

Asignación con flujo



Algoritmos y estructuras de datos III, 2do cuatrimestre 2023

Objetivos



Repaso

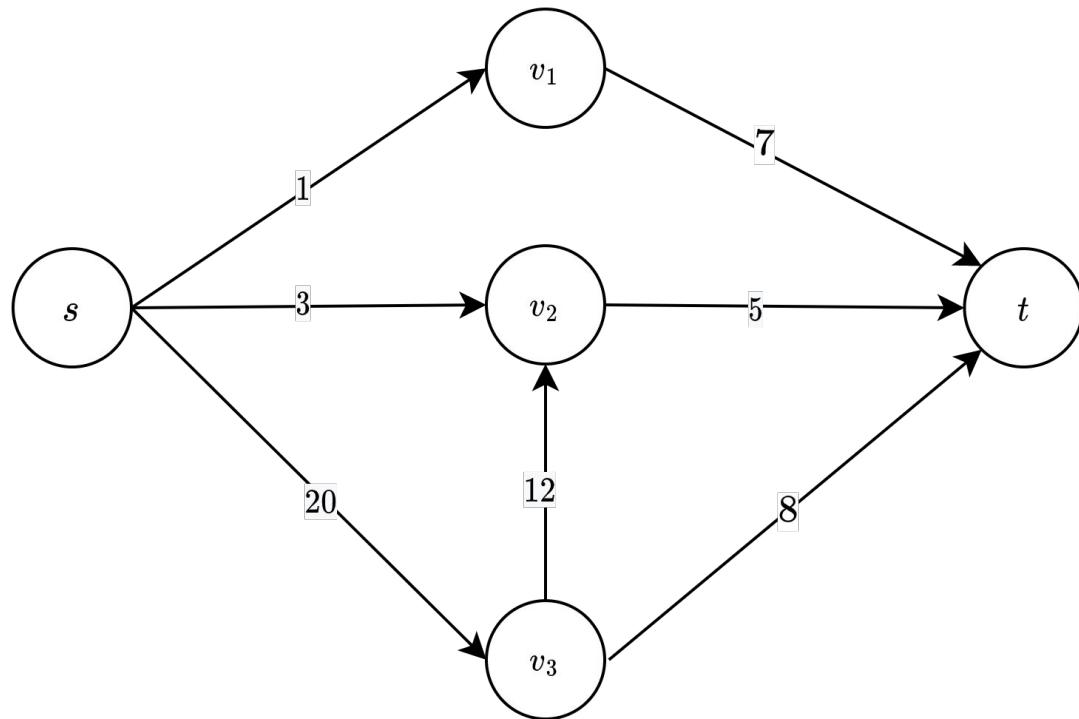
Conceptos básicos



Red

¿Qué características tiene una red de flujo con capacidades?

- Tenemos **s** el nodo distinguido donde “sale” el flujo.
- Tenemos **t** el nodo distinguido donde “entra” el flujo.
- Cada arista tiene una cierta capacidad.



Propiedades del flujo

- **Conservación:** Salvo por los nodos s (fuente) y t (sumidero), el flujo que entra por un nodo debe ser igual al que sale del mismo.
- **Restricción capacidad:** El flujo que pasa por una arista respeta que no es mayor a la capacidad de la misma.

El valor que tiene un flujo en una red es el que sale de s , o equivalentemente, que entra a t .

Algoritmos

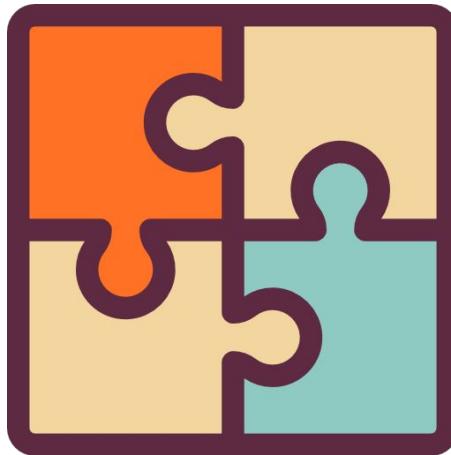
Considerando que para una red G tenemos:

- V = Cantidad de nodos.
- E = Cantidad de aristas.
- U = Capacidad máxima de una arista.
- F = Máximo flujo obtenible.

Tenemos las siguientes implementaciones del algoritmo de flujo máximo:

- **Ford & Fulkerson:** $O(|E| * F)$.
- **Edmonds Karp:** $O(|V| E^2)$.

¡Asignemos!



Food truck orders



[Link](#)

Enunciado

Enunciado

Miso tiene un food truck que produce una variedad de platos, cada uno con su cantidad. Quiere asignarle almuerzos a sus **C** clientes, donde:

- El cliente c_i tiene un conjunto de platos preferidos C_i .

Además Miso produce **P** platos distintos, donde:

- Hay $q \square$ platos del tipo $p \square$.

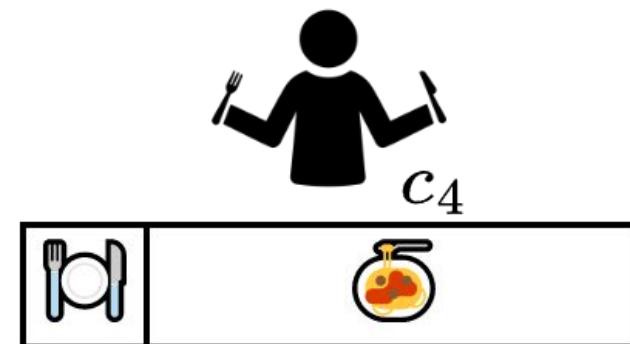
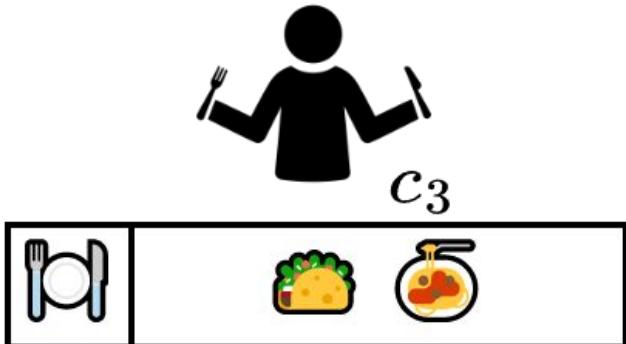
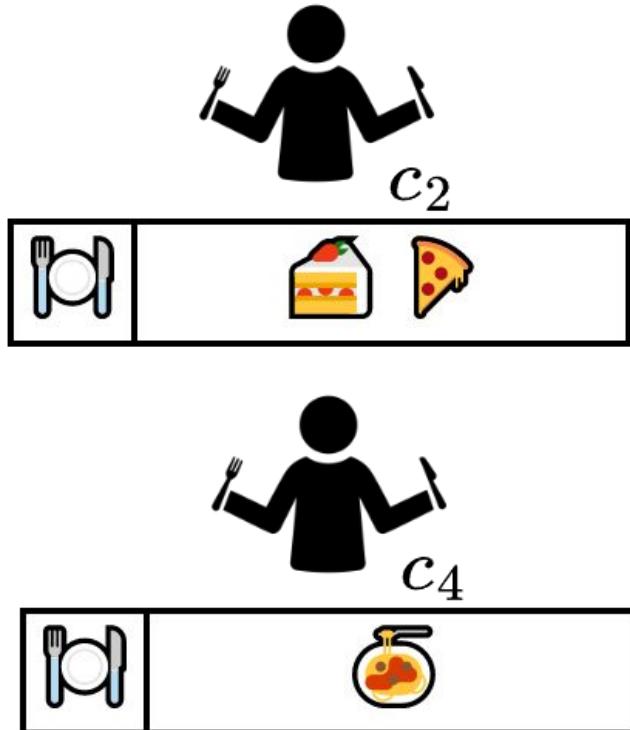
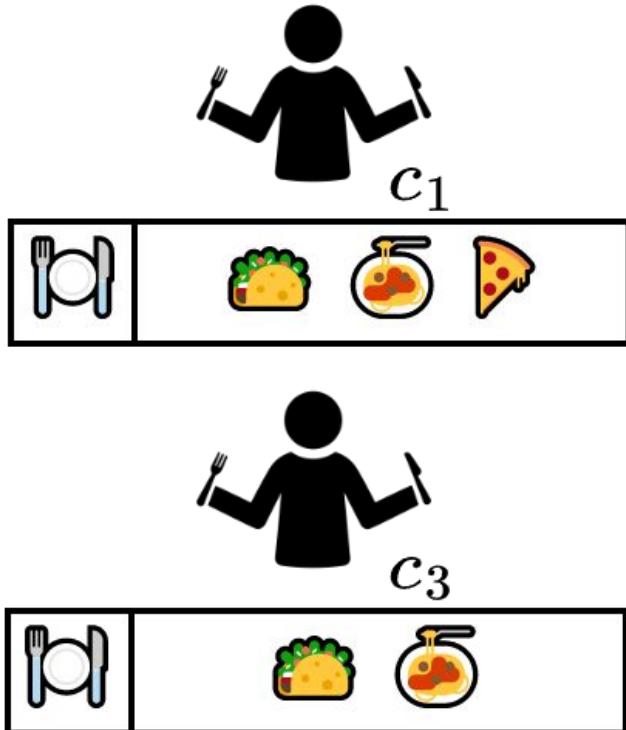
Si no se le puede asignar un plato preferido a un cliente, entonces se le paga un voucher.

El objetivo es minimizar la cantidad de vouchers, o equivalentemente, **maximizar clientes con platos**, cumpliendo con todas las restricciones.

Ejemplo

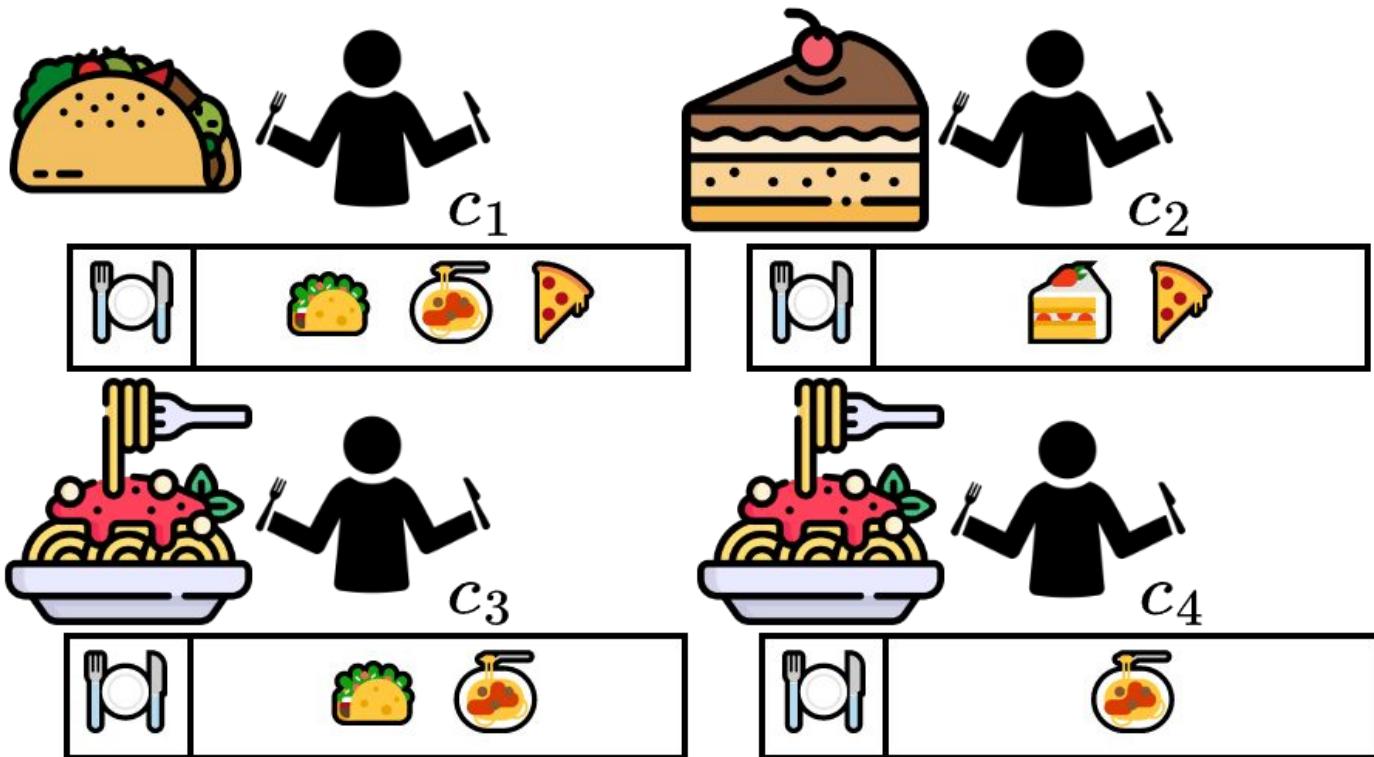
Ejemplo: Comidas y clientes

p_j	q_j
Taco	3
Cake	2
Pasta	1
Pizza	2



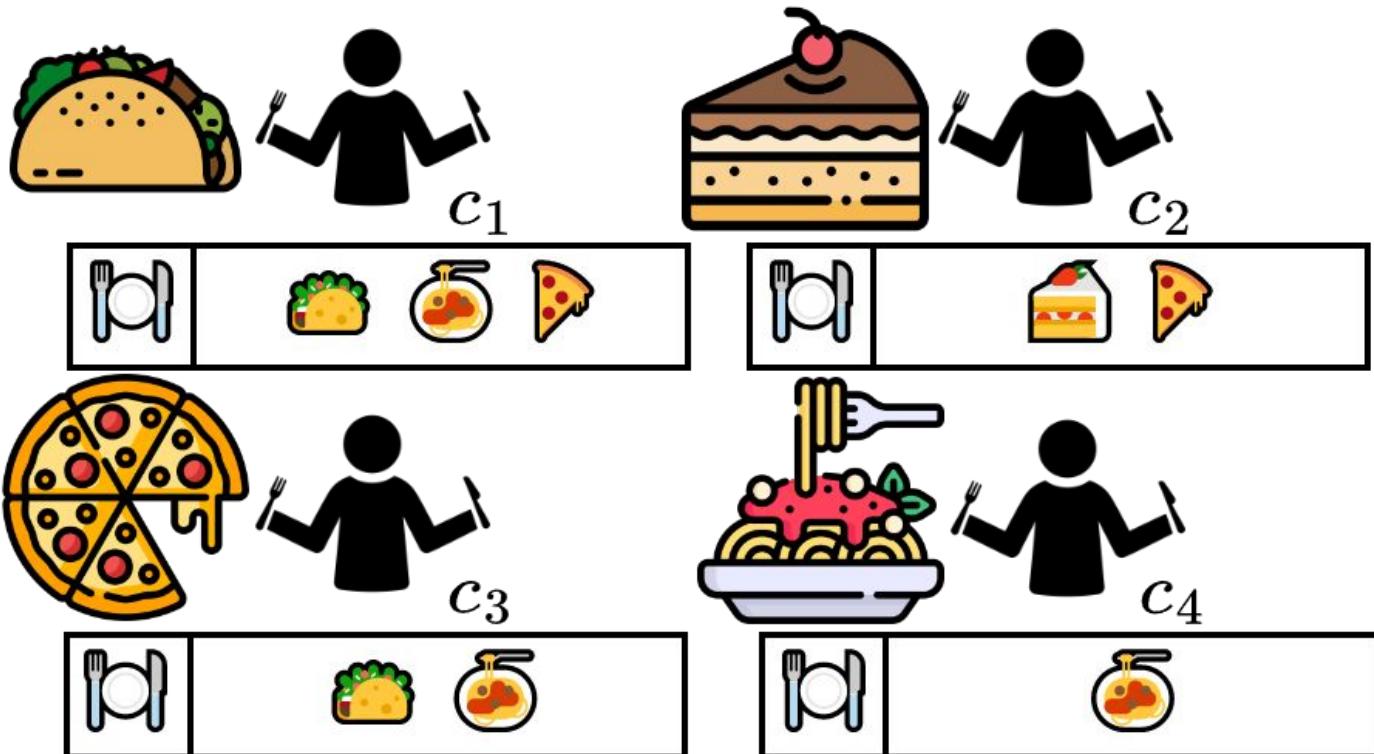
Ejemplo: Inválida por cantidades

p_j	q_j
	3
	2
	1
	2



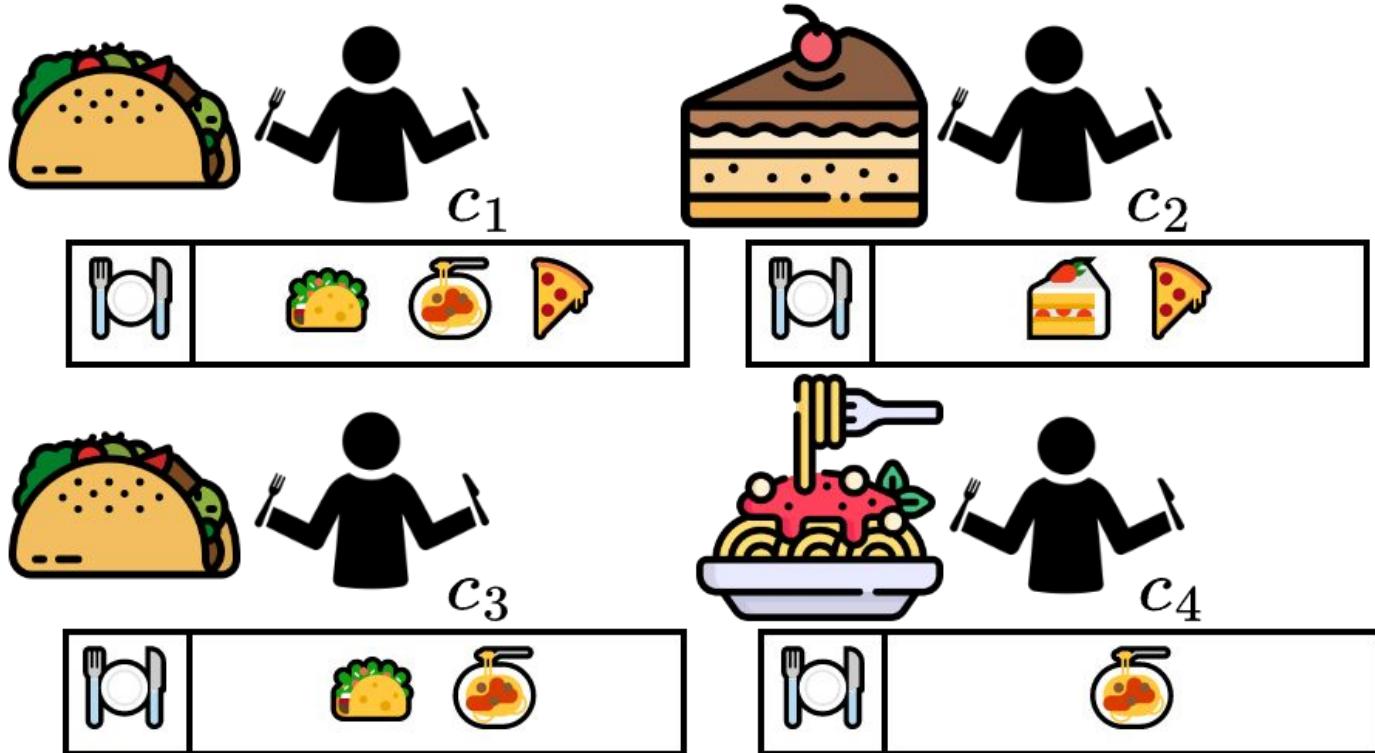
Ejemplo: Inválida por preferencias

p_j	q_j
	3
	2
	1
	2



Ejemplo: Posible solución

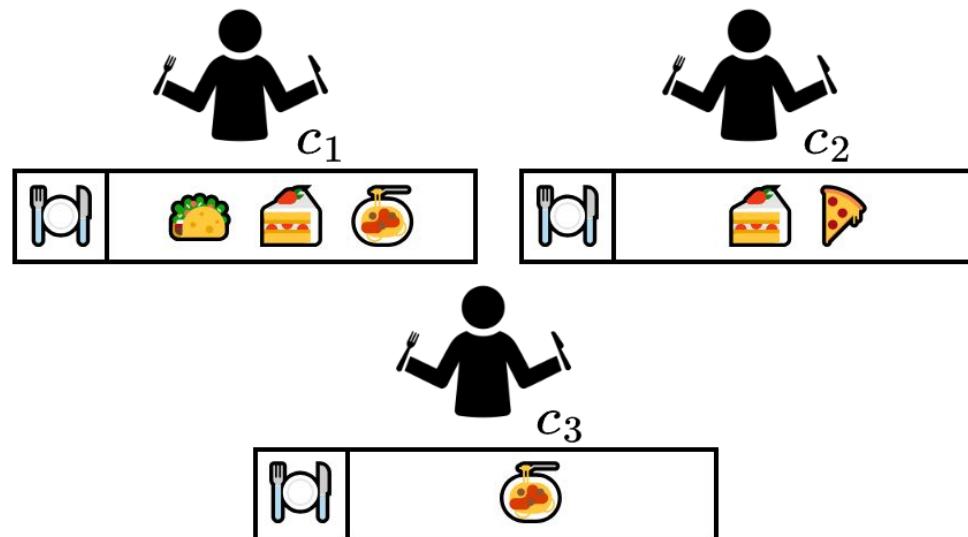
p_j	q_j
	3
	2
	1
	2



Datos de ejemplo para la resolución

Para modelar la red vamos a estar utilizando estos platos y clientes:

p_j	q_j
	3
	2
	1
	4

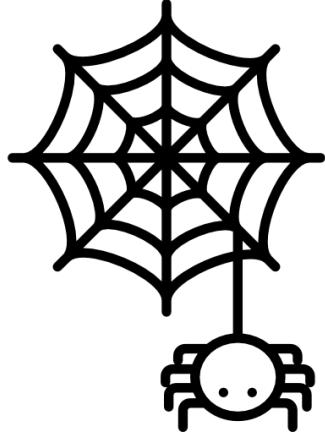




Receta

Pasos para resolver un problema de flujo máximo.

- Modelo de red.
 - Semántica de unidad de flujo.
 - Conexión de los mundos problema ↔ modelo.
 - Algoritmo que resuelva el modelo.
 - Complejidad del algoritmo en base a parámetros del problema.
-



Paso 1

Modelo de red

Vamos a definir un modelo de la red en donde especificamos:

- Quién es **s**.
 - Quién es **t**.
 - Qué es cada nodo distinto a **s** y **t**.
 - Qué aristas hay entre nodos.
 - Cuál es la capacidad de cada nodo.
-



Tip #1

Elemento a asignar como unidad de flujo



¿Qué queremos asignar?

Lo primero que queremos reconocer en un problema de asignación, es el elemento que se está asignando.

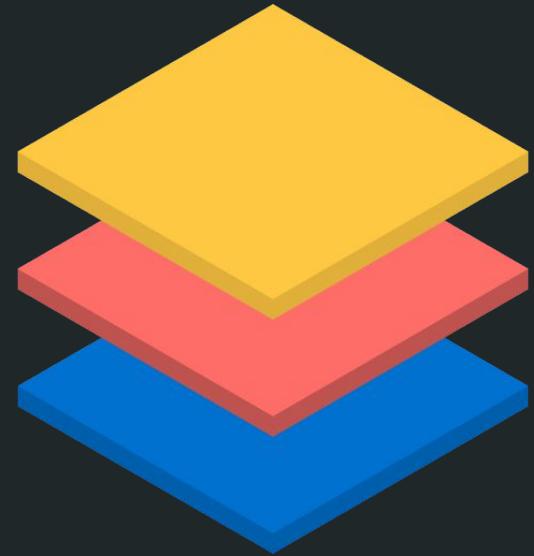
Esto va a terminar siendo lo que representa nuestra unidad de flujo, y nos va a permitir razonar más fácil sobre como modelar la red.

Para este caso particular, lo que queremos asignar es un plato a un cliente.



Tip #2

Entidades como capas del modelo



¿Qué entidades están involucradas en esta votación?

Las entidades involucradas son:

- Clientes
- Platos



Cliente



Plato



Tip #3



Interpretación del orden de las capas

¿Cómo ordenamos las capas?

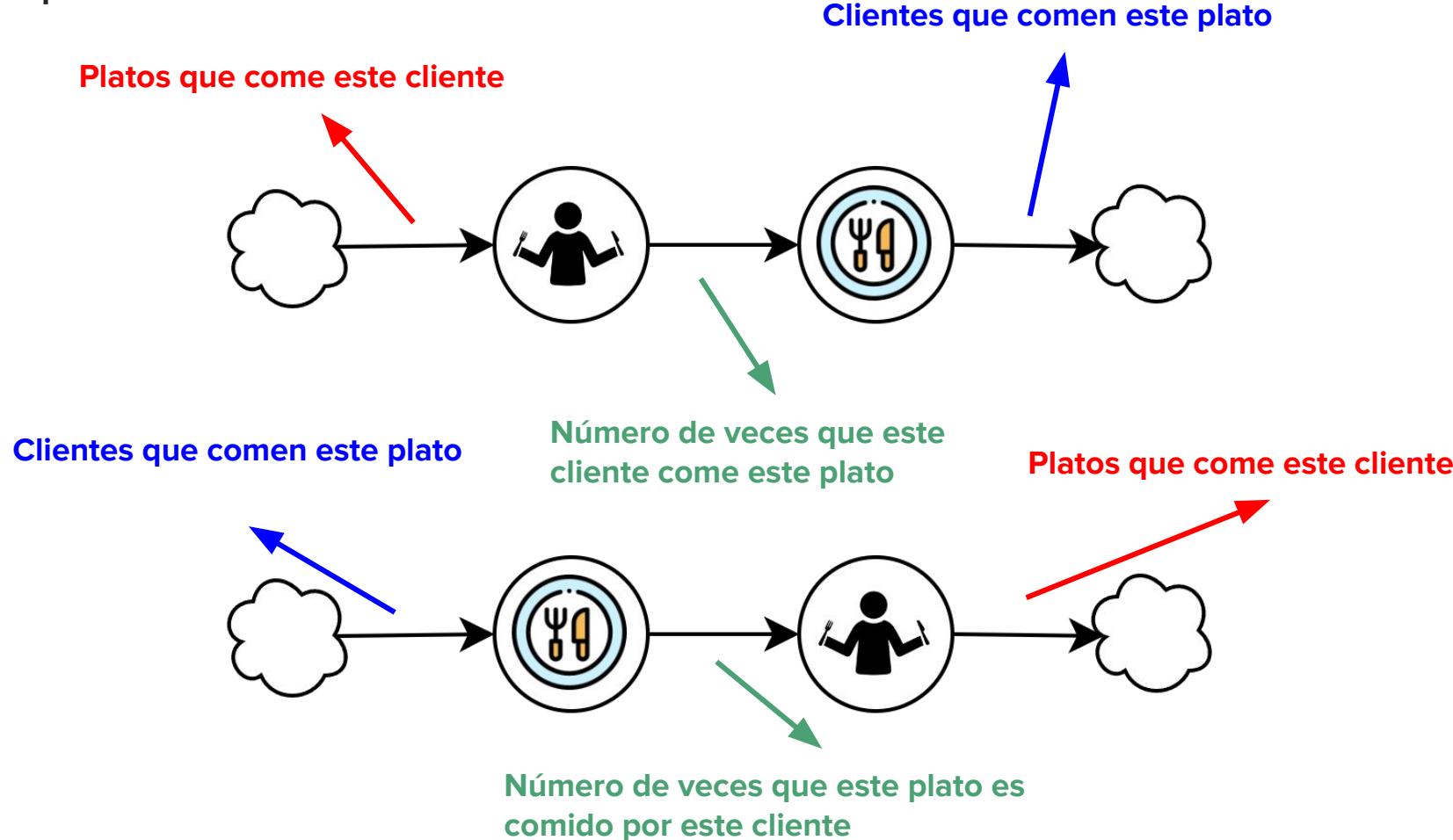
Dependiendo de cómo ordenemos las capas, la entrada y salida de cada una va a tener información distinta.

A su vez, este orden puede hacer que la información que sale de una capa no nos sirva, porque perdemos la información valiosa.

En general cuando solo se tienen 2 capas, el orden es indistinto porque termina siendo lo mismo en sentido inverso.

Veamos algunos ejemplos para este problema.

Ejemplos de órdenes



Entidades como capas del modelo

Pongamos las entidades anteriores en capas para nuestro modelo de red, junto con los nodos **s** y **t**.

p_j	q_j
	3
	2
	1
	4



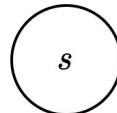
c_1



c_2



c_3



s



c_1



c_2



c_3



p_1



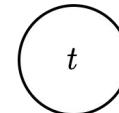
p_2



p_3



p_4



t





Tip #4

Restricciones como conexiones



Interpretando restricciones como conexiones

Las conexiones en el modelo de la red van a surgir de las restricciones que hay para las entidades del problema. A su vez, las capacidades van a estar relacionadas directamente con los números que se mencionan en cada una de las restricciones.

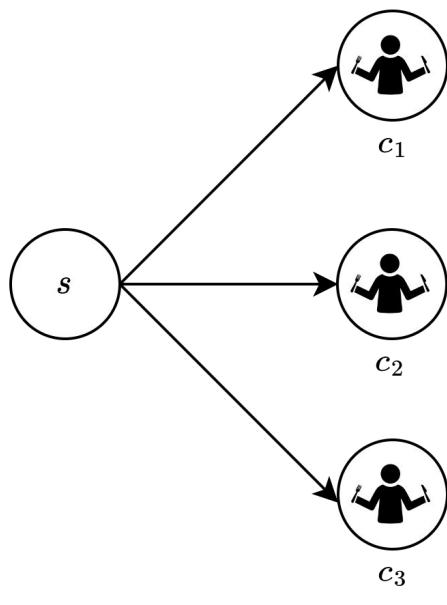
Para este caso, podemos pensar en las restricciones para luego agregar las conexiones:

1. Cada cliente c_i come un único plato.
2. Cada cliente c_i puede comer un conjunto de platos C_i .
3. Para cada plato $p \square$ hay una cantidad $q \square$ del mismo.

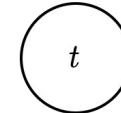
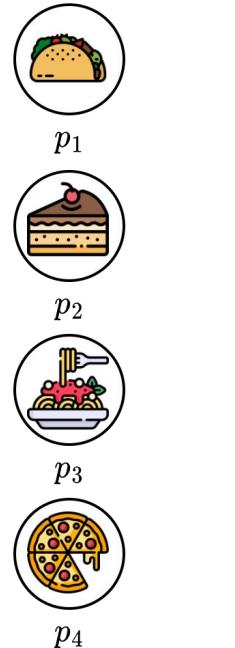
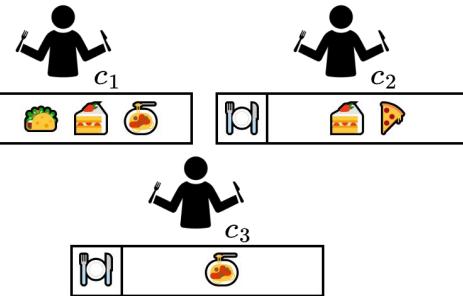
Veamos cómo incluir esto en nuestro modelo.

Conexiones

Cada cliente c_i come una única vez.

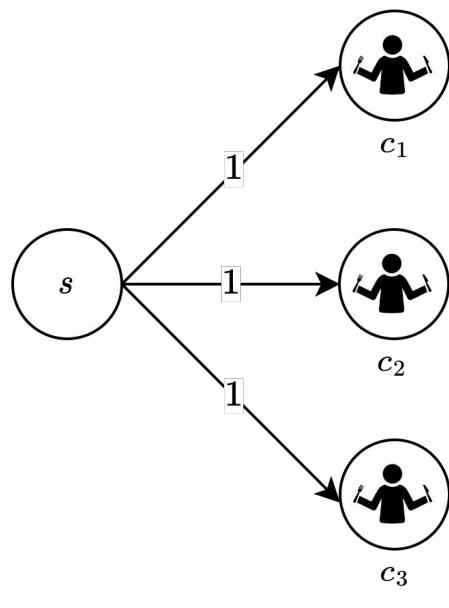


p_j	q_j
	3
	2
	1
	4

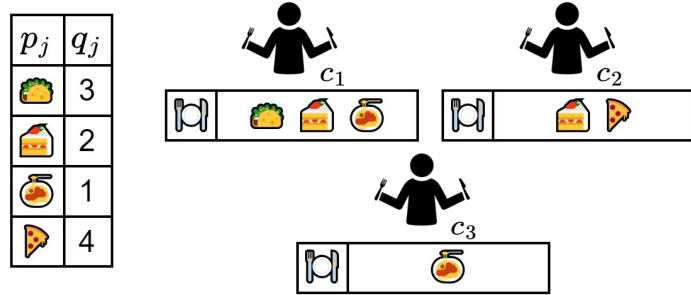


Restricciones

Cada cliente c_i come una única vez.



p_j	q_j
	3
	2
	1
	4



p_1



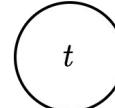
p_2



p_3



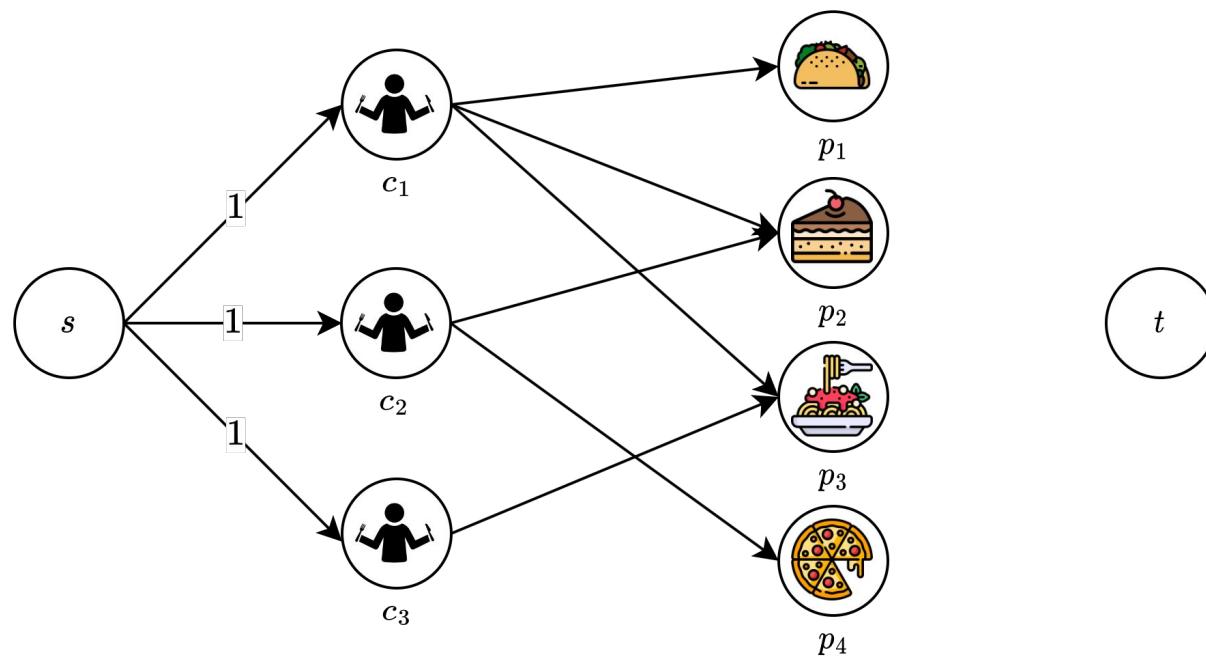
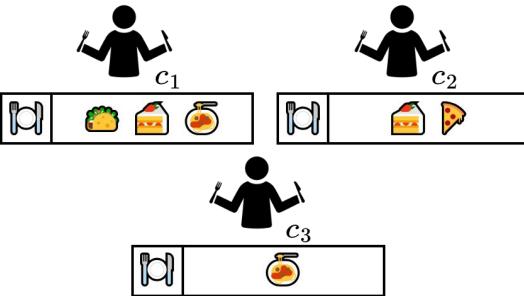
p_4



Conexiones

Cada cliente c_i puede comer un conjunto de platos C_i .

p_j	q_j
	3
	2
	1
	4





Tip #5

Poner infinito o no poner infinito

Esa es la cuestión...



Capacidades que dan “igual”

Nos vamos a encontrar con casos donde la capacidad que vamos a querer poner va a ser indistinta, y podría poner cualquier número, así también infinito.

No es incorrecto esto, ya que puede pasar que una restricción A sea más fuerte que otra B y ya nos cubra esta última.

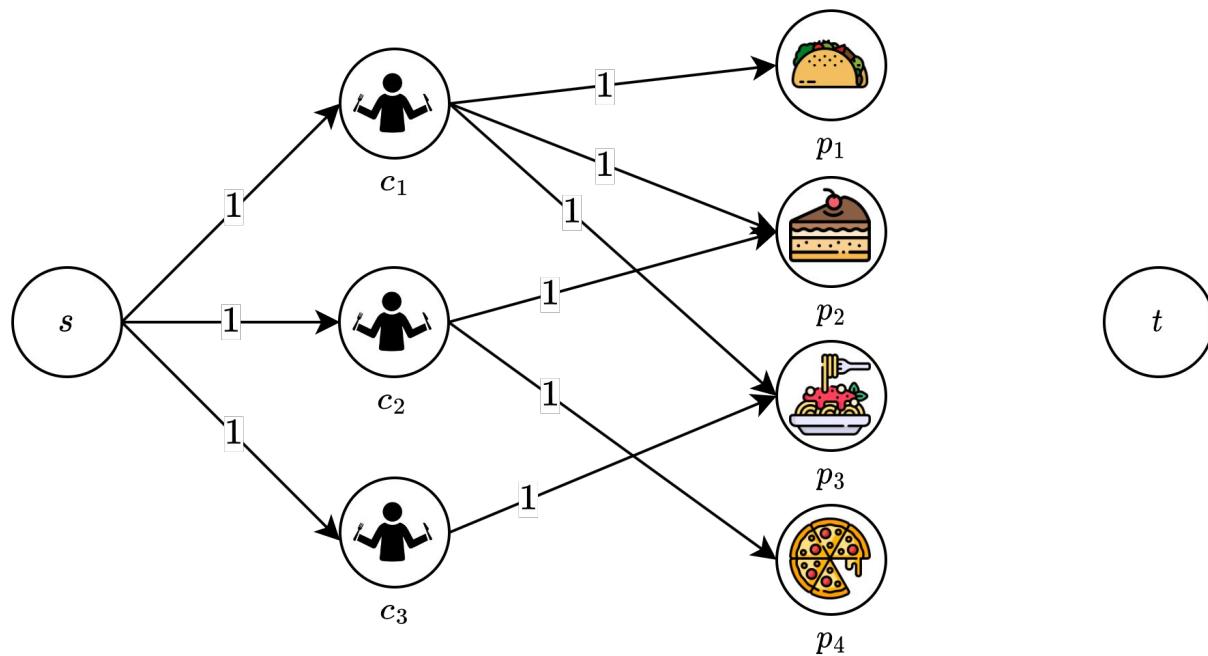
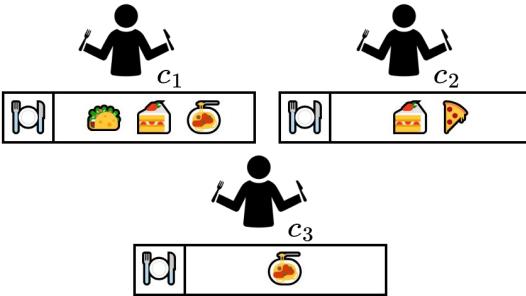
Para este problema por ejemplo, la conexión entre cliente - plato no requiere que pongamos un número específico.

