Programación 3

Instituto de Computación
Facultad de Ingeniería, UdelaR

August 8, 2024

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

El Curso I

- El curso se basa en algunos capítulos del libro Algorithm Design - Kleinberg & Tardos y se asume que cada estudiante lo lee a medida que los diferentes temas se presentan durante el semestre.
- La dedicación semanal esperada de cada estudiante es de unas 15hs, que deberán ser dedicadas a leer y comprender la serie de secciones del libro detalladas en el cronograma semanal y resolver los ejercicios propuestos del práctico.

El Curso II

- **Teórico**: Se tratarán y jerarquizarán algunos de los conceptos teóricos centrales asociados al tema semanal de forma expositiva e interactiva. **El teórico no reemplaza la lectura del libro, sino que la guía y profundiza en sus explicaciones de ser necesario.**
- **Monitoreo**: Habrá clases semanales basadas en los ejercicios del práctico, en las que el docente asignado podrá guiar y hacer un seguimiento del avance de cada estudiante. Cada estudiante que haya elegido la modalidad de evaluación continua tendrá un horario asignado. Algunas de las clases serán dedicadas a evaluar las tareas. Los integrantes de cada grupo deben estar asignados al mismo horario.

El Curso III

- **Talleres**: Espacio de trabajo guiado por docentes para la resolución de ejercicios del práctico y consultas en general.

Evaluaciones I

- **Básicas**: dos parciales (presenciales), el primero de 40 puntos y el segundo de 60 puntos.
- **Controles**: tres tareas de cuatro puntos cada una, que tienen como objetivo que cada estudiante lleve el curso al día. Las tareas se realizan en grupos de alrededor de tres estudiantes cada uno, y consisten en entregar la resolución de un conjunto de problemas, y defender la entrega realizada en una clase posterior. Esas clases serán en las semanas del 16/9-20/9, 21/10-25/10 y 18/11-22/11. Si bien la entrega la hace el grupo, los puntos se obtienen de manera individual dependiendo del rendimiento en la defensa.

Modalidades I

- **Básica**. Se aprueba el curso reuniendo al menos 25 puntos entre los dos parciales, y se exonera el curso reuniendo al menos 60 puntos entre los dos parciales.
- **Continua**. Se aprueba el curso reuniendo al menos 25 puntos entre controles y parciales, y un mínimo de 4 puntos en los controles. Se exonera el curso reuniendo al menos 60 puntos entre controles y parciales, y un mínimo de 6 puntos en los controles.

Por ejemplo, obteniendo dos puntos en cada tarea (6 en total), y 54 puntos entre los dos parciales, se exonera el curso.

Emparejamiento estable

- Dos conjuntos de personas, del mismo tamaño, $W = \{w_1 \dots w_n\}$, $M = \{m_1 \dots m_n\}$.
- Cada $w \in W$ tiene una lista de preferencia sobre M.
- Cada $m \in M$ tiene una lista de preferencia sobre W.
- *Emparejamiento*: subconjunto de $M \times W$ donde nadie aparece repetido.
- *Emparejamiento perfecto*: además nadie queda solo.
- *Emparejamiento estable*: además no existe ninguna inestabilidad, es decir, un par (m, w) que se prefieren mutuamente a sus respectivas parejas.

Algoritmo de Gale–Shapley

1 Marcar a todos en $M \cup W$ como libres
2 **while** *Existe* $m \in M$ *libre que no se ha propuesto a toda* $w \in W$ **do**
3 | Sea w la persona de mayor preferencia para m a la que aún no se propuso m
4 | **if** *w está libre* **then**
5 | | emparejar m con w
6 | **else if** *w prefiere m a su actual pareja* m' **then**
7 | | dejar m' libre y emparejar m con w
8 | **else**
9 | | w rechaza m
10 | **end**
11 **end**

Algoritmo de Gale–Shapley

- ¿Termina?
- Si termina, ¿produce un emparejamiento?
- ¿perfecto?
- ¿estable?
- ¿Todo esto es independiente del orden en que se escoja m libre?
- ¿Qué pasa si no existe una solución? ¿siempre existe una?

Objetivos del curso

1. Conocer algoritmos clásicos que constituyen una base para la resolución de problemas en computación y aplicarlos para la resolución de problemas concretos.
2. Dominar técnicas generales de diseño de algoritmos y aplicarlas.
3. Razonar con rigurosidad sobre cualidades como la corrección y complejidad de algoritmos.
4. Analizar rigurosamente problemas algorítmicos para identificar limitaciones de cómputo inherentes a cada problema, reconociendo diferentes clases de complejidad.

Cómo pensar y escribir una demostración

- ¡Escribirlas! No alcanza con “saber por dónde sale”.
- Pedir a alguien que las lea, por ejemplo los compañeros de grupo de monitoreo.
- Tener en cuenta el destinatario y el contexto donde se escribe; no es lo mismo un libro introductorio, que un boceto de solución hecho en clase, que una solución hecha en una evaluación.
- Lecturas sobre este tema:
  - “Fundamentos de Algoritmia”, G. Brassard, P. Bratley.
  - “Mathematics for Computer Science”, Eric Lehman, F. Thomson Leighton, Albert R. Meyer.
- Consideraciones similares aplican a la escritura de algoritmos.

Cómo escribir una demostración

- Anunciar el plan para la demostración (por inducción, por absurdo, etc.).
- Mantener un flujo ordenado de razonamiento, no un salpicón de ideas desordenadas (conviene empezar con un borrador y luego pasar en limpio).
- Escribir frases completas, no cadenas de ecuaciones libradas a interpretación por sí solas.
- No abusar de símbolos matemáticos, $\forall x \exists y : \forall z \ldots$. En general un texto puede ser mucho más claro y no menos riguroso.

Cómo escribir una demostración

- Definir toda la notación que se usa, el significado de las variables, etc. Ej.: Sea n la cantidad de ....
- Cuando nos referimos a un algoritmo, es conveniente numerar o etiquetar los pasos a los que nos vamos a referir en la demostración. Ej.: Si la condición del ciclo en el paso X no se verifica, entonces debemos tener $n \leq m$ y por lo tanto ...
- Dividir demostraciones complejas en resultados auxiliares.
- Terminar diciendo claramente cómo se concluye la tesis a partir de lo que hemos desarrollado. Ej. .... este hecho contradice la hipótesis X, por lo cual concluimos Y.

