos que lo hacen sobre A .

En la Figura 7.11, podemos ver gráficamente las relaciones entre los conjuntos en instancias PPU.

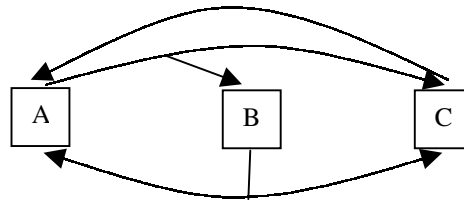


Figura 7.11 Preferencias entre los conjuntos en una instancia PPU.

Veamos un ejemplo de una instancia PPU.

Ejemplo 7.14

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P :

a_0 : $(b_0, c_1) (b_1, c_0) (b_0, c_0) (b_2, c_1) (b_2, c_2) (b_1, c_1) (b_2, c_0) (b_0, c_2) (b_1, c_2)$

a_1 : $(b_1, c_1) (b_0, c_0) (b_1, c_0) (b_2, c_1) (b_2, c_2) (b_0, c_1) (b_2, c_0) (b_0, c_2) (b_1, c_2)$

a_2 : $(b_1, c_0) (b_2, c_2) (b_0, c_0) (b_2, c_1) (b_0, c_1) (b_1, c_1) (b_2, c_0) (b_0, c_2) (b_1, c_2)$

b_0 : $(a_1, c_0) (a_0, c_0) (a_0, c_2) (a_2, c_1) (a_2, c_2) (a_1, c_1) (a_2, c_0) (a_0, c_1) (a_1, c_2)$

b_1 : $(a_1, c_0) (a_1, c_1) (a_0, c_2) (a_2, c_1) (a_2, c_2) (a_0, c_0) (a_2, c_0) (a_0, c_1) (a_1, c_2)$

b_2 : $(a_2, c_0) (a_2, c_2) (a_0, c_2) (a_2, c_1) (a_0, c_0) (a_1, c_1) (a_1, c_0) (a_0, c_1) (a_1, c_2)$

c_0 : $a_0 a_1 a_2$

c_1 : $a_0 a_1 a_2$

c_2 : $a_2 a_1 a_0$

Podemos dar el siguiente matching estable $M = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$. Mientras que el matching $M' = \{(a_0, b_2, c_0), (a_1, b_1, c_1), (a_2, b_0, c_2)\}$ es bloqueado por la terna (a_2, b_2, c_2) .

7.7.1 Subconjuntos de instancias: PPU con prioridad

Para el conjunto de listas por pares, podemos tratar separadamente al subconjunto de las instancias de esta familia que cumplen con la característica de priorizar uno de los conjuntos sobre el otro. A su vez subclasificaremos las instancias según cuál sea el conjunto priorizado tal como sigue:

- ⊃ $P_p P_p U$: los individuos de los conjuntos que expresan sus preferencias como listas de preferencias por pares se priorizan entre sí.
- ⊃ $P_U P_U U$: los individuos de los conjuntos que expresan sus preferencias como listas de preferencias por pares priorizan sobre el tercer conjunto.
- ⊃ $P_p P_U U$: los individuos de uno de los conjuntos que expresan sus preferencias como listas de preferencias por pares priorizan al otro y este último prioriza al conjunto que expresa sus preferencias como listas únicas.

7.7.1.1 $P_p P_p U$

Recordemos que, en instancias $P_p P_p U$, los agentes de los conjuntos A y B, definen sus preferencias sobre pares de los dos conjuntos restantes, pero priorizándose mutuamente entre sí. Los individuos del conjunto C tienen listas de preferencias únicas sobre alguno de los conjuntos A o B. Supongamos, sin pérdida de generalidad, que C elige sobre A.

Decimos que (a,b,c) es *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b \succ_a M_B(a)$ o $(b=M_B(a) \text{ y } c \succ_{a/b} M_C(a))$ y
2. $a \succ_b M_A(b)$ o $(a=M_A(b) \text{ y } c \succ_{b/a} M_C(b))$ y
3. $a \succ_c M_A(c)$

Veamos un ejemplo de una instancia de $P_p P_p U$.

Ejemplo 7.15

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $(b_0, c_0) (b_0, c_1) (b_0, c_2) (b_1, c_0) (b_1, c_1) (b_1, c_2) (b_2, c_0) (b_2, c_1) (b_2, c_2)$
 a_1 : $(b_0, c_1) (b_0, c_2) (b_0, c_0) (b_1, c_1) (b_1, c_2) (b_1, c_0) (b_2, c_1) (b_2, c_2) (b_2, c_0)$
 a_2 : $(b_2, c_1) (b_2, c_2) (b_2, c_0) (b_1, c_1) (b_1, c_2) (b_1, c_0) (b_0, c_1) (b_0, c_2) (b_0, c_0)$

b_0 : $(a_0, c_1) (a_0, c_2) (a_0, c_0) (a_2, c_1) (a_2, c_2) (a_2, c_0) (a_1, c_1) (a_1, c_2) (a_1, c_0)$
 b_1 : $(a_1, c_1) (a_1, c_0) (a_1, c_2) (a_2, c_1) (a_2, c_0) (a_2, c_2) (a_0, c_1) (a_0, c_0) (a_0, c_2)$
 b_2 : $(a_2, c_1) (a_2, c_0) (a_2, c_2) (a_1, c_1) (a_1, c_0) (a_1, c_2) (a_0, c_1) (a_0, c_0) (a_0, c_2)$

c_0 : $a_0 a_1 a_2$

c_1 : $a_2 a_1 a_0$

c_2 : $a_2 a_1 a_0$

Podemos dar el siguiente matching estable $M = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$. Mientras que el matching $M' = \{(a_2, b_0, c_0), (a_1, b_1, c_1), (a_0, b_2, c_2)\}$ es bloqueado por la terna (a_0, b_0, c_0) .

Algoritmo 7.9

- 1) Se encuentra un matching estable entre los conjuntos A y B por el algoritmo de Gale y Shapley.

Todos los c_j ($1 \leq j \leq n$) están sin asignar a los pares (a_i, b_k) formados en el punto anterior.

- 2) Se encuentra un matching estable bidimensional entre los individuos de A y los de C, considerando para estos últimos las listas relativas a los b elegidos en el punto 1.

El matching así obtenido es estable.

Teorema

El Algoritmo 7.9 es correcto y completo.

Demostración

Demostración de correctitud

- c) Probar que la solución que el algoritmo genera es un matching.

Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un b_k asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

Del paso 2 cada c_j es asignado una y sólo una vez a pares distintos de (a_i, b_k) .

- d) Probar que el matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M, tal que:

1. $b >_a M_B(a)$ o $(b=M_B(a) \text{ y } c >_{a/b} M_C(a))$ y
2. $a >_b M_A(b)$ o $(a=M_A(b) \text{ y } c >_{b/a} M_C(b))$ y
3. $a >_c M_A(c)$

distribuyendo nos queda:

- | | | |
|------|--|---|
| i) | $b >_a M_B(a)$ y $a >_b M_A(b)$ y $a >_c M_A(c)$ | o |
| ii) | $b >_a M_B(a)$ y $a=M_A(b)$ y $c >_{b/a} M_C(b)$ y $a >_c M_A(c)$ | o |
| iii) | $b=M_B(a)$ y $c >_{a/b} M_C(a)$ y $a >_b M_A(b)$ y $a >_c M_A(c)$ | o |
| iv) | $b=M_B(a)$ y $c >_{a/b} M_C(a)$ y $a=M_A(b)$ y $c >_{b/a} M_C(b)$ y $a >_c M_A(c)$ | |

Supongamos que existe tal terna.

Caso 1: la fórmula (i) dice $b >_a M_B(a)$ y $a >_b M_A(b)$, significa que existe un par (a,b) que bloquea al matching de pares M_{AB} obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 2: la fórmula (ii) dice $a = M_A(b) \Rightarrow b = M_B(a)$ luego no puede darse $b=M_B(a)$ y $b >_a M_B(a)$. Absurdo

Caso 3: la fórmula (iii) dice $b = M_B(a) \Rightarrow a = M_A(b)$ luego no puede darse $a=M_A(b)$ y $a >_b M_A(b)$. Absurdo

Caso 4: la fórmula (iv) dice $b=M_B(a)$ y $c >_{a/b} M_C(a)$ y $a >_c M_A(c) \Rightarrow c >_{a/M_B(a)} M_C(a)$ y $a >_c M_A(c)$, esto implica que el par (a,c) bloquea el matching M_{AC} con listas de A para C relativas a $M_B(a)$. Absurdo (por correctitud del algoritmo de Gale y Shapley)

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas $P_P P_U$ con prioridad.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

7.7.1.2 $P_U P_U$

Recordemos que, en instancias $P_U P_U$, los agentes de los conjuntos A y B, definen sus preferencias sobre pares de los dos conjuntos restantes, pero priorizando al conjunto que tiene listas únicas. Los individuos del conjunto C tienen listas de preferencias únicas sobre alguno de los conjuntos A o B. Supongamos, sin pérdida de generalidad, que C elige sobre A.

Decimos que (a,b,c) es *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $c \succ_a M_C(a)$ o $(c=M_C(a) \text{ y } b \succ_{a/c} M_B(a))$ y
2. $c \succ_b M_C(b)$ o $(c=M_C(b) \text{ y } a \succ_{b/c} M_A(b))$ y
3. $a \succ_c M_A(c)$

Veamos un ejemplo de una instancia $P_U P_U$.