Esto es así porque si bien es cierto que cada cliente puede comer 1 vez cualquier plato que le guste, ya está limitado por la restricción de que el cliente puede comer 1 plato en general.

Restricciones

Cada cliente c_i puede comer un conjunto de platos C_i .

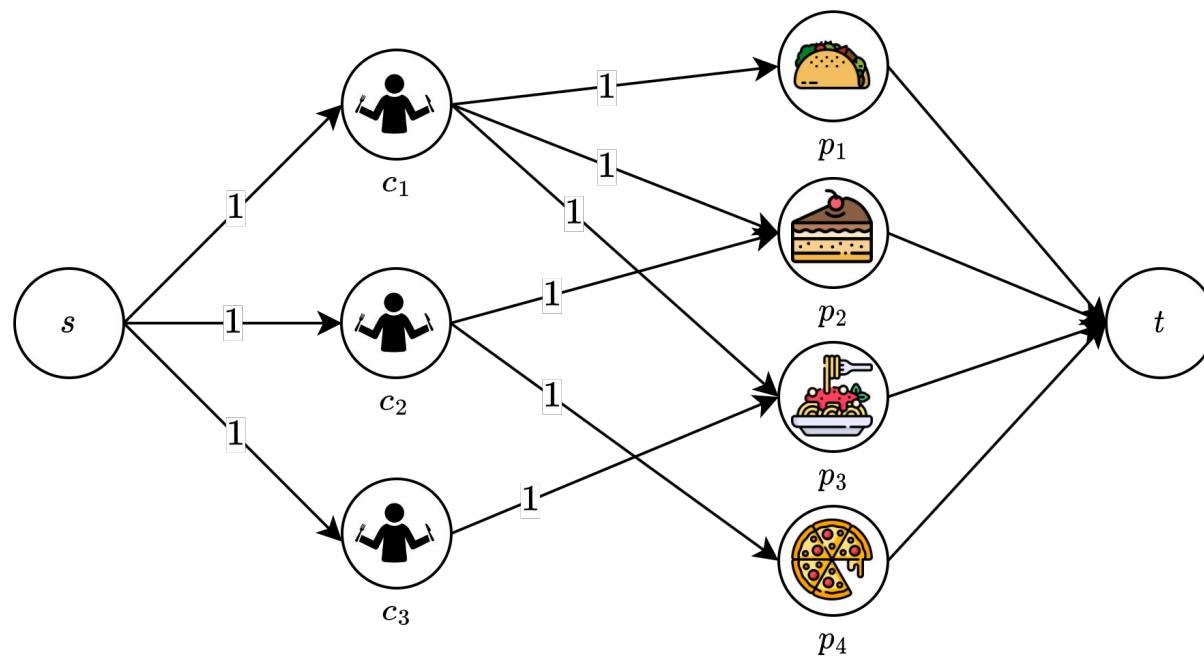
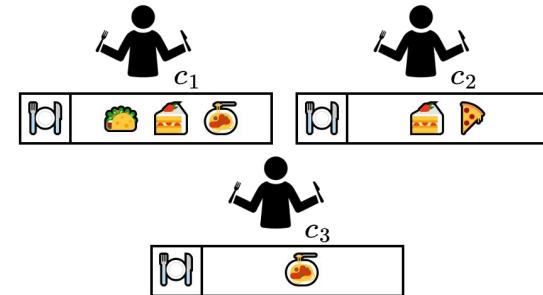
p_j	q_j
3	3
2	2
1	1
4	4



Conexiones

Para cada plato $p \square$ hay una cantidad $q \square$ del mismo.

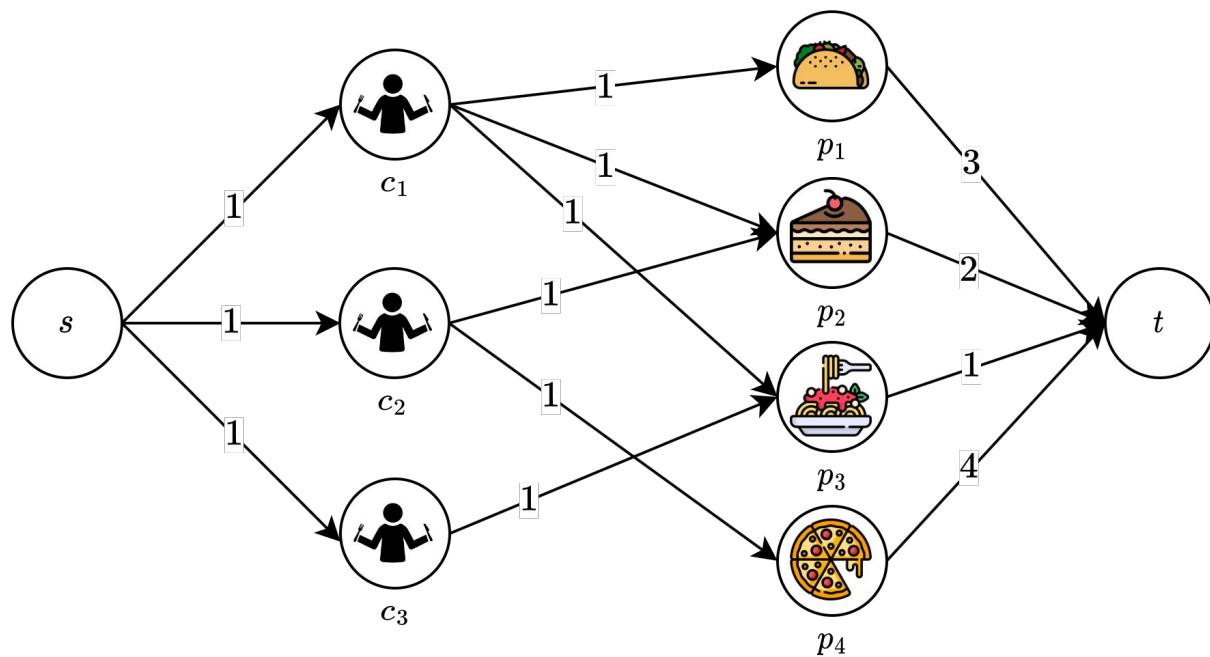
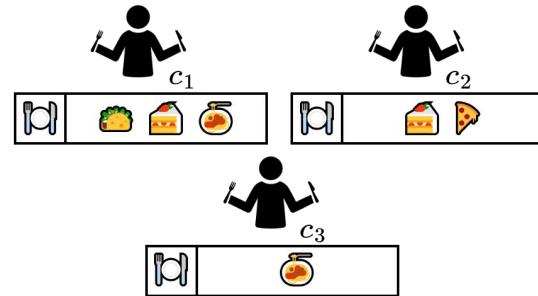
p_j	q_j
Taco	3
Ensalada	2
Pasta	1
Pizza	4



Restricciones

Para cada plato $p \square$ hay una cantidad $q \square$ del mismo.

p_j	q_j
Taco	3
Ensalada	2
Postre	1
Pizza	4





Paso 2

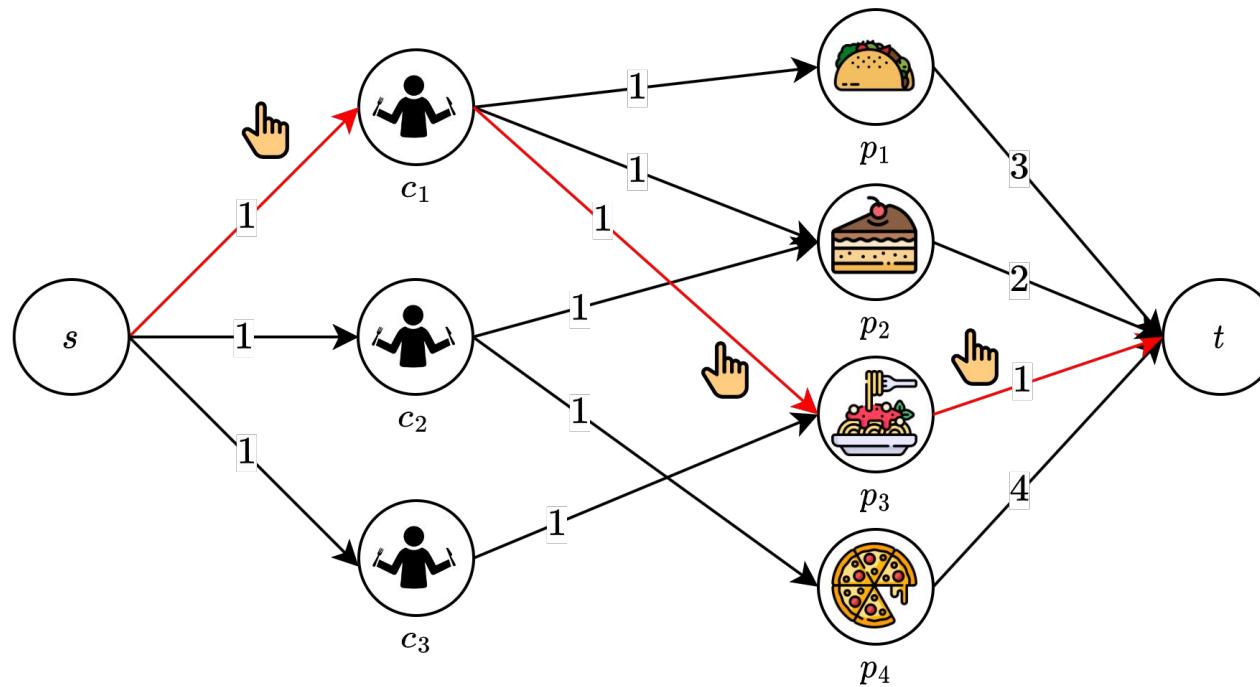
Semántica de unidad de flujo

En este paso tenemos que explicar cómo se interpreta una unidad de flujo en nuestro modelo.

¿Cómo interpretamos a la unidad de flujo en la red?

Representa una asignación de un cliente a un plato dentro de los preferidos por este.

Podemos pensarla gráficamente en nuestro modelo así:





Paso 3

Conexión
Problema ↔ Modelo

Acá queremos establecer una relación entre las soluciones de los 2 mundos:

- Problema original.
- Modelo de flujo.

Mostrando la conexión entre el modelo y el problema

Para hacer esto en el caso general podemos demostrar que:

1. “**Existe una solución válida al problema de tamaño $N \leftrightarrow$ Existe un flujo factible M** ”

Donde buscamos una relación entre N y M . ¿Por qué queremos esto?

- Si tenemos \rightarrow , dada una solución al problema, podemos construir un flujo en la red. Con esto sabemos que nuestra red modela bien soluciones válidas.
 - Esto no nos garantiza que toda solución de flujo se puede ver como una en el problema.
- Si tenemos \leftarrow , podemos transformar un flujo máximo en una solución del problema.
 - Esto no nos garantiza que no nos perdemos ninguna solución del problema original.

Notar que en particular con esto hay una relación 1:1 de **solución óptima : flujo máximo**.



Tip #6



¿Y si demuestro “solución óptima \leftrightarrow flujo factible máximo” directo?

Demostración alternativa: “solución óptima \leftrightarrow flujo máximo”

Podemos hacer esta otra demostración, pero puede resultar más sencillo ir por la anterior, ya que en este caso tenemos que mostrar que:

1. La solución óptima se condice con el flujo máximo obtenible en la red (habría que justificar por qué no puede obtenerse algo mejor).
2. El flujo máximo se puede ver como una solución óptima y que no podría conseguir otra solución mejor.

Básicamente tenemos que justificar máximos y óptimos, que es complicado a veces.

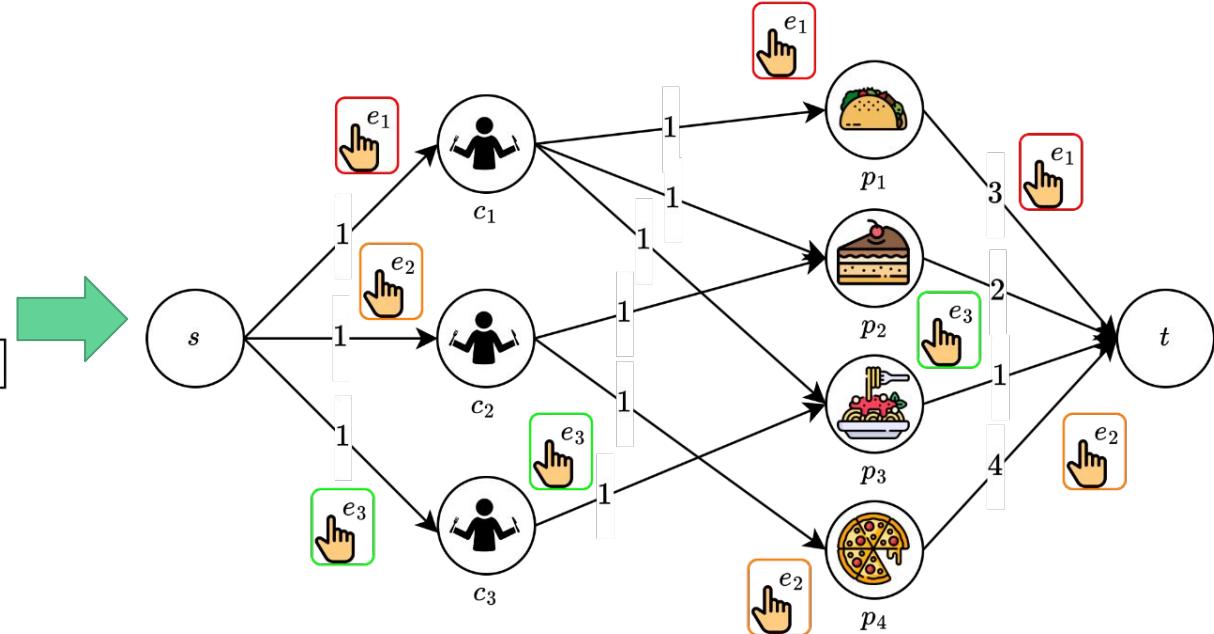
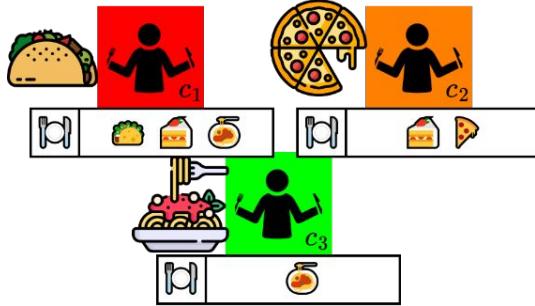
Mostrando la conexión entre el modelo y el problema

Para este caso puntual vamos a querer demostrar:

“Existe asignación de U clientes a platos \leftrightarrow Existe un flujo factible de U flujo”

Demostremos cada implicación guiándonos con un gráfico de cada caso.

p_j	q_j
taco	3
pasta	2
pizza	1
cake	4



Mostrando la conexión entre el modelo y el problema

“Existe asignación válida de U clientes a platos \rightarrow Existe un flujo factible de U flujo”

Tenemos una asignación al problema y la queremos transformar en una asignación de flujo en la red que respete las propiedades de flujo. Podemos hacer lo siguiente para cada asignación $(c_i, p\square)$:

- Asignamos 1 unidad de flujo en la arista $s \rightarrow c_i$.
- Asignamos 1 unidad de flujo en la arista $c_i \rightarrow p\square$.
- Asignamos 1 unidad de flujo en la arista $p\square \rightarrow t$.

Veamos que esta asignación es un flujo factible.

Mostrando la conexión entre el modelo y el problema

“Existe asignación válida de U clientes a platos \rightarrow Existe un flujo factible de U flujo”

Siempre existen las aristas mencionadas, en particular de cliente a plato, porque es una solución factible y la red tiene aristas de clientes a platos si lo prefieren.

¿El flujo respeta capacidades?

- $s \rightarrow c_i$: Siempre se asigna 1 máximo porque se le asigna 1 plato al cliente por ser factible.
- $c_i \rightarrow p_j$: Siempre se le asigna 1 máximo por lo mismo que antes.
- $p_j \rightarrow t$: Siempre se le asigna q_j clientes a este plato por ser factible.

¿El flujo respeta la conservación?

- En c_i cada asignación agrega 1 unidad en $s \rightarrow c_i$ y $c_i \rightarrow p_j$.
- En p_j cada asignación agrega 1 unidad en $c_i \rightarrow p_j$ y en $p_j \rightarrow t$.

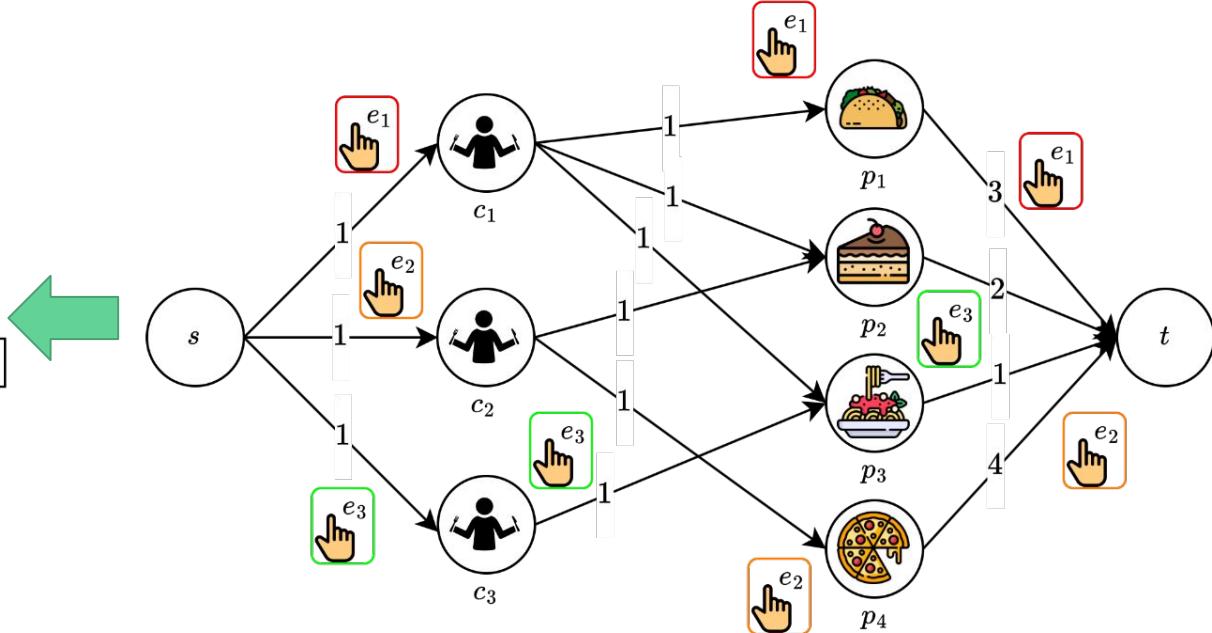
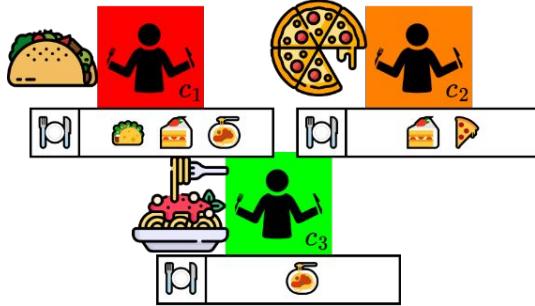
Mostrando la conexión entre el modelo y el problema

“Existe asignación válida de U clientes a platos → Existe un flujo factible de U flujo”

La asignación de flujo que hicimos es **U**?

- Si, porque asignamos **U** unidades de flujo saliendo desde s , una por cada cliente asignado a un plato.

p_j	q_j
taco	3
pasta	2
pizza	1
cake	4



Mostrando la conexión entre el modelo y el problema

“Existe asignación válida de U clientes a platos \leftrightarrow Existe un flujo factible de U flujo”

Por cada unidad de flujo que sale de s a un nodo c_i podemos:

- Asignar el p_j al c_i , donde j es el nodo del plato al que le llega la unidad de flujo que sale desde el nodo del cliente i .

Mostrando la conexión entre el modelo y el problema

“Existe asignación válida de U clientes a platos \leftrightarrow Existe un flujo factible de U flujo”

Veamos que esto respeta las restricciones:

- Solo se asigna 1 plato por c_i porque $s \rightarrow c_i$ tiene capacidad 1.
- Existía la arista del c_i al p_j porque era una preferencia.
- No se asignan más de q_j clientes al plato p_j porque $p_j \rightarrow t$ tiene capacidad q_j .

Mostrando la conexión entre el modelo y el problema

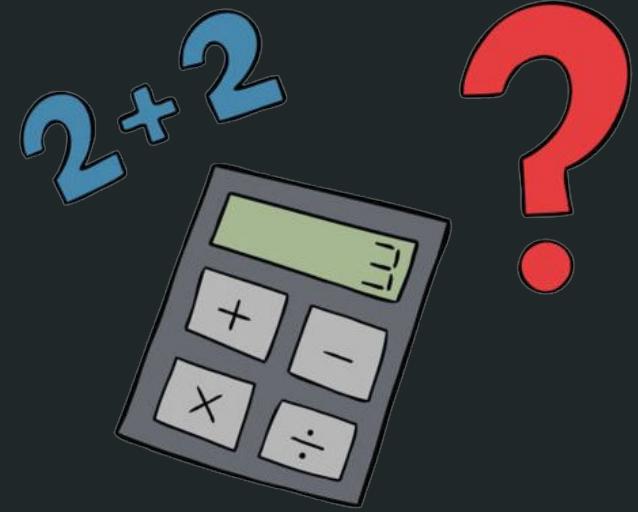
“Existe asignación válida de U clientes a platos \leftarrow Existe un flujo factible de U flujo”

La asignación que hicimos son **U** clientes a platos?

- Si, porque como había **U** de flujo, cada unidad representaba una asignación, y pudimos asignar todos.



Tip #7



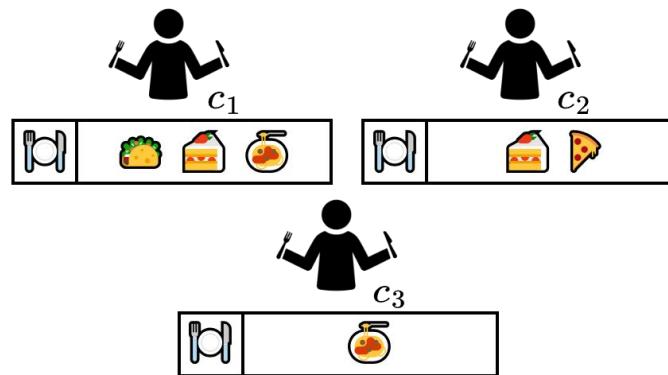
La demostración es imposible porque el modelo está mal

Demostración imposible

Lo que puede pasar al intentar hacer cualquiera de las 2 implicaciones anteriores, es que el modelo sea menos restrictivo / más laxo y nos impide asegurar que se cumplen las restricciones del problema / el flujo factible.

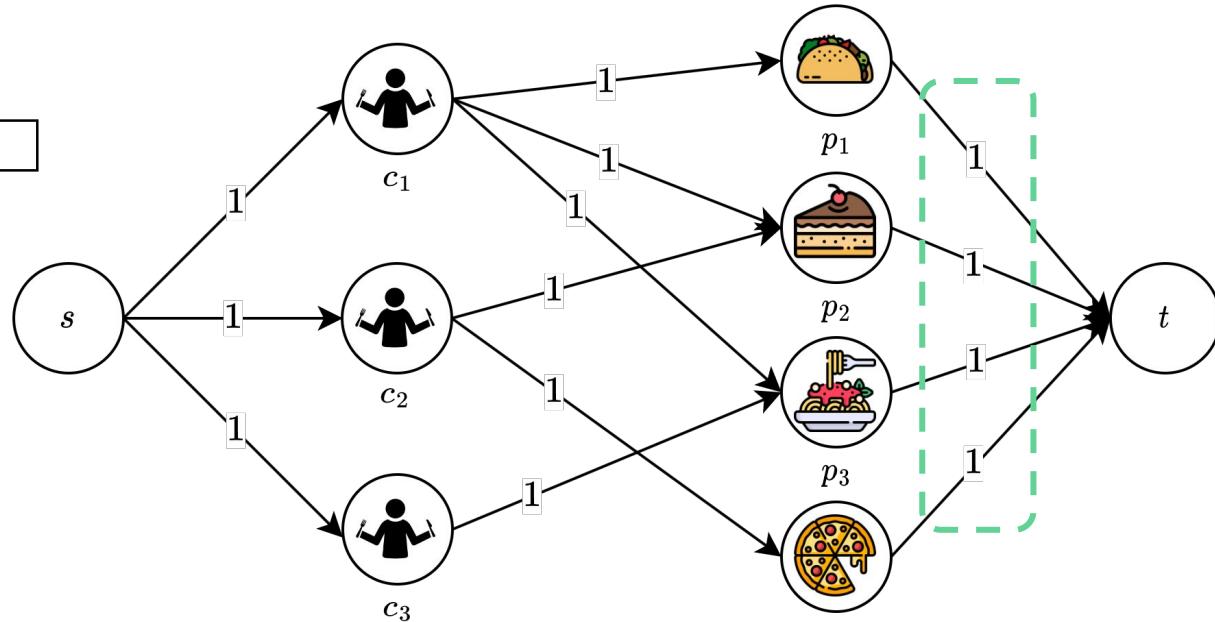
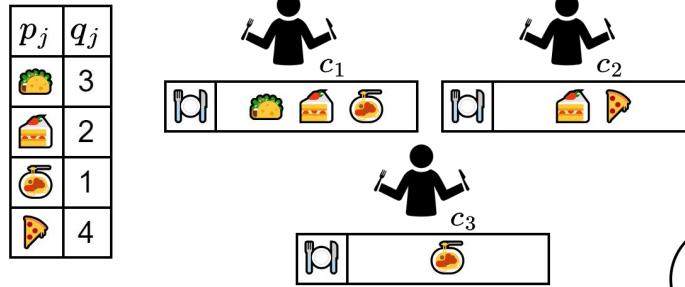
Veamos un ejemplo de cada una con un modelo mal diseñado, siguiendo con los mismos datos que antes:

p_j	q_j
🌮	3
🍰	2
🍕	1
🍕	4



Caso →

“Existe asignación válida de U clientes a platos → Existe un flujo factible de U flujo”



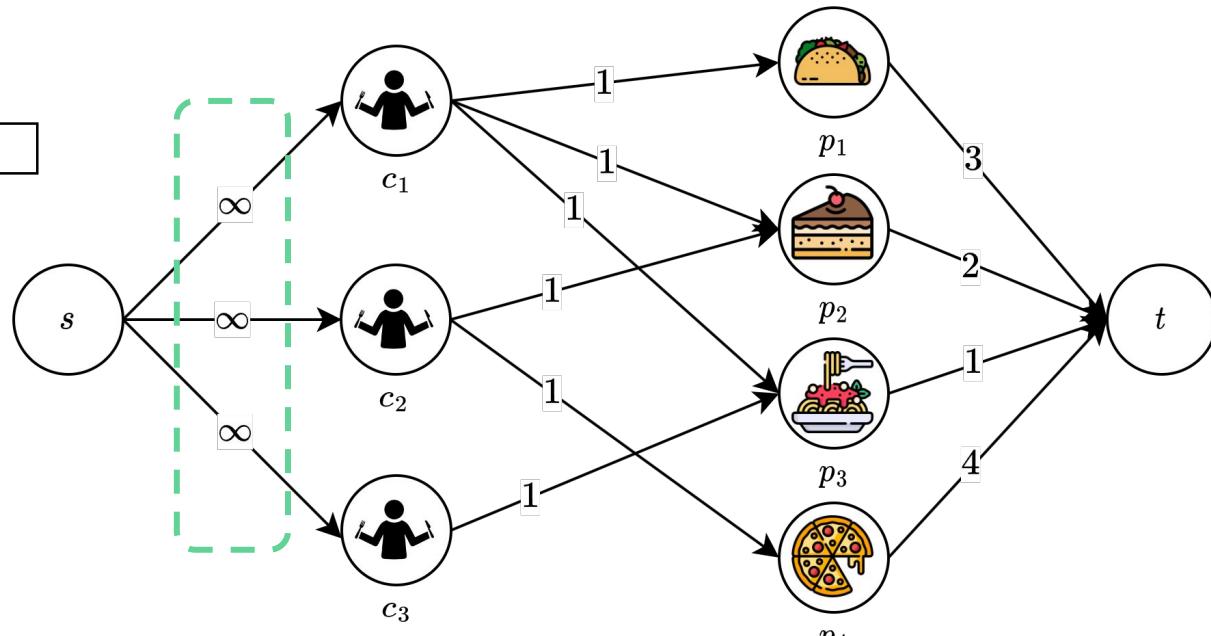
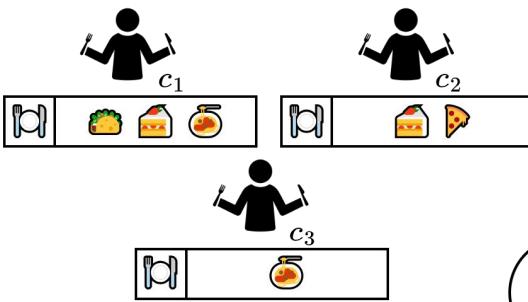
¿Qué es lo que no puedo demostrar acá?

- No puedo garantizar que se respeta la capacidad $p_i \rightarrow t$, porque siempre es 1 y una solución factible puede haber asignado a más de un cliente el mismo plato p_i .

Caso ←

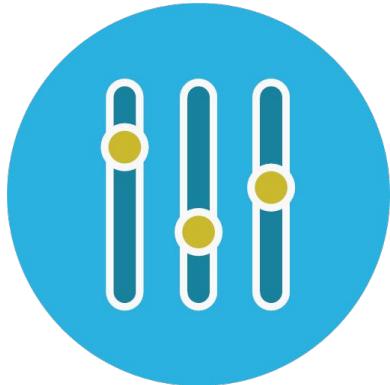
“Existe asignación válida de U clientes a platos ← Existe un flujo factible de U flujo”

p_j	q_j
Taco	3
Sandwich	2
Pasta	1
Pizza	4



¿Qué es lo que no puedo demostrar acá?

- No puedo garantizar que se respeta la restricción de que a cada cliente c_i se le asigna un solo plato, porque la capacidad de $s \rightarrow c_i$ es infinito.



Paso 4

Nodos, aristas y flujo máximo en base a parámetros del problema

En base a las características del problema, determinamos:

- La cantidad de aristas por su relación con el problema.
- La cantidad de vértices por su relación con el problema.
- Flujo máximo por las cotas definidas por el problema.

¿Cuántos nodos, aristas y flujo máximo tenemos?

- Nodos en cada capa:

- Capa de clientes: **C** porque es una por cliente.
- Capa de platos: **P** porque es una por plato.

Hay **O(C+P)** en total.

- Aristas entre cada capa:

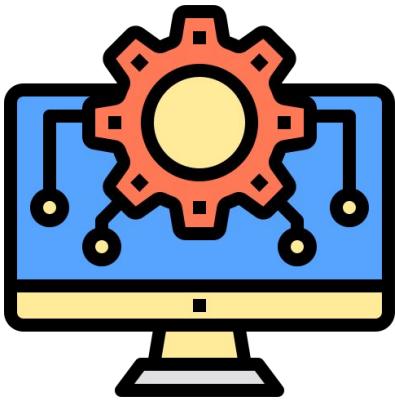
- Entre s y clientes: Hay **C** porque es una por cliente.
- Entre clientes y platos: Hay tantas como la suma de las preferencias de cada cliente, que es **C*P** máximo (si todos prefieren todos los platos).
- Entre platos y t: Hay **P**, porque es una por plato.

Hay **O(C*P)** en total.

- Flujo máximo:

- Puede salir **C** de flujo de s, porque de s a cada jugador hay capacidad 1.
- Puede entrar la suma de los **q** de flujo a t, porque de cada plato a t la capacidad es **q**. Sería la suma de las cantidades, digamos **Q**.

Hay **O(C)** en total, porque como máximo se puede asignar cada cliente a un plato.



Paso 5

Algoritmo y complejidad para resolver modelo

Elegimos, entre todas las opciones de algoritmos que resuelven flujo máximo, el que mejor se adapte al modelo en base a lo que calculamos en el paso anterior.

¿Qué implementación del algoritmo es mejor aca?

Tenemos estas opciones:

- **Ford & Fulkerson:** $O(|E| * F)$.
 - Reemplazando tenemos $O(CP * C) = O(C^2 P)$.
- **Edmonds Karp:** $O(|V| E^2)$.
 - Reemplazando tenemos $O((C+P) * (CP)^2) = O(C^3 P^2 + P^3 C^2)$.

Termina ganando **Ford & Fulkerson** en el caso general.



Monkeys in the Emei Mountain

[Link](#)

Enunciado

Enunciado

Tenemos **N** ($1 \leq N \leq 100$) monos que quieren tomar agua de un lago. El i -ésimo mono:

- Quiere tomar v_i ($0 \leq v_i \leq 50000$) agua.
- Puede tomar agua en un intervalo de tiempo $[a_i, b_i]$ ($0 \leq a_i, b_i \leq 50000$).

Todos estos toman una unidad de agua por unidad de tiempo.

La dificultad de todo esto es que solo pueden tomar **M** ($1 < M \leq 5$) monos al mismo tiempo del lago.

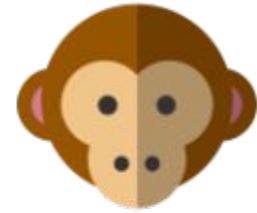
El objetivo es determinar si todos los monos pueden tomar la cantidad de agua que necesitan, cumpliendo con todas las restricciones.