Cómo pensar una demostración

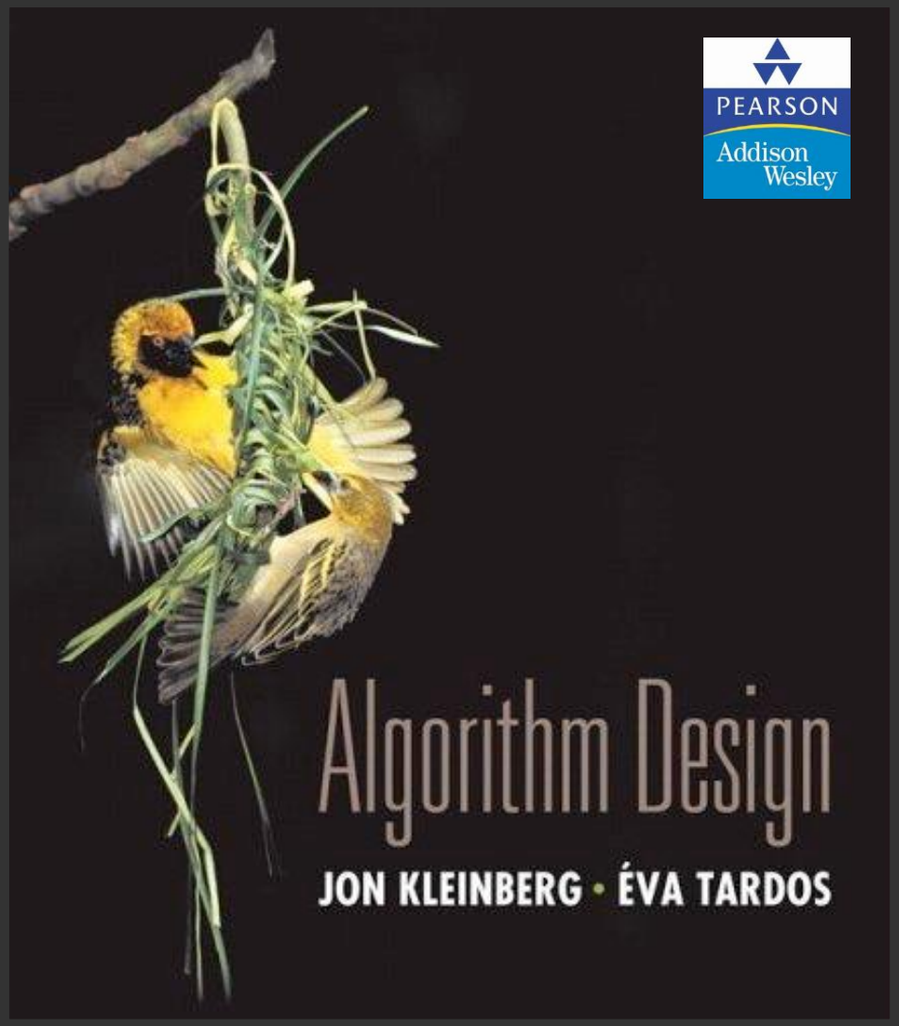
- Desconfiar de las obviedades.
- Implicación ($A \implies B$): Se puede asumir A y obtener B a partir de una secuencia de pasos lógicos, o probar el contrarecíproco ($\neg B \implies \neg A$).
- Equivalencias (Si y solo si, sii, iff, $A \iff B$): Se puede probar ambas implicaciones o construir una cadena de equivalencias. Nuestras pruebas de corrección a menudo son pruebas de equivalencia.
- Descomposición en casos. Ej.: Si $n = m$ .... Si $n \neq m$, uno de ellos es mayor que el otro; supongamos, sin pérdida de generalidad, que $n > m$. Entonces ...
- Por absurdo: Asumir que la proposición es falsa y arribar mediante un razonamiento lógico a una contradicción.
- Por inducción: Muy práctico para analizar algoritmos recursivos (pero no solo en esos casos).

Referencia disponible en línea

“Mathematics for Computer Science”,

Eric Lehman, F. Thomson Leighton, Albert R. Meyer.

Gracias.

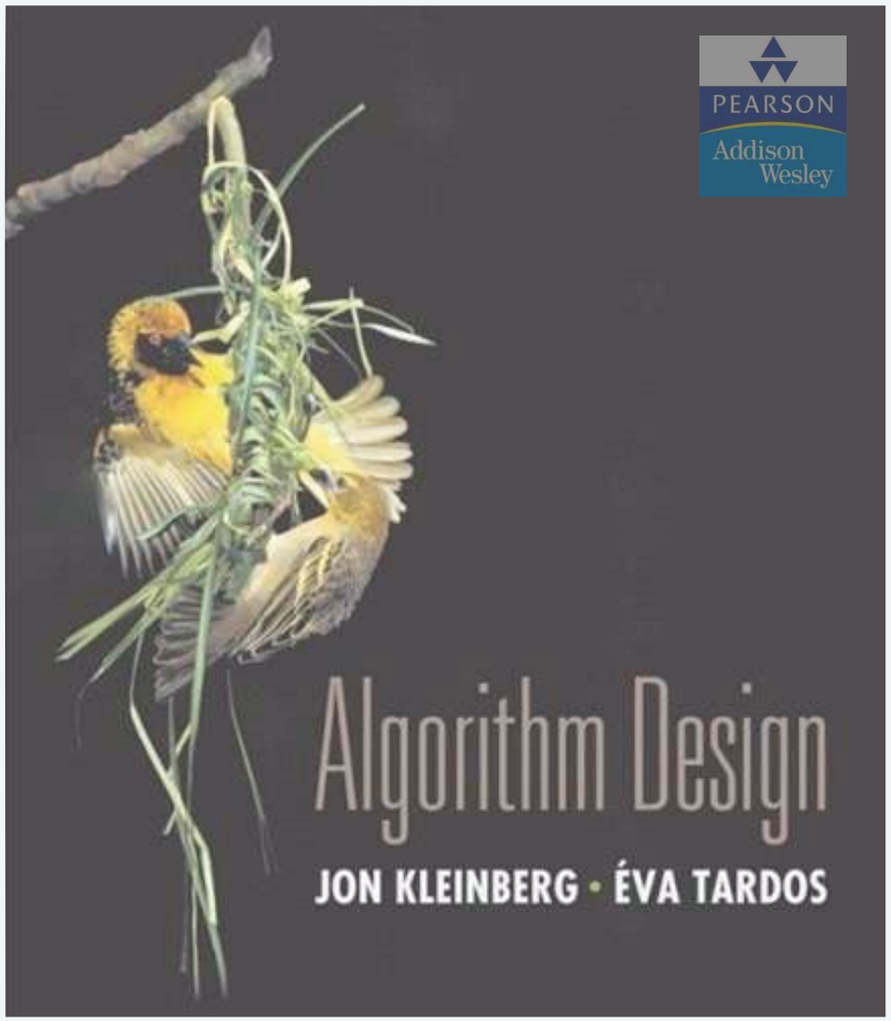


1. Problemas

REPRESENTATIVOS

- *emparejamiento estable*
- *cinco problemas representativos*

Last updated on 8/26/24 9:04 AM



1. PROBLEMAS

REPRESENTATIVOS

- *emparejamiento estable*
- *cinco problemas representativos*

Poner en contacto a estudiantes de medicina con hospitales

Goal. Given a set of preferences among hospitals and med-school students, design a self-reinforcing admissions process.

Unstable pair. Hospital h and student s form an unstable pair if both:

- h prefers s to one of its admitted students.
- s prefers h to assigned hospital.

Stable assignment. Assignment with no unstable pairs.

- Natural and desirable condition.
- Individual self-interest prevents any hospital–student side deal.

Poner en contacto a estudiantes de medicina con hospitales

Objetivo. Dado un conjunto de preferencias entre hospitales y estudiantes de medicina, diseñar un proceso de admisión que se refuerce a sí mismo.