Ejemplo 7.16

Dada la instancia $I = (A, B, C; P)$ donde $A=\{a_0, a_1, a_2\}$, $B=\{b_0, b_1, b_2\}$, $C=\{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $(b_1, c_0) (b_0, c_0) (b_2, c_0) (b_1, c_1) (b_0, c_1) (b_2, c_1) (b_1, c_2) (b_0, c_2) (b_2, c_2)$
 a_1 : $(b_1, c_0) (b_2, c_0) (b_0, c_0) (b_1, c_1) (b_2, c_1) (b_0, c_1) (b_1, c_2) (b_2, c_2) (b_1, c_2)$
 a_2 : $(b_0, c_2) (b_0, c_2) (b_2, c_2) (b_1, c_1) (b_0, c_1) (b_2, c_1) (b_1, c_0) (b_0, c_0) (b_2, c_0)$

b_0 : $(a_2, c_0) (a_0, c_0) (a_1, c_0) (a_2, c_1) (a_0, c_1) (a_1, c_1) (a_2, c_2) (a_0, c_2) (a_1, c_2)$
 b_1 : $(a_1, c_2) (a_0, c_2) (a_2, c_2) (a_1, c_1) (a_0, c_1) (a_2, c_1) (a_1, c_0) (a_0, c_0) (a_2, c_0)$
 b_2 : $(a_1, c_2) (a_2, c_2) (a_0, c_2) (a_1, c_1) (a_2, c_1) (a_0, c_1) (a_1, c_0) (a_2, c_0) (a_0, c_0)$

c_0 : $a_1 a_2 a_0$

c_1 : $a_1 a_2 a_0$

c_2 : $a_0 a_2 a_1$

Podemos dar el siguiente matching estable $M=\{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$. Mientras que el matching $M'=\{(a_0, b_0, c_0), (a_2, b_1, c_1), (a_1, b_2, c_2)\}$ es bloqueado por la terna (a_1, b_1, c_1) .

Damos a continuación un algoritmo que siempre encuentra un matching estable en instancias $P_U P_U$.

Algoritmo 7.10

- 1) Se encuentra un matching estable entre los individuos del conjunto A y los de C por el algoritmo de Gale y Shapley.

Todos los c_j ($1 \leq j \leq n$) están sin asignar a los pares (a_i, b_k) formados en el punto anterior.

- 2) Para cada b_j ($1 \leq j \leq n$)
 b_j se asigna a alguno de los pares formados en el paso anterior. Por ejemplo: elige el primer a_i de su lista de preferencias tal que no esté asignado a algún b .

El matching así obtenido es estable.

Teorema

El Algoritmo 7.10 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.
Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada a_i resulta con un c_k asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

Del paso 2 cada b_j es asignado una y sólo una vez a pares distintos de (a_i, c_k)

- b) Probar que el matching encontrado es estable.

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a, b, c) no perteneciente al matching M , tal que:

1. $c \succ_a M_C(a)$ o $(c = M_C(a) \text{ y } b \succ_{a/c} M_B(a))$ y
2. $c \succ_b M_C(b)$ o $(c = M_C(b) \text{ y } a \succ_{b/c} M_A(b))$ y
3. $a \succ_c M_A(c)$

distribuyendo nos queda:

- | | | |
|------|--|---|
| i) | $c \succ_a M_C(a)$ y $c \succ_b M_C(b)$ y $a \succ_c M_A(c)$ | o |
| ii) | $c \succ_a M_C(a)$ y $c = M_C(b)$ y $a \succ_{b/c} M_A(b)$ y $a \succ_c M_A(c)$ | o |
| iii) | $c = M_C(a)$ y $b \succ_{a/c} M_B(a)$ y $c \succ_b M_C(b)$ y $a \succ_c M_A(c)$ | o |
| iv) | $c = M_C(a)$ y $b \succ_{a/c} M_B(a)$ y $c = M_C(b)$ y $a \succ_{b/c} M_A(b)$ y $a \succ_c M_A(c)$ | |

Supongamos que existe tal terna.

Caso 1: las fórmulas (i) y (ii) dicen $c \succ_a M_C(a)$ y $a \succ_c M_A(c)$, significa que existe un par (a, c) que bloquea al matching de pares M_{AC} obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 2: la fórmula (iii) dice $c = M_C(a) \Rightarrow a = M_A(c)$ luego no puede darse $a = M_A(c)$ y $a \succ_c M_A(c)$. Absurdo

Caso 3: las fórmulas (iii) y (iv) dicen $b = M_B(a) \Rightarrow a = M_A(b)$ luego no puede darse $a = M_A(b)$ y $a \succ_b M_A(b)$. Absurdo

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas $P_U P_U U$ con prioridad.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$.

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: La asignación tiene una complejidad de $O(n)$

7.7.1.3 $P_P P_U U$

Notemos que para $P_P P_U U$ no son indistintas las elecciones del tercer conjunto, tenemos entonces una partición del conjunto de instancias de este tipo tal como sigue:

- $P_P P_U U_{P_P}$: el tercer conjunto elige sobre los individuos del primero.
- $P_P P_U U_{P_U}$: el tercer conjunto elige sobre los individuos del segundo.

7.7.1.3.1 $P_P P_U U_{P_U}$

Recordemos que, en instancias $P_P P_U U_{P_U}$, los agentes de los conjuntos A y B, definen sus preferencias sobre pares de los dos conjuntos restantes, pero en este caso uno de ellos prioriza al conjunto que tiene listas únicas y el otro al que tiene listas por pares. Los individuos del conjunto C tienen listas de preferencias únicas sobre el conjunto que lo prioriza a él.

Decimos que (a,b,c) es *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b \succ_a M_B(a)$ o $(b=M_B(a) \text{ y } c \succ_{a/b} M_C(a))$ y
2. $c \succ_b M_C(b)$ o $(c=M_C(b) \text{ y } a \succ_{b/c} M_A(b))$ y
3. $b \succ_c M_B(c)$

Veamos el siguiente ejemplo de una instancia $P_P P_U U_{P_U}$.

Ejemplo 7.17

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $(b_0, c_0) (b_0, c_1) (b_0, c_2) (b_1, c_0) (b_1, c_1) (b_1, c_2) (b_2, c_0) (b_2, c_1) (b_2, c_2)$

a_1 : $(b_1, c_2) (b_1, c_1) (b_1, c_0) (b_0, c_1) (b_0, c_2) (b_0, c_0) (b_2, c_1) (b_2, c_2) (b_2, c_0)$

a_2 : $(b_2, c_0) (b_2, c_2) (b_2, c_1) (b_1, c_1) (b_1, c_2) (b_1, c_0) (b_0, c_1) (b_0, c_2) (b_0, c_0)$

b_0 : $(a_1, c_1) (a_0, c_1) (a_2, c_1) (a_0, c_0) (a_1, c_0) (a_2, c_0) (a_1, c_2) (a_2, c_2) (a_0, c_2)$

$b_1:$ $(a_1, c_0) (a_2, c_0) (a_0, c_0) (a_2, c_1) (a_1, c_1) (a_0, c_1) (a_0, c_2) (a_1, c_2) (a_2, c_2)$
 $b_2:$ $(a_2, c_1) (a_1, c_1) (a_0, c_1) (a_1, c_0) (a_2, c_0) (a_0, c_0) (a_0, c_2) (a_1, c_2) (a_2, c_2)$

 $c_0:$ $b_0 b_1 b_2$
 $c_1:$ $b_2 b_1 b_0$
 $c_2:$ $b_2 b_1 b_0$

El matching $M = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable, mientras que el matching $M' = \{(a_0, b_2, c_2), (a_1, b_1, c_1), (a_2, b_0, c_0)\}$ es bloqueado por la terna (a_0, b_0, c_0) .

Damos a continuación un algoritmo que siempre encuentra un matching estable en instancias $P_P P_U P_{Pu}$.

Algoritmo 7.11

- 1) Se encuentra un matching estable entre los individuos del conjunto B y los de C por el algoritmo de Gale y Shapley.

Todos los a_j ($1 \leq j \leq n$) están sin asignar a los pares (b_i, c_k) formados en el punto anterior.

- 2) Para cada a_j ($1 \leq j \leq n$)
 - a_j se asigna a alguno de los pares formados en el paso anterior. Por ejemplo: elige el primer b_i de su lista de preferencias tal que no esté asignado a algún a .

El matching así obtenido es estable.

Teorema

El Algoritmo 7.11 es correcto y completo.

Demostración

Demostración de correctitud

- a) Probar que la solución que el algoritmo genera es un matching.
 Debemos ver que cada a_i tiene asignado un par (b_j, c_k) tal que todos los b_j son distintos entre sí y todos los c_k también.

Del paso 1 cada b_i resulta con un c_k asignado, para todo $1 \leq i \leq n$. (por correctitud del algoritmo de Gale y Shapley).