Ejemplo

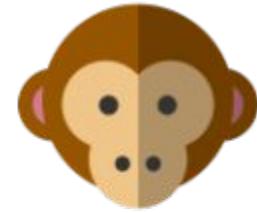
Ejemplo: Monos tomando agua



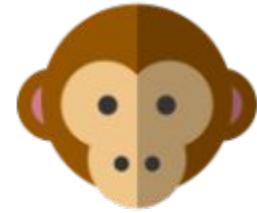
$$M = 2$$

 m_1

	[1,4]
	3

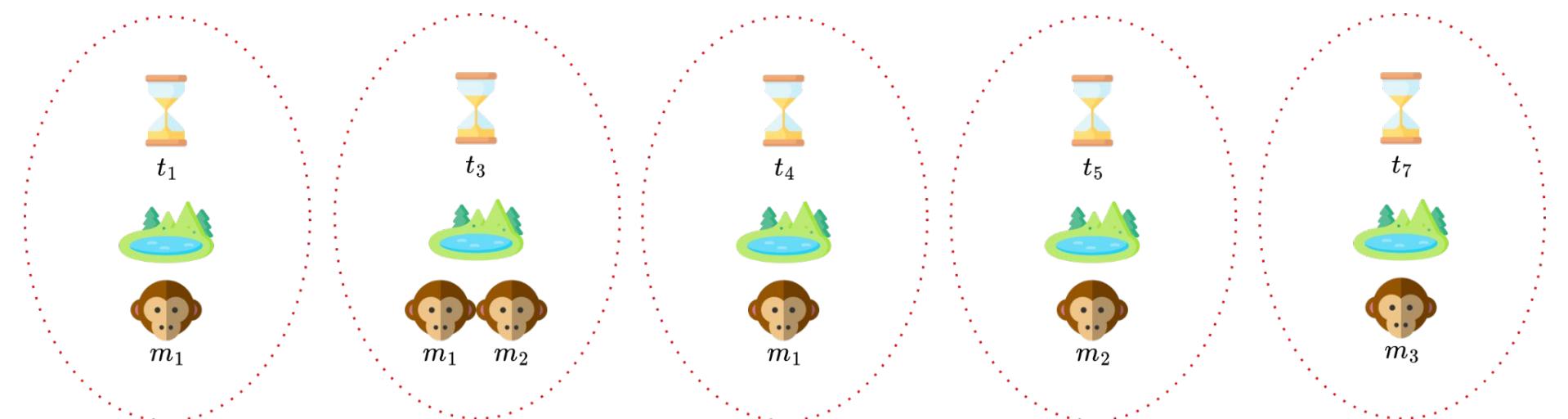
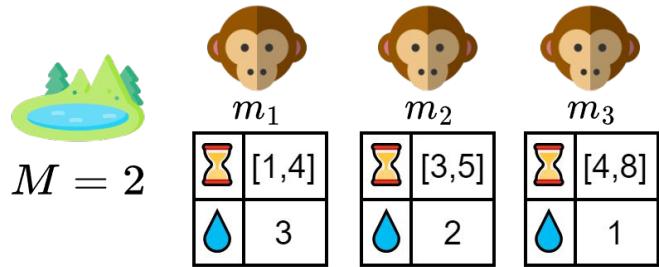
 m_2

	[3,5]
	2

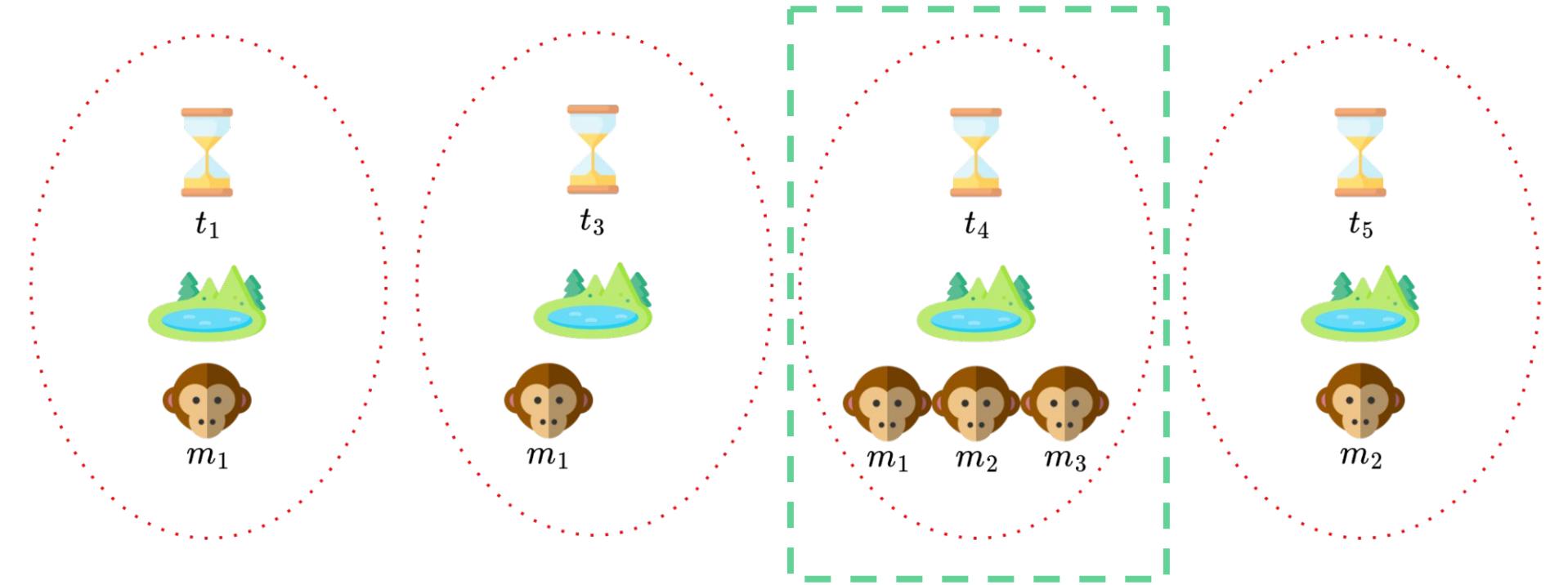
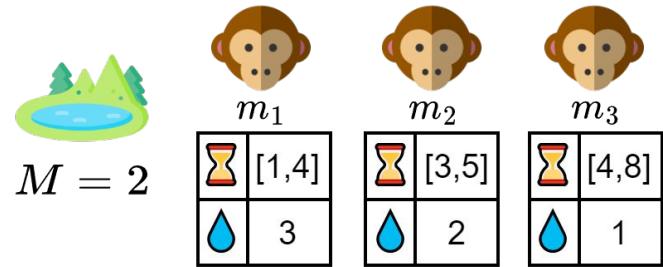
 m_3

	[4,8]
	1

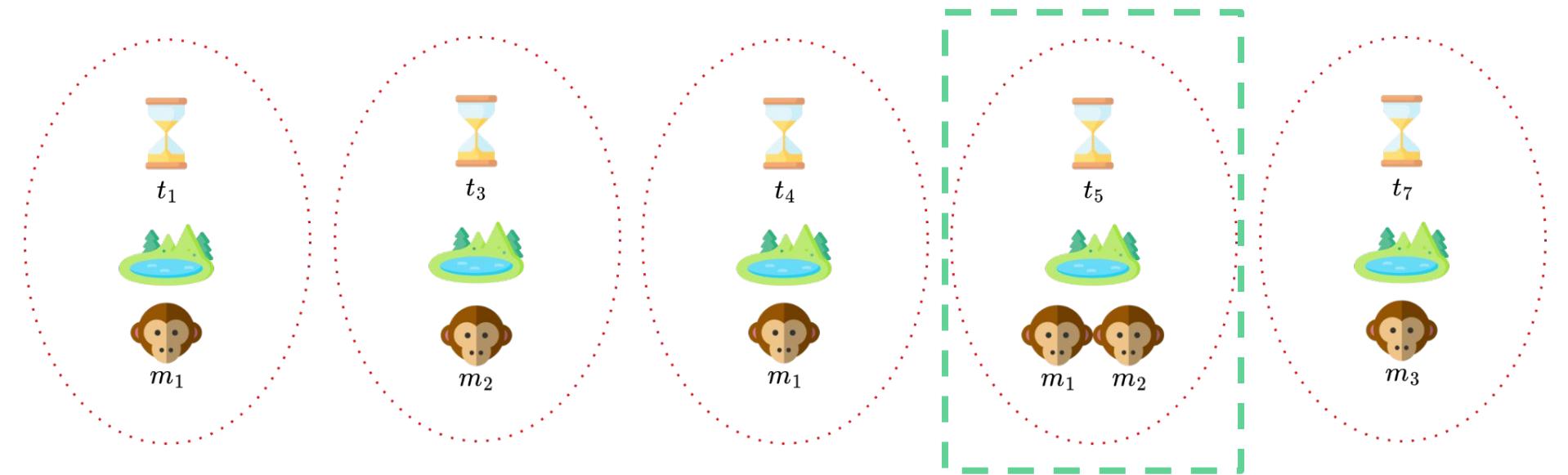
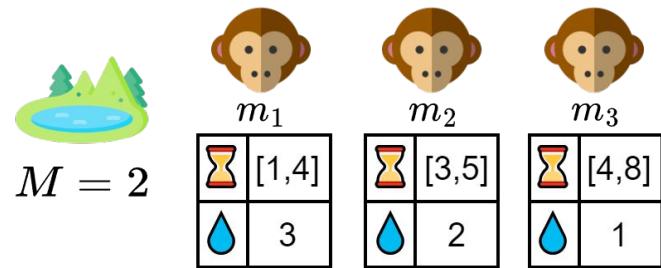
Ejemplo: Solucion valida



Ejemplo: Exceso de monos en lago



Ejemplo: Tomando en tiempo invalido



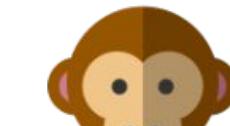
Resolución

Datos de ejemplo para la resolución

Para modelar la red vamos a estar utilizando estos monos y M.



$$M = 2$$



m_1

	[1,4]
	3



m_2

	[3,5]
	2



m_3

	[7]
	1

Modelo de red

¿Qué queremos asignar?

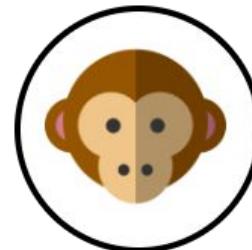
Tenemos monos turnándose para tomar agua, donde cada uno puede tomar en cierto intervalo de tiempo y pueden haber **M** simultáneamente tomando en una unidad de tiempo.

Queremos asignar entonces cada sorbo de agua que quiere tomar cada mono a una unidad de tiempo.

¿Qué entidades están involucradas?

Las entidades involucradas son:

- Monos
- Unidad de tiempo



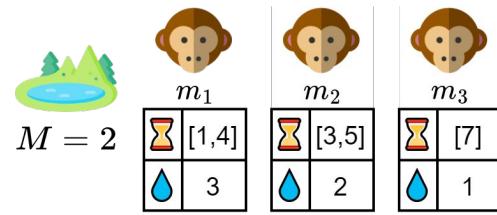
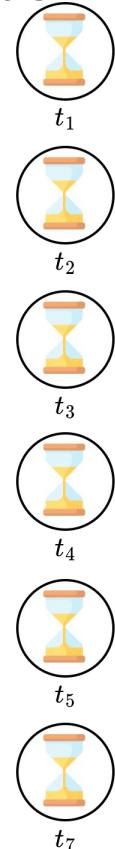
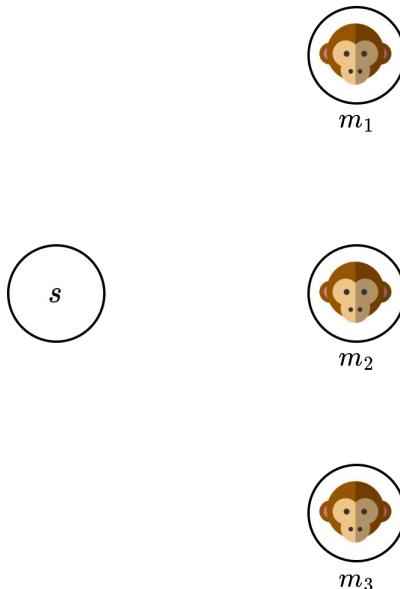
Mono



Tiempo

Entidades como capas del modelo

Pongamos las entidades anteriores en capas para nuestro modelo de red, junto con los nodos s y t.



¿Qué restricciones tenemos?

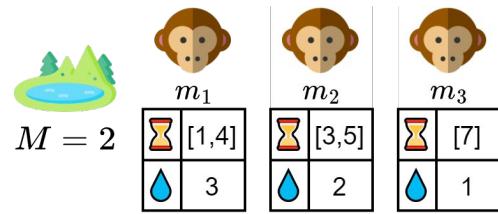
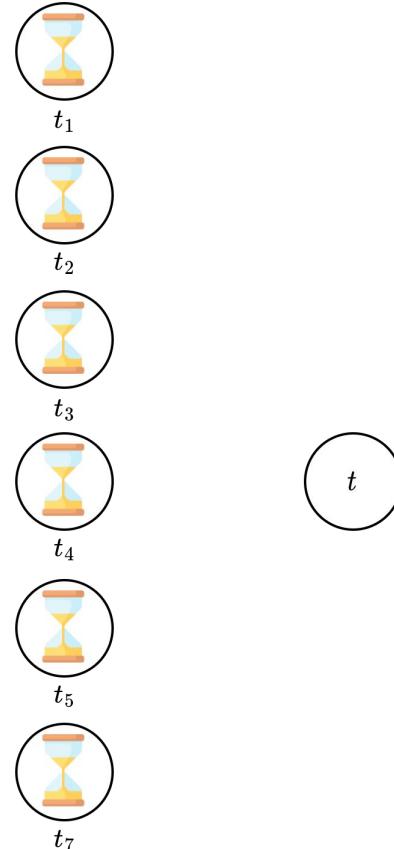
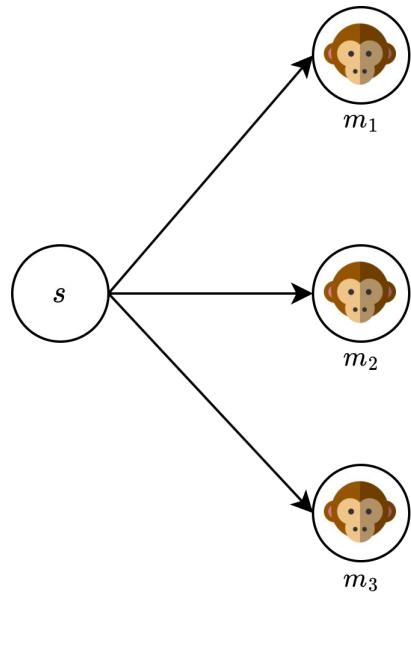
Pensando en las restricciones del problema, podríamos decir que:

1. Cada mono quiere tomar una cantidad, ni más ni menos.
2. Cada mono puede tomar en un cierto intervalo contiguo de tiempo.
3. Cada mono no puede tomar más de un sorbo de agua por unidad de tiempo.
4. No pueden tomarse más de **M** sorbos de agua al mismo tiempo.

Veamos cómo incluir esto en nuestro modelo.

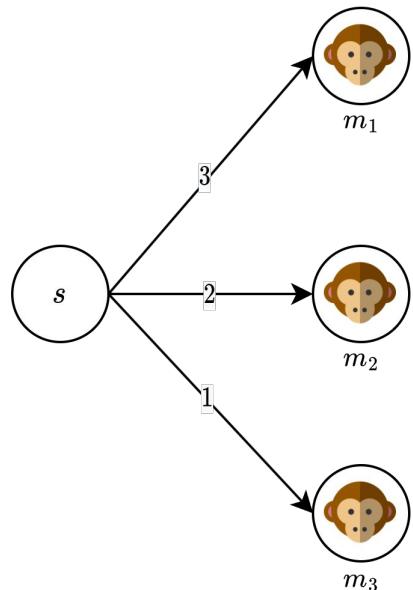
Conexiones

Cada mono quiere tomar una cantidad, ni más ni menos.



Restricciones

Cada mono quiere tomar una cantidad, ni más ni menos.



t_1



t_2



t_3



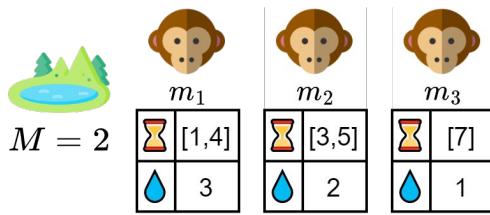
t_4



t_5

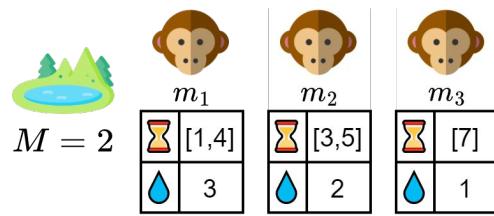
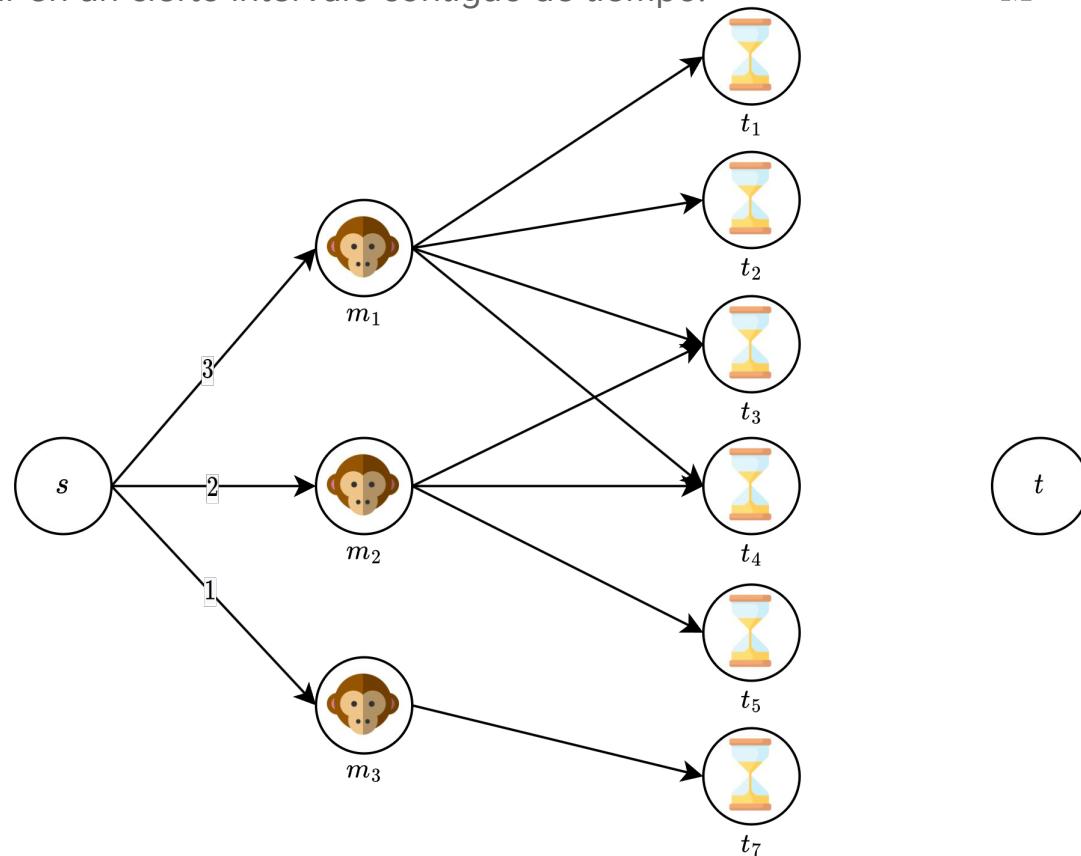


t_7



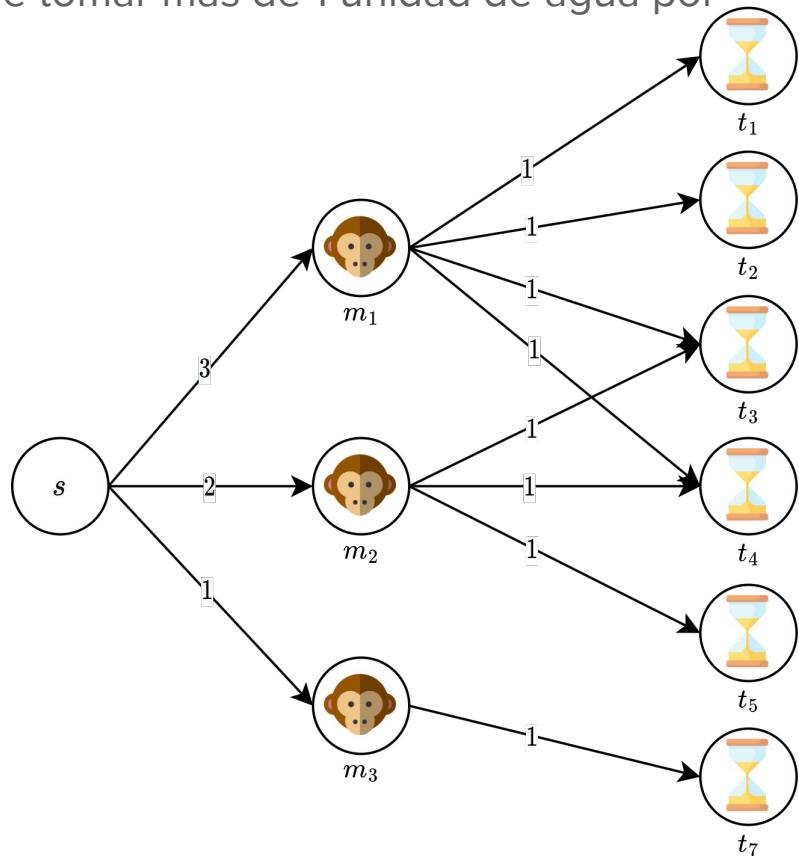
Conexiones

Cada mono puede tomar en un cierto intervalo contiguo de tiempo.



Restricciones

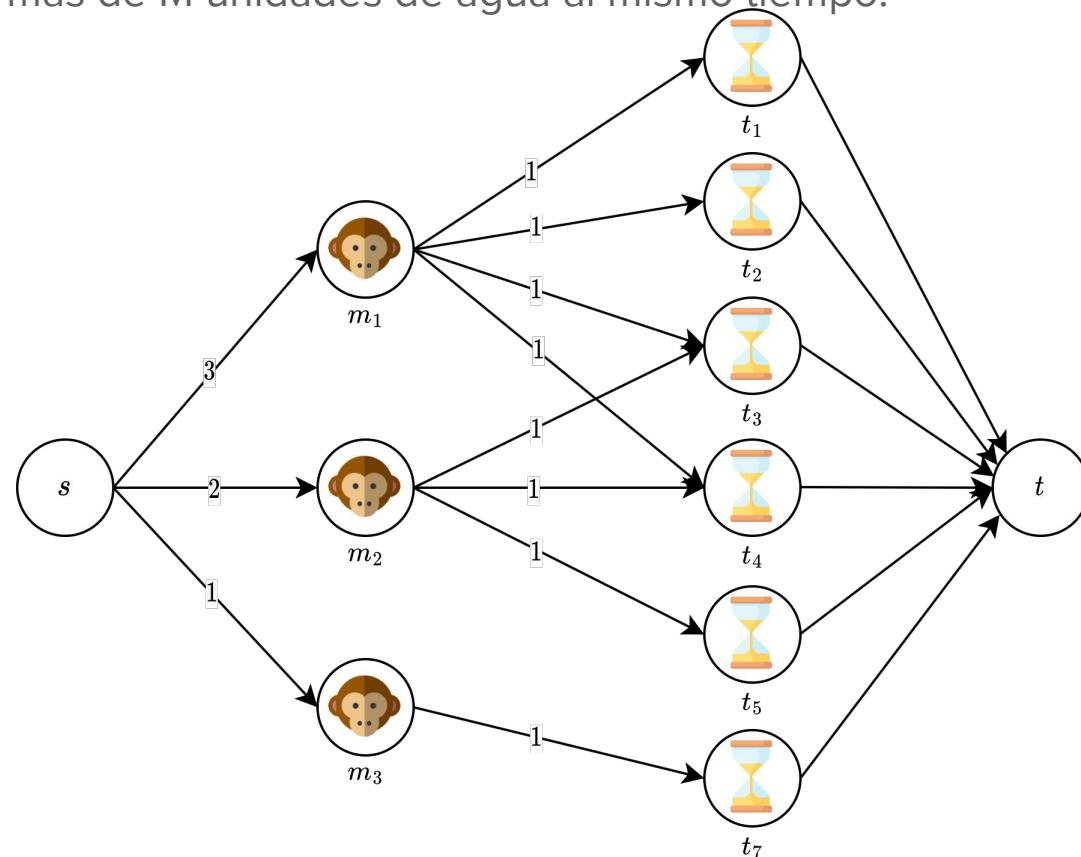
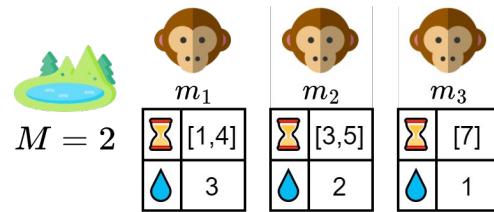
Cada mono no puede tomar más de 1 unidad de agua por unidad de tiempo.



$M = 2$	m_1	m_2	m_3
[1,4]	3	[3,5]	[7]
3	2	1	

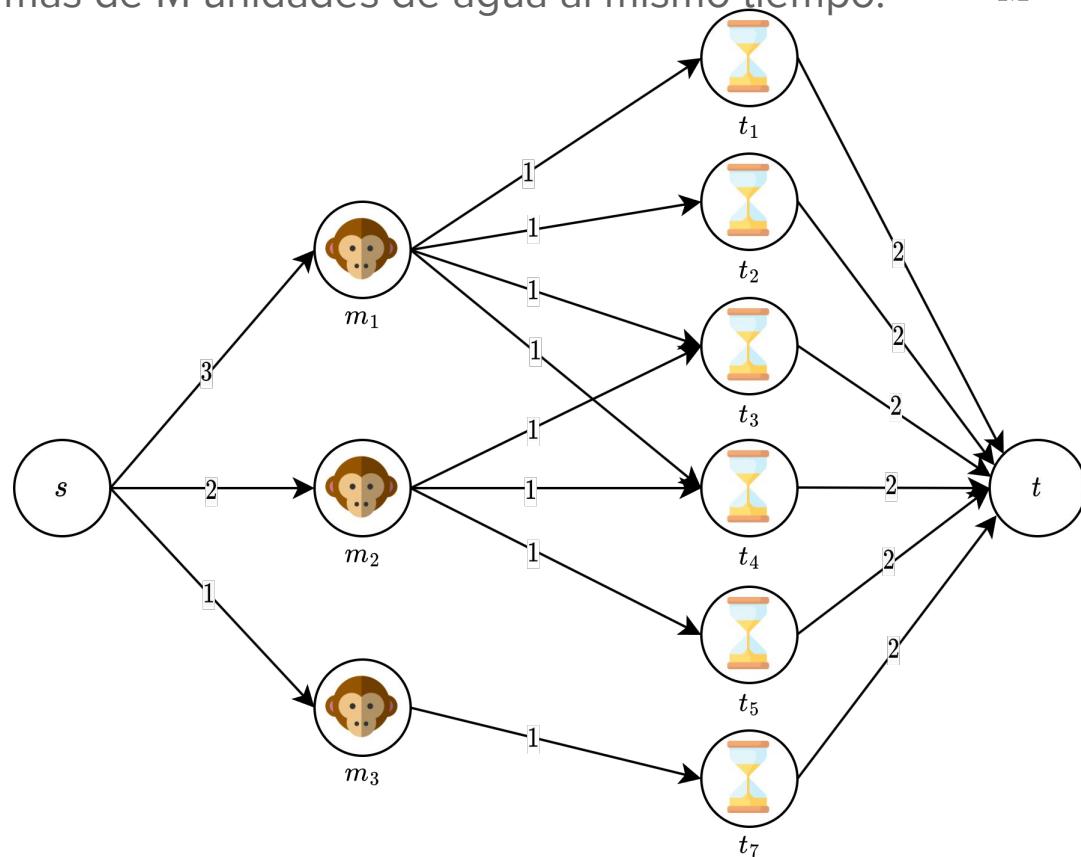
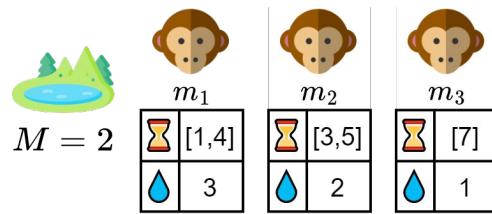
Conexiones

No pueden tomarse más de M unidades de agua al mismo tiempo.



Restricciones

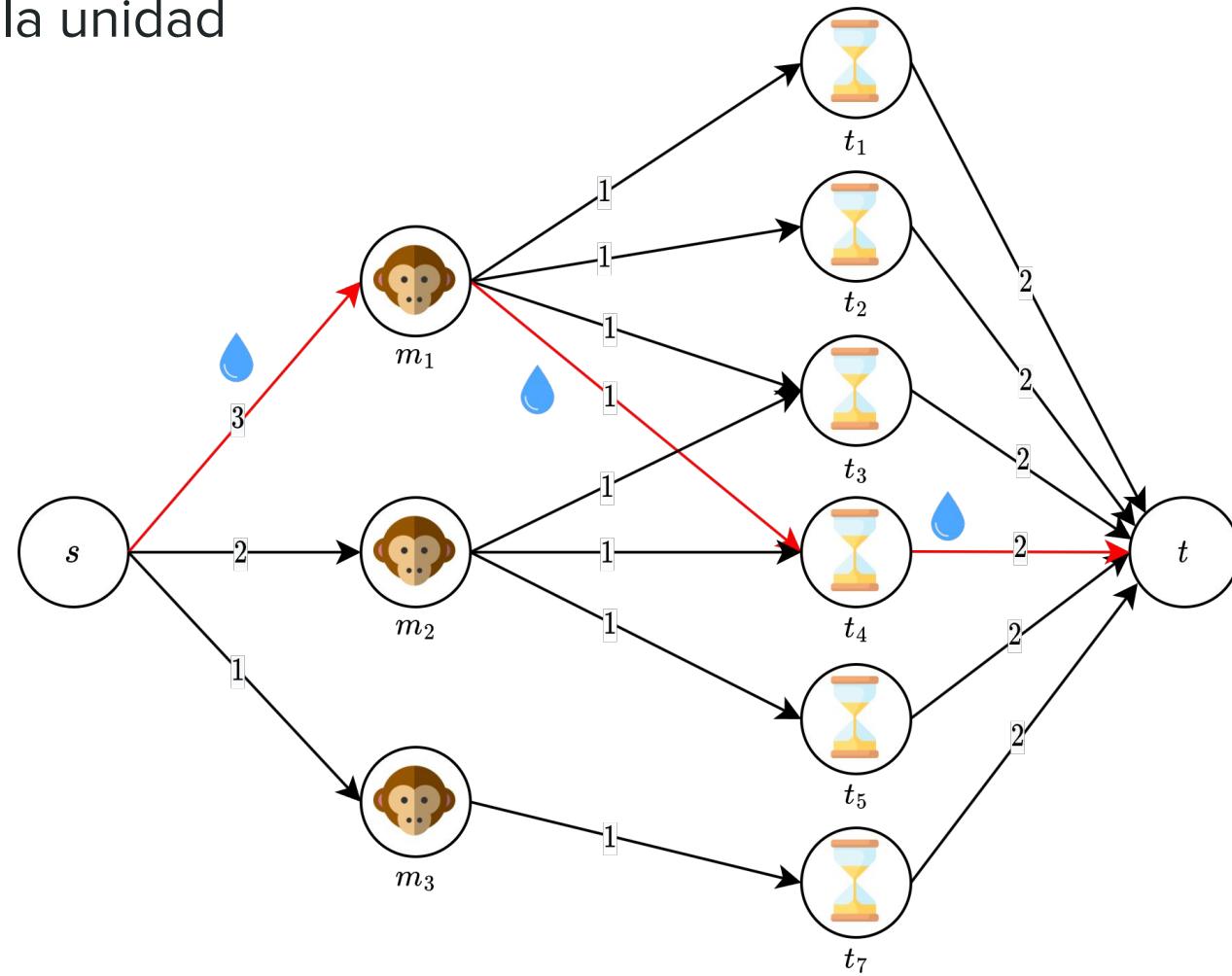
No pueden tomarse más de M unidades de agua al mismo tiempo.



Semántica de unidad de flujo

¿Cómo interpretamos a la unidad de flujo en la red?

Como lo pensamos cuando empezamos el modelo, podemos interpretarlo como una unidad de agua asignada a un mono en un instante de tiempo.



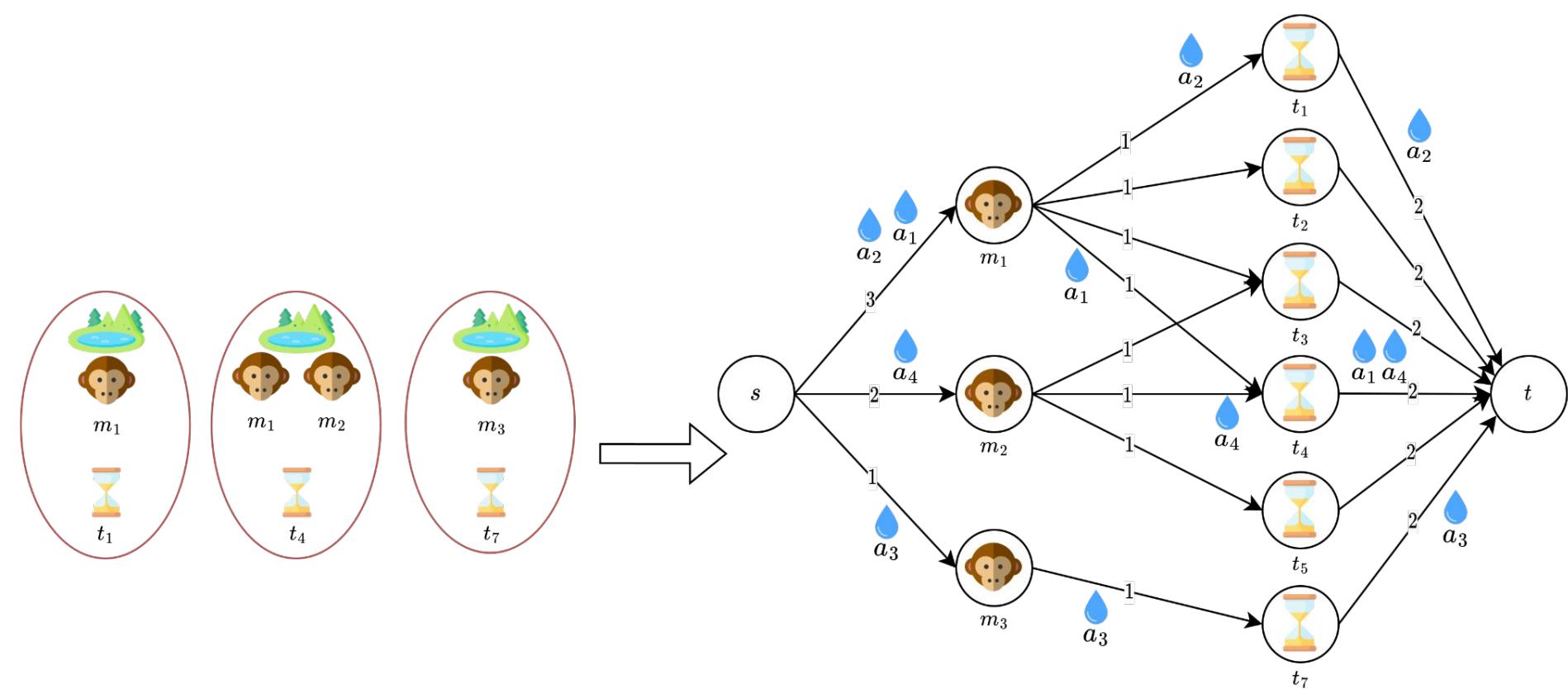
Conexión Problema ↔ Modelo

Mostrando la conexión entre el modelo y el problema

Vamos a querer demostrar esta afirmación:

“*Existe una forma de asignar U sorbos* \leftrightarrow *Existe un flujo factible de U flujo*”

Demostremos cada implicación, guiándonos con un gráfico de cada caso.



Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U sorbos → Existe un flujo factible de U flujo”

Como existe una solución para asignar U sorbos, por cada uno que hace que el mono m_i tome en el instante $t\square$, construimos un camino de flujo 1 de la siguiente forma:

- Asignamos 1 unidad de flujo en la arista $s \rightarrow m_i$.
- Asignamos 1 unidad de flujo en la arista $m_i \rightarrow t\square$.
- Asignamos 1 unidad de flujo en la arista $t\square \rightarrow t$.

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U sorbos → Existe un flujo factible de U flujo”

Siempre existen las aristas mencionadas, en particular del mono al tiempo, porque es una solución factible y la red tiene aristas de monos a tiempos si pueden tomar en ese instante.

¿El flujo respeta capacidades?

- $s \rightarrow m_i$: Siempre se asigna v_i máximo porque un mono puede tomar hasta v_i en la solución.
- $m_i \rightarrow t\Box$: Siempre se le asigna 1 máximo porque un mono puede tomar 1 sorbo por unidad de tiempo.
- $t\Box \rightarrow t$: Siempre se le asigna M máximo porque no puede haber más de M monos tomando a la vez, por ser factible.

¿El flujo respeta la conservación y entra lo mismo que sale?

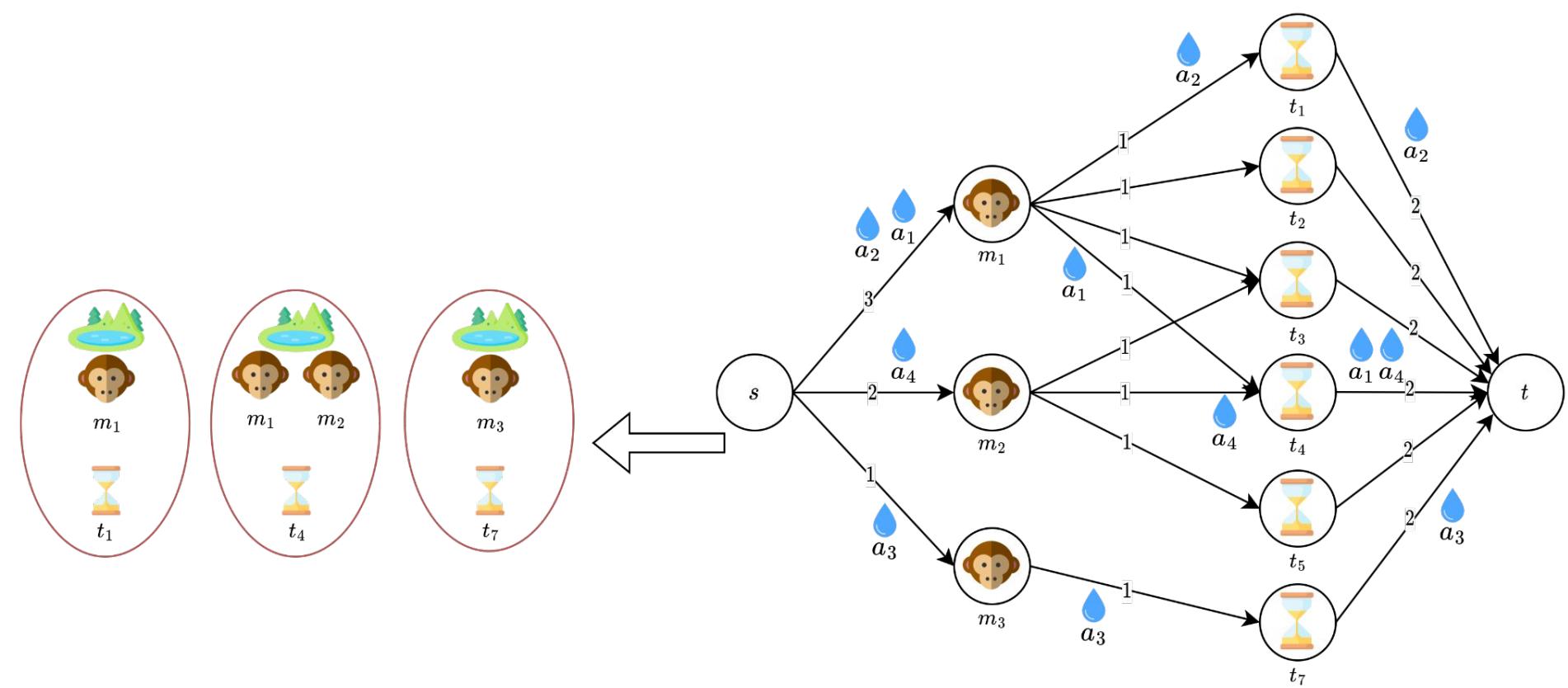
- En m_i cada asignación agrega 1 unidad en $s \rightarrow m_i$ y $m_i \rightarrow t\Box$.
- En $t\Box$ cada asignación agrega 1 unidad en $m_i \rightarrow t\Box$ y en $t\Box \rightarrow t$.

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U sorbos → Existe un flujo factible de U flujo”

La asignación de flujo que hicimos es U?

- Si, porque como la asignación de sorbos es válida, había U sorbos y asignamos U unidades de flujo saliendo desde s, una por cada sorbo.



Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U sorbos \leftarrow Existe un flujo factible de U flujo”

Por cada unidad de flujo que sale de s a un nodo mono m_i y pasa por un instante $t \square$ podemos:

- Hacer que el mono m_i tome un sorbo en el instante $t \square$, donde $t \square$ es el nodo.

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U sorbos \leftarrow Existe un flujo factible de U flujo”

Veamos que esto respeta las restricciones:

- Se asigna hasta v_i sorbos por mono m_i porque $s \rightarrow m_i$ tiene capacidad v_i .
- Existía la arista del mono m_i al instante $t \square$ porque este ultimo esta en el intervalo de tiempo que toma el mono.
- No se asignan mas de M monos a tomar en el instante $t \square$ porque $t \square \rightarrow t$ tiene capacidad M .

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U sorbos \leftarrow Existe un flujo factible de U flujo”

La asignación que hicimos son U sorbos?

- Si, porque como había U de flujo, cada unidad representaba un sorbo, y pudimos asignar todos.

Nodos, aristas y flujo máximo en base a parámetros del problema

¿Cuántos nodos, aristas y flujo máximo tenemos?

- Nodos en cada capa:
 - Capa de monos: **N** porque es una por mono.
 - Capa de instantes de tiempo: **T**, siendo todos los instantes cubiertos por algún intervalo de tiempo de los monos.

Hay $O(N+T)$ en total.

- Aristas entre cada capa:
 - Entre s y monos: Hay **N** porque es una por mono.
 - Entre monos e instantes: Hay **N*T** máximo (si todos tienen el intervalo más grande posible).
 - Entre instantes y t: Hay **T**, porque es una por instante.

Hay $O(N*T)$ en total.

- Flujo máximo:
 - Puede salir **V** de flujo de s, siendo la suma de los v_i de los monos, porque de s a cada mono hay capacidad v_i .
 - Puede entrar **T*M** de flujo a t, porque de cada instante a t la capacidad es **M**, y hay **T**.

Hay $O(V)$ en total, considerando que buscamos satisfacer a todos los monos.

Algoritmo y complejidad para resolver modelo

¿Qué implementación del algoritmo es mejor aca?

Tenemos estas opciones:

- **Ford & Fulkerson**: $O(|E| * F)$.
 - Reemplazando tenemos $O(NT * V) = O(NTV)$.
- **Edmonds Karp**: $O(|V| |E|^2)$.
 - Reemplazando tenemos $O(NT * (NT)^2) = O(N^3 T^3)$.

Tenemos que:

- **Ford & Fulkerson** gana si $NTV \ll N^3 T^3 \leftrightarrow V \ll N^2 T^2$.
- **Edmonds Karp** en caso contrario.



EASY

OH
NO!



HA-HA!



¿Esto esta bien?

El modelo no está mal diseñado, funciona bien la idea.

El problema es otro, la complejidad final.

Recordemos los números del enunciado acerca de las variables:

- Cantidad de monos: $1 \leq N \leq 100$.
- Agua a tomar e intervalos de tiempo de cada mono: $0 \leq v_i, a_i, b_i \leq 50000$.
- Cantidad de monos tomando a la vez: $1 < M \leq 5$.

Ahora reemplazamos esto en las 2 posibles complejidades como peor caso.

¿Esto esta bien?

Considerando en el peor caso que:

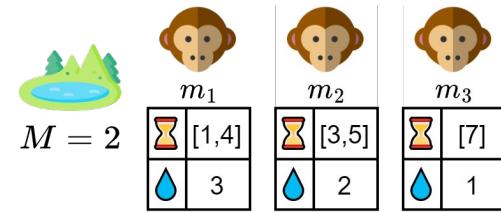
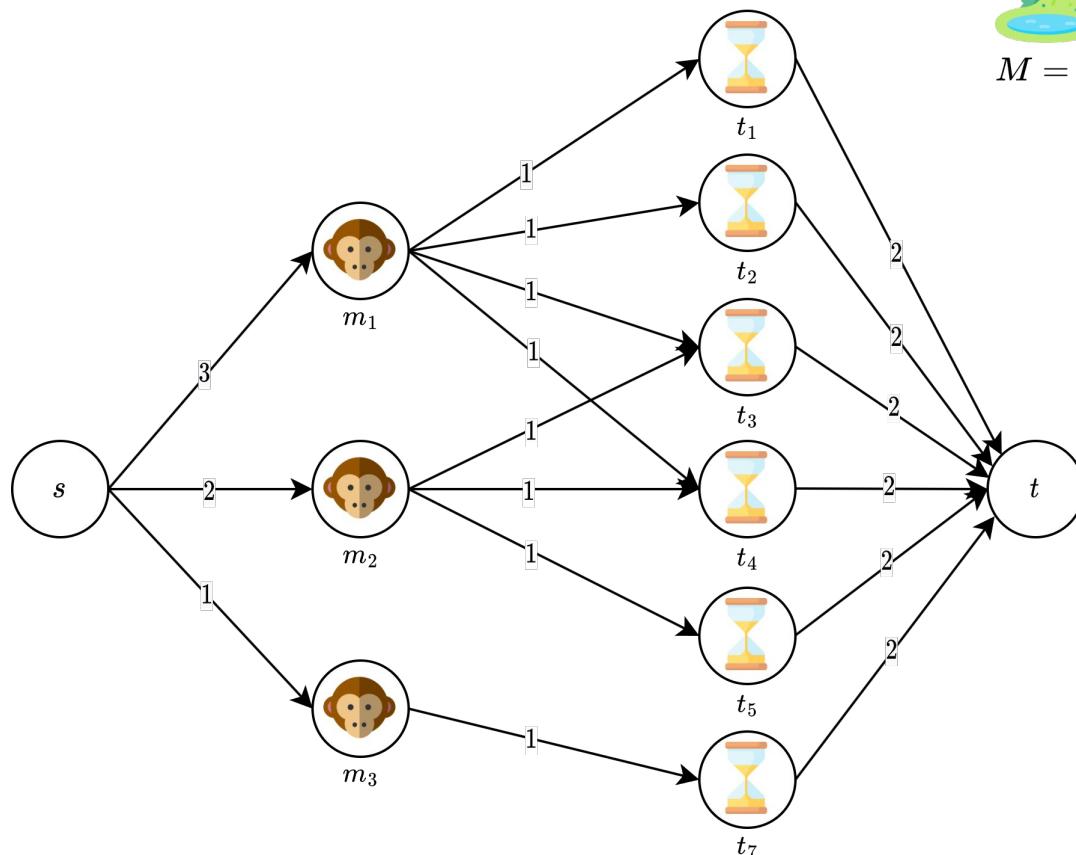
- $N = 100$.
- $M = 5$.
- $T = 50000$ (todo el tiempo disponible).
- $V = 100 * 50000$ (todos los monos quieren tomar lo máximo).

Nos queda:

- **Ford & Fulkerson:** $NTV = 100 * 50000 * (100 * 50000) = 25 * 10^{12} = \text{💀}$
- **Edmonds Karp:** $(TN)^3 = (50000 * 100)^3 = 5 * 10^{18} = \text{💀}$

R.I.P 

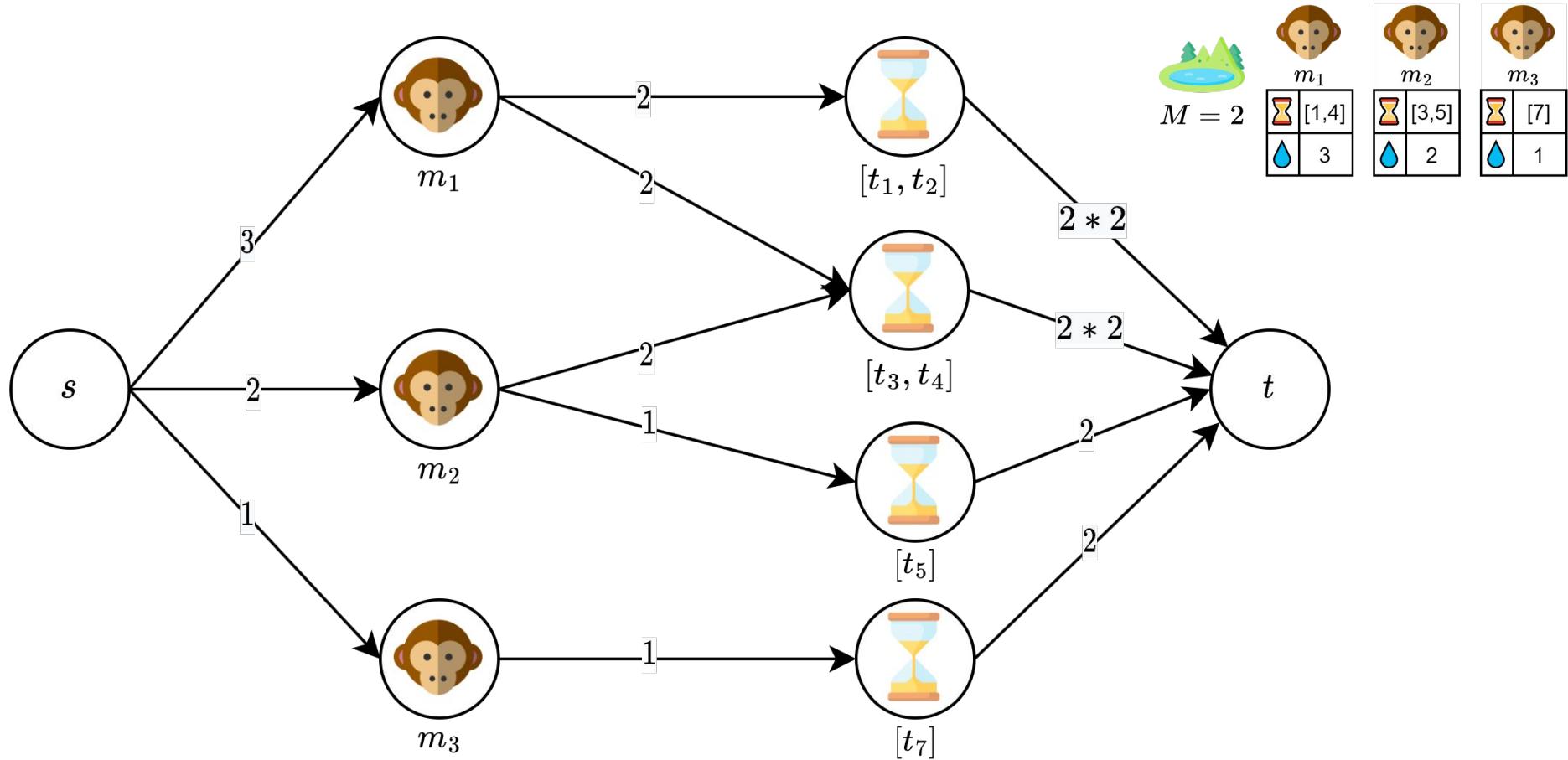
Pensemos un segundo si se nos pasó algo...



Pista:

Nos está arruinando que haya un nodo por instante de tiempo ¿Se podrán colapsar?

Colapsando instantes en intervalos sin cambios de monos



Recalculemos nodos, aristas y flujo máximo

- Nodos en cada capa:
 - Capa de monos: **N** porque es una por cliente.
 - **Capa de intervalos de instantes de tiempo: $2N$, porque a lo sumo cada intervalo de tiempo que contiene un conjunto de monos cambia cada vez que un nuevo mono se puede sumar o cuando uno existente se va.**

Hay $O(N)$ en total.

- Aristas entre cada capa:
 - Entre s y monos: Hay N porque es una por mono.
 - **Entre monos e instantes: Pongamos una cota de $N * 2N$ irreal (todos los monos a todos los intervalos).**
 - **Entre intervalos de instantes y t : Hay $2N$, porque es una por intervalo.**

Hay $O(N^2)$ en total.

- Flujo máximo:
 - Puede salir V de flujo de s , siendo la suma de los v_i de los monos, porque de s a cada mono hay capacidad 1.
 - **Puede entrar $T*M$ de flujo a t , siendo T los instantes cubiertos por algún mono.**

Hay $O(V)$ en total, considerando que buscamos satisfacer a todos los monos.

Retomando competencia de algoritmos

Tenemos estas opciones:

- **Ford & Fulkerson:** $O(|E| * F)$.
 - Reemplazando tenemos $O(N * V) = O(NV)$.
- **Edmonds Karp:** $O(|V| E^2)$.
 - Reemplazando tenemos $O(N * N^4) = O(N^5)$.

Podemos usar **Edmonds Karp**, porque **Ford & Fulkerson** queda atrás dado que se cumple que $NV \gg N^4 \leftrightarrow V \gg N^3$, y V puede ser N^*50000 en el peor caso, por lo que $N^*50000 \gg N^3 \leftrightarrow 50000 \gg N^2$.

Notemos que la complejidad se redujo a $O(N^5) = 10^{10}$. Con esto vemos la importancia de iterar un modelo y refinarlo.

EXTRA: Esto se podría mejorar aún más usando el algoritmo de **Dinic**, que tiene complejidad $O(|V|^2 E)$, y nos dejaría un $O(N^4) = 10^8 = 😊$.

Espera... no terminamos, ¿El modelo es correcto?



¿Cómo demuestro que este modelo es equivalente?

Aca tenemos, por ejemplo, 2 opciones:

1. Retomamos la misma demostración que antes, pero “parcheandola” acorde a los cambios que hubo en el modelo.
2. Demostramos equivalencia entre el modelo viejo y este, y nos sale gratis que el modelo representa al problema, porque:

“Existe una forma de asignar U sorbos ↔

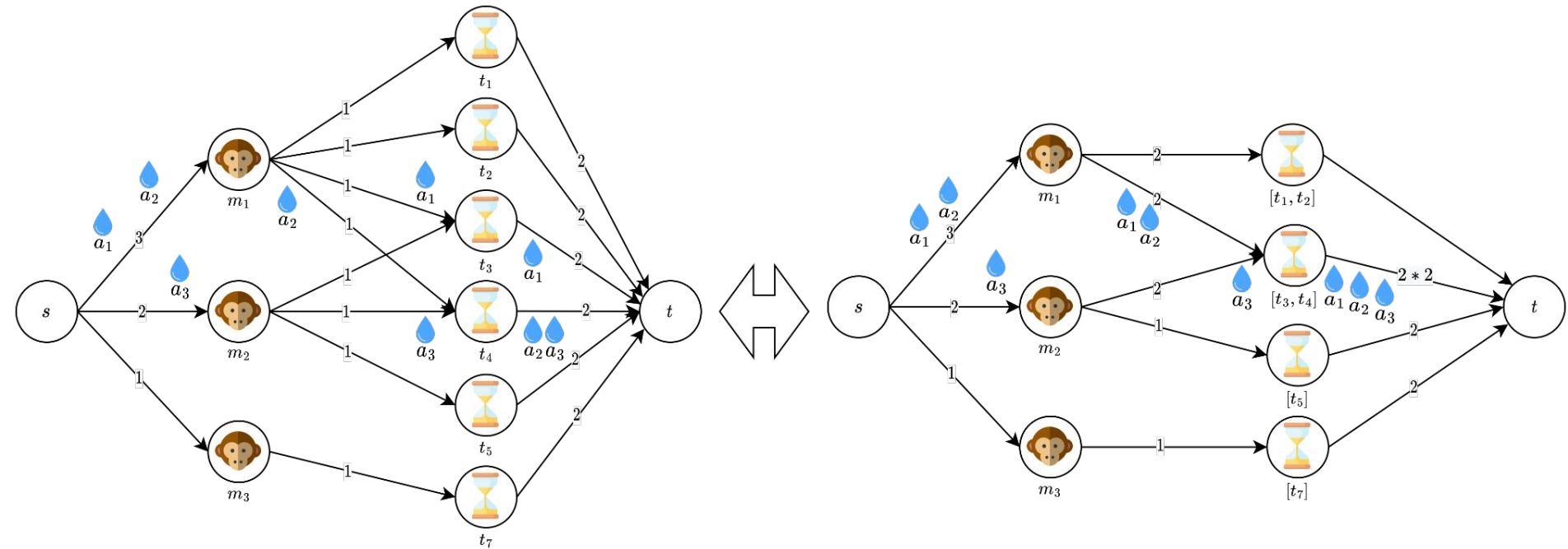
Existe un flujo factible de U flujo en el modelo viejo ↔

Existe un flujo factible de U flujo en el modelo nuevo”

Donde el primer sii ya lo tenemos, y solo tenemos que probar el segundo.

Veamos un ejemplo visual de esto último y pensemos en una posible demostración.

Mostrando la conexión entre ambos modelos



Interpretando la conexión

Existe un flujo factible de U flujo en el modelo viejo \leftrightarrow

Existe un flujo factible de U flujo en el modelo nuevo”

Podemos pensar que colapsamos todos los nodos del modelo que eran indistinguibles (mismos vecinos y hasta misma capacidades de las aristas).

Todo el flujo que pasaba por un conjunto de nodos que colapsó en 1, lo puedo hacer que pase por este.

- Funciona porque si pasaba X de flujo en total seguro que al colapsarlo, como las capacidades se suman, puede pasar esa cantidad de flujo.

Lo mismo puedo hacer a la inversa.

- Como el conjunto de nodos separados suman una capacidad igual al individual, puedo distribuir el flujo entre ellos, y se que siempre voy a poder por como es la relación de las capacidades.

Arca de Noé



Inspirado en segundo parcial 1er cuatrimestre 2023

Enunciado

Enunciado

¡Se viene una gran inundación y tenemos **A** animales **a_i** que están en peligro!

Por suerte nos adelantamos y compramos el **Arca de Noé** por , que cuenta con **R** habitaciones **r_j**, y cada una puede tener **c(r_j)** animales.

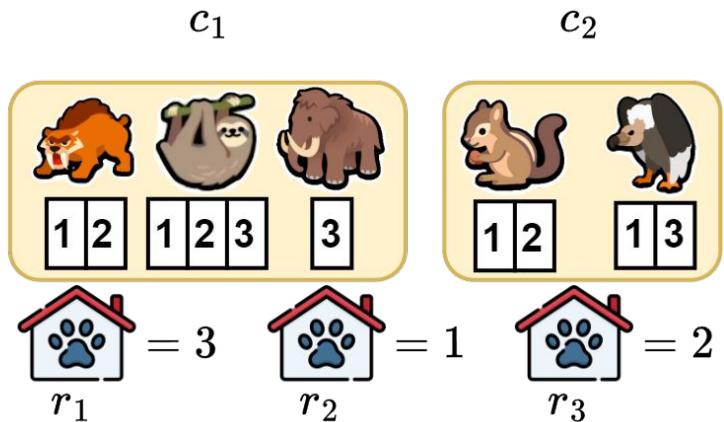
Si bien el espacio de cada habitación ya era un problema, tenemos otros adicionales:

- Cada animal **a_i** es exquisito, y solo admite ser ubicado en un conjunto de habitaciones.
- Hay **C** conflictos **c_{ij}** entre animales, lo cual implica que no podemos ubicarlos nunca en la misma habitación (porque bueno... se matan 😅, y el objetivo era salvarlos). Siempre un animal **a_i** pertenece a solo un conflicto **c_{ij}**.

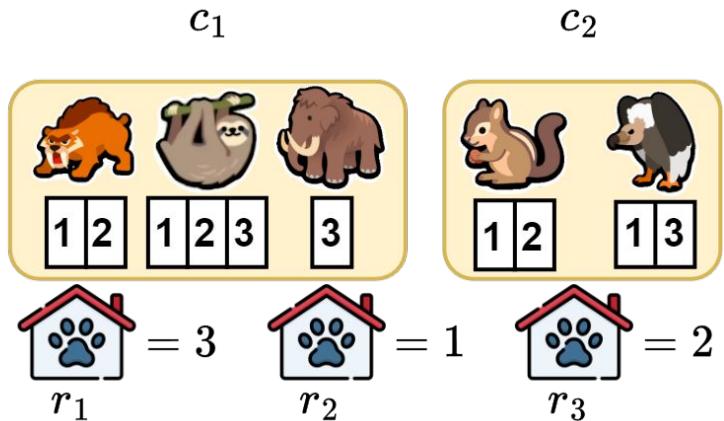
¿Nos ayudas a encontrar una forma de ubicarlos a todos en caso de ser posible?

Ejemplo

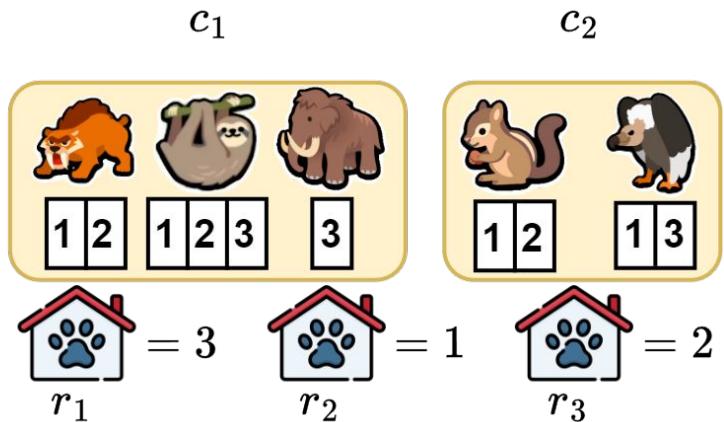
Ejemplo: Invalido por habitación



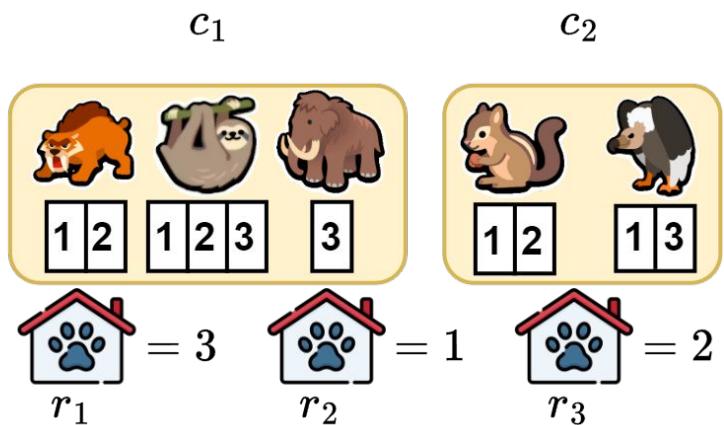
Ejemplo: Invalido por tamaño



Ejemplo: Invalido por conflicto



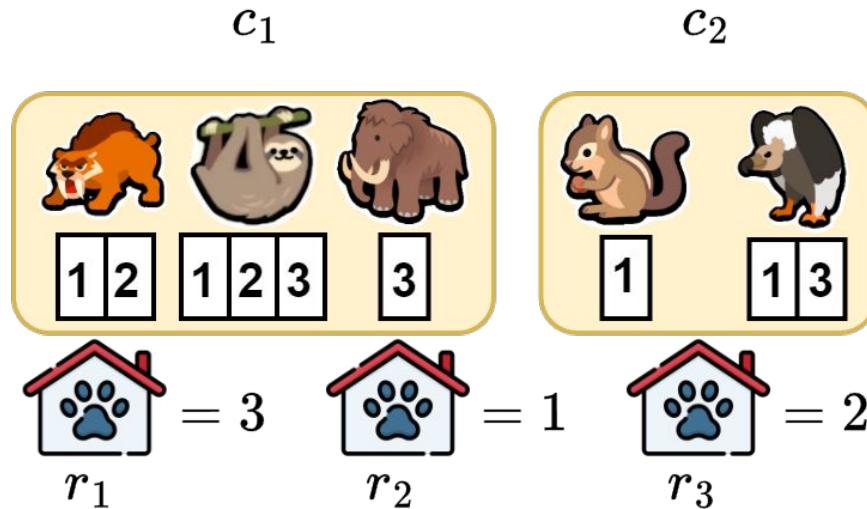
Ejemplo: Valido



Resolución

Datos de ejemplo para la resolución

Para modelar la red vamos a utilizar estos animales, conflictos y habitaciones:



Modelo de red

¿Qué queremos asignar?

Tenemos un **Arca de Noé**, donde los animales se ubican en habitaciones.

Lo que queremos asignar es un animal **a_i** a una habitación **r_j**.

¿Qué entidades están involucradas en esta arca?

Las entidades involucradas son:

- Animales.
- Habitaciones.
- ¿Conflictos?



Animal



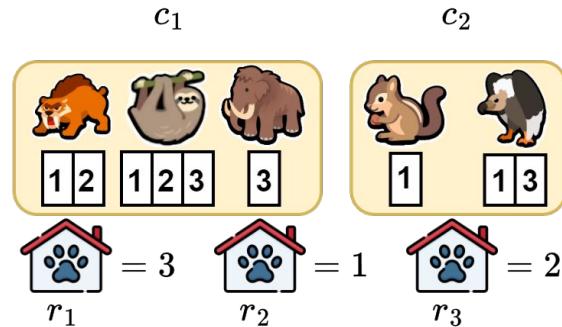
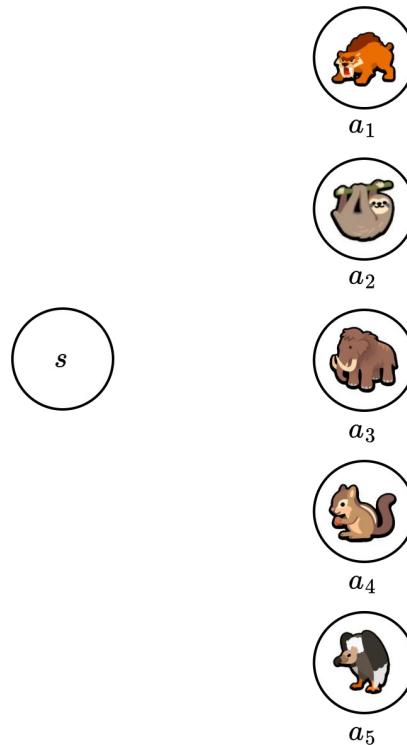
¿Conflictos?



Habitacion

Entidades como capas del modelo

Pongamos las entidades anteriores en capas para nuestro modelo de red, junto con los nodos **s** y **t**.



¿Qué restricciones tenemos?

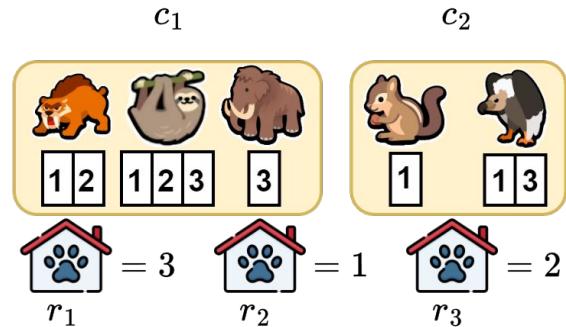
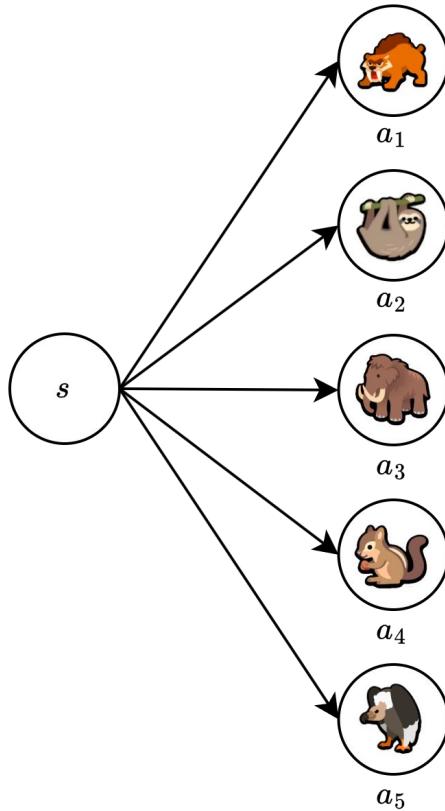
Pensando en las restricciones, podríamos decir que:

1. Cada animal a_i puede ubicarse una sola vez.
2. Cada animal a_i puede ubicarse en posibles habitaciones r^{\square} .
3. Cada habitación r_i tiene una capacidad máxima de animales.
4. No puede haber más de 1 animal a_i del mismo conflicto c^{\square} en una habitación r^{\square} .

Veamos cómo incluir esto en nuestro modelo.

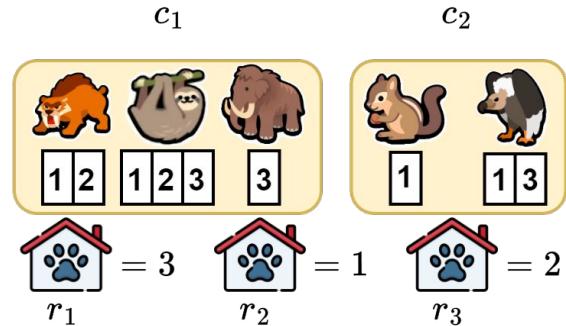
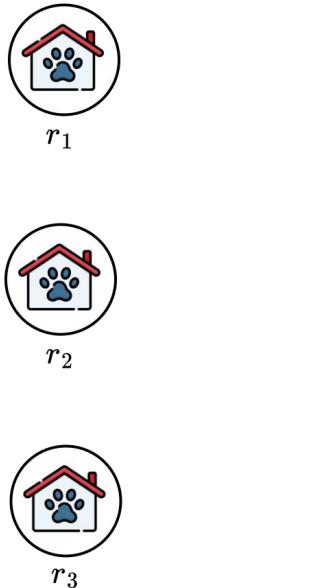
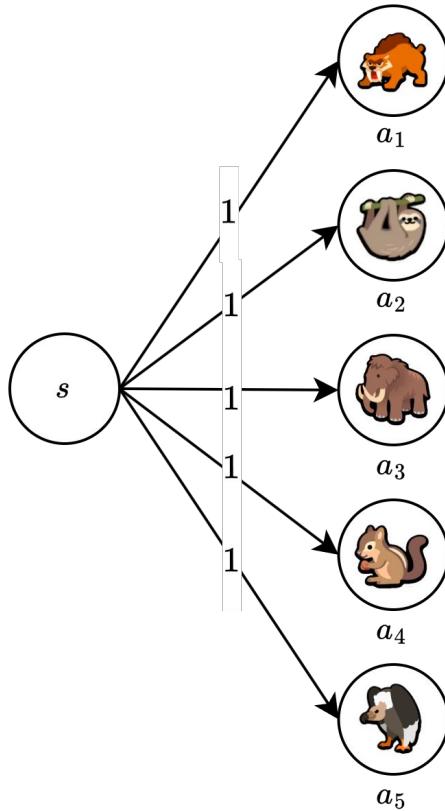
Conexiones

Cada animal a_i puede ubicarse una sola vez.



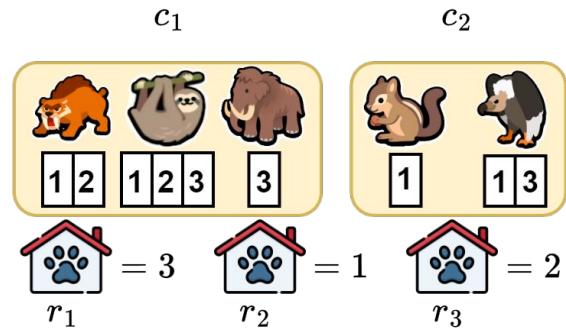
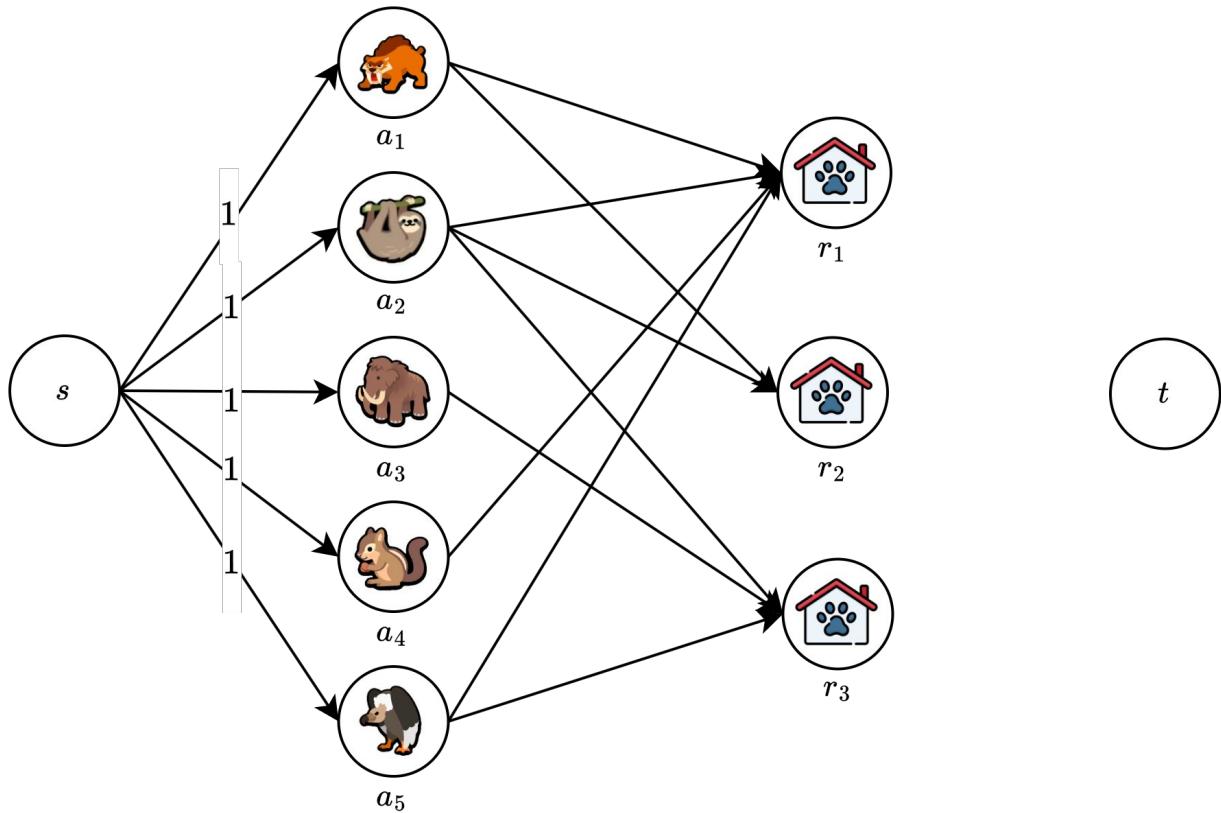
Restricciones

Cada animal a_i puede ubicarse una sola vez.



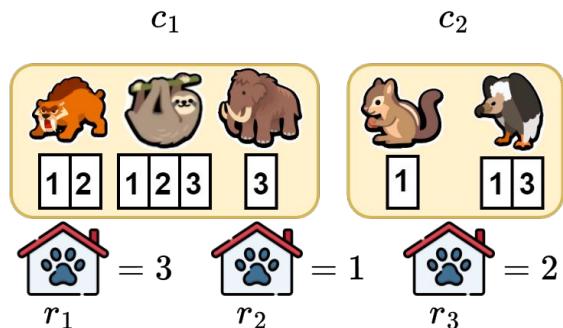
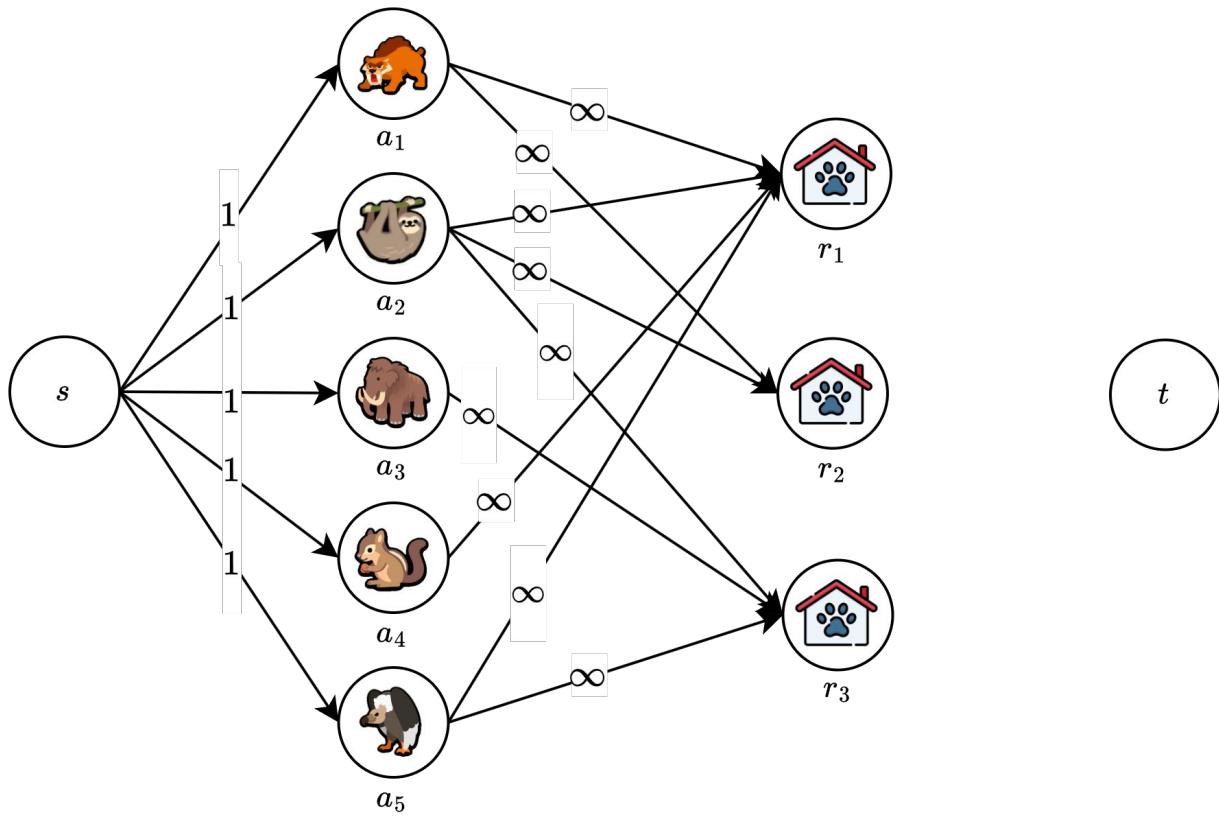
Conexiones

Cada animal a_i puede ubicarse en posibles habitaciones $r \square$.