Par inestable. El hospital h y el alumno s forman un par inestable si:

- h prefiere s en lugar de uno de sus estudiantes admitidos, **y además**
- s prefiere h en lugar del hospital asignado.

Asignación estable. Asignación sin pares inestables.

- Estado natural y deseable.
- El interés personal impide cualquier acuerdo entre el hospital y los estudiantes.

Problema de emparejamiento estable: entrada

Entrada. Un conjunto de n hospitales H y un conjunto de n estudiantes S.

- Cada hospital $h \in H$ clasifica a los estudiantes.
- Cada alumno $s \in S$ clasifica los hospitales.

un estudiante por hospital (por ahora)

favorito ↓ menos favorito ↓

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

listas de preferencias de los hospitales

favorito ↓ menos favorito ↓

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

listas de preferencias de los estudiantes

Emparejamiento perfecto

Def. Un emparejamiento M es un conjunto de pares ordenados h-s con $h \in H$ y $s \in S$ s.t.

- Cada hospital $h \in H$ aparece como máximo en un par de M.
- Cada alumno $s \in S$ aparece como máximo en un par de M.

Def. Un emparejamiento M es perfecto si $|M| = |H| = |S| = n$.

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

un emparejamiento perfecto M = { A-Z, B-Y, X-X }

Par inestable

Def. Dado un emparejamiento perfecto M, el hospital h y el estudiante s forman una
par inestable si se da que

- h prefiere s en lugar del estudiante emparejado, **y**
- s prefiere h en lugar del hospital emparejado.

Punto clave. Una pareja inestable h-s podría mejorar mediante una acción conjunta de sus miembros.

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

A-Y es un par inestable

Problema de emparejamiento estable

Def. Un emparejamiento estable es un emparejamiento perfecto sin pares inestables.

Problema de emparejamiento estable. Dadas las listas de preferencias de n hospitales y n estudiantes, encontrar un emparejamiento estable (si existe).

- Condición natural, deseable y que se refuerza a sí misma.
- El interés individual impide a cualquier pareja hospital-alumno romper su compromiso.

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

un emparejamiento estable M = { A-X, B-Y, C-Z }

Problema de compañero de piso estable

Q. ¿Existen siempre emparejamientos estables?

A. A priori no es obvio.

Problema de compañero de piso estable.

- 2 n personas; cada persona clasifica a las demás de 1 a 2 n - 1.
- Asignar parejas de compañeros de habitación de modo que no haya parejas inestables.

	1st	2nd	3rd
A	B	C	D
B	C	A	D
C	A	B	D
D	A	B	C

ninguna coincidencia perfecta es estable

A-B, C-D $\Rightarrow$ B-C inestable

A-C, B-D $\Rightarrow$ A-B inestable

A-D, B-C $\Rightarrow$ A-C inestable

Observación. No es necesario que existan coincidencias estables.

Algoritmo de aceptación diferida Gale-Shapley

Un método intuitivo que garantiza encontrar un emparejamiento estable.

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

  $s \leftarrow$ first student on h's list to whom h has not yet proposed.

  IF (s is unmatched)

    Add h–s to matching M.

  ELSE IF (s prefers h to current partner h')

    Replace h'–s with h–s in matching M.

  ELSE

    s rejects h.

RETURN stable matching M.

Prueba de corrección: terminación

Observación 1. Los hospitales proponen a los estudiantes por orden decreciente de preferencia.

Observación 2. Una vez que un alumno es emparejado, nunca deja de estarlo; sólo "sube de nivel".

Afirmación. El algoritmo termina después de un máximo de n^2 iteraciones del bucle *while*.

Pf. Cada vez a través del bucle *while* un hospital propone a un nuevo alumno. Sólo hay n^2 propuestas posibles. ▪

	1st	2nd	3rd	4th	5th
Atlanta	A	B	C	D	E
Boston	B	C	D	A	E
Chicago	C	D	A	B	E
Dallas	D	A	B	C	E
Eugene	A	B	C	D	E

	1st	2nd	3rd	4th	5th
Val	W	X	Y	Z	V
Wayne	X	Y	Z	V	W
Xavier	Y	Z	V	W	X
Yolanda	Z	V	W	X	Y
Zeus	V	W	X	Y	Z

n(n-1) + 1 propuestas necesarias

Prueba de corrección: perfección

Afirmación. Gale-Shapley produce un emparejamiento.

Pf. Hospital propone sólo si no está emparejado; estudiante

Afirmación. En el emparejamiento Gale-Shapley, todos los hospitales son emparejados.

Pf. [por contradicción]

- Supongamos, en aras de la contradicción, que algún hospital $h \in H$ es no emparejado a la terminación del algoritmo de Gale-Shapley.
- Entonces algún alumno, digamos $s \in S$, no es emparejado al terminar.
- Por Observación 2, s nunca se propuso.
- Pero, h propone a todos los estudiantes, ya que h termina sin emparejar.

Afirmación. En el emparejamiento Gale-Shapley, todos los estudiantes son emparejados.

Pf.

- Según la afirmación anterior, se emparejan todos los n hospitales.
- Así, todos los n estudiantes quedan emparejados. ▪

Prueba de corrección: estabilidad

Afirmación. En el emparejamiento M* *de* Gale-Shapley, no hay pares inestables.

Pf. Supongamos, que la asignación producida por el algoritmo no es estable.

Esto significaría que existe una pareja (h, s) tal que:

- h está emparejado con s', y
- s está emparejado con h', pero
- h y s prefieren estar emparejados entre sí.

Dado que h prefiere a s sobre s', h habrá propuesto a s antes que a s'.

Hay dos posibles escenarios:

- **h nunca le propuso a s**: entonces **h** le propuso primero a **s'.**
- s **lo rechazó por** h'' : Cómo s está emparejado con h', o bien $h'' = h'$, o bien s prefiere a h' sobre h'' y por lo tanto a h' sobre h.

En ambos casos, llegamos a una contradicción.