Del paso 2 cada a_j es asignado una y sólo una vez a pares distintos de (b_i, c_k)

- b) Probar que el matching encontrado es estable-

Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a, b, c) no perteneciente al matching M , tal que:

1. $b \succ_a M_B(a)$ o $(b = M_B(a) \text{ y } c \succ_{a/b} M_C(a))$ y
2. $c \succ_b M_C(b)$ o $(c = M_C(b) \text{ y } a \succ_{b/c} M_A(b))$ y
3. $b \succ_c M_B(c)$

distribuyendo nos queda:

- | | | |
|------|--|---|
| i) | $b \succ_a M_B(a) \text{ y } c \succ_b M_C(b) \text{ y } b \succ_c M_B(c)$ | o |
| ii) | $b \succ_a M_B(a) \text{ y } c = M_C(b) \text{ y } a \succ_{b/c} M_A(b) \text{ y } b \succ_c M_B(c)$ | o |
| iii) | $b = M_B(a) \text{ y } c \succ_{a/b} M_C(a) \text{ y } c \succ_b M_C(b) \text{ y } b \succ_c M_B(c)$ | o |
| iv) | $b = M_B(a) \text{ y } c \succ_{a/b} M_C(a) \text{ y } c = M_C(b) \text{ y } a \succ_{b/c} M_A(b) \text{ y } b \succ_c M_B(c)$ | |

Supongamos que existe tal terna.

Caso 1: las fórmulas (i) y (iii) dicen $c >_b M_C(b)$ y $b >_c M_B(c)$, significa que existe un par (b, c) que bloquea al matching de pares M_{BC} obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 2: la fórmula (ii) dice $c = M_C(b) \Rightarrow b = M_B(c)$ luego no puede darse $b = M_B(c)$ y $b >_c M_B(c)$. Absurdo

Caso 3: la fórmula (iv) dice $b = M_B(a) \Rightarrow$ el par $(a,b) \in M$ y $c = M_C(B) \Rightarrow$ el par $(b,c) \in M$, luego la terna $(a,b,c) \in M$. Absurdo

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas $P_P P_U U_{P_P}$ con prioridad.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$.

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: La asignación tiene una complejidad de $O(n)$

7.7.1.3.2 $P_P P_U U_{P_P}$

Recordemos que, en instancias $P_P P_U U_{P_P}$, los agentes de los conjuntos A y B, definen sus preferencias sobre pares de los dos conjuntos restantes, pero en este caso uno de ellos prioriza al conjunto que tiene listas únicas y el otro al que tiene listas por pares. Los individuos del conjunto C tienen listas de preferencias únicas sobre el conjunto que no lo prioriza a él.

Decimos que (a,b,c) es *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $b >_a M_B(a)$ o $(b = M_B(a) \text{ y } c >_{a/b} M_C(a))$ y
2. $c >_b M_C(b)$ o $(c = M_C(b) \text{ y } a >_{b/c} M_A(b))$ y
3. $a >_c M_A(c)$

Veamos un ejemplo de una instancia $P_P P_U U_{P_P}$.

Ejemplo 7.18

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a_0 : $(b_0, c_0) (b_0, c_1) (b_0, c_2) (b_1, c_0) (b_1, c_1) (b_1, c_2) (b_2, c_0) (b_2, c_1) (b_2, c_2)$

$a_1:$ $(b_1, c_2) (b_1, c_1) (b_1, c_0) (b_0, c_1) (b_0, c_2) (b_0, c_0) (b_2, c_1) (b_2, c_2) (b_2, c_0)$
 $a_2:$ $(b_2, c_0) (b_2, c_2) (b_2, c_1) (b_1, c_1) (b_1, c_2) (b_1, c_0) (b_0, c_1) (b_0, c_2) (b_0, c_0)$

 $b_0:$ $(a_1, c_1) (a_0, c_1) (a_2, c_1) (a_2, c_0) (a_1, c_0) (a_0, c_0) (a_1, c_2) (a_2, c_2) (a_0, c_2)$
 $b_1:$ $(a_1, c_0) (a_2, c_0) (a_0, c_0) (a_2, c_1) (a_1, c_1) (a_0, c_1) (a_0, c_2) (a_1, c_2) (a_2, c_2)$
 $b_2:$ $(a_2, c_1) (a_1, c_1) (a_0, c_1) (a_1, c_0) (a_2, c_0) (a_0, c_0) (a_0, c_2) (a_1, c_2) (a_2, c_2)$

 $c_0:$ $a_0 a_1 a_2$
 $c_1:$ $a_2 a_1 a_0$
 $c_2:$ $a_2 a_1 a_0$

El matching $M = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable, mientras que el matching $M' = \{(a_0, b_2, c_2), (a_1, b_0, c_1), (a_2, b_0, c_0)\}$ es bloqueado por la terna (a_0, b_0, c_0) .

De la misma manera que observamos en el análisis de instancias $U_U P_U P_{U_U}$, podemos ver que el problema de encontrar un matching estable en instancias $P_P P_U U_{P_P}$ es una extensión del mismo problema en LUC, del cual conjeturamos que es NP-Completo. Observemos que las condiciones que definen una terna bloqueante incluyen las mismas condiciones que para LUC, incorporando además nuevas posibles ternas bloqueantes.

Esto nos lleva a basar nuestro estudio de instancias $P_P P_U U_{P_P}$ en lo estudiado oportunamente para LUC, incorporando las nuevas restricciones.

Lo enunciado previamente queda reflejado por la siguiente propiedad, la cual nos da una condición suficiente para encontrar un matching estable en instancias $P_P P_U U_{P_P}$.

Propiedad 7.2

Sean las siguientes instancias:

$I = (A, B, C; P)$ una instancia de $P_P P_U U_{P_P}$

$I' = (A, B, C; P')$ una instancia de LUC tal que $P' = \{P_B(a)/a \in A\} \cup \{P_C(b)/b \in B\} \cup \{P(c)/c \in C\}$.

$I_{AB} = (A, B; P'')$ una instancia del problema de matrimonios estables tal que $P'' = \{P_B(a)/a \in A\} \cup \{P_A(b) / b \in B\}$

$I_{AC} = (A, C; P''')$ una instancia del problema de matrimonios estables tal que $P''' = \{P_C(a)/a \in A\} \cup \{P(c) / c \in C\}$

Entonces, M estable en I' y M_{AB} estable en I_{AB} y M_{AC} estable en $I_{AC} \Rightarrow M$ estable en I .

Demostración

Supongamos M estable en I' y M_{AB} estable en I_{AB} y M_{AC} estable en I_{AC} , pero M no estable en I , entonces $\exists (a,b,c)$ terna bloqueante de M en I , es decir $\exists (a,b,c)$ tal que:

$(b >_a M_B(a) \text{ o } (b = M_B(a) \text{ y } c >_{a/b} M_C(a))) \text{ y}$

$(c >_b M_C(b) \text{ o } (c = M_C(b) \text{ y } a >_{b/c} M_A(b))) \text{ y}$

$a >_c M_A(c)$

\Rightarrow

$b >_a M_B(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c)$ o

$b >_a M_B(a) \text{ y } c = M_C(b) \text{ y } a >_{b/c} M_A(b) \text{ y } a >_c M_A(c)$ o

$b = M_B(a) \text{ y } c >_{a/b} M_C(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c)$ o

$b = M_B(a) \text{ y } c >_{a/b} M_C(a) \text{ y } c = M_C(b) \text{ y } a >_{b/c} M_A(b) \text{ y } a >_c M_A(c)$

Si $b >_a M_B(a) \text{ y } c >_b M_C(b) \text{ y } a >_c M_A(c) \Rightarrow (a,b,c)$ bloquea a M en I' . Absurdo, por hipótesis.

Si $b >_a M_B(a) \text{ y } c = M_C(b) \text{ y } a >_{b/c} M_A(b) \text{ y } a >_c M_A(c) \Rightarrow b >_a M_B(a) \text{ y } a >_{b/M_C(b)} M_A(b) \Rightarrow (a,b)$ bloquea a M_{AB} en I_{AB} . Absurdo, por hipótesis.

Si $b=M_B(a)$ y $c >_{a/b} M_C(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c) \Rightarrow c >_{a/M_B(a)} M_C(a)$ y $a >_c M_A(c) \Rightarrow (a,c)$ bloquea a M_{AC} en I_{AC} . Absurdo, por hipótesis.

Si $b=M_B(a)$ y $c >_{a/b} M_C(a)$ y $c=M_C(b)$ y $a >_{b/c} M_A(b)$ y $a >_c M_A(c) \Rightarrow b=M_B(a)$ y $c=M_C(b) \Rightarrow (a,b,c) \in M$. Absurdo, por hipótesis, ya que (a,b,c) bloquea a M .

Luego, M estable en I' y M_{AB} estable en I_{AB} y M_{AC} estable en $I_{AC} \Rightarrow M$ estable en I . ♦

7.7.2 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir el conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones.

Sea:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

Daremos a continuación la formulación entera que define el conjunto de todos los matchings estables en instancias PPU.

Las restricciones 1, 2, 3 y 5 garantizan que la solución es un matching, por lo tanto son comunes a todas las familias de instancias dentro de PPU. La cuarta restricción, la cual garantiza la estabilidad del matching, varía dependiendo de la subfamilia que estemos analizando.