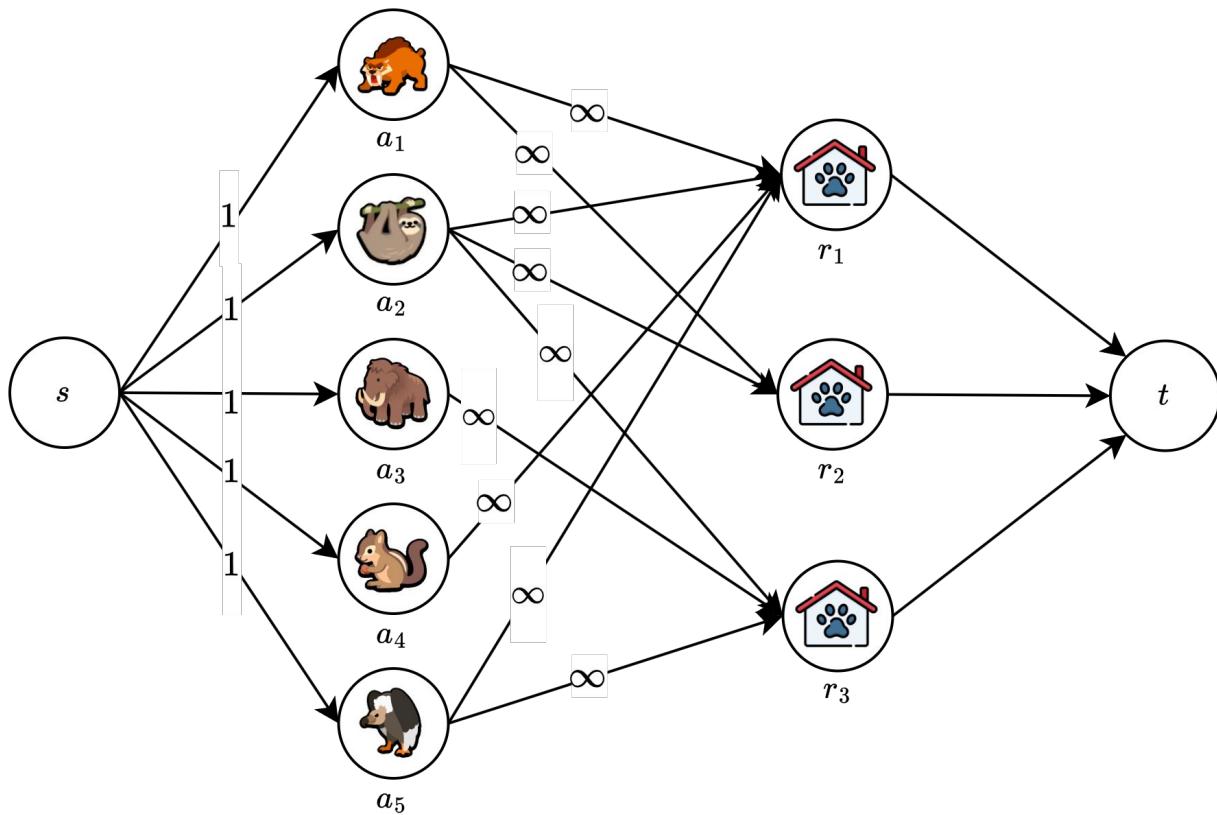
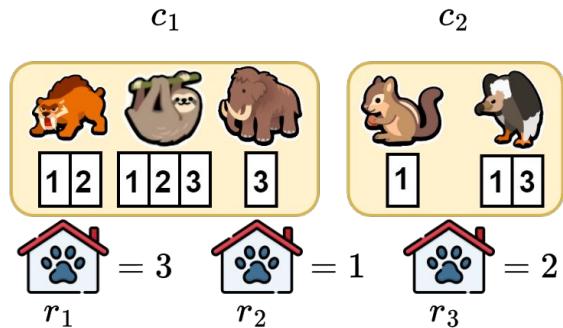
Restricciones

Cada animal a_i puede ubicarse en posibles habitaciones $r \square$.



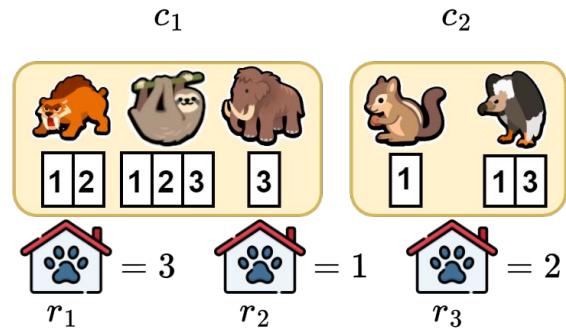
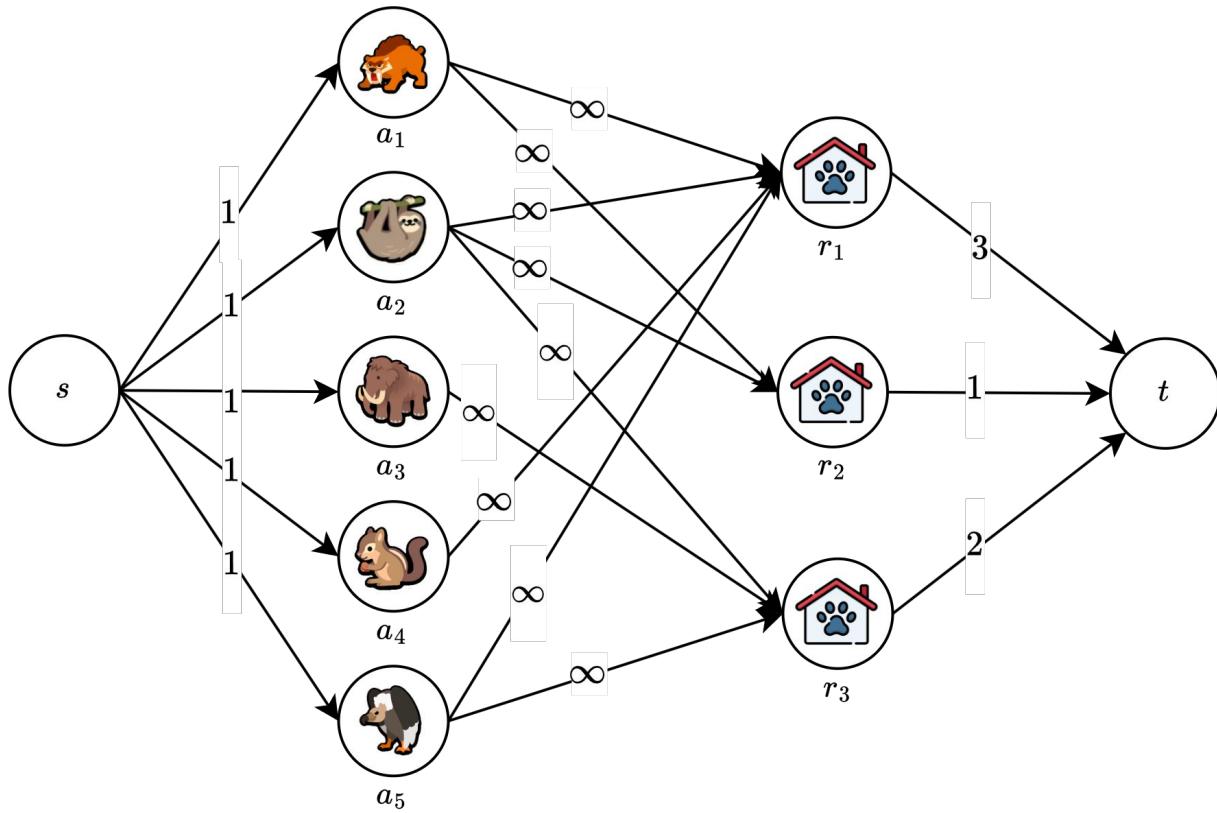
Conexiones

Cada habitación r_i tiene una capacidad máxima de animales.



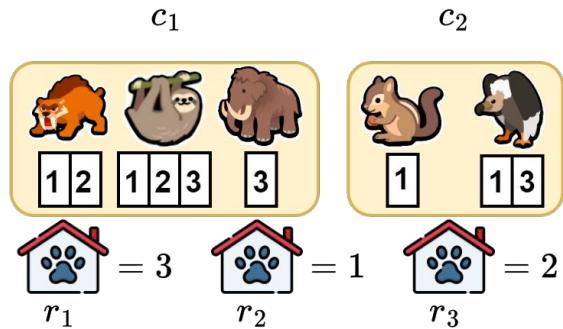
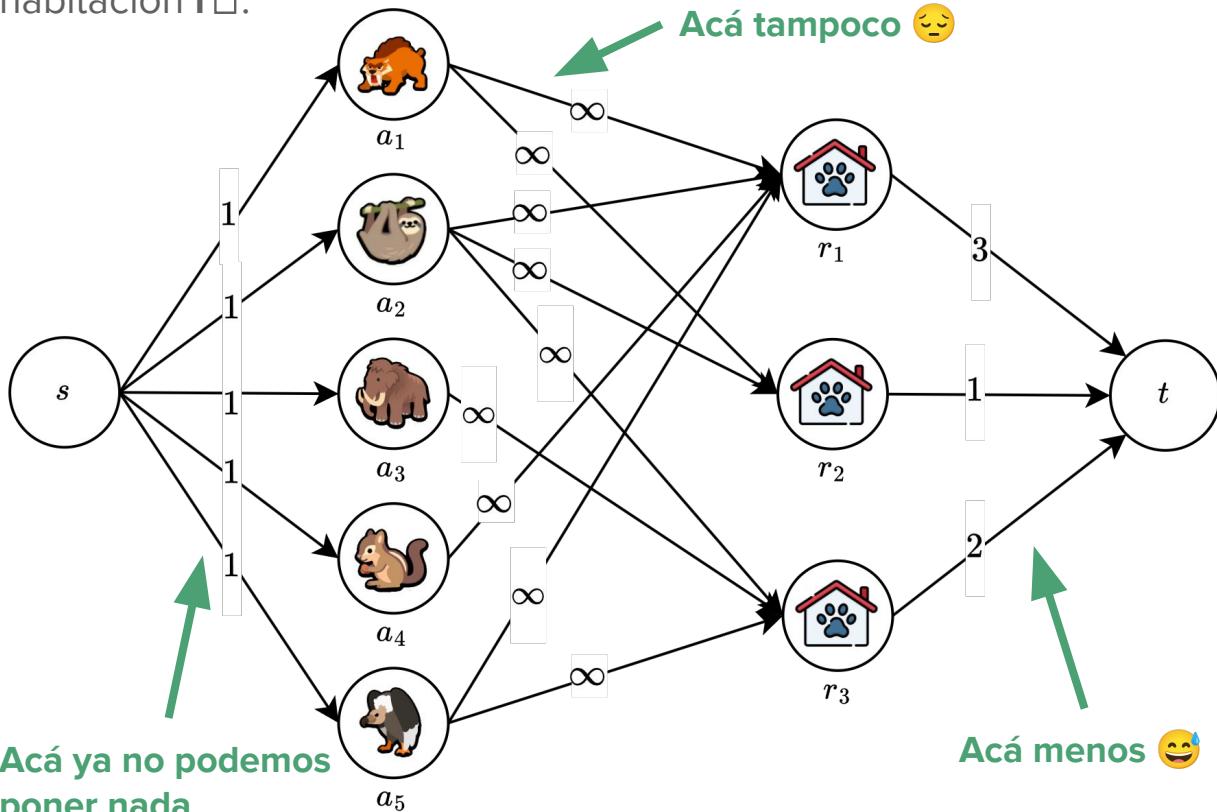
Restricciones

Cada habitación r_i tiene una capacidad máxima de animales.



Conexiones

No puede haber más de 1 animal a_i del mismo conflicto c_i en una habitación r_i .





Tip #8

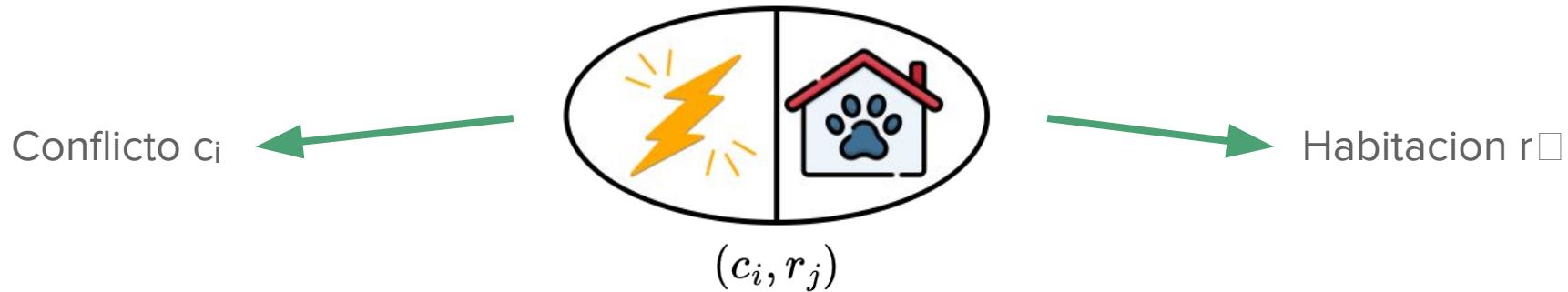
Nodos como combinaciones de entidades



Nodos como combinaciones de entidades

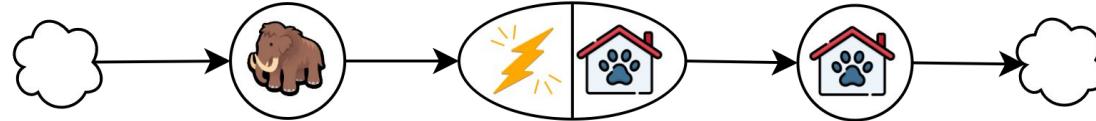
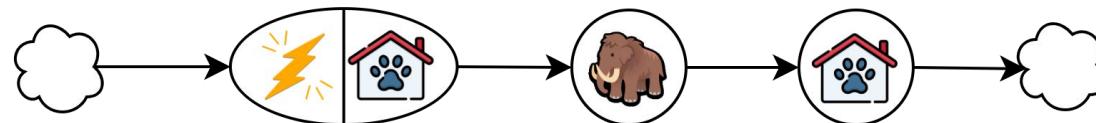
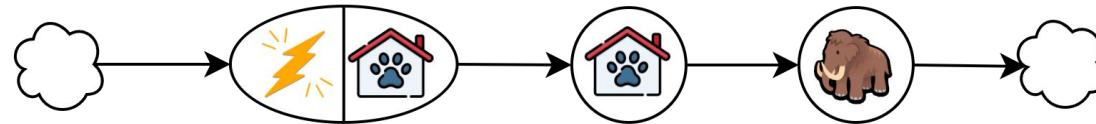
Puede pasar a veces que no sea suficiente con las entidades concretas del problema como capas para modelar las restricciones, sino que tenemos que hacer alguna combinación entre ellas, o con entidades abstractas.

En este caso, podríamos representar el conflicto (**entidad abstracta**) en conjunto con una cierta habitación (**entidad concreta**).

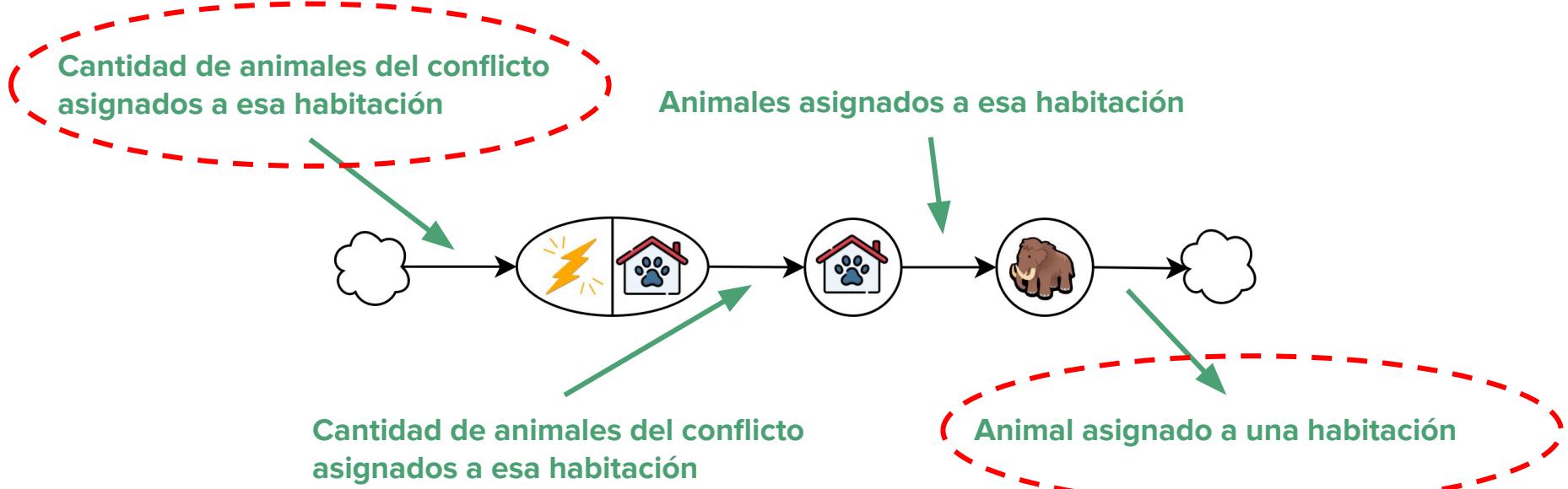


¿Nuevo orden de capas?

Tenemos que pensar en que parte ubicamos la nueva capa, para lo cual tenemos 3 opciones (el resto son las mismas pero invertidas):

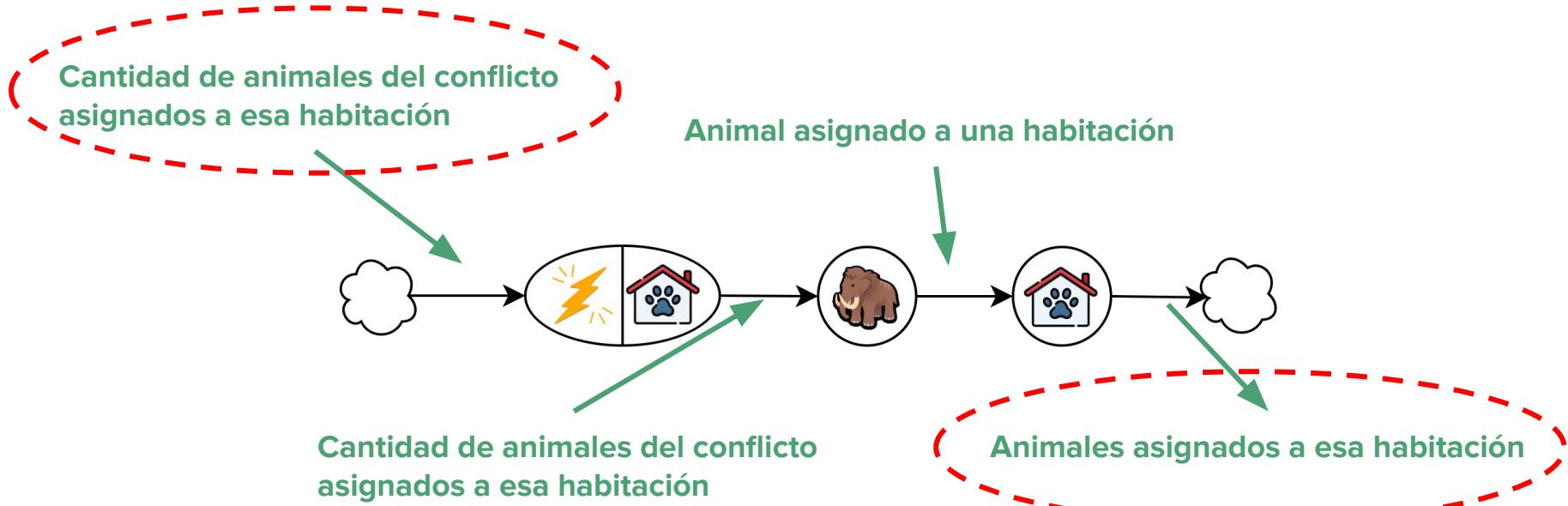


Orden 1: (conflicto, habitación) → habitación → animal



La unidad de flujo empezó representando un animal de un conflicto **asignado a una habitación X**, y pudo terminar representando **otro animal que no pertenece al conflicto original**.
Esto pasó porque la información que salió de la capa de la habitación solo me decía **la cantidad de animales asignados ahí, y no de qué conflicto eran**.

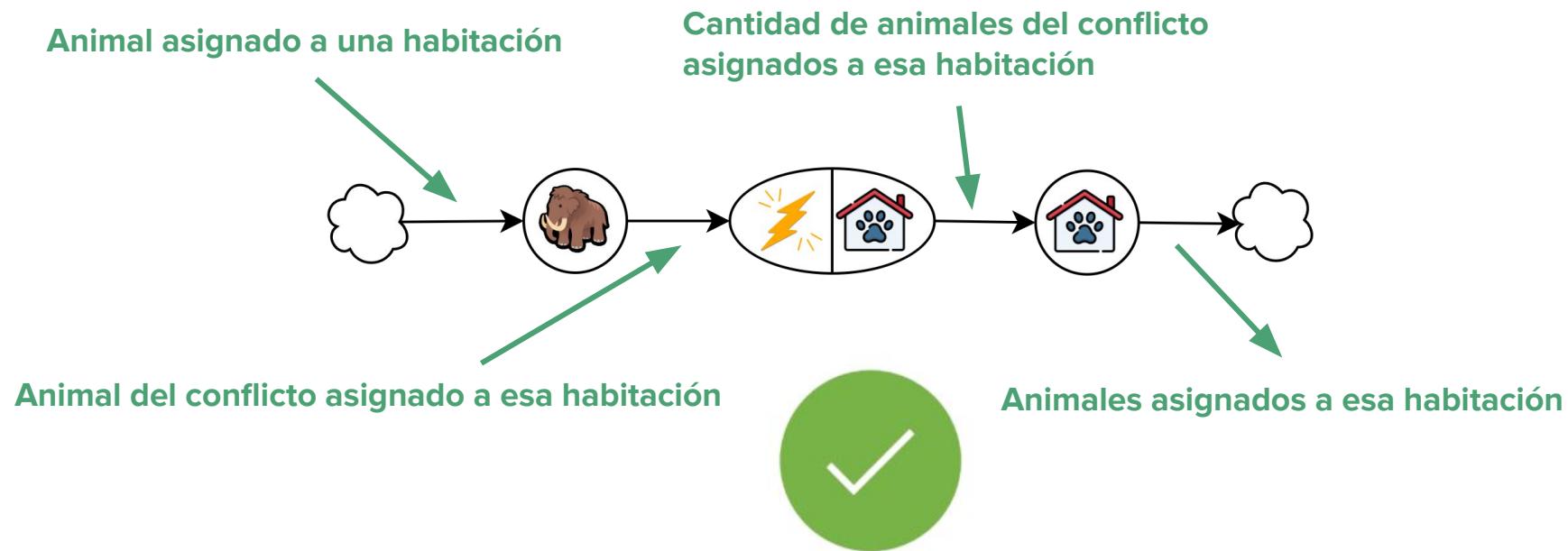
Orden 2: (conflicto, habitación) → animal → habitación



La unidad de flujo empezó representando un animal de un conflicto **asignado a una habitación X**, y pudo terminar representando al mismo animal **pero asignado a otra habitación Y**.

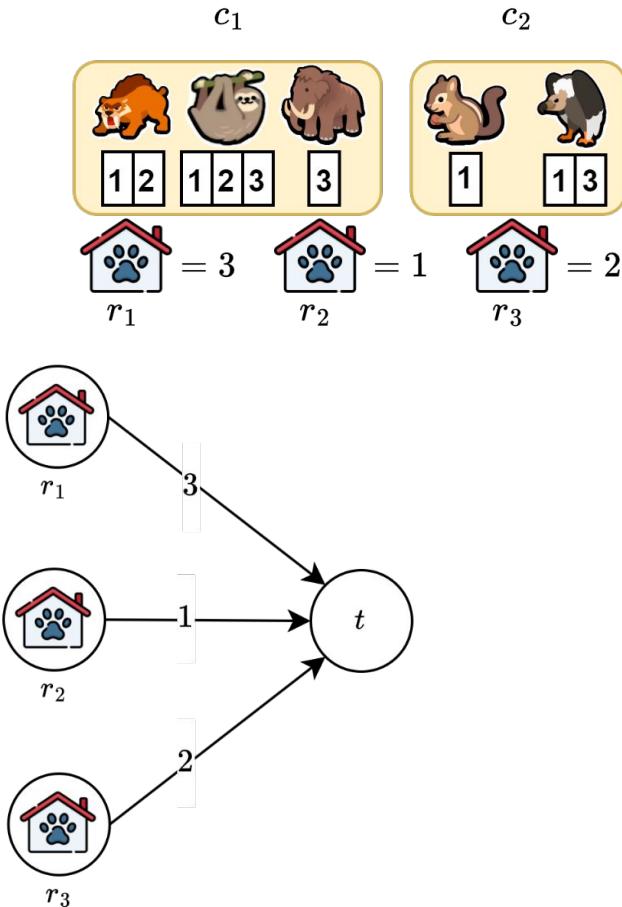
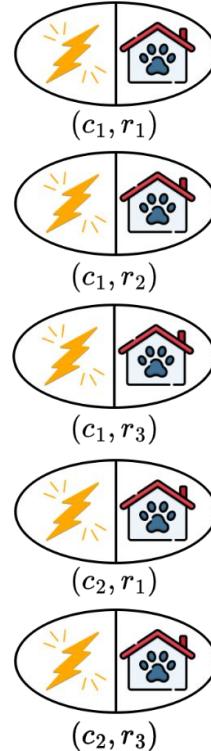
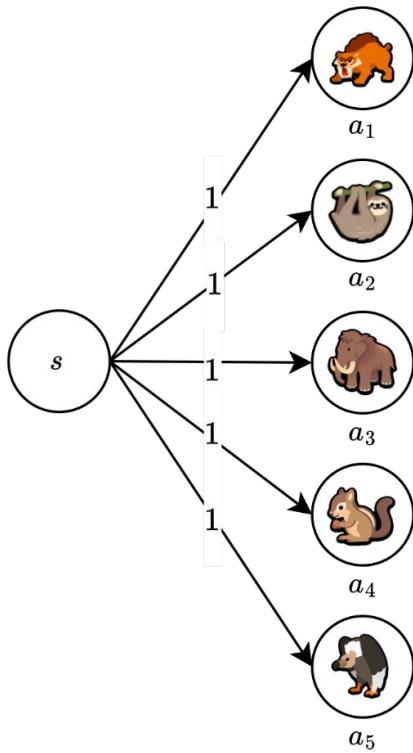
Esto pasó porque la información que salió de la capa del animal solo me decía que **el animal fue asignado a alguna habitación, pero no a cual**.

Orden 3: animal → (conflicto, habitación) → habitación



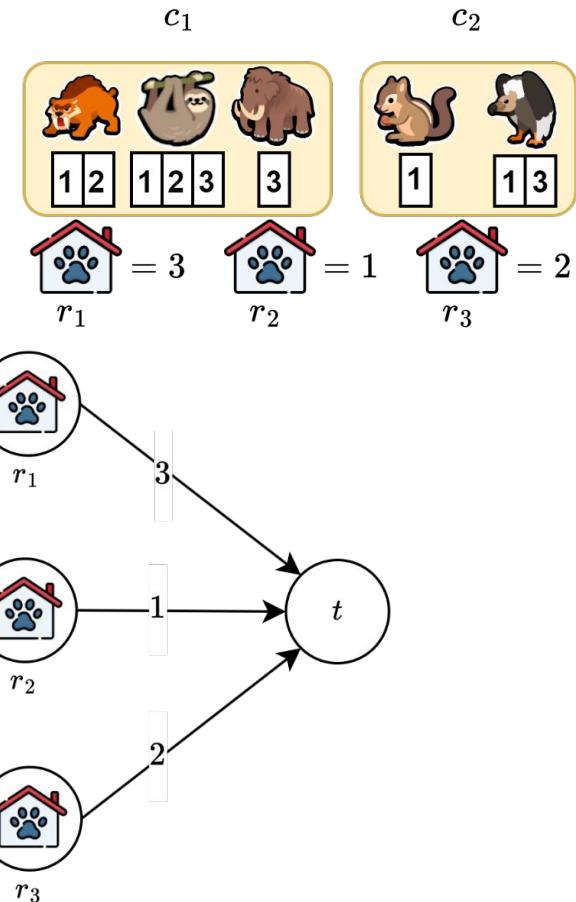
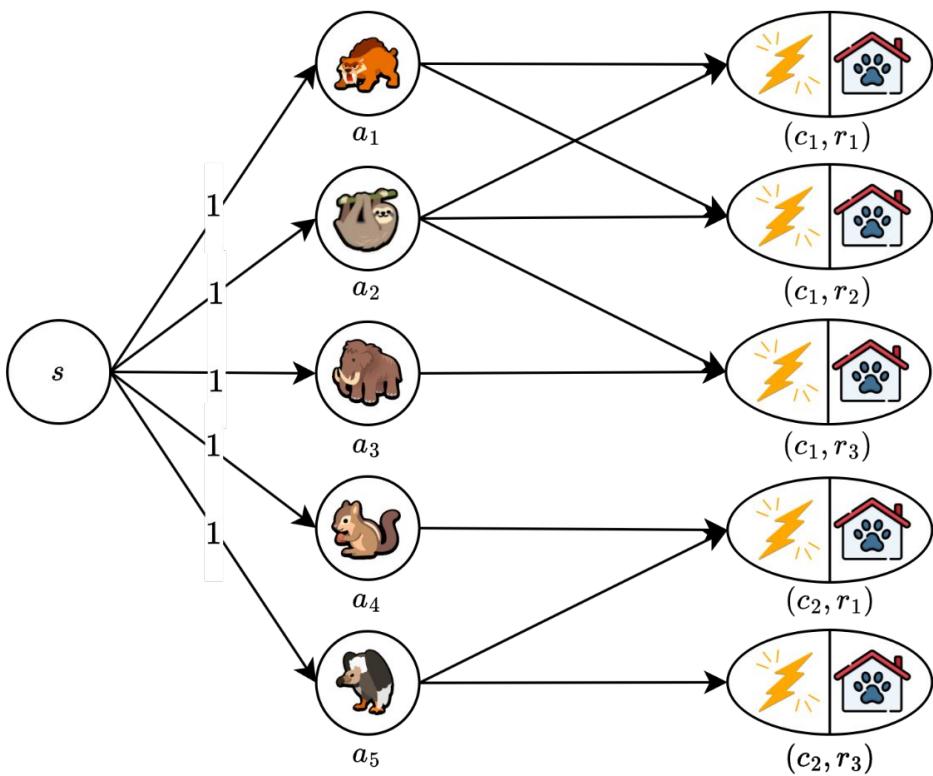
Retomando...

Ahora que introdujimos esta nueva capa en el lugar correcto, tenemos que repensar cómo quedan las conexiones y capacidades.



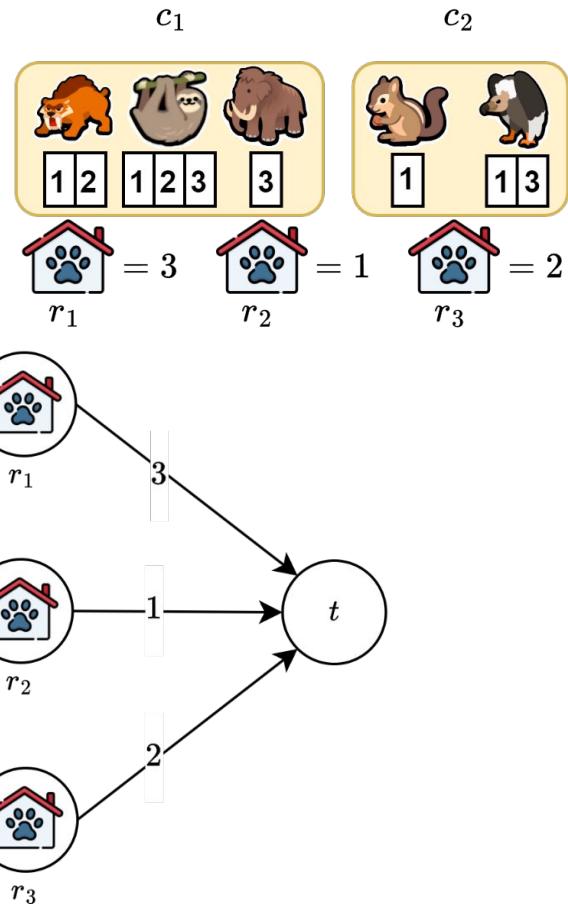
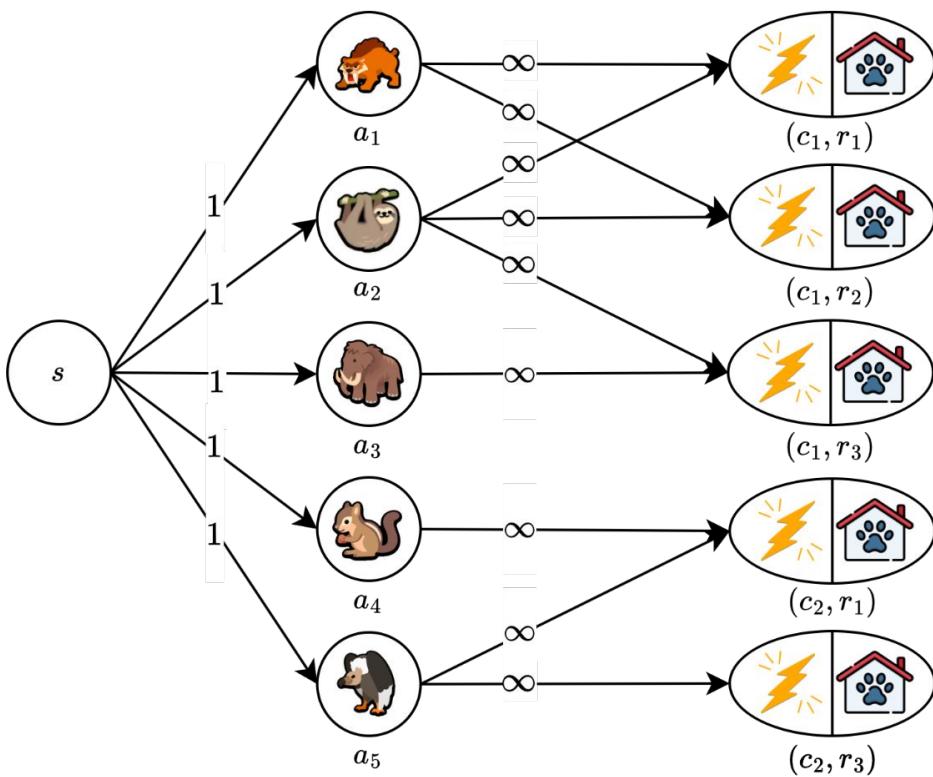
Restricciones

Cada animal a_i puede ubicarse en posibles habitaciones $r \square$.



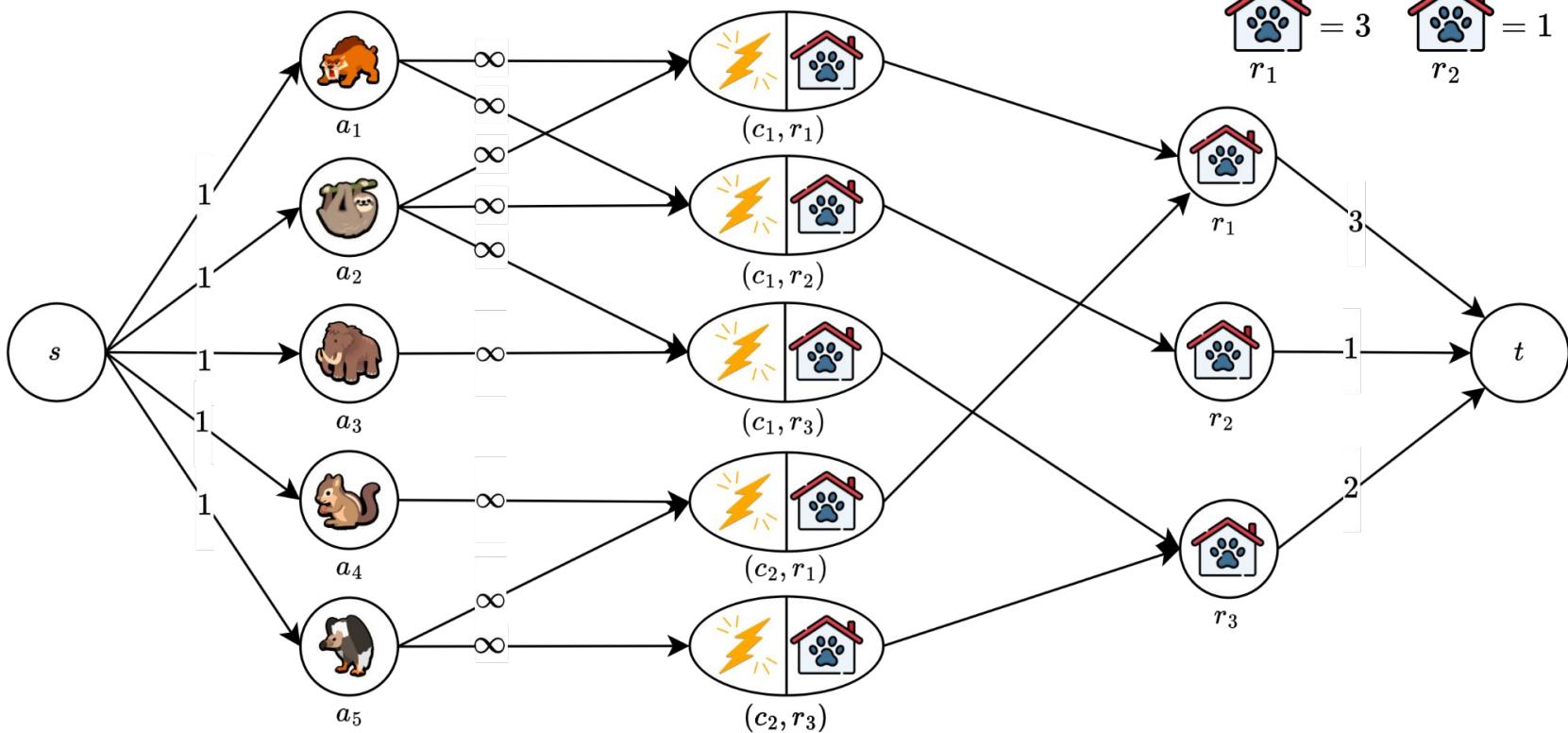
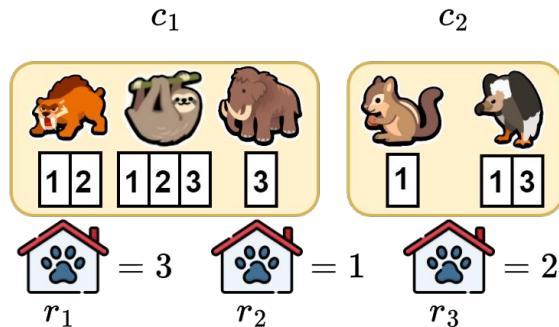
Conexiones

Cada animal a_i puede ubicarse en posibles habitaciones $r \square$.