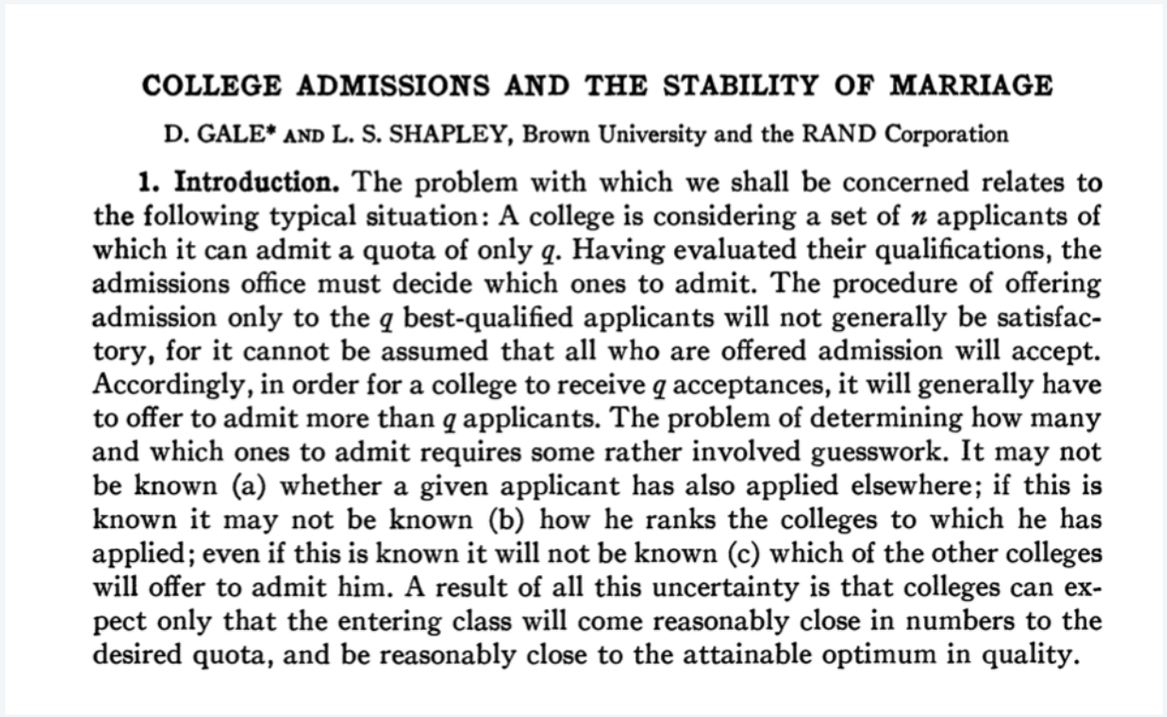
Resumen

Problema de emparejamiento estable. Dados n hospitales y n estudiantes, y sus preferencias, encontrar un emparejamiento estable si existe.

Teorema. [Gale-Shapley 1962] El algoritmo de Gale-Shapley garantiza encontrar un emparejamiento estable para cualquier instancia del problema.

Q. ¿Cómo implementar el algoritmo Gale-Shapley de forma eficiente?

Q. Si hay varias coincidencias estables, ¿cuál encuentra Gale-Shapley?



COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only q. Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the q best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept. Accordingly, in order for a college to receive q acceptances, it will generally have to offer to admit more than q applicants. The problem of determining how many and which ones to admit requires some rather involved guesswork. It may not be known (a) whether a given applicant has also applied elsewhere; if this is known it may not be known (b) how he ranks the colleges to which he has applied; even if this is known it will not be known (c) which of the other colleges will offer to admit him. A result of all this uncertainty is that colleges can expect only that the entering class will come reasonably close in numbers to the desired quota, and be reasonably close to the attainable optimum in quality.

Aplicación eficaz

Implementación eficiente. Una implementación en tiempo $O(n^2)$.

Representación de hospitales y estudiantes. hospitales y estudiantes indexados $1, \ldots, n$.

Representación del emparejamiento.

- Mantener una lista de hospitales libres (en una pila o cola).
- Mantener dos arrays $estudiante[h]$ y $hospital[s]$.
  - si h emparejado con s, entonces $alumno[h] = s$ y $hospital[s] = h$
  - utilice el valor 0 para indicar que el hospital o el estudiante no están emparejados

Hospitales proponen.

- Para cada hospital, mantenga una lista de estudiantes, ordenados por preferencia.
- Para cada hospital, mantener un puntero a los estudiantes en la lista para la próxima propuesta.

Aplicación eficiente (continuación)

Estudiantes que rechazan/aceptan.

- ¿Prefiere el alumno s el hospital h al hospital h'?
- Para cada estudiante, crear la lista inversa de preferencias de hospitales.
- Acceso en tiempo constante para cada consulta tras un preprocesamiento $O(n)$.

pref[]

1st	2nd	3rd	4th	5th	6th	7th	8th
8	3	7	1	4	5	6	2

el estudiante prefiere el hospital 3 al 6

ya que inverso[3] < inverso[6]

inversa[]

1	2	3	4	5	6	7	8
4th	8th	2nd	5th	6th	7th	3rd	1st

```
for i = 1 to n
    inverso[pref[i]] = i
```

Comprender la solución

Para un problema determinado, puede haber varias coincidencias estables.

- ¿Todas las ejecuciones de Gale-Shapley producen el mismo emparejamiento estable?
- En caso afirmativo, ¿cuál?

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

una instancia con dos emparejamientos estables: S = { A-X, B-Y, C-Z } y S′ = { A-Y, B-X, C-Z }

Comprender la solución

Def. El alumno s es un par válido para el hospital h si existe cualquier emparejamiento estable en el que h y s estén emparejados.

Ex.

- Tanto Xavier como Yolanda son socios válidos para Atlanta.
- Tanto Xavier como Yolanda son socios válidos para Boston.
- Zeus es el único socio válido para Chicago.

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

una instancia con dos emparejamientos estables: S = { A-X, B-Y, C-Z } y S′ = { A-Y, B-X, C-Z }

Comprender la solución

Def. El alumno s es un par válido para el hospital h si existe cualquier emparejamiento estable en el que h y s estén emparejados.

Asignación óptima por hospital. Cada hospital recibe el mejor socio válido.

- ¿Es perfecto?
- ¿Es estable?

Afirmación. Todas las ejecuciones de Gale-Shapley producen una asignación hospitalaria óptima.

Corolario. La asignación óptima de hospitales es un emparejamiento estable.

Optimalidad hospitalaria

Afirmación. El emparejamiento S^* de Gale-Shapley es óptimo para hospitales.

Pf. [por contradicción] Supongamos que se empareja a un hospital con un estudiante que no es el mejor par válido.

- Los hospitales proponen en orden decreciente de preferencia
  ⇒ algún hospital es rechazado por un par válido durante Gale-Shapley.
- Sea h el primero al que le pasa, y sea s el primer estudiante válido de que rechaza h.
- Cuando s rechaza h en Gale-Shapley, s forma (o reafirma) compromiso con un hospital, digamos $\boldsymbol{h'}$.
  ⇒ s prefiere $\boldsymbol{h'}$ a h.
- Sea S algún emparejamiento estable en el que h y s están emparejados.
- Sea $\boldsymbol{s'}$ pareja de $\boldsymbol{h'}$ en S.
- $\boldsymbol{h'}$ no había sido rechazada por ningún interlocutor válido (incluido $\boldsymbol{s'}$) en el momento en que h es rechazada por s.
- Así pues, $\boldsymbol{h'}$ aún no se había declarado a $\boldsymbol{s'}$ cuando $\boldsymbol{h'}$ se declaró a s.
  ⇒ $\boldsymbol{h'}$ prefiere s a $\boldsymbol{s'}$.
- Por lo tanto, $\boldsymbol{h'}$-$\boldsymbol{s}$ es inestable en S, una contradicción. ▪

h - s

h' - s'

⋮

emparejamiento estable S

porque esta es la primera rechazo por un interlocutor válido

"Pesimalidad" estudiantil

Q. ¿La optimización hospitalaria va en detrimento de los estudiantes?

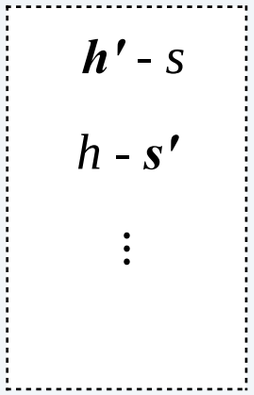
A. Sí.