Supongamos, sin pérdida de generalidad, que los agentes de C eligen sobre los individuos de A . La formulación entera para definir el conjunto de matchings estables en instancias $P_P P_U$ queda definida de la siguiente forma:

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{k > c \\ a}} x_{a,b,k} + \sum_{\substack{i > a \\ b \\ k \in C}} x_{i,b,k} + \sum_{\substack{k > c \\ b}} x_{a,b,k} + \sum_{\substack{i > a \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

En instancias $P_P P_U P_P$ la restricción (4) es reemplazada por:

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{k > c \\ a}} x_{a,b,k} + \sum_{\substack{k > c \\ b \\ i \in A}} x_{i,b,k} + \sum_{\substack{i > a \\ b}} x_{i,b,c} + \sum_{\substack{i > a \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, (a,b,c) \in A \times B \times C \quad (4)$$

En instancias $P_P P_U U_{P_U}$ la restricción (4) es reemplazada por:

$$\sum_{\substack{j > b \\ a \\ k \in C}} x_{a,j,k} + \sum_{\substack{k > c \\ a}} x_{a,b,k} + \sum_{\substack{k > c \\ b \\ i \in A}} x_{i,b,k} + \sum_{\substack{i > a \\ b}} x_{i,b,c} + \sum_{\substack{j > b \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, (a,b,c) \in A \times B \times C \quad (4)$$

En instancias $P_U P_U U$, y suponiendo que los agentes de C eligen sobre A . la restricción (4) es reemplazada por:

$$\sum_{\substack{k > c \\ a \\ j \in B}} x_{a,j,k} + \sum_{\substack{j > b \\ a}} x_{a,j,c} + \sum_{\substack{k > c \\ b \\ i \in A}} x_{i,b,k} + \sum_{\substack{i > a \\ b}} x_{i,b,c} + \sum_{\substack{j > b \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, (a,b,c) \in A \times B \times C \quad (4)$$

7.8 PPS

Una instancia del problema de PPS está formada por tres conjuntos A , B y C y un conjunto P de listas de preferencias, con las siguientes estructuras:

- $\forall a \in A$, a tiene listas de preferencias por pares,
- $\forall b \in B$, b tiene listas de preferencias por pares y
- $\forall c \in C$, c tiene listas de preferencias simples.

Esta familia no será subclasificada ya que tanto las listas por pares como las listas simples incluyen los otros 2 conjuntos de individuos.

Decimos que (a,b,c) es *terna bloqueante* del matching M si $(a,b,c) \notin M$ y:

1. $(b, c) >_a M_{BC}(a)$ y
2. $(a, c) >_b M_{AC}(b)$ y
3. $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$

En la Figura 7.12 podemos ver gráficamente las relaciones entre las preferencias de los conjuntos en una instancia PPS.

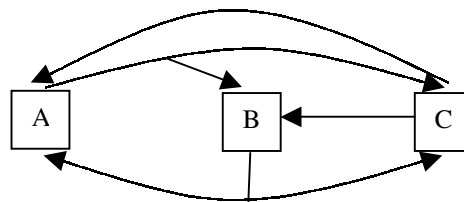


Figura 7.12 Preferencias entre los conjuntos en una instancia PPS.

Veamos un ejemplo de una instancia PPS.

Ejemplo 7.19

Dada la instancia $I = (A, B, C; P)$ donde $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:
 a_0 : $(b_1, c_0) (b_1, c_1) (b_0, c_0) (b_2, c_2) (b_0, c_1) (b_1, c_2) (b_2, c_0) (b_2, c_1) (b_0, c_2)$
 a_1 : $(b_1, c_2) (b_2, c_1) (b_1, c_0) (b_0, c_1) (b_2, c_2) (b_0, c_0) (b_1, c_1) (b_0, c_2) (b_2, c_0)$
 a_2 : $(b_2, c_0) (b_2, c_2) (b_1, c_1) (b_2, c_1) (b_1, c_2) (b_0, c_0) (b_0, c_1) (b_0, c_2) (b_1, c_0)$

b_0 : $(a_1, c_2) (a_0, c_1) (a_2, c_1) (a_2, c_0) (a_1, c_0) (a_0, c_0) (a_1, c_1) (a_2, c_2) (a_0, c_2)$
 b_1 : $(a_1, c_1) (a_2, c_0) (a_0, c_0) (a_2, c_2) (a_1, c_0) (a_0, c_1) (a_0, c_2) (a_1, c_2) (a_2, c_1)$
 b_2 : $(a_2, c_0) (a_1, c_1) (a_0, c_0) (a_1, c_0) (a_2, c_1) (a_0, c_1) (a_0, c_2) (a_1, c_2) (a_2, c_2)$

c_0 : $a_0 a_1 a_2$
 $b_2 b_0 b_1$
 c_1 : $a_2 a_1 a_0$
 $b_1 b_0 b_2$
 c_2 : $a_2 a_1 a_0$
 $b_2 b_1 b_0$

El matching $M = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable mientras que el matching $M' = \{(a_0, b_2, c_2), (a_1, b_1, c_1), (a_2, b_0, c_0)\}$ es bloqueado por la terna (a_2, b_2, c_0) .

7.8.1 Subconjuntos de instancias: PPS con prioridad

Para los conjuntos de listas por pares, podemos tratar separadamente al subconjunto de las instancias de esta familia que cumplen con la característica de priorizar uno de los conjuntos sobre el otro. A su vez subclasificaremos las instancias según cuál sea el conjunto priorizado tal como sigue:

- ⇐ $P_p P_p S$: los individuos de los conjuntos que expresan sus preferencias como listas de preferencias por pares se priorizan entre sí. Entonces, decimos que una terna $(a, b, c) \notin M$ bloquea M si:
 1. $b >_a M_B(a)$ o $(b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a))$ y
 2. $a >_b M_A(b)$ o $(a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b))$ y
 3. $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$
- ⇐ $P_p P_s S$: los individuos de uno de los conjuntos que expresan sus preferencias como listas de preferencias por pares priorizan al otro y este último prioriza al conjunto que expresa sus preferencias como listas simples. Entonces, decimos que una terna $(a, b, c) \notin M$ bloquea M si:
 1. $b >_a M_B(a)$ o $(b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a))$ y
 2. $c >_b M_C(b)$ o $(c = M_C(b) \text{ y } (a, c) >_b M_{AC}(b))$ y
 3. $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$
- ⇐ $P_s P_s S$: los individuos de los conjuntos que expresan sus preferencias como listas de preferencias por pares priorizan sobre el tercer conjunto. Entonces, decimos que una terna $(a, b, c) \notin M$ bloquea M si:
 1. $c >_a M_C(a)$ o $(c = M_C(a) \text{ y } (b, c) >_a M_{BC}(a))$ y
 2. $c >_b M_C(b)$ o $(c = M_C(b) \text{ y } (a, c) >_b M_{AC}(b))$ y
 3. $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$

Veamos un ejemplo de una instancia PPS con prioridad.

Ejemplo 7.20

Dada la instancia $I = (A, B, C; P)$, en la cual A tiene listas por pares con prioridad sobre B , B tiene listas por pares con prioridad sobre A y C tiene listas simples, con $A = \{a_0, a_1, a_2\}$, $B = \{b_0, b_1, b_2\}$, $C = \{c_0, c_1, c_2\}$ y las siguientes listas de preferencias:

P:

a₀: (b₀, c₀) (b₀, c₁) (b₀, c₂) (b₁, c₀) (b₁, c₁) (b₁, c₂) (b₂, c₀) (b₂, c₁) (b₂, c₂)

a₁: (b₁, c₂) (b₁, c₁) (b₁, c₀) (b₀, c₁) (b₀, c₂) (b₀, c₀) (b₂, c₁) (b₂, c₂) (b₂, c₀)

a₂: (b₂, c₀) (b₂, c₂) (b₂, c₁) (b₁, c₁) (b₁, c₂) (b₁, c₀) (b₀, c₁) (b₀, c₂) (b₀, c₀)

b₀: (a₀, c₁) (a₀, c₂) (a₀, c₀) (a₂, c₁) (a₂, c₂) (a₂, c₀) (a₁, c₁) (a₁, c₂) (a₁, c₀)

b₁: (a₁, c₁) (a₁, c₀) (a₁, c₂) (a₂, c₁) (a₂, c₀) (a₂, c₂) (a₀, c₁) (a₀, c₀) (a₀, c₂)

b₂: (a₂, c₁) (a₂, c₀) (a₂, c₂) (a₁, c₁) (a₁, c₀) (a₁, c₂) (a₀, c₁) (a₀, c₀) (a₀, c₂)

c₀: a₀ a₁ a₂
b₂ b₀ b₁

c₁: a₂ a₁ a₀
b₁ b₀ b₂

c₂: a₂ a₁ a₀
b₂ b₁ b₀

El matching $M = \{(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2)\}$ es estable mientras que el matching $M' = \{(a_0, b_2, c_2), (a_1, b_1, c_1), (a_2, b_0, c_0)\}$ es bloqueado por la terna (a_0, b_2, c_0) .

A continuación, daremos un algoritmo que permite encontrar un matching estable en instancias PPS con prioridad.

Algoritmo 7.12

Sean las siguientes preferencias entre conjuntos: A y B listan sus preferencias sobre pares de los otros dos conjuntos priorizando sobre uno de ellos.

- 1) Se encuentra un matching estable bidimensional entre B y el conjunto priorizado por los agentes en B por el algoritmo de Gale y Shapley.
- 2) Se encuentra un matching estable bidimensional entre el tercer conjunto y el conjunto priorizado (si se trata de C puede optar por cualquiera de los otros dos conjuntos, A o B). Si el conjunto priorizado es B, trabajar con las listas de B relativas a la elección realizada en el punto anterior.
- 3) Se forman las ternas realizando la junta entre los matchings obtenidos en los pasos 1 y 2.

El matching así obtenido es estable.