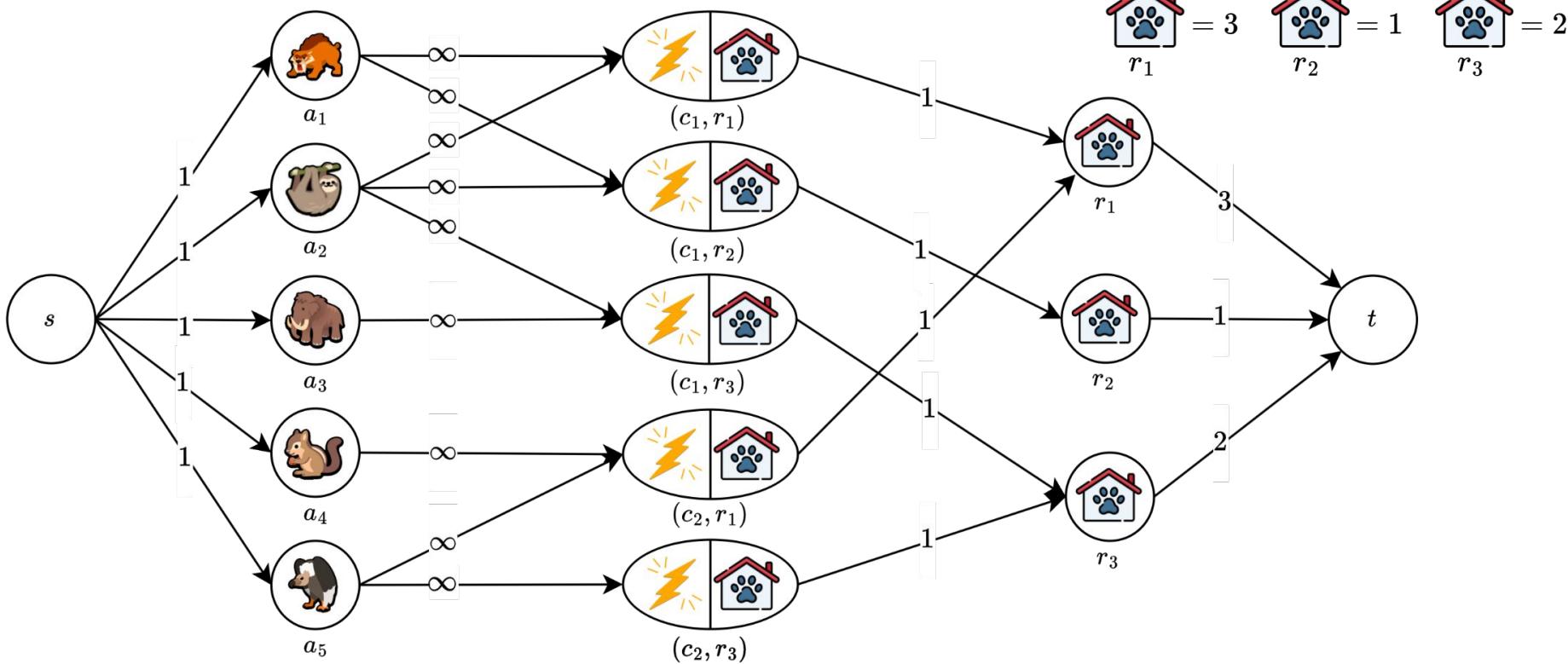
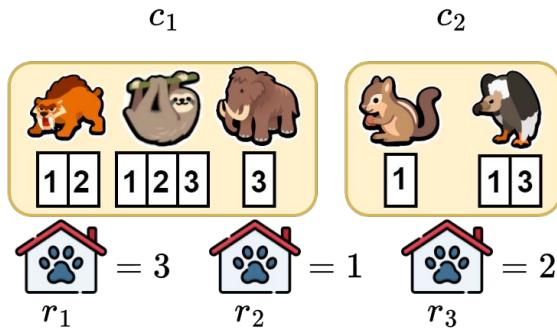
Conexiones

No puede haber más de 1 animal a_i del mismo conflicto c_i en una habitación r_j .



Restricciones

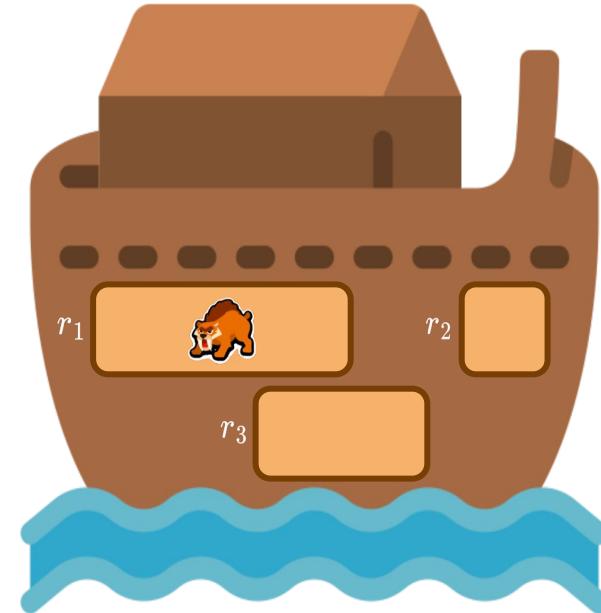
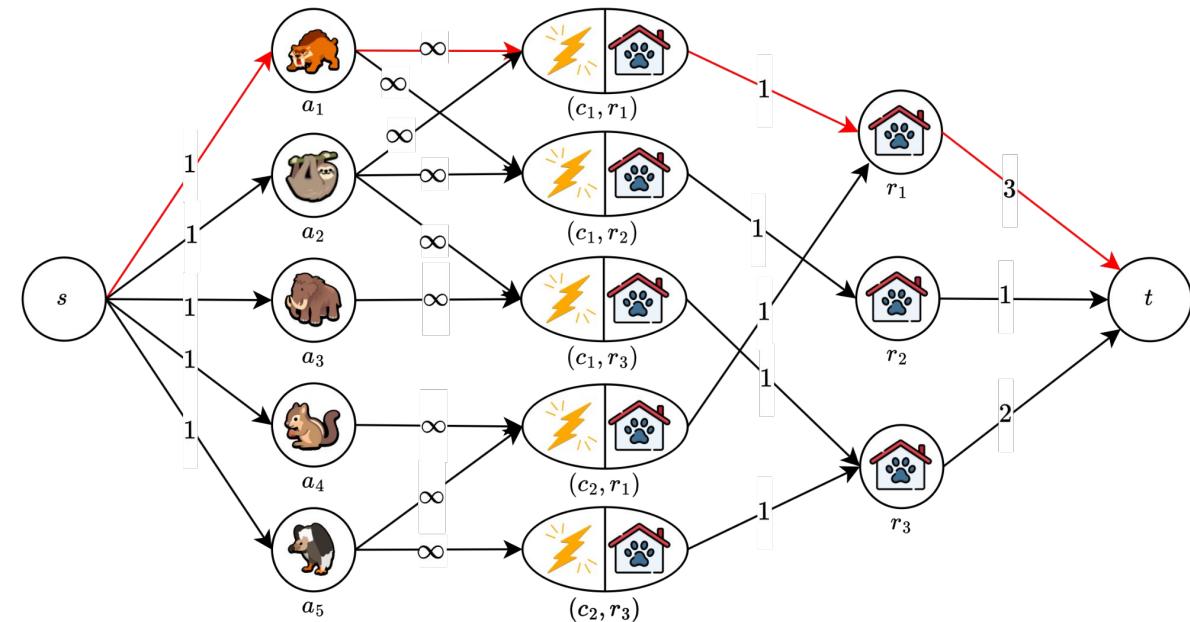
No puede haber más de 1 animal a_i del mismo conflicto c_j en una habitación r_k .



Semántica de unidad de flujo

¿Cómo interpretamos a la unidad de flujo en la red?

Es una asignación de un animal a_i a una habitación $r \square$.



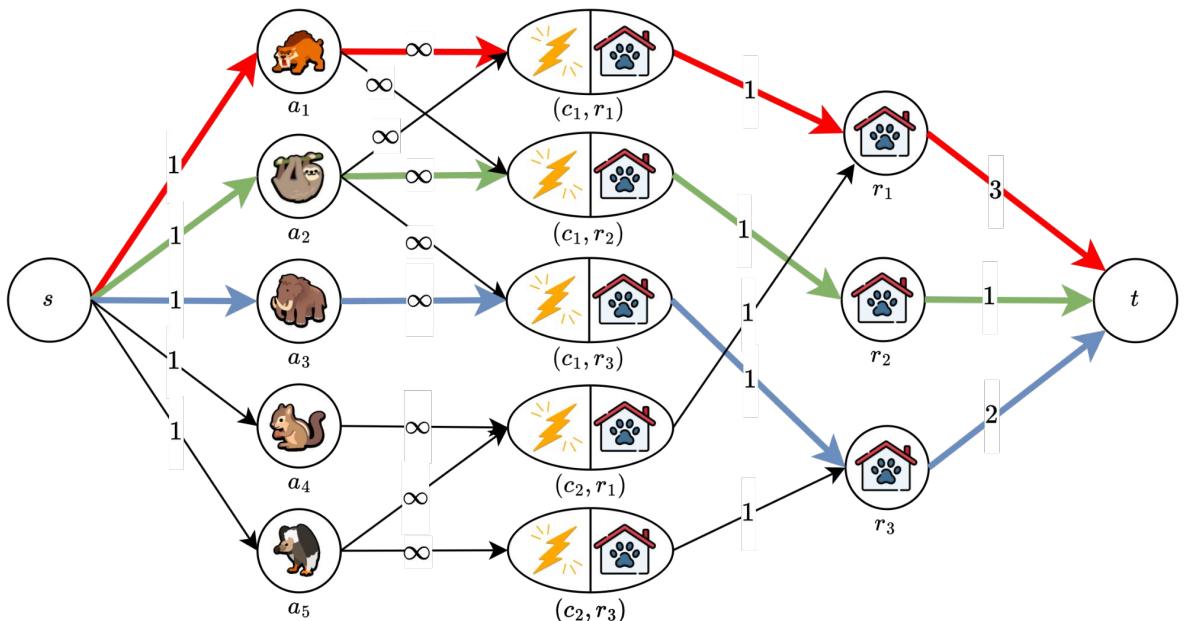
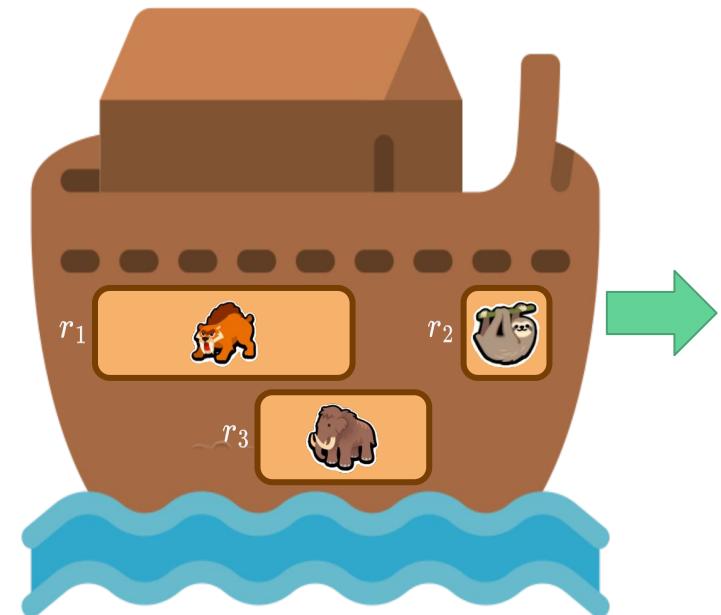
Conexión Problema ↔ Modelo

Mostrando la conexión entre el modelo y el problema

Vamos a querer demostrar esta afirmación:

“Existe una forma de asignar U animales cumpliendo las restricciones \leftrightarrow Existe un flujo factible de U flujo”

Demosmos cada implicación, guiándonos con un gráfico de cada caso.



Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U animales cumpliendo las restricciones

→ **Existe un flujo factible de U flujo”**

Por cada animal a_i perteneciente al conflicto c_i asignado a una habitación r_i construimos un camino de flujo 1 de la siguiente forma:

- De s al nodo del a_i .
- Del nodo a_i al nodo (c_i, r_i) .
- Del nodo (c_i, r_i) al nodo r_i donde pertenece.
- Del nodo del r_i al nodo t .

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U animales cumpliendo las restricciones → Existe un flujo factible de U flujo”

¿El flujo respeta capacidades?

- $s \rightarrow a_i$: Siempre se asigna 1 máximo porque cada animal se ubica una vez.
- $a_i \rightarrow (c_r, r)$: La arista existe porque la ubicación es válida, y asignamos menos que ∞ .
- $(c_r, r) \rightarrow r$: La arista existe porque coinciden en la habitación r , y hay como máximo 1 unidad porque es una ubicación válida y no pueden haber 2 animales del mismo conflicto c_r en la habitación r .
- $r \rightarrow t$: Hay máximo $c(r)$ unidades, porque la ubicación es válida y no puede tener más de esa cantidad de animales la habitación r .

¿El flujo respeta la conservación?

- Por cada unidad de flujo que llega a un animal asignado a_i , sale para (c_r, r) .
- Idem de (c_r, r) con r .
- Idem de r con t .

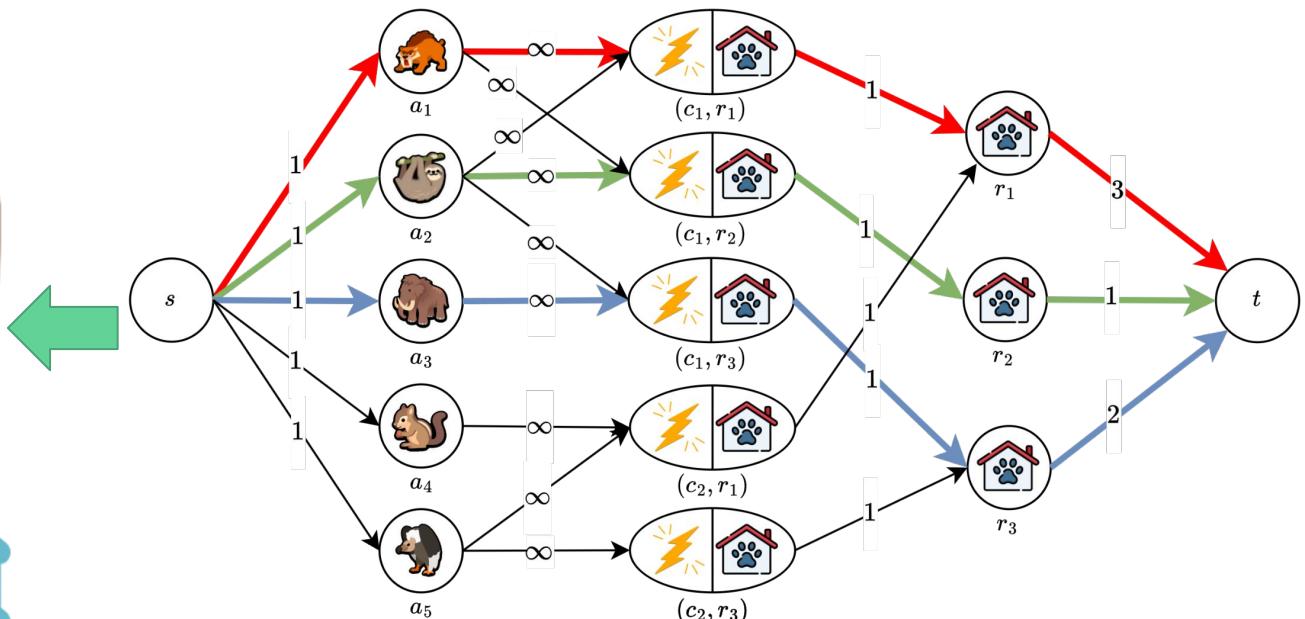
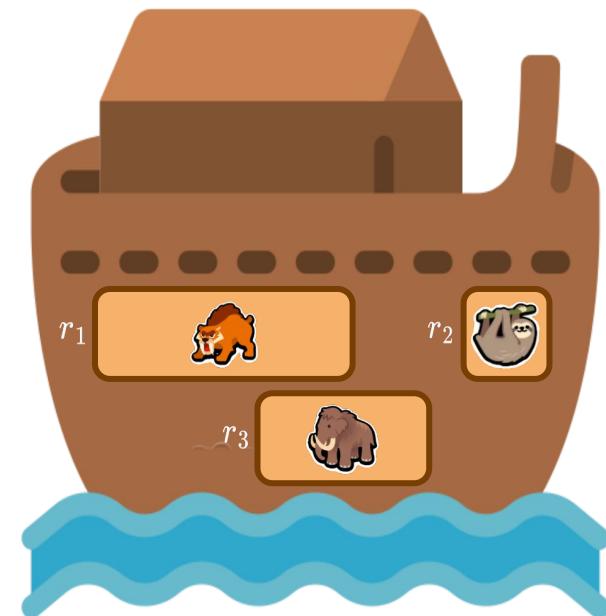
Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U animales cumpliendo las restricciones

→ Existe un flujo factible de U flujo”

La asignación de flujo que hicimos es \mathbf{U} ?

- Si, porque como la asignación era válida, había U animales, asignamos U unidades de flujo saliendo desde s , una por cada animal.



Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U animales cumpliendo las restricciones ← Existe un flujo factible de U flujo”

Por cada unidad de flujo que sale de s a un nodo a_i y pasa por (c_i, r_i) podemos:

- Asignar el animal a_i a r_i .

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U animales cumpliendo las restricciones ← Existe un flujo factible de U flujo”

Veamos que esto respeta las restricciones:

- Solo se asigna 1 vez cada animal a_i porque el flujo es factible.
- El animal a_i se asigna a una habitación r_j válida porque solo existe la arista si a_i podía ser ubicado en r_j .
- No se asignan más de $c(r_j)$ animales a un r_j porque desde el nodo r_j a t hay capacidad $c(r_j)$.
- No se asigna más de 1 animal del conflicto c_i a la habitación r_j porque desde el nodo (c_i, r_j) a r_j está esa misma capacidad.

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U animales cumpliendo las restricciones ← Existe un flujo factible de U flujo”

La asignación que hicimos son **U** animales?

- Si, porque hay **U** de flujo, cada unidad representa un animal en una habitación, y pudimos asignarlos a todos.

Nodos, aristas y flujo máximo en base a parámetros del problema

¿Cuántos nodos, aristas y flujo máximo tenemos?

- Nodos en cada capa:
 - Capa de **animales**: **A**, una por animal.
 - Capa de **(conflicto, habitacion)**: **CR** maximo, porque por cada conflicto puede haber un animal para cada habitación.
 - Capa de **habitaciones**: **R**, una por habitación.

Hay **O(A + CR)** en total.

- Aristas entre cada capa:
 - Entre **s** y **animales**: Hay **A** porque es una por jugador que vota.
 - Entre **animales** y **(conflicto, habitacion)**: Hay **AR** máximo (si todos los animales pueden ubicarse en todas las habitaciones) y **A** mínimo (cada animal se puede asignar a 1 habitación).
 - Entre **(conflicto, habitacion)** y **habitaciones**: Hay igual cantidad que **(conflicto, habitacion)**, que es **CR** maximo.
 - Entre **habitaciones** y **t**: Hay **R**.

Hay **O(AR + CR) = O(R(A+C))** en total.

- Flujo máximo:
 - Puede salir **A** de flujo de **s**, porque de **s** a cada animal hay capacidad 1.
 - Puede entrar como mínimo **A** de flujo a **t**, porque sino no hay suficiente espacio para todos los animales.

Hay **O(A)** en total.

Algoritmo y complejidad para resolver modelo

¿Qué implementación del algoritmo es mejor aca?

Tenemos estas opciones:

- **Ford & Fulkerson:** $O(|E| * F)$.
 - Reemplazando tenemos $O(R(A+C) * A)$.
- **Edmonds Karp:** $O(|V| E^2)$.
 - Reemplazando tenemos $O(A * (R(A+C))^2)$.

Termina ganando **Ford & Fulkerson** en el caso general.



EXTRA

Votacion

Segundo parcial 2do cuatrimestre 2022



Enunciado

Enunciado

Tuki fue uno de los afortunados en entrar a la casa de GH (Grupo Humano). En este juego cada semana los jugadores votan por la eliminación de otro jugador. Cada jugador vota a un único jugador y los jugadores con más votos son enviados a un desafío.

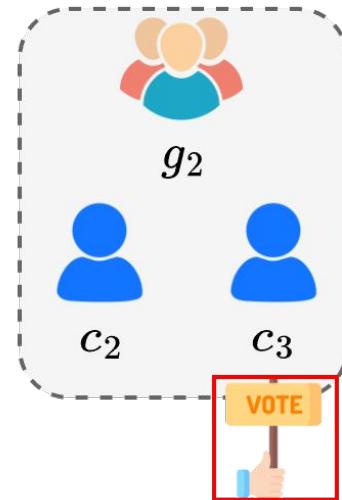
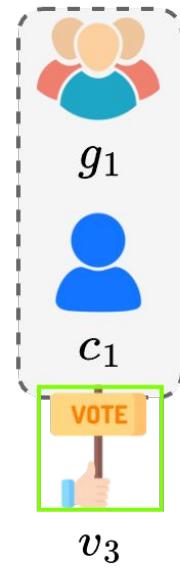
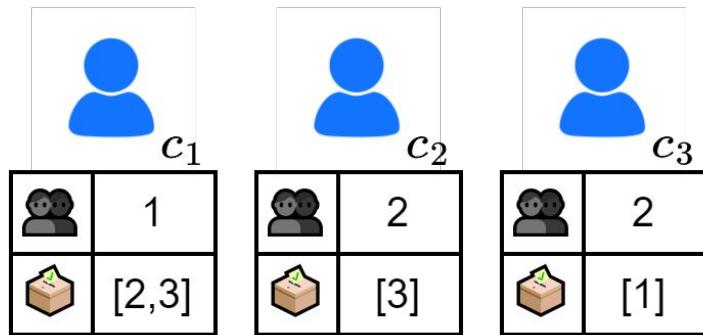
Tuki sabe que conviene mantener la votación lo más pareja posible, para evitar problemas en la casa. El sabe que:

- Hay n jugadores y m grupos.
- El jugador i pertenece al grupo g_j ,
- El jugador i está dispuesto a votar al conjunto de jugadores N_i .
- Si algún grupo g_j recibe más votos que el doble de la cantidad de jugadores en el grupo, entonces se va a sentir atacado.
- Si un jugador individual i recibe más de 3 votos, entonces también se siente atacado.

Quiere decidir, dados los n jugadores junto a sus conjuntos N_i y la descripción de los m grupos, si es posible que la votación logre que ningún grupo ni jugador se sienta atacado.

Ejemplo

Ejemplo



Resolución

Datos de ejemplo para la resolución

Para modelar la red vamos a estar utilizando estos 3 jugadores.



c_1

	1
	[2,3]



c_2

	2
	[3]



c_3

	2
	[1]

Modelo de red

¿Qué queremos asignar?

Tenemos una votación, donde los jugadores pueden votar a otros jugadores.

Queremos asignar entonces el voto de cada jugador X a otro jugador Y.

¿Qué entidades están involucradas en esta votación?

Las entidades involucradas son:

- Jugadores votantes.
- Jugadores candidatos (son los mismos que los otros, pero está bueno verlo así).
- Grupos de jugadores.



Votante



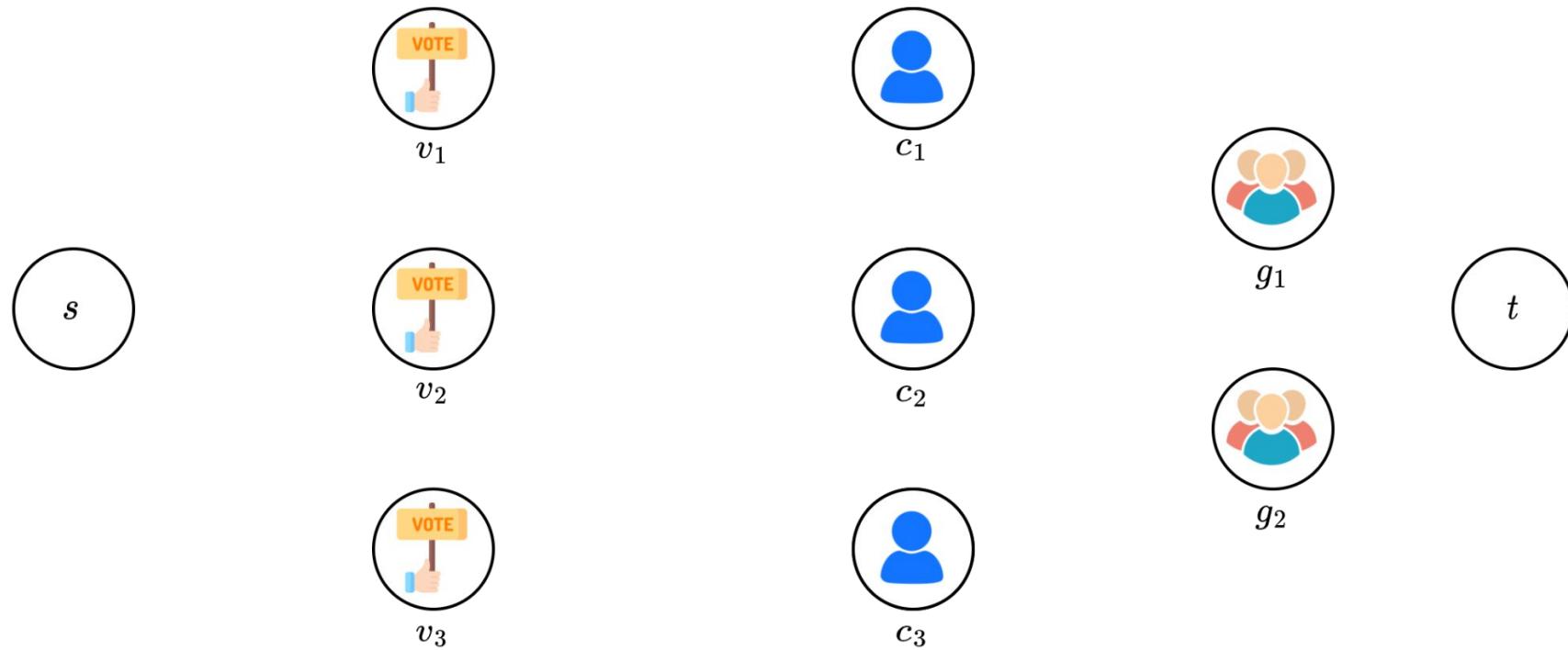
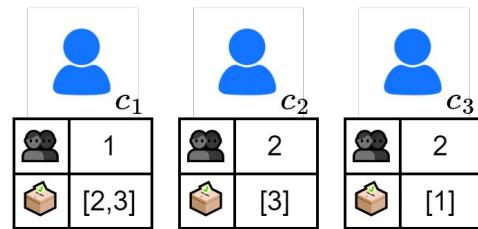
Candidato



Grupo

Entidades como capas del modelo

Pongamos las entidades anteriores en capas para nuestro modelo de red, junto con los nodos s y t.



¿Qué restricciones tenemos en esta votación?

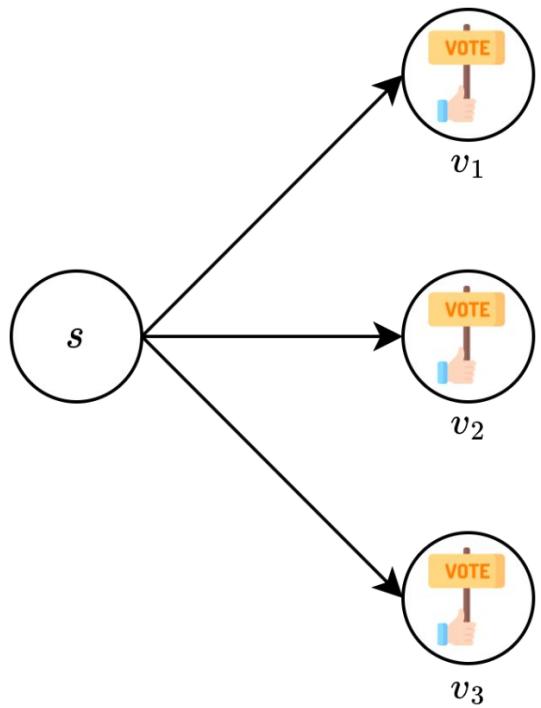
Pensando en las restricciones, podríamos decir que:

1. Cada jugador puede votar una vez.
2. Cada jugador i puede votar a un conjunto de jugadores N_i .
3. No puede votarse más de 3 veces un candidato.
4. No puede recibir más de 2^*g_i votos el grupo i .

Veamos cómo incluir esto en nuestro modelo.

Conexiones

Cada jugador puede votar una vez.



c_1



c_2



c_3

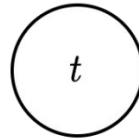
	c_1
	1
	[2,3]
	2
	[3]
	2
	[1]



g_1



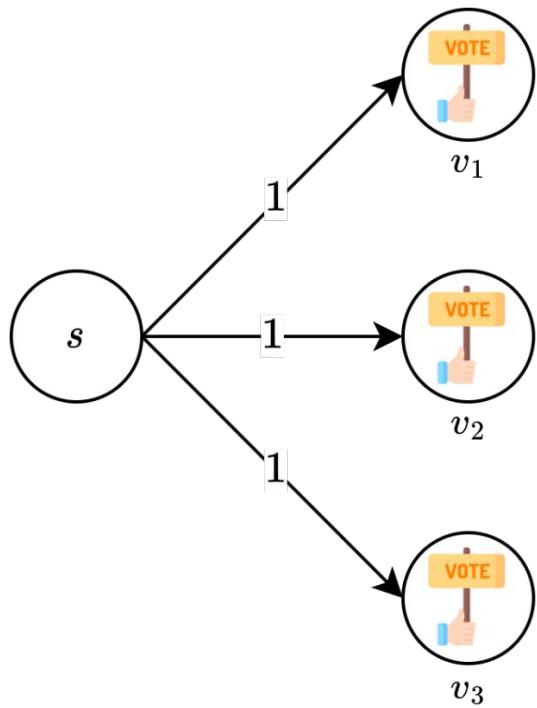
g_2



t

Restricciones

Cada jugador puede votar una vez.



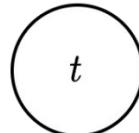
	c_1
	1
	[2,3]
	2
	[3]
	2
	[1]



g_1



g_2



t

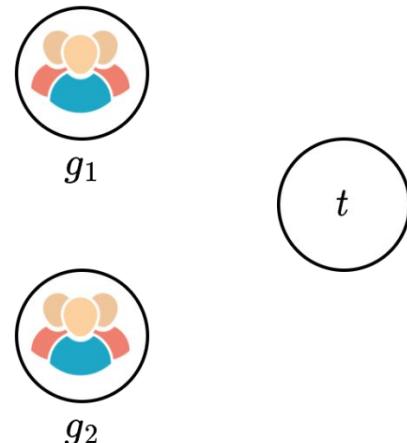
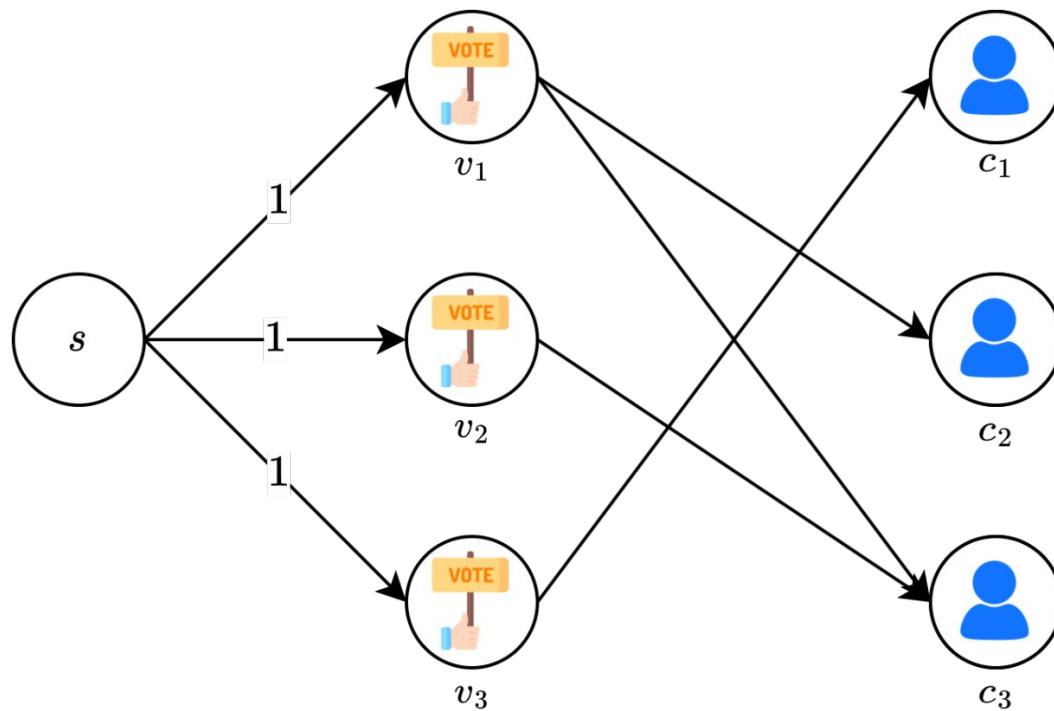
Conexiones

Cada jugador i puede votar a un conjunto de jugadores N_i .

	c_1
	1
	[2,3]

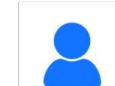
	c_2
	2
	[3]

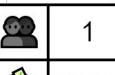
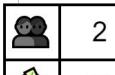
	c_3
	2
	[1]

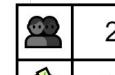
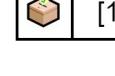


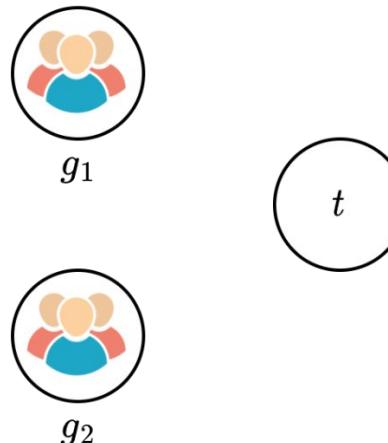
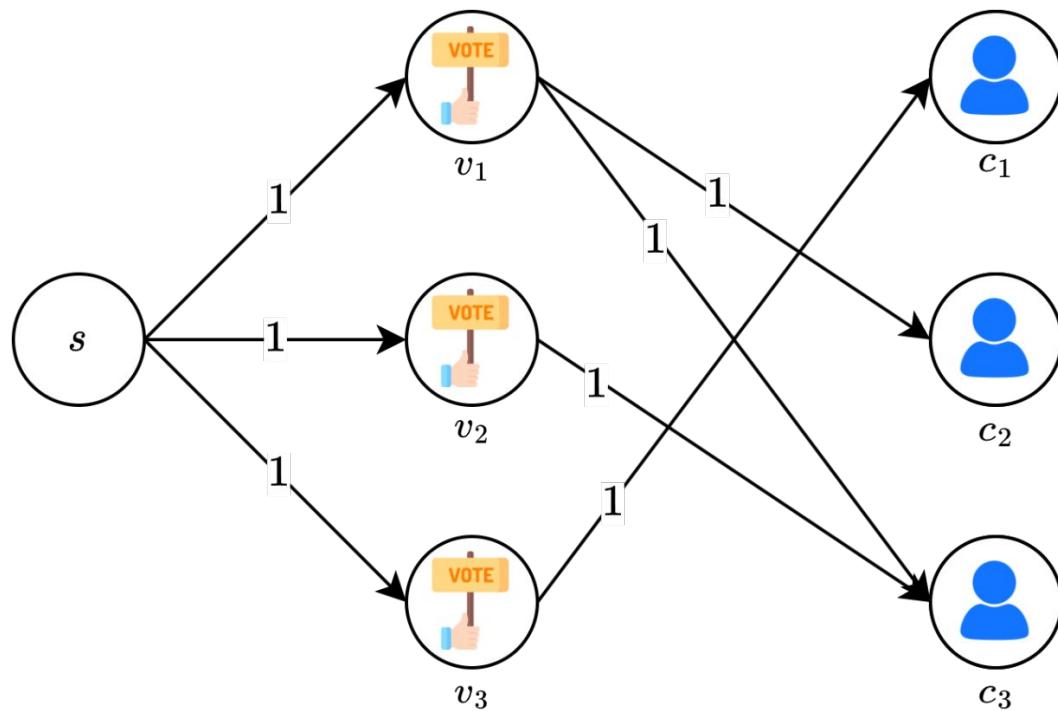
Restricciones

Cada jugador i puede votar a un conjunto de jugadores N_i .