Peor Asignación para los estudiantes. Cada estudiante recibe peor pareja válida.

Afirmación. Gale-Shapley encuentra el emparejamiento estable estudiante-pesimal S*.

Pf. [por contradicción] Supongamos que h-s existe en S* pero h no es la peor pareja válida para s.

- Existe un emparejamiento estable S en el que s está emparejado con un hospital, digamos $\boldsymbol{h'}$, al que s prefiere menos que h.
  ⇒ s prefiere h a $\boldsymbol{h'}$.
- Sea $\boldsymbol{s'}$ el compañero de h en S. Por *optimalidad hospitalaria*, s es el mejor compañero válido para h.
  ⇒ h prefiere s a $\boldsymbol{s'}$.
- Por tanto, h-s es un par inestable en S, una contradicción. ▪



emparejamiento estable M

Extensiones

Extensión 1. Algunos participantes declaran inaceptables a otros.

Ampliación 2. Algunos hospitales tienen más de un puesto.

Ampliación 3. Número desigual de puestos y alumnos.

estudiante de medicina poco dispuesto a trabajar en Cleveland

más de 43.000 estudiantes estudiantes; sólo 31.000 plazas

Def. El emparejamiento S es inestable si existe un hospital h y un alumno s tales que:

- h y s son aceptables entre sí; y
- O bien s no está emparejado, o bien s prefiere h al hospital asignado; y
- O bien h no tiene todas sus plazas ocupadas, o bien h prefiere que s al menos uno de sus alumnos asignados.

Contexto histórico

Programa nacional de emparejamiento de residentes (NRMP).

- Centro de intercambio de información para poner en contacto a estudiantes de medicina con hospitales.
- Comenzó en 1952 para solucionar el desbarajuste de las fechas de las ofertas.
- Originalmente utilizaba el algoritmo "Boston Pool".
- Algoritmo revisado en 1998.

hospitales empezaron a hacer ofertas cada vez más tempranas, hasta con 2 años de antelación

- estudiante de medicina óptimo ⟵ ya no está garantizada la coincidencia estable
- aborda diversas restricciones laterales (por ejemplo, permite que las parejas coincidan)

The Redesign of the Matching Market for American Physicians: Some Engineering Aspects of Economic Design

By Alvin E. Roth and Elliott Peranson*

We report on the design of the new clearinghouse adopted by the National Resident Matching Program, which annually fills approximately 20,000 jobs for new physicians. Because the market has complementarities between applicants and between positions, the theory of simple matching markets does not apply directly. However, computational experiments show the theory provides good approximations. Furthermore, the set of stable matchings, and the opportunities for strategic manipulation, are surprisingly small. A new kind of "core convergence" result explains this; that each applicant interviews only a small fraction of available positions is important. We also describe engineering aspects of the design process. (*JEL* C78, B41, J44)

THE MATCH NATIONAL RESIDENT MATCHING PROGRAM®

Premio Nobel de Economía 2012

Lloyd Shapley. Teoría de la correspondencia estable y algoritmo de Gale-Shapley.

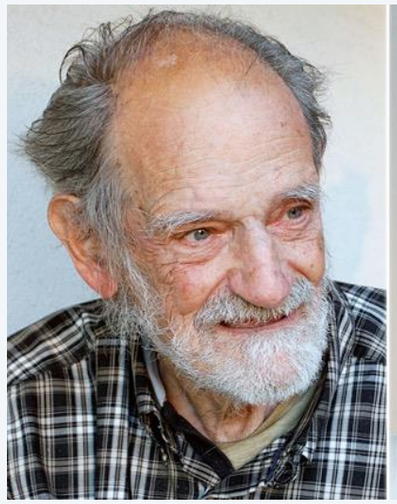
COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only q. Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the q best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept.

← solicitudes originales: college admissions and "matrimonio tradicional"

Alvin Roth. Aplicó Gale-Shapley para emparejar estudiantes de medicina con hospitales, estudiantes con escuelas y donantes de órganos con pacientes.



Lloyd Shapley

Alvin Roth

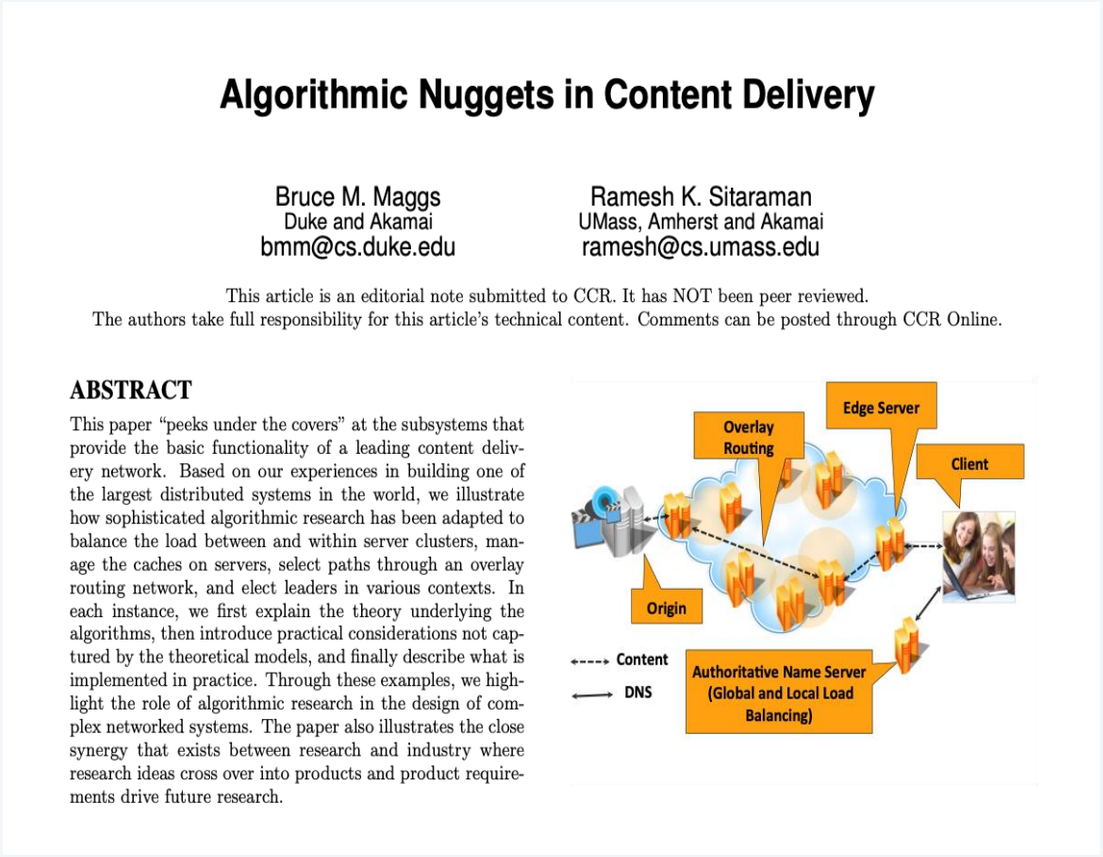
Una aplicación moderna

Redes de distribución de contenidos. Distribuyen gran parte del contenido mundial en la web.