Teorema

El Algoritmo 7.12 es correcto y completo.

Demostración

Demostración de correctitud

a) Probar que la solución que el algoritmo genera es un matching.
Debemos ver que cada a tiene asignado un par (b, c) tal que todos los b son distintos entre sí y todos los c también.

Del paso 1 cada b_i resulta con un agente asignado del conjunto priorizado por los agentes en B (por correctitud del algoritmo de Gale y Shapley).

Del paso 3 cada agente del tercer conjunto resulta con asignado a un agente de B o de su priorizado. (por correctitud del algoritmo de Gale y Shapley).

b) Probar que el matching encontrado es estable. En este punto se hace necesario distinguir según los conjuntos sobre los que los agentes expresan sus preferencias. Dijimos anteriormente que tenemos 3 posibilidades:

- b.1) $P_P P_P S$: los agentes en A priorizan sobre B y los de B sobre A.
- b.2) $P_P P_S S$: los agentes en A priorizan sobre B y los de B sobre C.
- b.3) $P_S P_S S$: los agentes en A priorizan sobre C y los de B sobre C.

b.1) $P_P P_P S$:

En este caso el algoritmo halla un matching estable B-A y hace la junta con el matching estable entre C y cualquiera de los otros dos conjuntos, sin pérdida de generalidad, asumamos A. Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M, tal que:

- $$b >_a M_B(a) \text{ o } (b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a)) \text{ y}$$
- $$a >_b M_A(b) \text{ o } (a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b)) \text{ y}$$
- $$((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$$
- └
- i) $b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$ o
 - ii) $b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a >_c M_A(c) \text{ y } b \geq_c M_B(c)$ o
 - iii) $b >_a M_B(a) \text{ y } a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$ o
 - iv) $b >_a M_B(a) \text{ y } a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b) \text{ y } a >_c M_A(c) \text{ y } b \geq_c M_B(c)$ o
 - v) $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \text{ y } a >_b M_A(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$ o
 - vi) $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \text{ y } a >_b M_A(b) \text{ y } a >_c M_A(c) \text{ y } b \geq_c M_B(c)$ o
 - vii) $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \text{ y } a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$ o
 - viii) $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \text{ y } a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b) \text{ y } a >_c M_A(c) \text{ y } b \geq_c M_B(c)$ o

Supongamos que existe tal terna.

Caso 1: $b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$

Como $b >_a M_B(a) \text{ y } a >_b M_A(b)$ significa que existe un par (a, b) que bloquea al matching de pares A-B obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 2: $b >_a M_B(a) \text{ y } a >_b M_A(b) \text{ y } a >_c M_A(c) \text{ y } b \geq_c M_B(c)$

Como $b >_a M_B(a) \text{ y } a >_b M_A(b)$ significa que existe un par (a, b) que bloquea al matching de pares A-B obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 3: $b >_a M_B(a) \text{ y } a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$

Como $a = M_A(b) \Rightarrow b = M_B(a)$. Absurdo ($b >_a M_B(a)$)

Caso 4: $b >_a M_B(a) \text{ y } a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b) \text{ y } a >_c M_A(c) \text{ y } b \geq_c M_B(c)$

Como $a = M_A(b) \Rightarrow b = M_B(a)$. Absurdo ($b >_a M_B(a)$)

Caso 5: $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \text{ y } a >_b M_A(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$

Como $b = M_B(a) \Rightarrow a = M_A(b)$. Absurdo ($a >_b M_A(b)$)

Caso 6: $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \text{ y } a >_b M_A(b) \text{ y } a >_c M_A(c) \text{ y } b \geq_c M_B(c)$

Como $b = M_B(a) \Rightarrow a = M_A(b)$. Absurdo ($a >_b M_A(b)$)

Caso 7: $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \text{ y } a = M_A(b) \text{ y } (a, c) >_b M_{AC}(b) \text{ y } a \geq_c M_A(c) \text{ y } b >_c M_B(c)$

Como $b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a) \Rightarrow c >_{a/b} M_C(a)$.

Si $a >_c M_A(c)$, significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 3. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Si $a = M_A(c) \Rightarrow c = M_C(a)$. Absurdo (la terna (a,b,c) pertenecería al matching)

Caso 8: $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $a = M_A(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

Como $b = M_B(a)$, $(b, c) >_a M_{BC}(a) \Rightarrow c >_{a/b} M_C(a)$.

Como $a >_c M_A(c)$, significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 3. Absurdo (por correctitud del algoritmo de Gale y Shapley).

b.2) P_PP_SS:

En este caso el algoritmo halla un matching estable B-C y hace la junta con el matching estable entre A y B. Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M , tal que:

- $b >_a M_B(a)$ o $(b = M_B(a) \text{ y } (b, c) >_a M_{BC}(a))$ y
- $c >_b M_C(b)$ o $(c = M_C(b) \text{ y } (a, c) >_b M_{AC}(b))$ y
- $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$
- i) $b >_a M_B(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$ o
 - ii) $b >_a M_B(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$ o
 - iii) $b >_a M_B(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$ o
 - iv) $b >_a M_B(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$ o
 - v) $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$ o
 - vi) $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$ o
 - vii) $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$ o
 - viii) $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$ o

Supongamos que existe tal terna.

Caso 1: $b >_a M_B(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

Como $b >_c M_B(c)$ y $c >_b M_C(b)$ significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 2: $b >_a M_B(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

Si $b >_c M_B(c)$, y como $c >_b M_C(b)$ significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Si $b = M_B(c) \Rightarrow c = M_C(b)$. Absurdo ($c >_b M_C(b)$)

Caso 3: $b >_a M_B(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

Como $c = M_C(b) \Rightarrow b = M_B(c)$. Absurdo ($b >_c M_B(c)$)

Caso 4: $b >_a M_B(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

Como $c = M_C(b)$, $(a, c) >_b M_{AC}(b) \Rightarrow a >_{b/c} M_A(b)$.

Si $b >_a M_B(a)$, significa que existe un par (a, b) que bloquea al matching de pares B-A obtenido en el paso 3. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 5: $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

Como $b >_c M_B(c)$ y $c >_b M_C(b)$, significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 6: $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

Si $b >_c M_B(c)$, y como $c >_b M_C(b)$ significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Si $b = M_B(c) \Rightarrow c = M_C(b)$. Absurdo ($c >_b M_C(b)$)

Caso 7: $b = M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

Como $c = M_C(b) \Rightarrow b = M_B(c)$. Absurdo ($b >_c M_B(c)$)

Caso 8: $b=M_B(a)$ y $(b, c) >_a M_{BC}(a)$ y $c=M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$
 Como $c = M_C(b)$ y $b = M_B(a)$, significa que la terna (a,b,c) pertenece al matching. Absurdo

b.3) $P_S P_S S$:

En este caso el algoritmo halla un matching estable B-C y hace la junta con el matching estable entre A y C. Sea M el matching encontrado por el algoritmo.

Debemos ver que no existe terna (a,b,c) no perteneciente al matching M, tal que:

- $c >_a M_C(a)$ o $(c = M_C(a) \text{ y } (b, c) >_a M_{BC}(a))$ y
 $c >_b M_C(b)$ o $(c = M_C(b) \text{ y } (a, c) >_b M_{AC}(b))$ y
 $((a \geq_c M_A(c) \text{ y } b >_c M_B(c)) \text{ o } ((a >_c M_A(c) \text{ y } b \geq_c M_B(c)))$
 \int
 i) $c >_a M_C(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$ o
 ii) $c >_a M_C(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$ o
 iii) $c >_a M_C(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$ o
 iv) $c >_a M_C(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$ o
 v) $c = M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$ o
 vi) $c = M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$ o
 vii) $c = M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$
 o
 viii) $c = M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

Supongamos que existe tal terna.

Caso 1: $c >_a M_C(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

Si $a >_c M_A(c)$, como $c >_a M_C(a)$ significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Si $a = M_A(c) \Rightarrow c = M_C(a)$. Absurdo

Caso 2: $c >_a M_C(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

Como $a >_c M_A(c)$ y $c >_a M_C(a)$ significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 3: $c >_a M_C(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

Si $a >_c M_A(c)$, como $c >_a M_C(a)$ significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Si $a = M_A(c) \Rightarrow c = M_C(a)$. Absurdo

Caso 4: $c >_a M_C(a)$ y $c = M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

Como $a >_c M_A(c)$ y $c >_a M_C(a)$ significa que existe un par (a, c) que bloquea al matching de pares A-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley).

Caso 5: $c = M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

Como $c >_b M_C(b)$ y $b >_c M_B(c)$ significa que existe un par (b, c) que bloquea al matching de pares B-C obtenido en el paso 1. Absurdo (por correctitud del algoritmo de Gale y Shapley)

Caso 6: $c = M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c >_b M_C(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

$c = M_C(a) \int a = M_A(c)$, además $a >_c M_A(c)$. Absurdo

Caso 7: $c=M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c=M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a \geq_c M_A(c)$ y $b >_c M_B(c)$

$c = M_C(a)$ y $c = M_C(b) \Rightarrow (a,b,c)$ pertenece al matching. Absurdo

Caso 8: $c=M_C(a)$ y $(b, c) >_a M_{BC}(a)$ y $c=M_C(b)$ y $(a, c) >_b M_{AC}(b)$ y $a >_c M_A(c)$ y $b \geq_c M_B(c)$

$c = M_C(a)$ y $c = M_C(b) \Rightarrow (a,b,c)$ pertenece al matching. Absurdo

Demostración de completitud

Para probar la completitud del algoritmo, debemos probar que siempre que existe solución, el algoritmo la encuentra.