	c_1
	c_2
	c_3

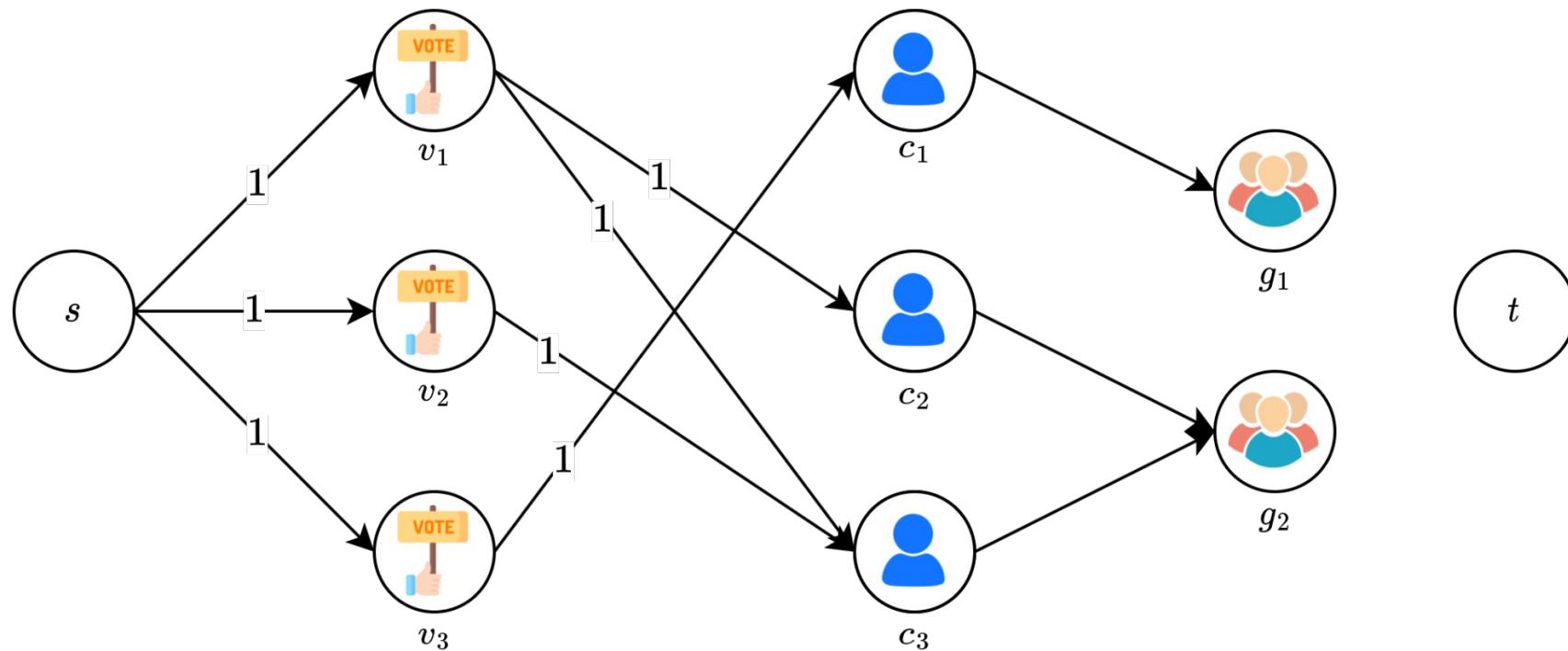
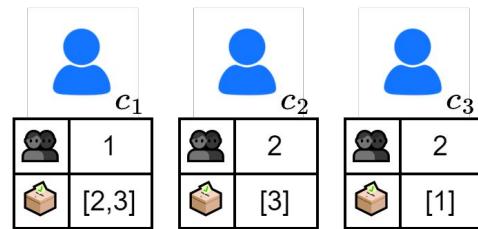
	1
	[2,3]

	2
	[1]



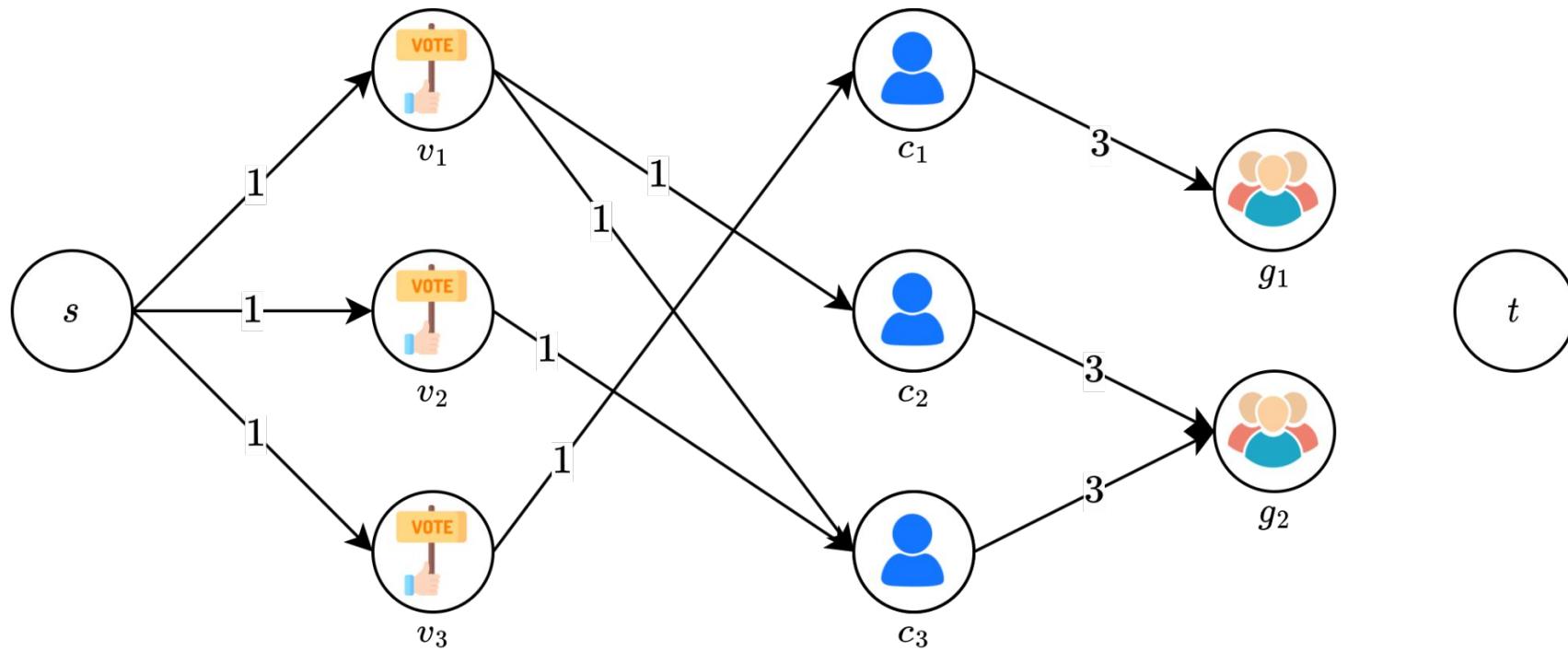
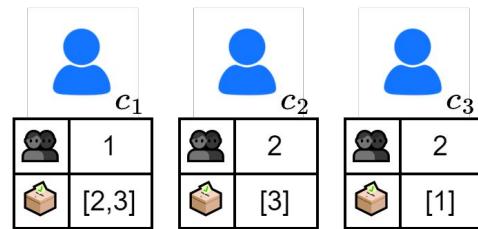
Conexiones

No puede votarse más de 3 veces un candidato.



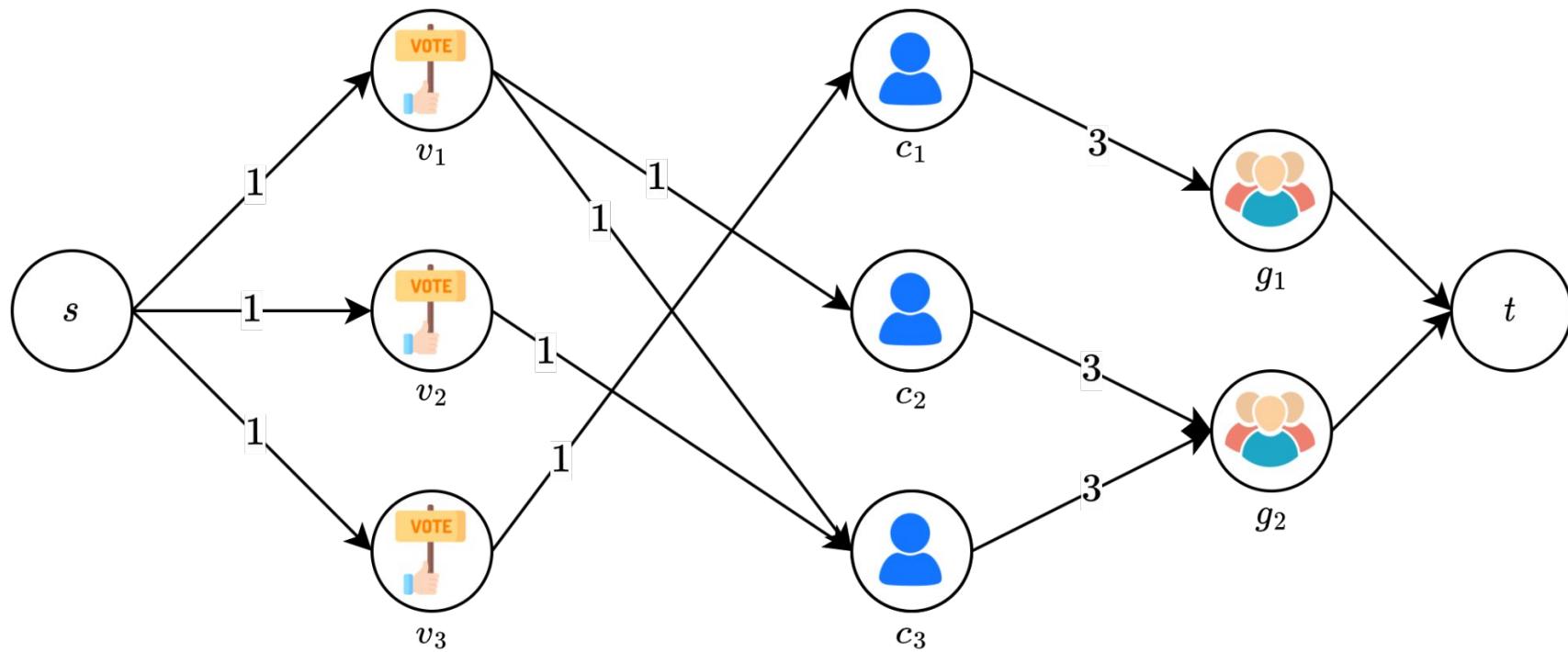
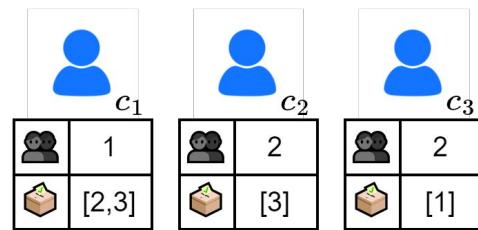
Restricciones

No puede votarse más de 3 veces un candidato.



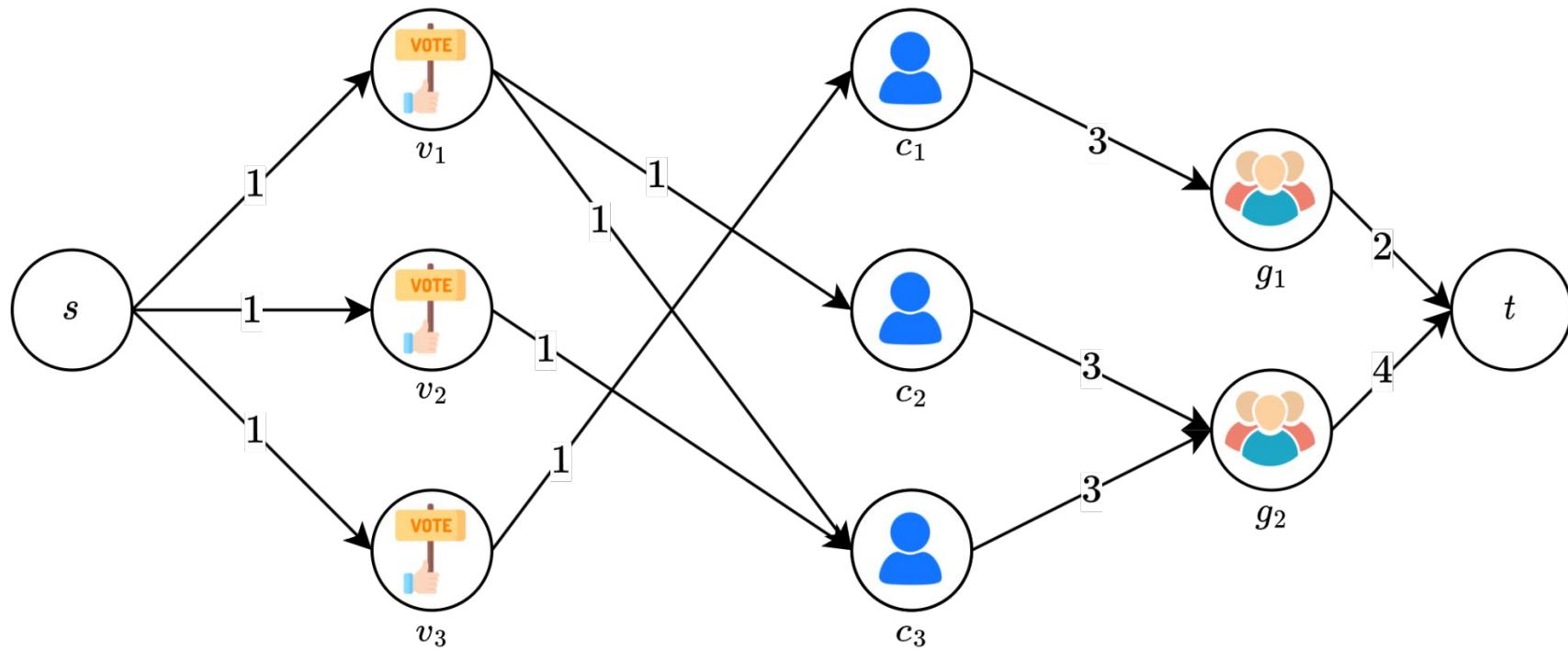
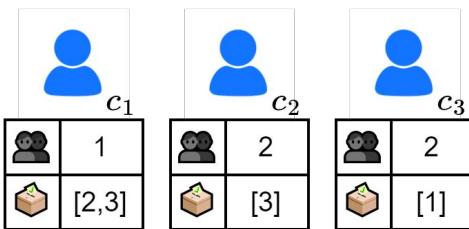
Conexiones

No puede recibir más de 2^*g_i votos el grupo i .



Restricciones

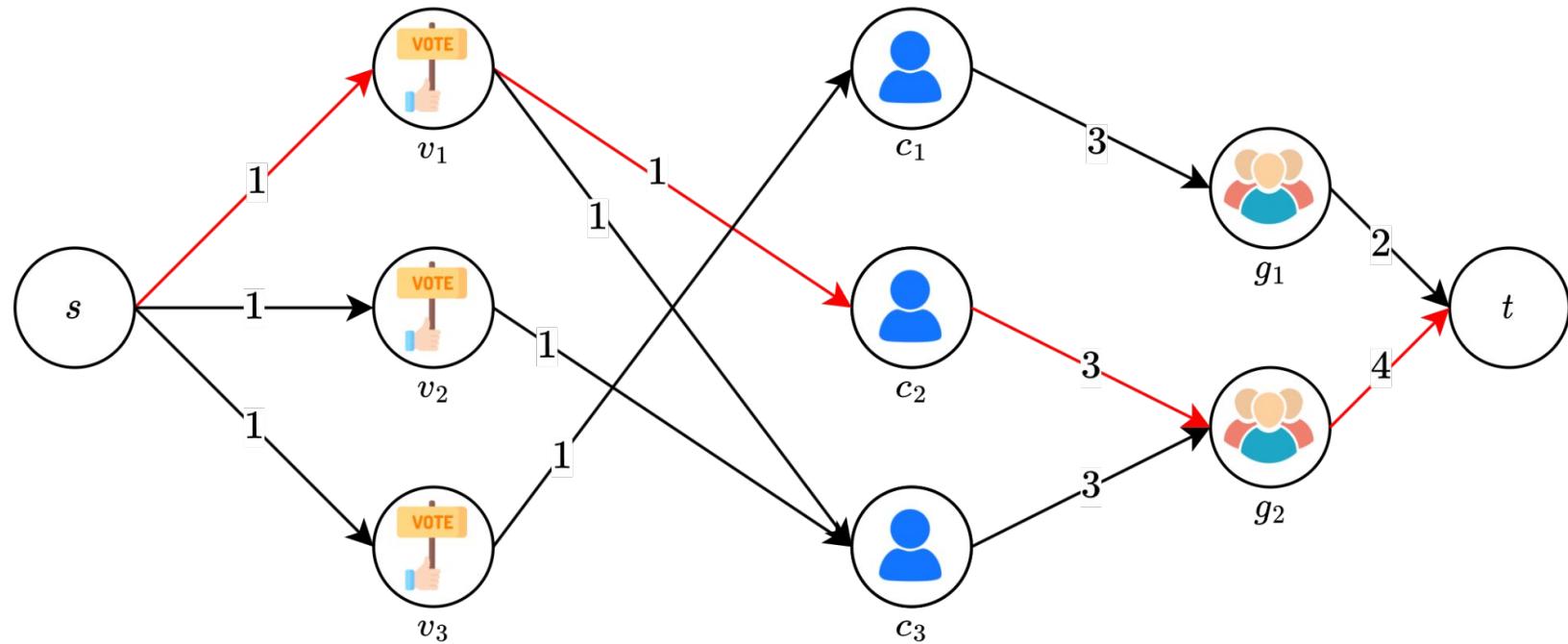
No puede recibir más de $2 \cdot g_i$ votos el grupo i .



Semántica de unidad de flujo

¿Cómo interpretamos a la unidad de flujo en la red?

Como lo pensamos cuando empezamos el modelo, podemos interpretarlo como un voto de un votante asignado a un candidato. A su vez podemos ver este voto que termina asignado al grupo del candidato.



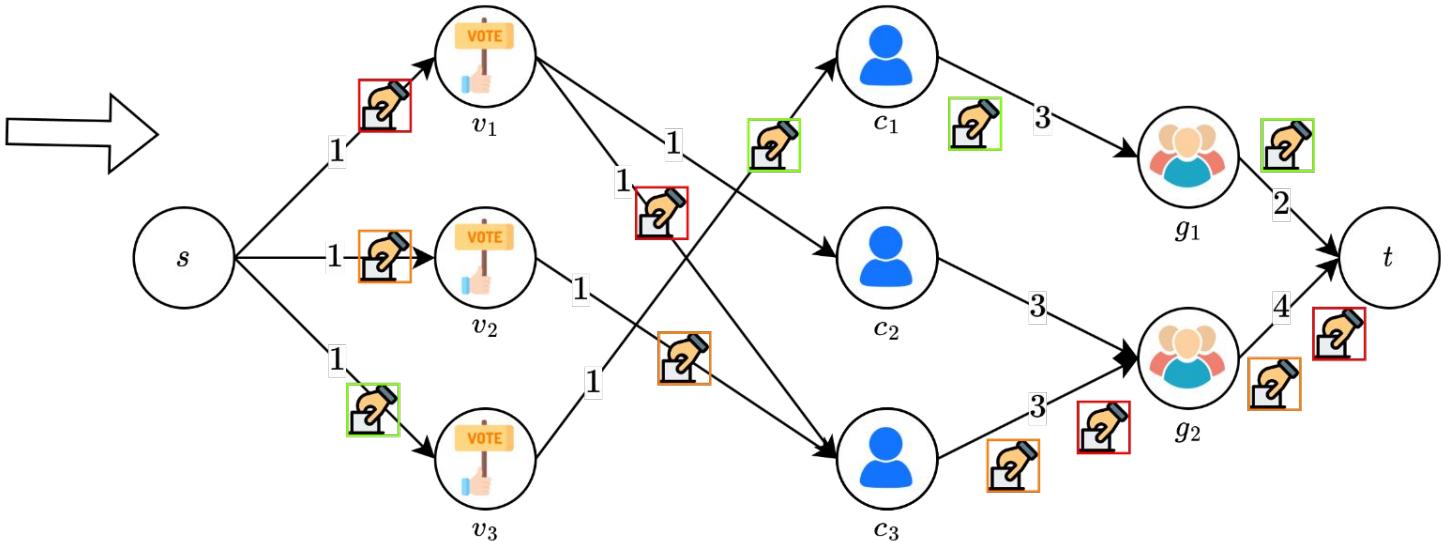
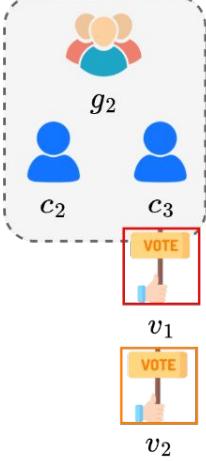
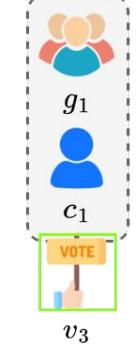
Conexión Problema ↔ Modelo

Mostrando la conexión entre el modelo y el problema

Vamos a querer demostrar esta afirmación:

“Existe una forma de asignar U votos sin generar conflictos \leftrightarrow Existe un flujo factible de U flujo”

Demosmos cada implicación, guiándonos con un gráfico de cada caso.



Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U votos sin generar conflictos \rightarrow Existe un flujo factible de U flujo”

Por cada voto del v_i que vota al c_j construimos un camino de flujo 1 de la siguiente forma:

- De s al nodo del v_i .
- Del nodo v_i al nodo del c_j .
- Del nodo c_j al nodo g_k donde pertenece.
- Del nodo del g_k al nodo t .

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U votos sin generar conflictos → Existe un flujo factible de U flujo”

¿El flujo respeta capacidades?

- $s \rightarrow v_i$: Siempre se asigna 1 máximo porque cada votante vota una vez.
- $v_i \rightarrow c_j$: La arista existe porque es un voto válido y siempre se le asigna 1 máximo por lo mismo que antes.
- $c_j \rightarrow g_k$: La arista existe porque pertenece al grupo ese, y hay como máximo 3 unidades porque es una votación valida y no se votó más de 3 veces a un candidato c_j .
- $g_k \rightarrow t$: Hay máximo $2^*(\text{cant de candidatos en } g_k)$ unidades, porque la votación es valida y no puede tener más de esa cantidad de votos el grupo g_k .

¿El flujo respeta la conservación?

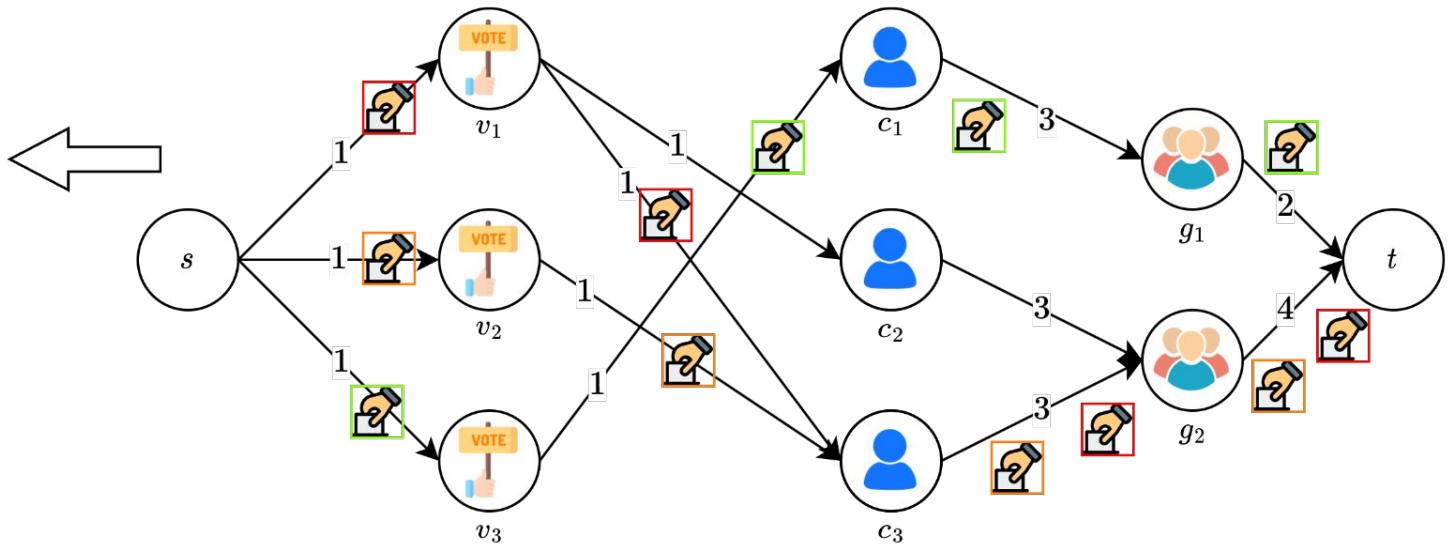
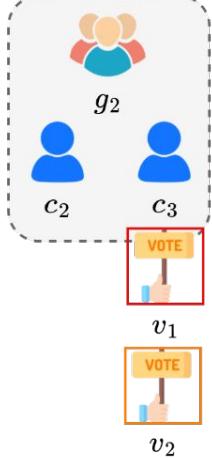
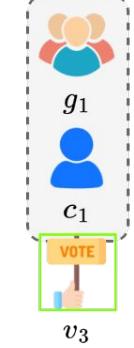
- Por cada voto asignado a v_i , sale para c_j .
- Idem de c_j con g_k .
- Idem de g_k con t .

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U votos sin generar conflictos → Existe un flujo factible de U flujo”

La asignación de flujo que hicimos es \mathbf{U} ?

- Si, porque como la votación era válida, había \mathbf{U} votos, asignamos \mathbf{U} unidades de flujo saliendo desde s , una por cada voto.



Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U votos sin generar conflictos \leftrightarrow Existe un flujo factible de U flujo”

Por cada unidad de flujo que sale de s a un nodo v_i podemos:

- Asignar el voto de v_i a c_j , donde j es el nodo candidato al que le llega la unidad de flujo que sale desde el nodo del v_i .

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U votos sin generar conflictos \leftrightarrow Existe un flujo factible de U flujo”

Veamos que esto respeta las restricciones:

- Solo se asigna 1 voto por v_i porque el flujo es factible.
- El v_i vota a un c_j valido porque solo existe la arista si era un posible candidato a votar por v_i .
- No se vota más de 3 veces a un c_j porque desde el nodo del c_j al grupo g_k hay capacidad 3, entonces no puede haber más de 3 votos.
- No se asignan más de $2^*(\text{cantidad de candidatos en } g_k)$ votos al grupo g_k porque desde el nodo del grupo g_k al t está esa misma capacidad.

Mostrando la conexión entre el modelo y el problema

“Existe una forma de asignar U votos sin generar conflictos \leftrightarrow Existe un flujo factible de U flujo”

La asignación que hicimos son U votos?

- Si, porque como había U de flujo, cada unidad representaba un voto, y pudimos asignar todos.

Nodos, aristas y flujo máximo en base a parámetros del problema

¿Cuántos nodos, aristas y flujo máximo tenemos?

- Nodos en cada capa:

- Capa de votantes: N porque es una por jugador que vota.
- Capa de candidatos: N porque es una por jugador que puede ser votado.
- Capa de grupos: Hay N máximo, si hay un grupo por jugador.

Hay $3N$ en total.

- Aristas entre cada capa:

- Entre s y votantes: Hay N porque es una por jugador que vota.
- Entre votantes y candidatos: Hay N^2 máximo (si todos votan a todos) y N mínimo (todos votan a 1).
- Entre candidatos y grupos: Hay N porque es una por jugador que pertenece a un único grupo.
- Entre grupos y t : Hay N maximo, si hay un grupo por jugador.

Hay $O(N^2)$ en total.

- Flujo máximo:

- Puede salir N de flujo de s , porque de s a cada jugador hay capacidad 1.
- Puede entrar $2N$ de flujo a t , porque de cada grupo a t la capacidad es el doble de jugadores que contiene.

Hay $O(N)$ en total.

Algoritmo y complejidad para resolver modelo

¿Qué implementación del algoritmo es mejor aca?

Tenemos estas opciones:

- **Ford & Fulkerson:** $O(|E| * F)$.
 - Reemplazando tenemos $O(N^2 * N) = O(N^3)$.
- **Edmonds Karp:** $O(|V| E^2)$.
 - Reemplazando tenemos $O(N * (N^2)^2) = O(N^5)$.

Termina ganando **Ford & Fulkerson** en el caso general.



Costo mínimo

Segundo parcial 1er cuatrimestre 2022

Enunciado

Enunciado

Un camión tiene una ruta fija de clientes v_1, \dots, v_n a la que se planificó enviarle paquetes en ese orden. Dentro del camión se encuentra un drone que va a ser usado para visitar a $n - 1$ clientes de otro conjunto $W = w_1, \dots, w_m$ tal que $m \geq n - 1$.

La operación del camión y el drone es la siguiente:

- En cada v_i , el drone despega del camión, visita a uno de los clientes de W que aun no fue visitado, y se encuentra con el camión en el cliente v_{i+1} .
- El vehículo que llega primero al cliente v_{i+1} se queda esperando al vehículo más lento.
- El camión demora t_i en viajar de v_i a v_{i+1} , y el drone demora d_{ij} en viajar desde v_i hasta w_j y luego ir a v_{i+1}

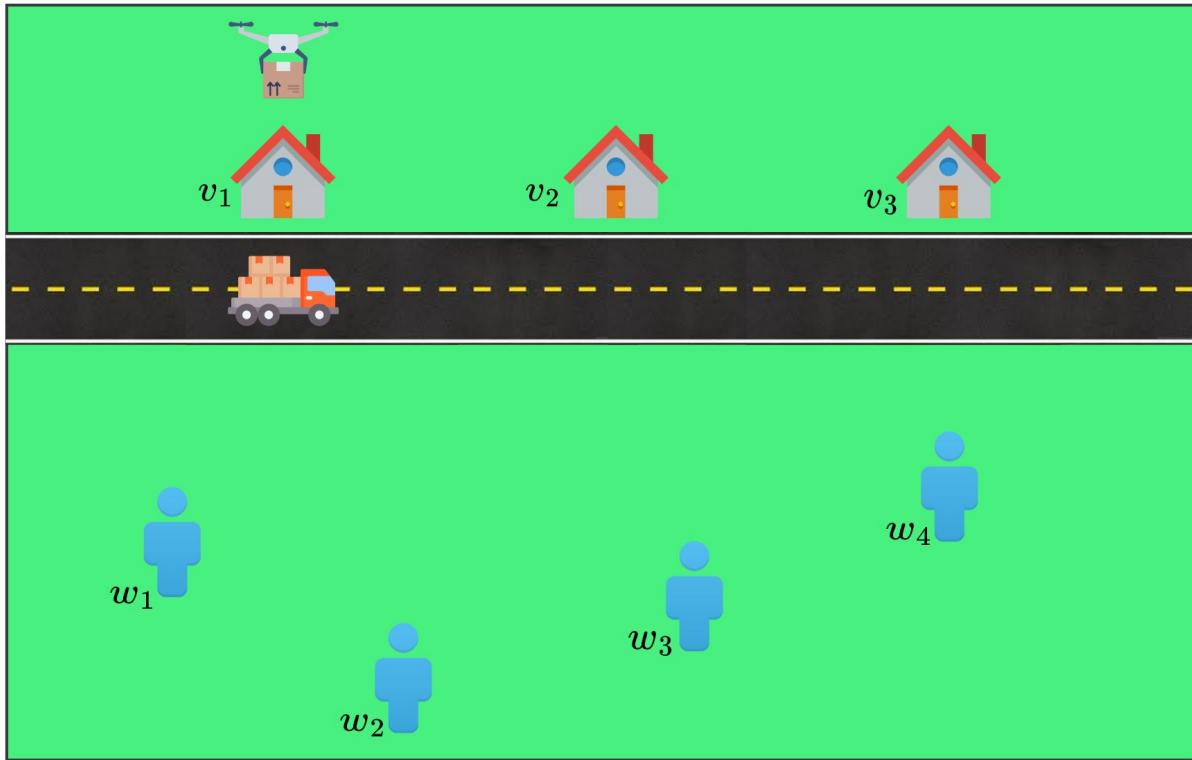
Queremos encontrar el camino que pueda recorrerse en el menor tiempo posible.

Enunciado

Los pasos a realizar son:

1. Modelar el problema como un problema de flujo máximo con costo mínimo o, alternativamente, como uno de matching bipartito de peso mínimo y justificar.
2. Proponer un algoritmo polinomial que resuelva el modelo propuesto.
3. Expresar la complejidad del algoritmo en función de n y m , indicando la cota más ajustada que sea posible.

Ejemplo

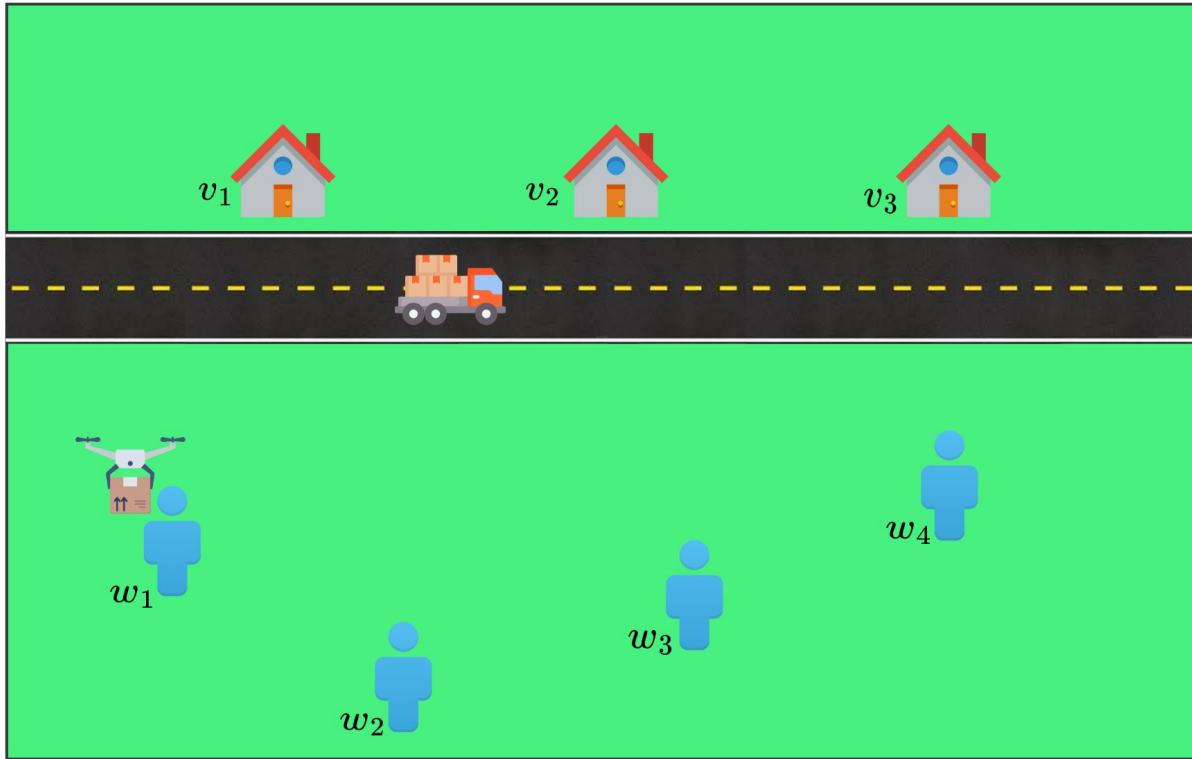


Camino : v_1

d_{ij}	w_1	w_2	w_3	w_4
v_1	5	11	3	8
v_2	7	5	1	6

t_i	
v_1	4
v_2	3

El recorrido empieza en el cliente v_1 , donde están el camión y el drone.

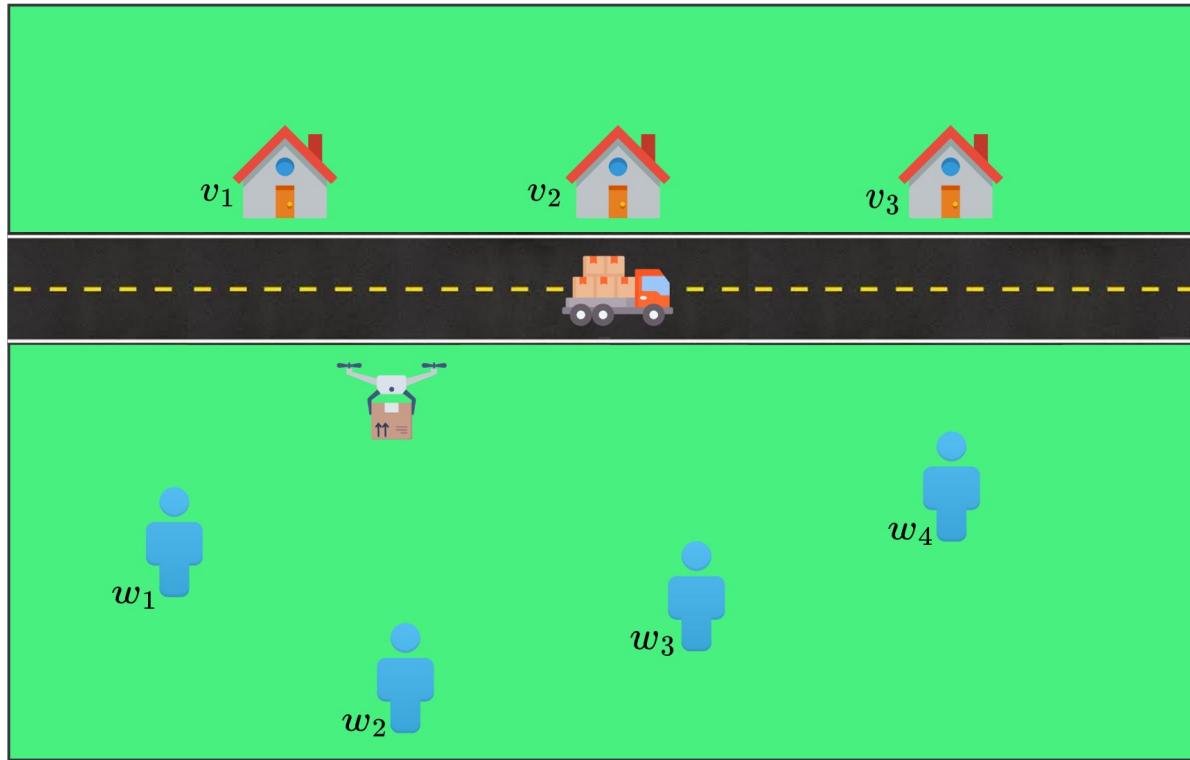


d_{ij}	w_1	w_2	w_3	w_4
v_1	5	11	3	8
v_2	7	5	1	6

t_i	
v_1	4
v_2	3

Camino : $v_1 \rightarrow w_1 \rightarrow v_2$

El dron pasa por w_1 , y le toma más tiempo que al camión para llegar a v_2 .