Usuario. Prefiere un servidor web que proporcione un tiempo de respuesta rápido.

Servidor web. Prefiere servir a los usuarios con bajo coste.

Objetivo. Asignar miles de millones de usuarios a servidores, cada 10 segundos.



Algorithmic Nuggets in Content Delivery

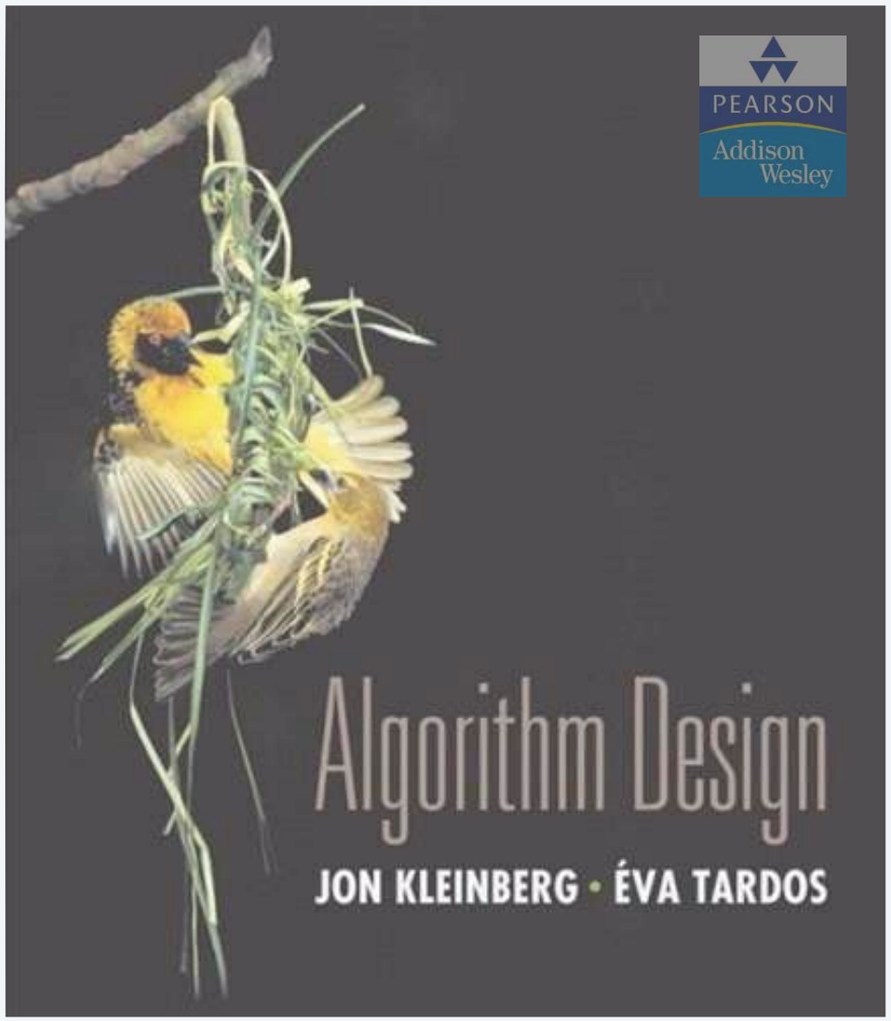
Bruce M. Maggs
Duke and Akamai
bmm@cs.duke.edu

Ramesh K. Sitaraman
UMass, Amherst and Akamai
ramesh@cs.umass.edu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

This paper "peeks under the covers" at the subsystems that provide the basic functionality of a leading content delivery network. Based on our experiences in building one of the largest distributed systems in the world, we illustrate how sophisticated algorithmic research has been adapted to balance the load between and within server clusters, manage the caches on servers, select paths through an overlay routing network, and elect leaders in various contexts. In each instance, we first explain the theory underlying the algorithms, then introduce practical considerations not captured by the theoretical models, and finally describe what is implemented in practice. Through these examples, we highlight the role of algorithmic research in the design of complex networked systems. The paper also illustrates the close synergy that exists between research and industry where research ideas cross over into products and product requirements drive future research.



1. PROBLEMAS REPRESENTATIVOS

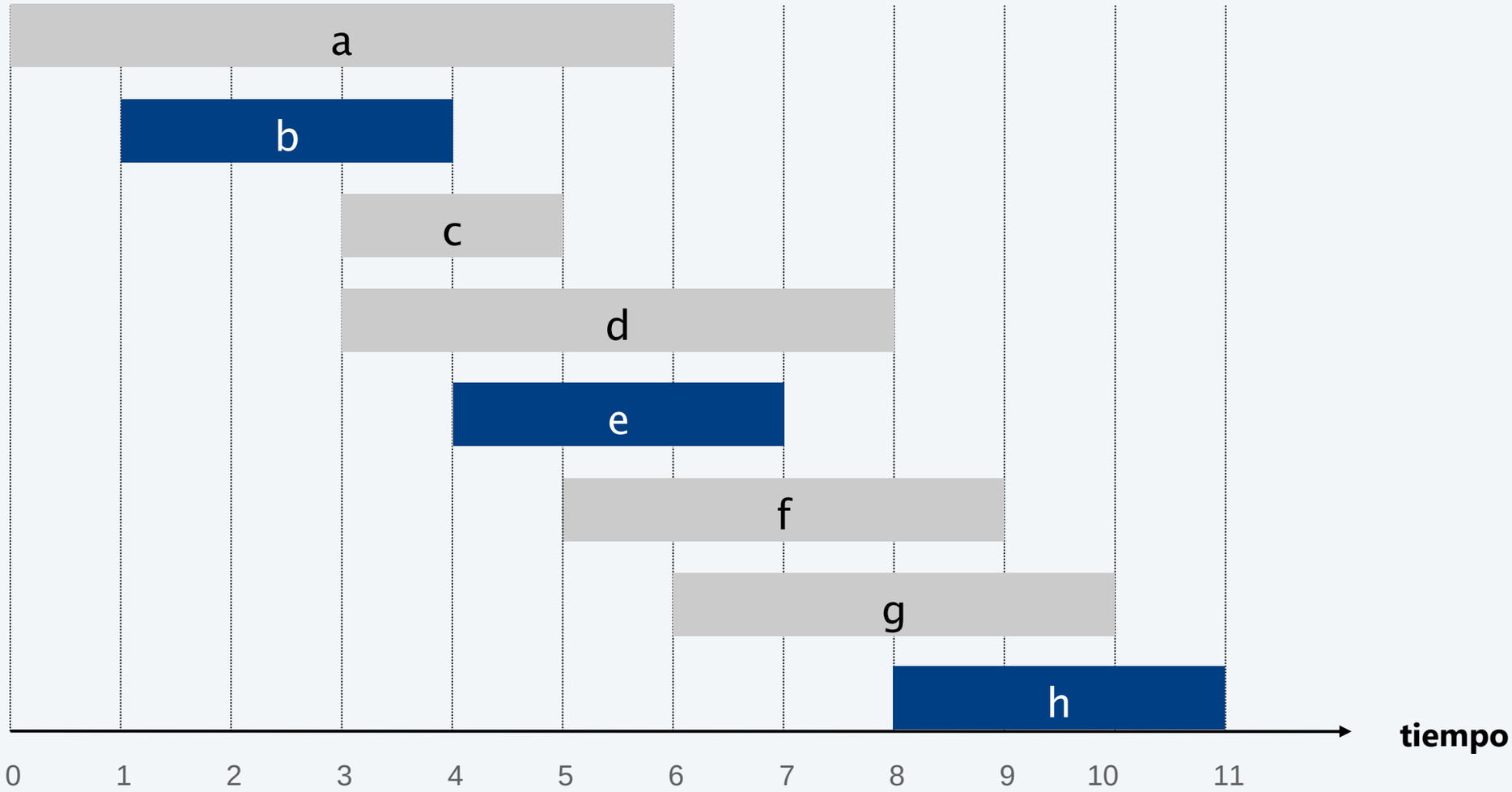
- *emparejamiento estable*
- *cinco problemas representativos*

Programación por intervalos

Entrada. Conjunto de trabajos con horas de inicio y finalización.