Sabemos que el algoritmo es correcto, es decir que si llega a un resultado, éste es solución (matching estable). También sabemos que, si el algoritmo termina, es porque encontró solución (por completitud de Gale y Shapley).

Sólo quedaría demostrar que el algoritmo termina, lo cual queda demostrado sabiendo que el algoritmo de Gale y Shapley termina. ♦

Corolario

Siempre existe solución al problema de hallar un matching estable en instancias con listas de preferencias combinadas PPS con prioridad.

Complejidad e Implementación

El algoritmo tiene complejidad de $O(n^2)$

Paso 1: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 2: El algoritmo de Gale y Shapley tiene una complejidad de $O(n^2)$

Paso 3: La Junta por un conjunto tiene una complejidad de $O(n)$

7.8.2 El conjunto de todos los Matchings Estables

Desde el punto de vista de la programación entera, podemos definir el conjunto de matchings estables como los puntos enteros factibles en el poliedro definido por un sistema de inecuaciones lineales o restricciones. La formulación entera para definir el conjunto de matchings estables en instancias PPS queda definida de la siguiente forma:

Sea:

$$x_{a,b,c} = \begin{cases} 1, & \text{si } (a,b,c) \text{ pertenece al matching} \\ 0, & \text{si } (a,b,c) \text{ no pertenece al matching} \end{cases} \quad \text{con } (a,b,c) \in A \times B \times C$$

$$\sum_{\substack{b \in B \\ c \in C}} x_{a,b,c} = 1, \quad \forall a \in A \quad (1)$$

$$\sum_{\substack{a \in A \\ c \in C}} x_{a,b,c} = 1, \quad \forall b \in B \quad (2)$$

$$\sum_{\substack{a \in A \\ b \in B}} x_{a,b,c} = 1, \quad \forall c \in C \quad (3)$$

$$\sum_{\substack{(j,k) > (b,c) \\ k > c \\ a}} x_{a,j,k} + \sum_{\substack{(i,k) > (a,c) \\ k \geq c \\ a}} x_{i,b,k} + \sum_{\substack{i \geq a \\ c}} x_{i,j,c} + \sum_{\substack{i > a \\ c}} x_{i,j,c} + x_{a,b,c} \geq 1, \quad (a,b,c) \in A \times B \times C \quad (4)$$

$$x_{a,b,c} \in \{0,1\}, \quad (a,b,c) \in A \times B \times C \quad (5)$$

7.9 El conjunto de todos los Matchings Estables

A lo largo del presente capítulo, hemos visto para cada clase particular de instancias con listas combinadas las correspondientes restricciones de programación lineal entera.

En esta sección veremos cómo se comporta el conjunto de todos los matchings estables de acuerdo a la relación de dominación que se define entre ellos, basándonos en lo analizado para cada tipo de listas de preferencias en los capítulos anteriores.

Para ello, analizaremos el comportamiento de las diferentes clases de instancias conforme a las distintas estructuras de listas de preferencias que combinen, para lo cual distinguimos cuatro categorías:

- Listas Únicas + Listas Simples
- Listas Únicas + Listas por Pares
- Listas Simples + Listas por Pares
- Listas Únicas + Listas Simples + Listas por Pares

Volviendo a la clasificación de instancias con listas combinadas detallada en el punto 7.1, nos quedan las siguientes relaciones:

– Listas Únicas + Listas Simples	→	UUS	
			SSU
– Listas Únicas + Listas por Pares	→	UUP	
			PPU
– Listas Simples + Listas por Pares	→	SSP	
			PPS
– Listas Únicas + Listas Simples + Listas por Pares	→	USP	

Veamos, para categoría mencionada, cómo se comporta el conjunto de todos los matchings estables. Veremos que el comportamiento varía, dentro de un mismo tipo de instancias, dependiendo del conjunto desde el cual se analice.

Listas Únicas + Listas Simples

Sabemos que para instancias con listas únicas existe un orden parcial definido por la relación de dominación entre clases de equivalencia, no así en la relación entre matchings.

Para instancias con listas simples, como vimos en el Capítulo 6, la relación de dominación definida entre los matchings determina un orden parcial.

En ambos casos, el conjunto de soluciones no conforma un reticulado.

De acuerdo a esto, el conjunto de matchings estables en instancias de UUS y SSU no forma un reticulado. Desde el punto de vista de los conjuntos con listas simples, la relación de dominación define un orden parcial entre matchings, mientras que para los conjuntos con listas únicas, este orden queda definido para las clases de equivalencia.

Listas Únicas + Listas por Pares

Recordemos que para listas de preferencias por pares, la estructura del conjunto de todos los matchings estables no conforma un reticulado, si bien la relación de dominación entre los mismos define un orden parcial.

Luego, el conjunto de matchings estables en instancias de UUP y PPU no forma un reticulado. Desde el punto de vista de los conjuntos con listas por pares, la relación de dominación define un orden parcial entre matchings, mientras que para los conjuntos con listas únicas, este orden queda definido para las clases de equivalencia.

Listas Simples + Listas por Pares

De la misma manera que lo detallado para los casos anteriores, se deduce que el conjunto de matchings estables en instancias de SSP y PPS no forma un reticulado. Sin embargo, en estas instancias la relación de dominación entre los matchings estables define un orden parcial desde el punto de vista de los tres conjuntos.

Listas Unicas + Listas Simples + Listas por Pares

Por último, podemos inferir que el conjunto de matchings estables en instancias de USP no forma un reticulado. Desde el punto de vista de los conjuntos con listas por pares y con listas simples, la relación de dominación define un orden parcial entre matchings, mientras que para el conjunto con listas únicas, este orden queda definido para las clases de equivalencia.

Capítulo 8: Conclusiones, Resultados y Problemas

Abiertos

Resultados y Conclusiones

En esta tesis analizamos una extensión del problema de los matrimonios estables a tres conjuntos. En trabajos previos se trató el problema de hallar un matching estable con listas de preferencias por pares completas, en donde se trata de encontrar un matching conformado por ternas de agentes pertenecientes a tres conjuntos disjuntos, que satisfaga la condición de estabilidad. El mismo fue planteado como un problema abierto en [Knu/76] y fue analizado posteriormente en [NH/95] donde se demuestra que determinar si una instancia dada tiene solución es NP-Completo. En nuestro estudio observamos que el conjunto de instancias analizadas por [NH/95], a las que llamamos instancias de listas por pares, forma parte de un universo de instancias más amplio, en donde el objetivo es el mismo, pues se trata de hallar un matching estable tridimensional, pero contempla diferentes estructuras de las listas de preferencias. Como resultado de este análisis, definimos y clasificamos este universo de instancias en cuatro grandes grupos según sus listas de preferencias: listas únicas (LU), listas simples (LS), listas por pares (LP) y listas combinadas (LC). También trabajamos en subconjuntos acotados, en donde analizamos la existencia y búsqueda de un matching estable, y la estructura del conjunto de todos los matchings estables. El análisis de esta clasificación puede verse en el Capítulo 3, mientras que el estudio de cada una de estas clases, puede verse en los Capítulos 4, 5, 6 y 7 respectivamente.

En la Figura 8.1 vemos el resultado de la clasificación del conjunto de instancias del problema de hallar un matching estable tridimensional sobre tres conjuntos según el formato de las listas de preferencias. La subclasificación correspondiente a listas combinadas es más compleja y se muestra en la Figura 8.2.

Definimos el problema general de hallar un matching estable con listas únicas (LU) como una generalización del problema de 3GSM circular que se deja abierto en [NH/95], ampliando el universo de instancias y clasificándolo en dos categorías de acuerdo a la preferencia entre los conjuntos: listas únicas mutuas (LUM) y listas únicas circulares (LUC). Para el primer caso, presentamos una condición necesaria y suficiente para determinar si una instancia tiene matching estable y definimos un nuevo tipo de estabilidad, a la que llamamos estabilidad débil, que tiene solución en todos los casos. Brindamos un algoritmo polinomial que encuentra siempre un matching estable débil. Acerca del problema en instancias de LUC, cuya complejidad es un tema abierto, conjeturamos que el mismo es NP-Completo y brindamos tanto la formulación entera que lo resuelve en forma exacta, como así también un algoritmo polinomial no completo que, para algunas instancias, puede encontrar solución en tiempos razonables para valores de n grandes.

En el transcurso de este trabajo demostramos que, tanto las instancias con listas únicas (LU) como las instancias con listas simples (LS) cumplen con la propiedad de consistencia, mientras que para el caso de instancias con listas por pares (LP), analizamos separadamente el subconjunto de instancias (LPC) que cumplen esta propiedad. Con respecto a estas últimas, encontramos que existe una relación que permite definir un mapeo entre las instancias de listas simples y las instancias de listas por pares consistentes, en donde una instancia de LS se corresponde o está embebida en una o varias instancias de LPC. Esto permite afirmar que si un matching es estable en una instancia de LPC es estable en la instancia correspondiente de LS.

Vimos también que los matching canónicos son estables cuando trabajamos con instancias LS y LP. Para el caso particular de instancias LU mostramos que los matchings canónicos tienen estabilidad débil.