Objetivo. Encontrar el subconjunto de máxima cardinalidad de trabajos mutuamente compatibles.

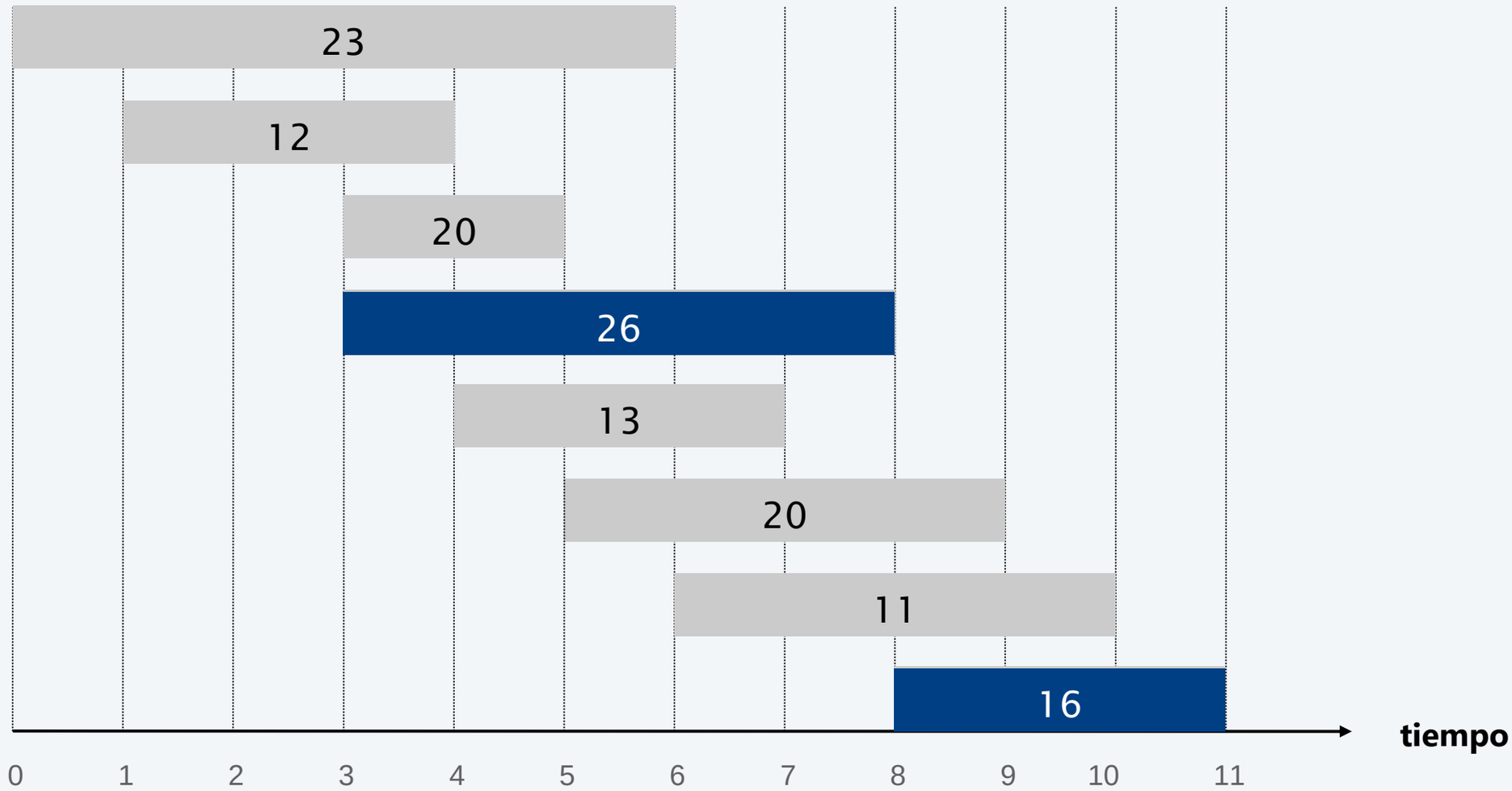
los trabajos no se solapan



Programación por intervalos ponderados

Entrada. Conjunto de trabajos con horas de inicio, horas de finalización y pesos.

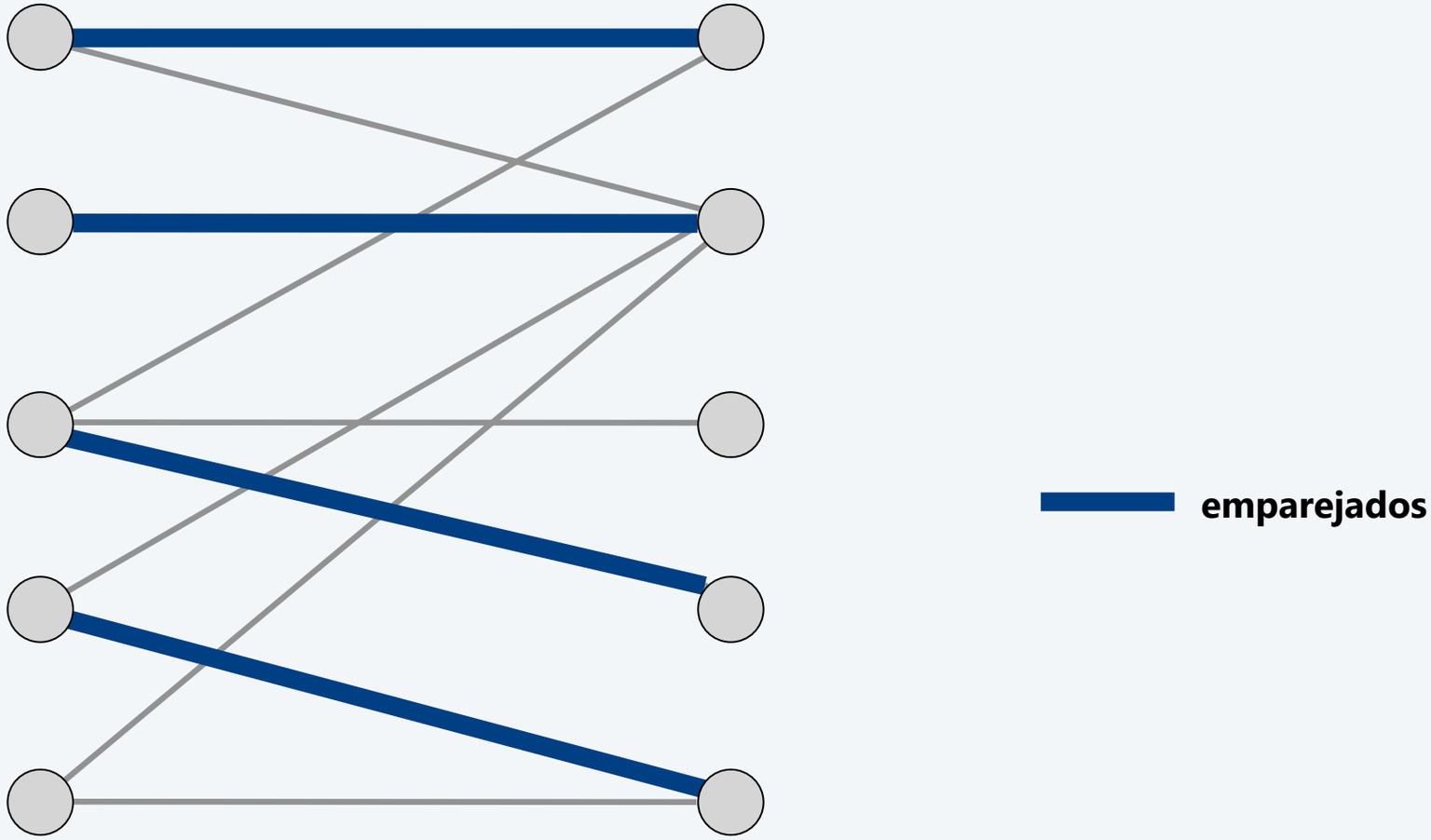
Objetivo. Encontrar el subconjunto de peso máximo de trabajos compatibles entre sí.