Dentro del conjunto de instancias de listas por pares, además de estudiar las listas consistentes, trabajamos con las listas con prioridad circular (LPPC), para las cuales conjeturamos que el problema es NP-Completo. Para este tipo de instancias, encontramos una relación con las listas únicas circulares (LUC), que permite definir un mapeo de instancias de LPPC a instancias de LUC. Esto fue la base para definir una propiedad que permite encontrar un matching estable en LPPC a partir de la solución en su instancia abarcativa LUC. También mostramos que si un matching es estable en una instancia LPPC, entonces es estable en la instancia abarcativa correspondiente.

Para resolver el problema en instancias con listas simples, listas únicas mutuas, listas por pares con prioridades mutuas y la mayoría de los casos con listas combinadas presentamos algoritmos exactos polinomiales, y demostramos que para estas instancias siempre existe solución.

Con respecto al estudio de la estructura de todas las soluciones de una instancia, en los distintos capítulos abordamos el problema mediante Programación Lineal Entera lo que permite definir el conjunto de todos los matchings estables para cada una de las clases de instancias. Esto permite no sólo maximizar o minimizar una determinada función sobre el conjunto de soluciones, sino también permitiría encontrar todas las soluciones.

En el estudio de la existencia de un orden entre las soluciones, demostramos que el conjunto de todos los matchings estables para listas simples y listas por pares conforma un orden que permite decir cuando un matching domina o es mejor que otro con respecto a uno de los conjuntos. También mostramos que esta relación de orden no forma un reticulado distributivo, a diferencia de lo que ocurre con la relación de dominación en el conjunto de todos los matchings estables bidimensionales, y los grafos dirigidos que representan esta relación pueden ser no conexos. En cambio, para el caso de listas únicas, vimos que la relación de dominación definida entre los matchings estables no es un orden. No obstante, para este último tipo de listas definimos un orden entre las clases de equivalencia que conforman una partición del conjunto de todos los matchings estables. En estas instancias, el conjunto de matchings estables tampoco forma un reticulado distributivo bajo esta nueva relación de orden, excepto en el caso del conjunto de soluciones fuertemente estables para instancias de LUM.

Dentro del conjunto de instancias de LC estudiamos todas las combinaciones entre los diferentes tipos de listas de preferencias, es decir Listas Únicas con Listas Simples(LU+LS), Listas Únicas con Listas por Pares(LU+LP), Listas Simples con Listas por Pares (LS+LP) y las que incluyen los tres diferentes tipos de listas(LU+LS+LP). Para el primer caso dimos algoritmos polinomiales de complejidad $O(n^2)$ que siempre encuentran un matching estable. En instancias combinadas de LU+LP, trabajamos sobre las listas por pares con prioridad y clasificamos las mismas en ocho categorías, para seis de las cuales encontramos algoritmos polinomiales correctos y completos. En los dos casos restantes, dimos propiedades que establecen condiciones suficientes para encontrar un matching estable relacionándolas con instancias LUC y matchings bidimensionales. En las instancias de LS+LP también analizamos los casos de listas por pares con prioridad, donde dimos algoritmos de orden n^2 que siempre encuentran un matching estable. Finalmente, en aquellas instancias que combinan los tres diferentes tipos de listas, considerando las listas por pares con prioridad encontramos algoritmos polinomiales, correctos y completos, que encuentran un matching estable, así como también para uno de los casos de listas por pares sin considerar prioridad.

Como vimos en el Capítulo 1, existen algunos casos de aplicación que pueden ser modelados como el problema de hallar un matching estable tridimensional sobre tres conjuntos, como por ejemplo armar grupos de trabajo o comisiones de tres personas pertenecientes a áreas distintas.

Problemas Abiertos

En esta sección mencionamos algunas variantes y posibles enfoques del problema que fue objeto de estudio en la presente tesis, así como los problemas abiertos que consideramos de mayor interés para trabajos a futuro.

En esta tesis trabajamos con agentes de tres conjuntos disjuntos. Se puede extender este análisis a instancias en las cuales todos los agentes pertenezcan a un mismo conjunto. También se puede analizar una variante de ambos problemas permitiendo que las listas de preferencias sean incompletas. Con esto un agente estaría expresando que prefiere quedar sólo, antes que con un par o agente que no contemple en sus preferencias. Otra de las variantes que se puede estudiar es la de permitir que el orden en la lista de preferencias acepte la igualdad o indiferencia entre dos pares o agentes.

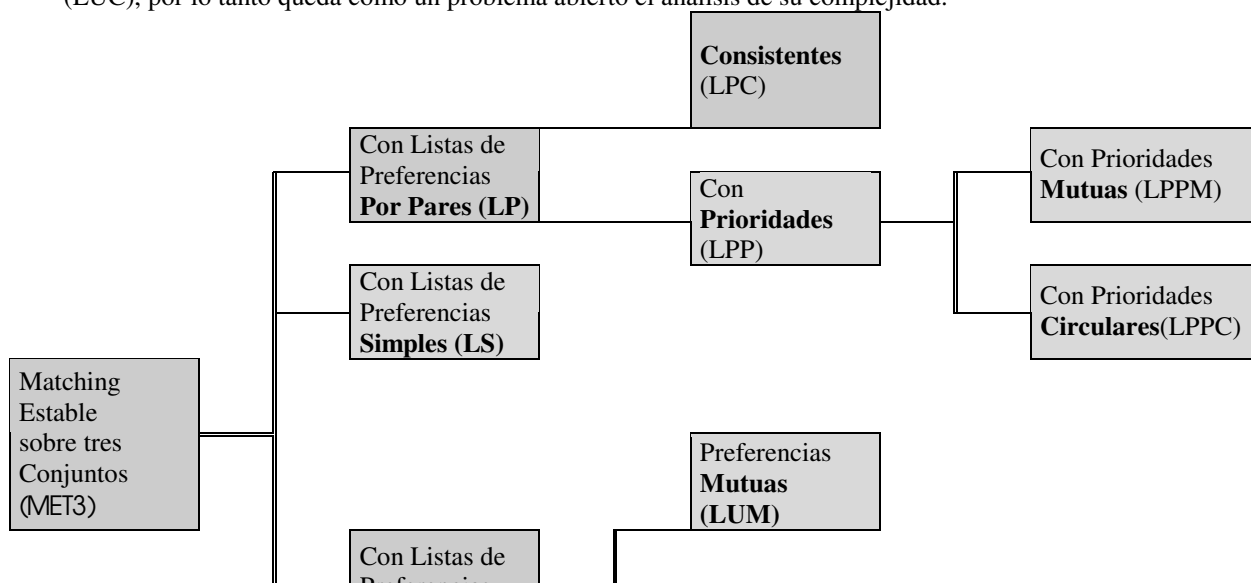
Puede resultar de interés trabajar el problema que desarrollamos en esta tesis desde el punto de vista de la teoría de hipergrafos, donde la instancia se puede modelar como un trigrafo tripartito completo. Es un trigrafo porque cada arista contiene tres vértices, es tripartito pues el conjunto de nodos puede verse como una partición en tres subconjuntos y es completo porque cada arista tiene un vértice en cada conjunto que compone la partición.

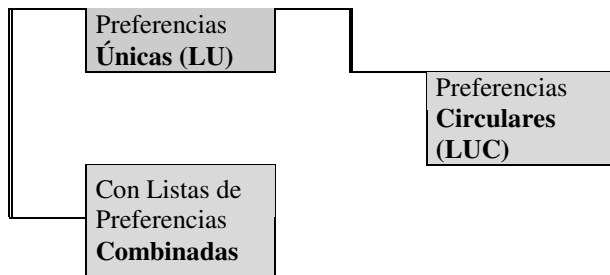
La generalización del problema de los matrimonios estables a matching estable sobre tres conjuntos abre las puertas al estudio de los matchings estables k -dimensionales sobre k conjuntos disjuntos, donde el problema sería hallar un matching estable sobre más conjuntos de agentes. Todas las variantes mencionadas para MET3 podrían aplicarse a este nuevo problema, como ser listas incompletas, planteo desde la teoría de hipergrafos, etc.

También se puede profundizar el análisis del problema visto desde la Programación Lineal Entera, sobre todo para aquellas instancias donde la complejidad de hallar una solución es un problema NP-Completo. Proponemos el análisis del poliedro definido con las formulaciones lineales enteras dadas en cada capítulo. También proponemos la realización de estudios de maximización y/o minimización de funciones sobre el conjunto de todos los matchings estables en función de las restricciones dadas.


Otro problema abierto es la cuantificación o el cálculo de cotas para determinar la cantidad de matchings estables en instancias de MET3.

En nuestro trabajo conjeturamos la NP-Complejidad del problema de hallar un matching estable en instancias con listas de preferencias por pares con prioridad circular (LPPC) y de listas únicas circulares (LUC), por lo tanto queda como un problema abierto el análisis de su complejidad.





Notación:

 Denota relación de clasificación

 Denota relación de inclusión

Figura 8.1 Clasificación de MET3.

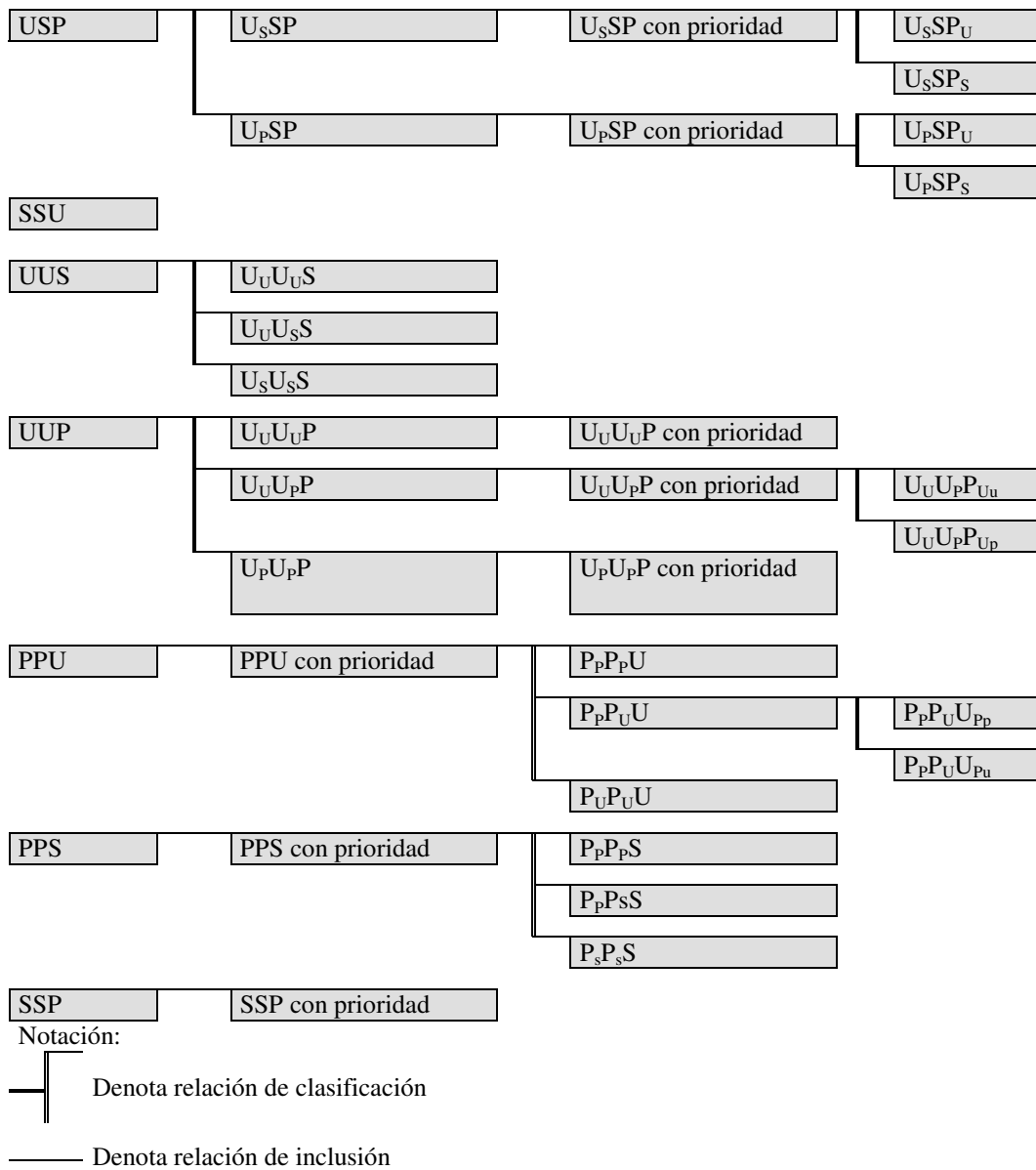


Figura 8.2 Subclasificación de instancias con listas de preferencias combinadas.

Bibliografía

- [Abe/93] Hernán Abeledo, The stable Matching Problems, Tesis de doctorado, New Brunswick Rutgers, The state University of New Jersey, mayo de 1993.
- [AC/95] Brian Aldershof, Olivia Carducci, Stable Matching with couples, Discrete Applied Mathematics, 68 1-2 pp. 203-207, 1996.
- [Alk/87] Ahmet Alkan, Nonexistence of stable threesome matchings, Mathematical Social Sciences 16 207-209, 1987.
- [AR/94] Hernán Abeledo, Uriel Rothblum, Paths to marriage stability, Discrete Applied Mathematics, 63 pp. 1-12, 1995.
- [Bla/88] Charles Blair, The lattice structure of the set of stable matchings with multiple partners, Mathematics of operations research, Vol 13, No. 4, noviembre de 1988.
- [BR/97] Michel Balinski, Guillaume Ratier, Of stable marriage and graphs, and strategy and politypes, SIAM Rev. Vol 39, No 4, pp 575-604, diciembre de 1997.
- [Chu/98] Kim-Sau Chung, On the existence of stable Roommate Matchings, Technical Report enviado a Games and Economic Behavior, octubre de 1998.
- [Dia/00] Vania Maria Félix Dias, O Problema dos casamentos estáveis com restricoes de pares, Tese do grau de mestre em Ciencias En engenharia de sistemas e computacao, Rio de Janeiro Brasil, febrero de 2000.
- [Duc/95] Pierre Duchet, Handbook of Combinatorics - Chapter 7 Hipergraphs, Edited by Rr. Graham, M. Grotschel y L. Lovász - Elsevier Science B.V., 1995.
- [GI/89] Dan Gusfield and Robert W. Irving, The Stable Marriage Problem: Structure and Algorithms, The MIT Press Cambridge, Massachusetts, London, England, 1989.
- [GS/62] D.Gale and L.S. Shapley , College admissions and the stability of marriage , American Mathematical Monthly, 69, pp9-15. , 1962.
- [GS/85] David Gale , Marilda Sotomayor, Ms.Machiavelli and the stable matching problem, American Mathematical Monthly, 92, pp261-268, abril de 1985.
- [Gus/85] Dan Gusfield, Three Fast Algorithms for four Problems in stable Marriage, SIAM Journal on Computing, 16, pp111-128, febrero de 1987.
- [IL/1986] Robert Irving, Paul Leather, The Complexity of Counting Stable Marriages, SIAM Journal Comput. Vol 15, No 3 pp 655-667, agosto de 1986.
- [ILG/86] Robert Irving, Paul Leather, Dan Gusfield, An efficient algorithm for the optimal stable marriage , Journal of Algorithms, 6 pp577-595, julio de 1986.
- [Ito/81] Stephen Y. Itoga , A generalization of the Stable Marriage Problem, J. Opl Res. Soc. Vol 32 pp. 1069 to 1074, 1981.
- [Knu/76] Donald Knuth, Marriages Stables, Les Presses de l' Université de Montréal, 1976.

- [MR/91] Susan Mongell and Alvin Roth , Soroty Rush as a two-sided matching Mechanism, American Economic Review, vol 81 pp441-464, junio de 1991.
- [MRMO/00] Martinez, Ruth, Jordi Masso, Alejandro Neme, and Jorge Oviedo, Single Agents and the Set of Many-to-One Stable Matchings, Journal of Economic Theory 91, 91-105, 2000, <http://www.itam.mx/lames/papers/contrses/oviedo.pdf>
- [NH/90] Cheng Ng, Daniel Hirschberg, Lower Bounds for the stable marriage problem and its variants, SIAM Journal Comput. Vol 19, No 1 pp 71-77, febrero de 1990.
- [NH/95] Cheng Ng, Daniel Hirschberg, Three-dimensional stable Matching Problems, Department of Information and Computer Science, University of California, Irvine, 1995.
- [Ron/90] Eytan Ronn, NP-Complete Stable Matching Problems, Journal of Algorithms, 11(2) pp285-304, junio de 1997, <http://SunSITE.Informatik.RWTH-Aachen.DE/dblp/db/indices/./journals/jal/jal11.html>
- [Rot/84] Alvin Roth , The evolution of the Labor Market for Medical Interns and Residents: A Case Study in Game Theory , J. Political Economy 92, pp991-1016, 1984.
- [Rot/90] Alvin Roth , New Physicians: A natural experiment in market organization, Science, 250. 1524-1528, 1990, <http://www.pitt.edu/~alroth/science.html>
- [Rot/95] Alvin Roth , Report on the design and testing of an applicant proposing matching algorithm, and comparison with the existing NRMP algorithm, Consultant' s report, 1995, <http://www.pitt.edu/~alroth/phase1.html>
- [Rot/96] Alvin Roth , The NRMP as a Labor Market: understanding the current study of the match, Journal of the American Medical Association, 275 pp 1054-1056, abril de 1996.
- [Rot/97] Alvin Roth , The effects of the Change in The NRMP Matching Algorithm , Journal of the American Medical Association, septiembre de 1997, JAMA.
- [RS/92] Alvin Roth, Marilda Sotomayor, Two-Sided Matching: A Study in Game-Theoretic Modeling and Analisis, Cambridge University Press, julio de 1992.
- [She/96] Katerina Sherstyuk , Multisided Matching Games, Department of Economics, Melbourne University, diciembre de 1996.
- [Sot/98] Marilda Sotomayor, Three remarks on the many-to-many stable matching problem, Mathematical Social Sciences 38 55-70, 1999.
- [VW/71] D.G.McVitie, L.B.Wilson , The stable Marriage Problem, Communications of the ACM, 14, No 7, pp486-492, julio de 1971.
- [WV/70] D.G.McVittie and L.B.Wilson, Stable marriage assignment for unequal sets, BIT, 10, pp295-309, febrero de 1970.