Emparejamiento bipartito

Problema. Dado un grafo bipartito $G = (L \cup R, E)$, encontrar un emparejamiento de cardinalidad máxima.

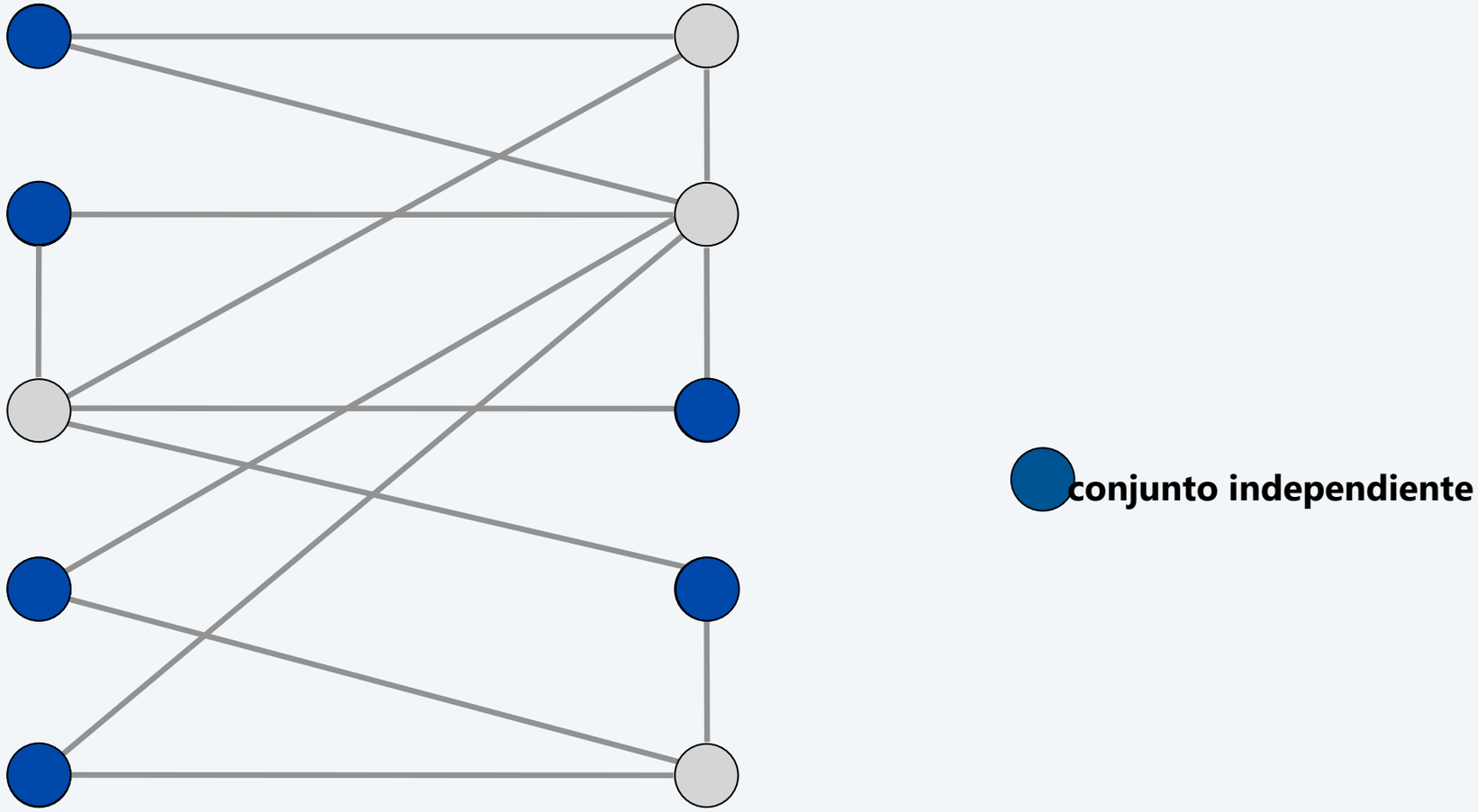
Def. Un subconjunto de aristas $M \subseteq E$ es un emparejamiento si cada nodo aparece en exactamente una arista de M.



Conjunto independiente

Problema. Dado un grafo $G = (V, E)$, encontrar un conjunto independiente de cardinalidad máxima.

Def. Un subconjunto $S \subseteq V$ es independiente si para cada $(u, v) \in E$, o bien $u \notin S$ o $v \notin S$ (o ambos).



Ubicación competitiva de las instalaciones

Entrada. Grafo con peso en cada nodo.

Juego. Dos jugadores se alternan en la selección de nodos.
No se permite seleccionar un nodo si alguno de sus vecinos ha sido seleccionado.

Objetivo. Seleccionar un subconjunto de nodos de peso máximo.

El segundo jugador puede garantizar 20, pero no 25.

Cinco problemas representativos

Variaciones sobre un tema: conjunto independiente.

Programación por intervalos: Algoritmo codicioso $O(n \log n)$.
Programación por intervalos ponderados: Algoritmo de programación dinámica $O(n \log n)$.
Emparejamiento bipartito: algoritmo basado en el flujo máximo $O(n^k)$.
Conjunto independiente: **NP-completo**.
Localización de instalaciones competitivas: **PSPACE-completo**.