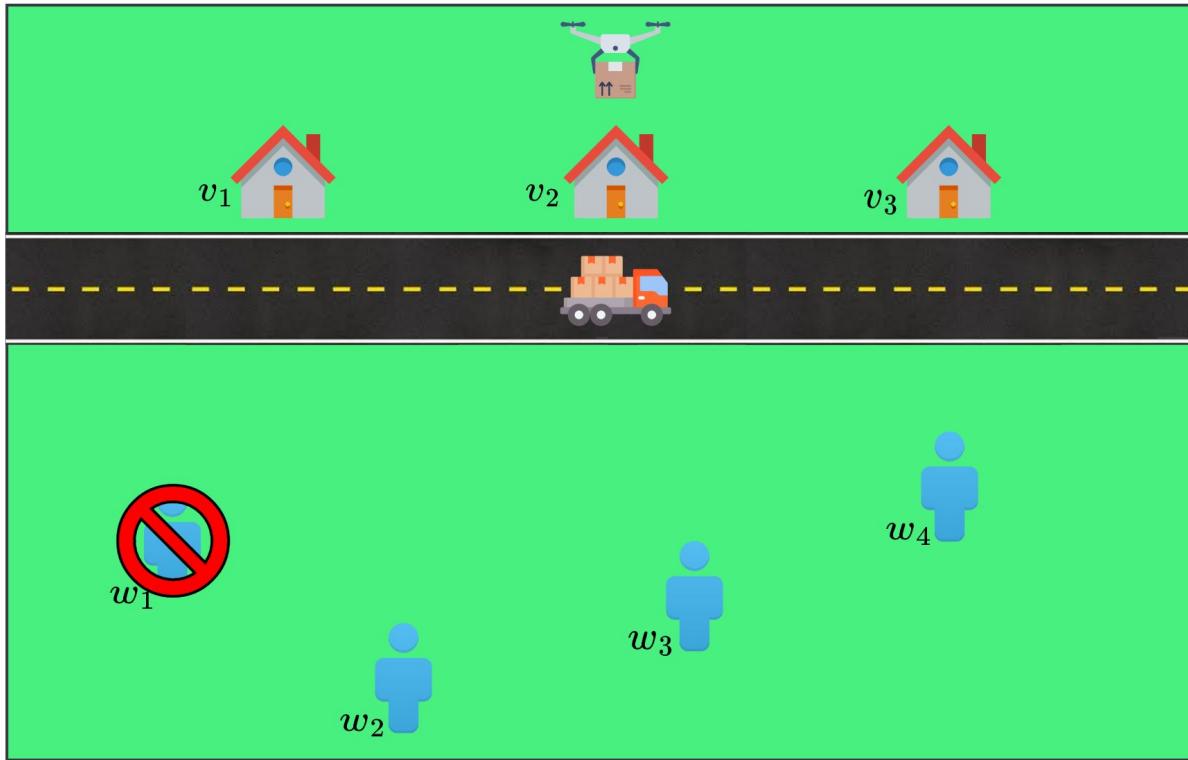


d_{ij}	w_1	w_2	w_3	w_4
v_1	5	11	3	8
v_2	7	5	1	6

t_i	
v_1	4
v_2	3

Camino : $v_1 \rightarrow w_1 \rightarrow v_2$

El dron pasa por w_1 , y le toma más tiempo que al camión para llegar a v_2 .

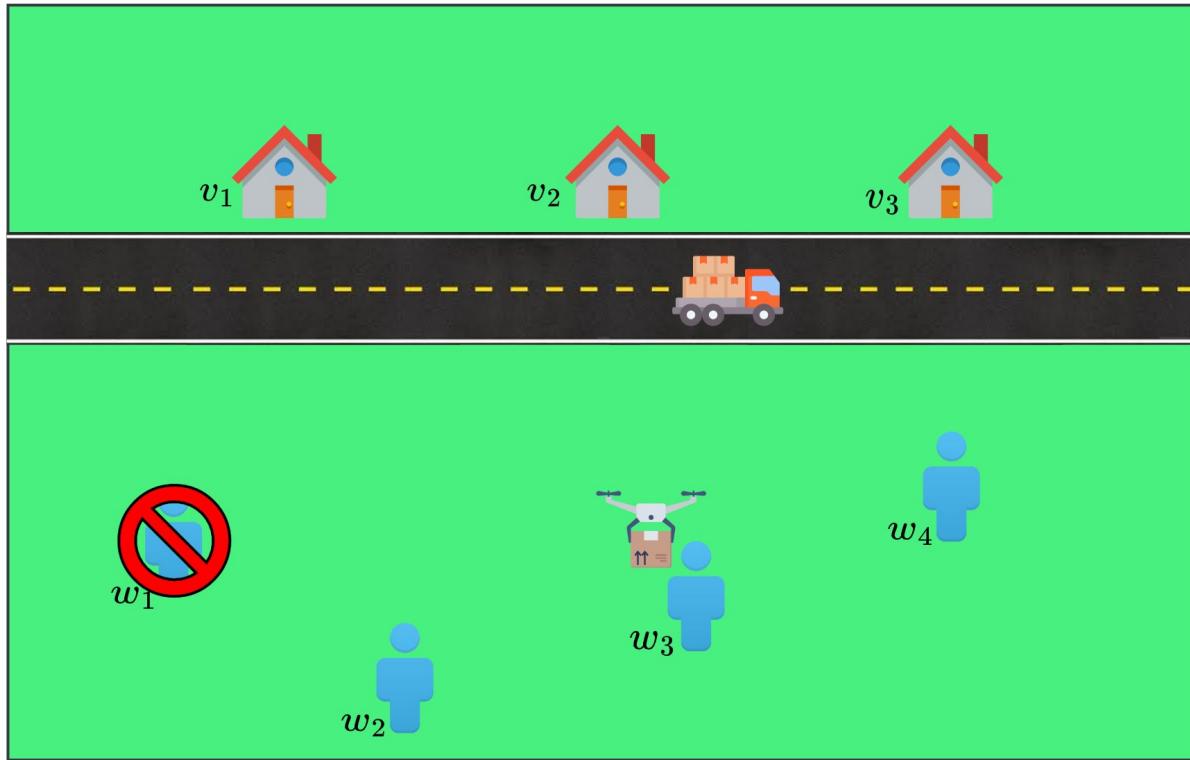


d_{ij}	w_1	w_2	w_3	w_4
v_1	5	11	3	8
v_2	7	5	1	6

t_i	
v_1	4
v_2	3

Camino : $v_1 \rightarrow w_1 \rightarrow v_2$

Llegamos a v_2 y el drone no puede pasar más por w_1 porque ya lo uso.

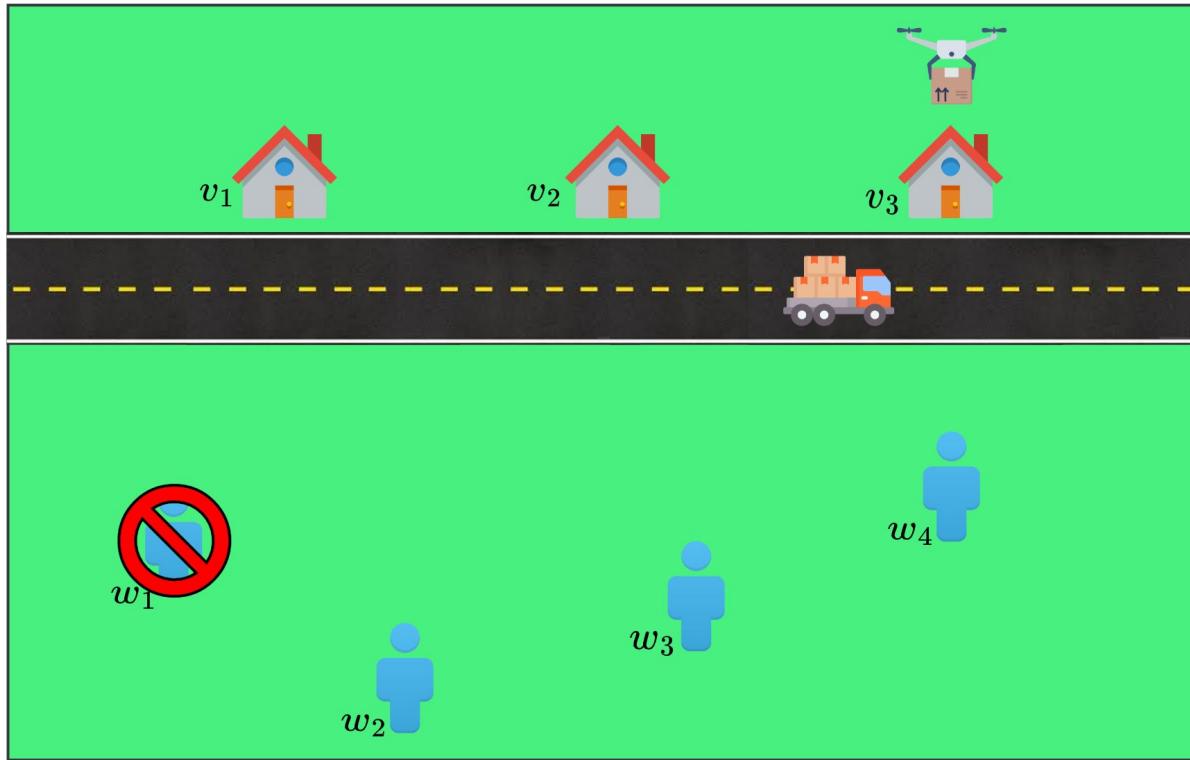


d_{ij}	w_1	w_2	w_3	w_4
v_1	5	11	3	8
v_2	7	5	1	6

t_i	
v_1	4
v_2	3

Camino : $v_1 \rightarrow w_1 \rightarrow v_2 \rightarrow w_3 \rightarrow v_3$

El dron pasa por w_3 , y le toma menos tiempo esta vez que al camión para llegar a v_3 .

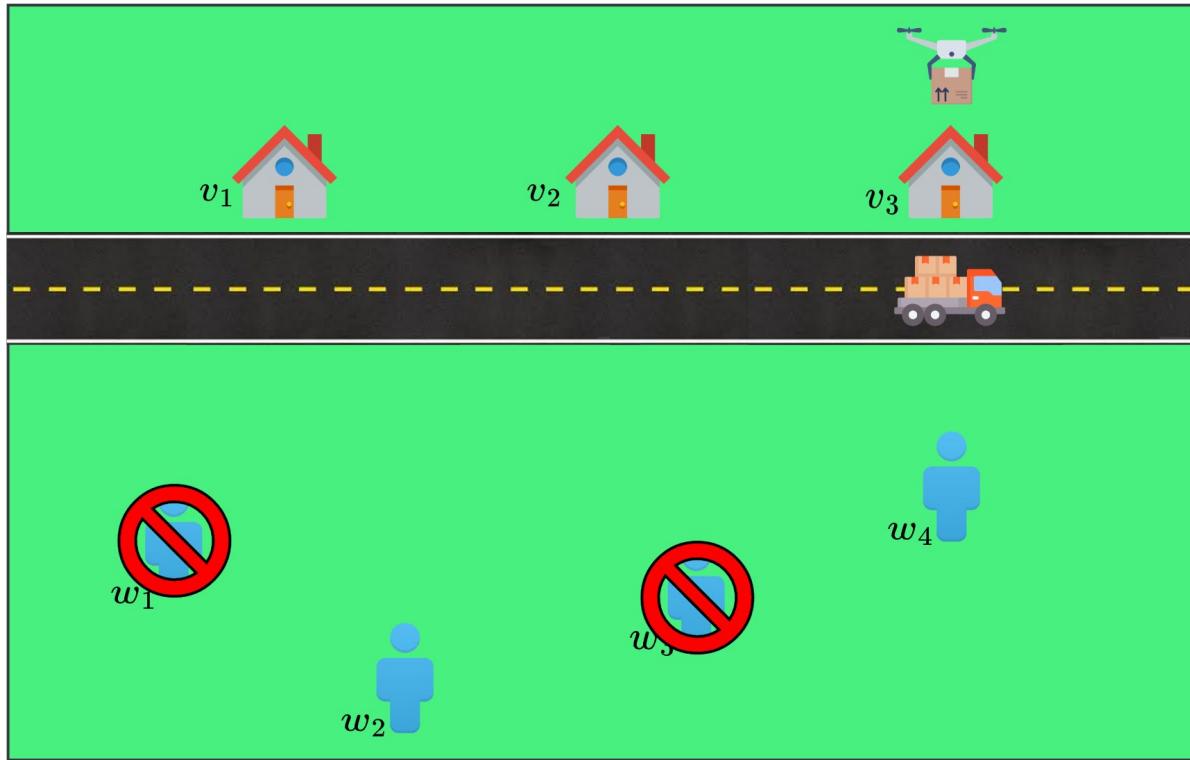


Camino : $v_1 \rightarrow w_1 \rightarrow v_2 \rightarrow w_3 \rightarrow v_3$

d_{ij}	w_1	w_2	w_3	w_4
v_1	5	11	3	8
v_2	7	5	1	6

t_i	
v_1	4
v_2	3

El dron pasa por w_3 , y le toma menos tiempo esta vez que al camión para llegar a v_3 .



d_{ij}	w_1	w_2	w_3	w_4
v_1	5	11	3	8
v_2	7	5	1	6

t_i	
v_1	4
v_2	3

Camino : $v_1 \rightarrow w_1 \rightarrow v_2 \rightarrow w_3 \rightarrow v_3$

El camino realizado tiene costo 8, y es el óptimo en este caso.

Resolución

Inciso (a)

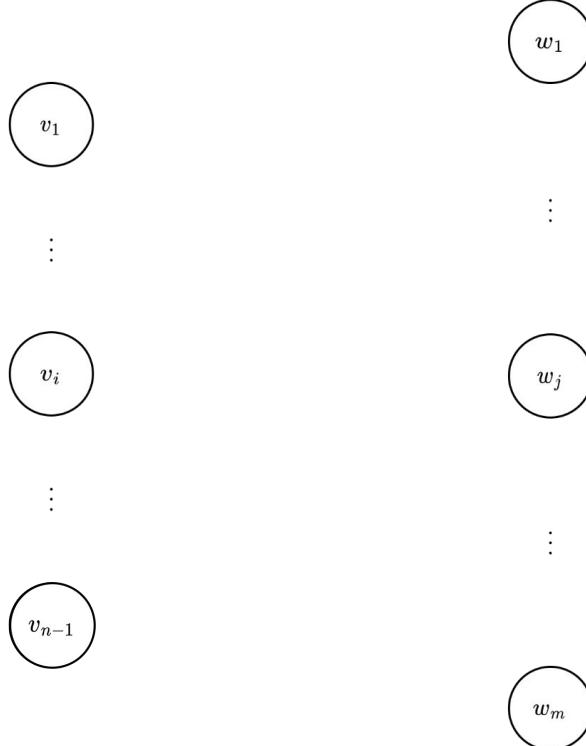
¿Cómo modelamos el problema?

Modelado: asignación

- Supongamos por un momento que no nos interesa el costo mínimo.
- Queremos encontrar una forma de asignarle al drone un cliente $w_i \in W$ para cada v_i que visita el camión, ignorando los tiempos que tarda cada uno.
- ¿Qué cosas nos interesan para resolver esto?
 - El camión tiene que pasar por todos los v_i , exceptuando el n , en ese orden fijo.
 - En cada cliente v_i , tiene que elegir a un $w_i \in W$ del conjunto W que no haya sido visitado para mandarle el drone. Este después se encuentra con el camión en el cliente $i + 1$ siempre.

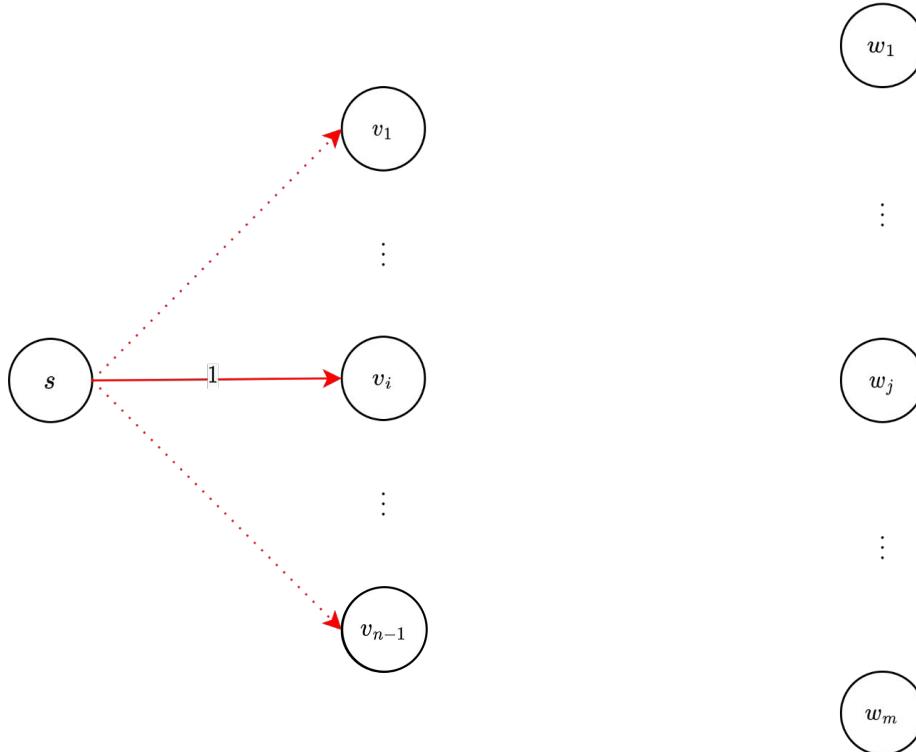
Modelado: asignación

Seguro queremos tener representados a los v_i y los w_j , así que empecemos por eso.



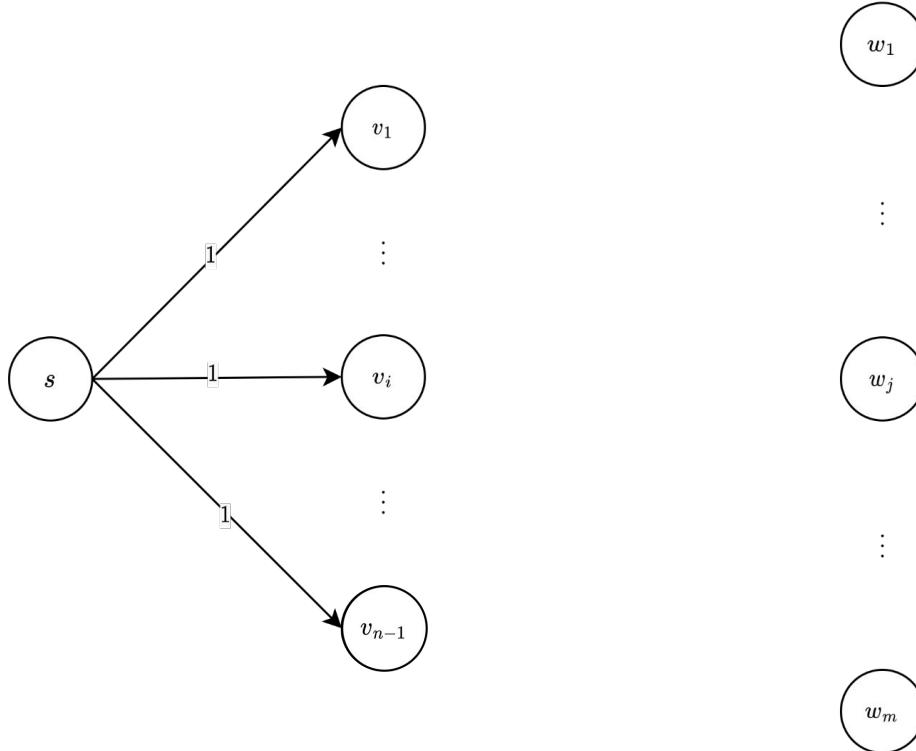
Modelado: asignación

Queremos agregar que cada v_i se visita una vez por el camión.



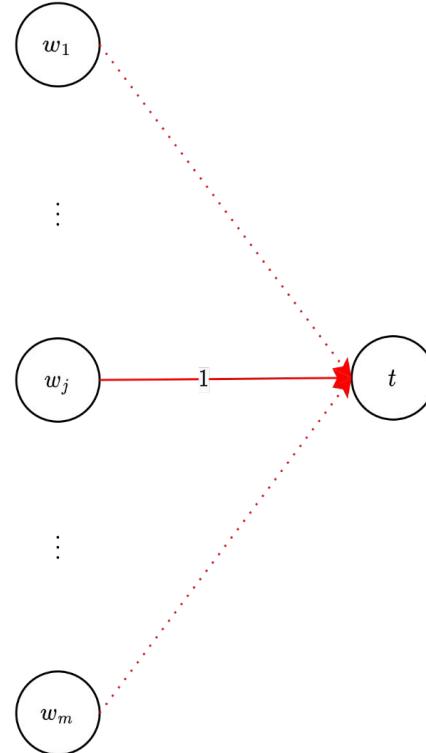
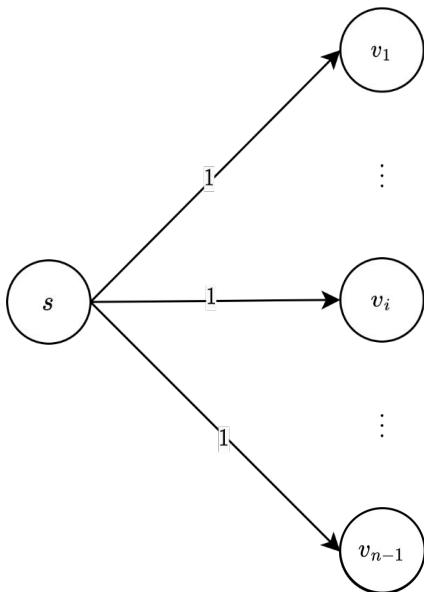
Modelado: asignación

Queremos agregar que cada v_i se visita una vez por el camión.



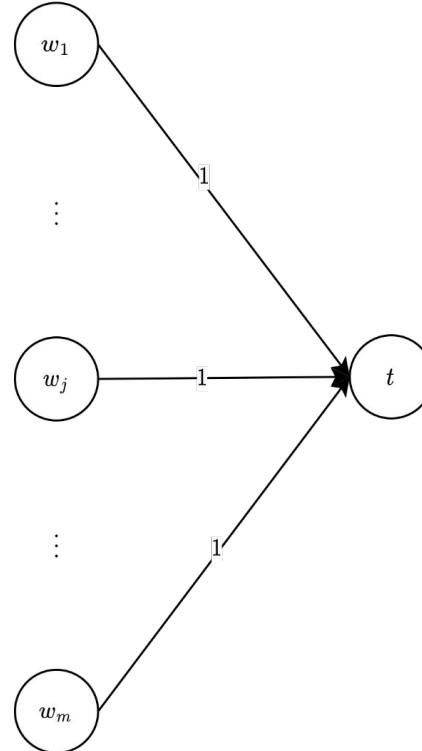
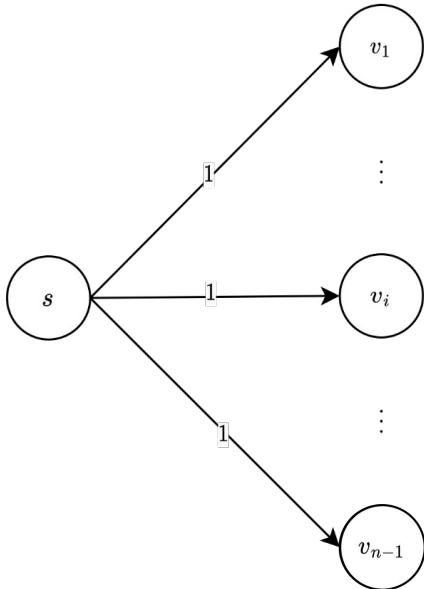
Modelado: asignación

A su vez cada cliente $w \square$ puede ser visitado una única vez.



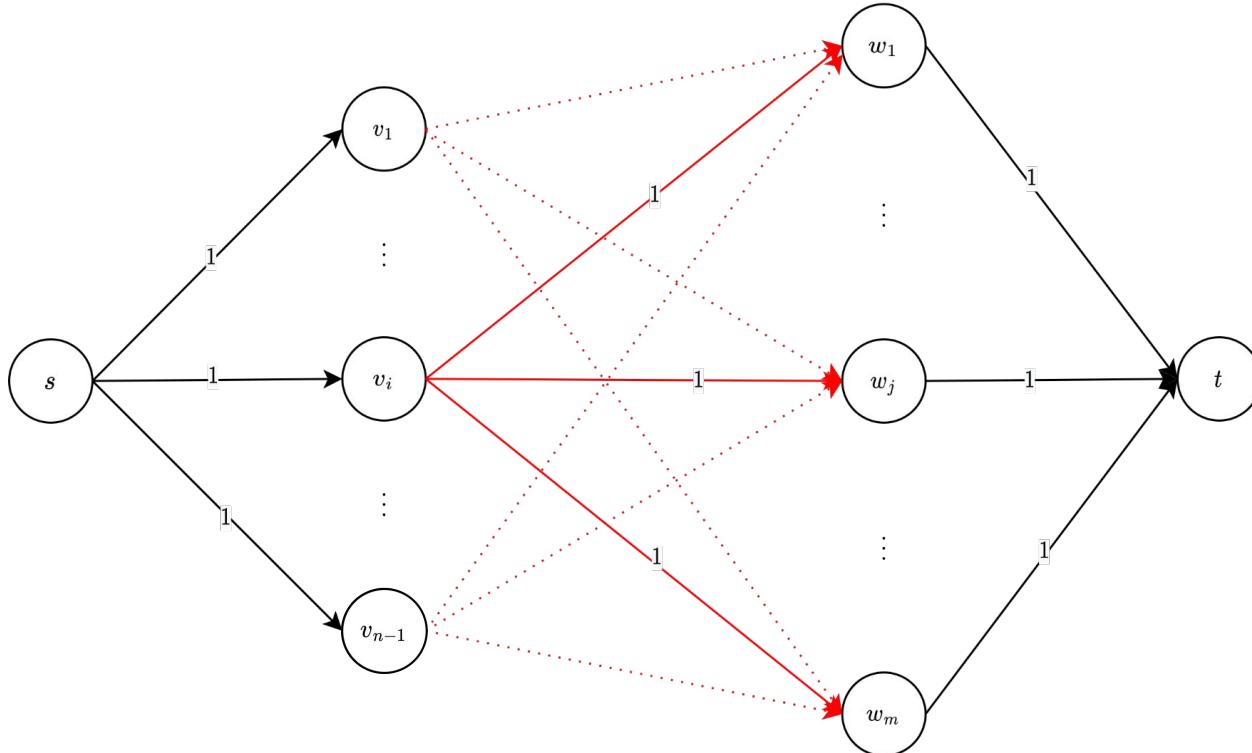
Modelado: asignación

A su vez cada cliente $w \square$ puede ser visitado una única vez.



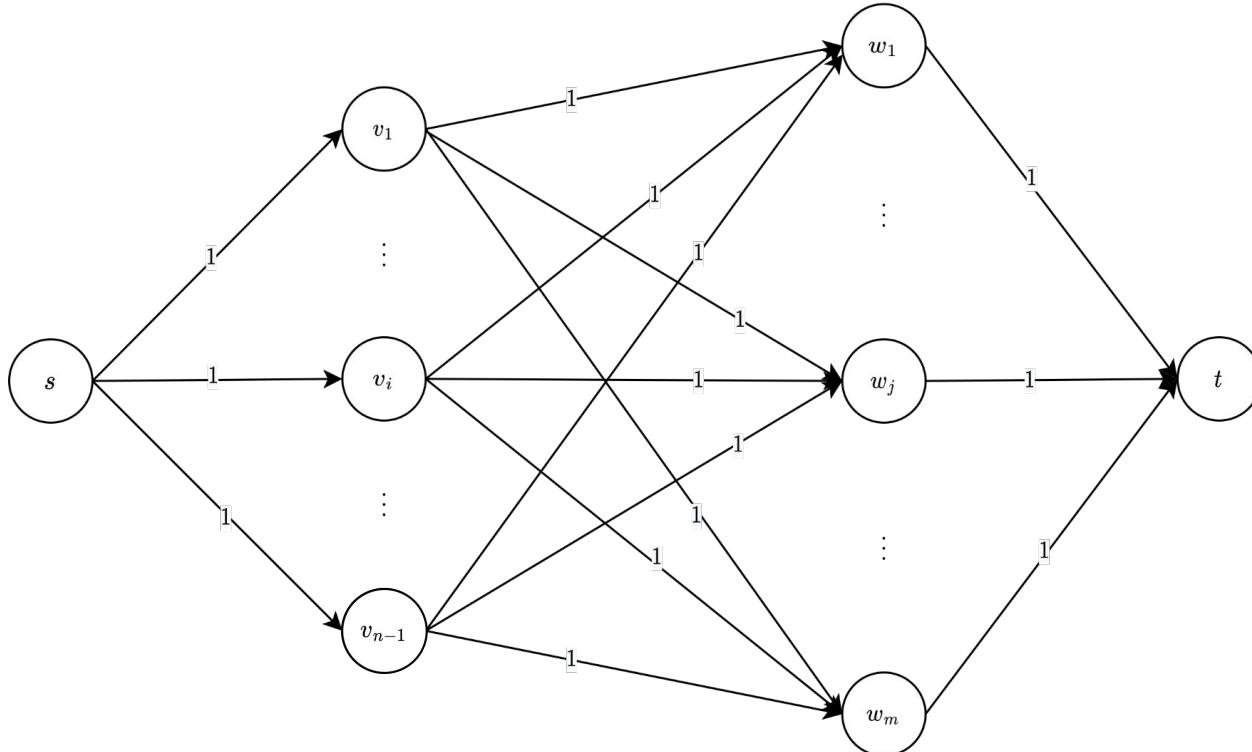
Modelado: asignación

Finalmente el drone puede salir desde un v_i a cualquier $w \square$ válido, así que...



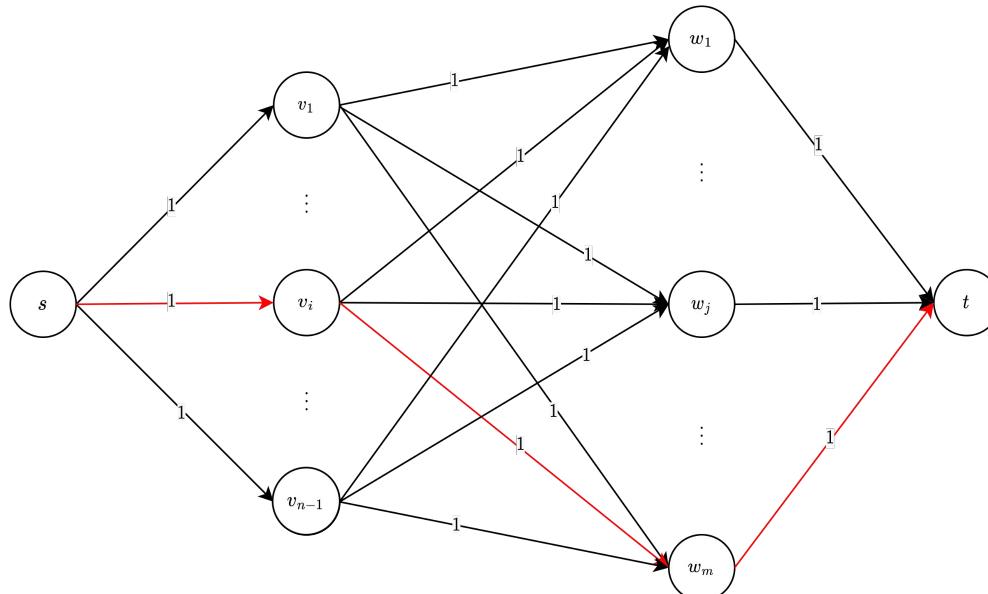
Modelado: asignación

Finalmente el drone puede salir desde un v_i a cualquier $w \square$ válido, así que...



Modelado: asignación

- Con este modelo podemos encontrar un camino válido que pasa por todos los v_i , y por $n - 1 w \square$, sin repetir 2 veces el mismo $w \square$.
- ¿Qué presenta una unidad de flujo acá?
 - Representa el cliente $w \square$ por el que pasa el drone cuando sale del camión que está en el cliente v_i .
 - Por ejemplo en rojo el drone sale del camión que está en v_i hacia el cliente $w \square$.



Modelado: asignación - justificación

- Queremos que nuestro modelo cumpla lo siguiente:
“Existe un camino válido \Leftrightarrow Existe un flujo factible máximo”
- Esto siempre ocurre en este caso, ya que como $m \geq n - 1$:
 - Siempre existe un camino válido porque podemos, por ejemplo, desde cada v_i visitar el w_i con el drone.
 - Siempre existe un flujo que satura todas las aristas que salen de s , porque al igual que antes, cada unidad de flujo puede hacer $s \rightarrow v_i \rightarrow w_i \rightarrow t$. Con esto obtenemos el flujo máximo $n - 1$.
- A continuación queda de referencia una demostración igualmente, que podría servir si tuviéramos alguna restricción del estilo “este v_i no puede visitar el w_i ”.

Modelado: asignación - justificación

“Existe un camino válido \Rightarrow Existe un flujo factible máximo”

Como el camino es válido, pasa por un $w \square$ diferente desde cada v_i , y podemos construir un flujo factible máximo con esto:

- Por cada cliente v_i del que sale el drone a $w \square$, asignamos una unidad de flujo que va de s a v_i , pasa por $w \square$, y termina en t .
- El flujo respeta capacidades porque son todas 1, y terminamos con $n - 1$ de flujo, que es el máximo posible.

Modelado: asignación - justificación

“Existe un camino válido \Leftarrow Existe un flujo factible máximo”

Como el flujo de cada arista es entero, podemos armar el siguiente camino:

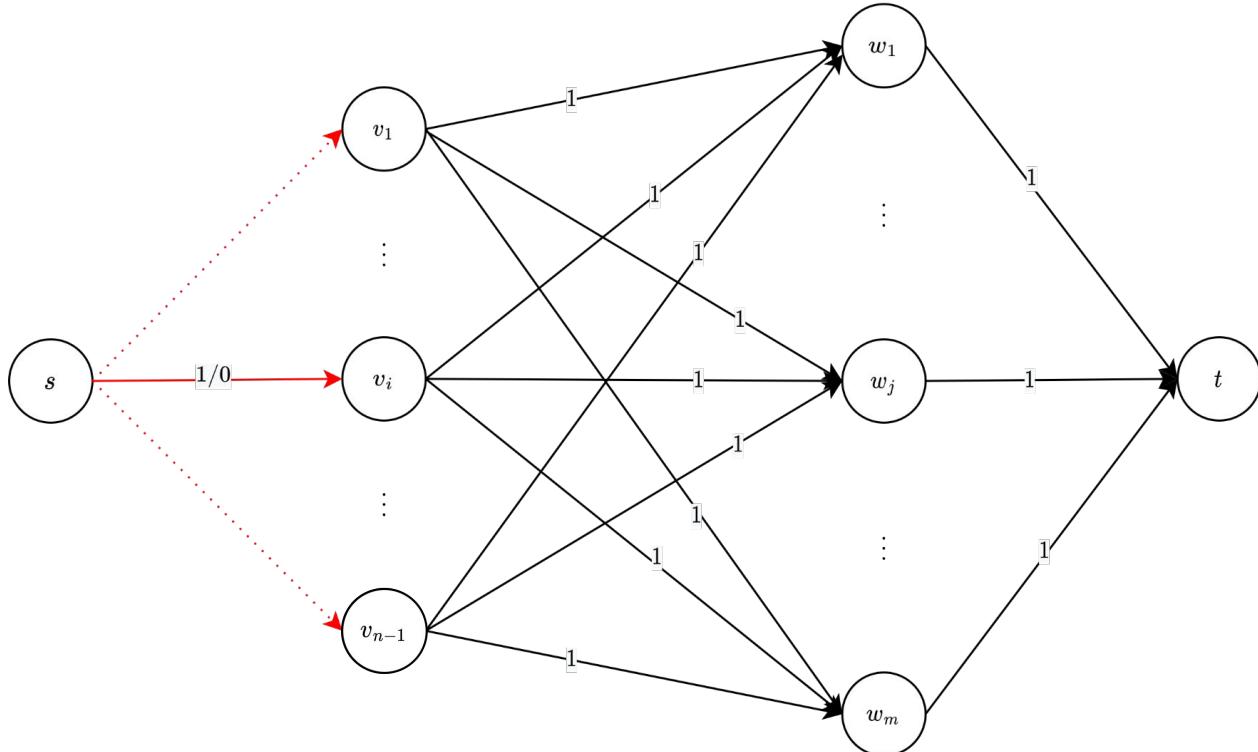
- Cada unidad de flujo pasa por un solo v_i y por un único $w\Box$.
 - Esto es así porque todas las capacidades son 1, entonces si ya pasa una unidad de s a v_i , esa arista está saturada y no puede pasar más, Lo mismo pasa con $w\Box$ a t .
- Nos construimos por cada unidad que pasa por un v_i y un $w\Box$, esa asignación.
- Como $m \geq n - 1$, siempre se pueden asignar todos los v_i a algún $w\Box$, entonces el flujo máximo es igual a $n - 1$, y tenemos una asignación completa.

Modelado: agregando costos

- Con este modelo podemos encontrar un camino válido que pasa por todos los v_i , y por $n - 1$ $w \square$, sin repetir 2 veces el mismo $w \square$.
- Ahora transformemos el modelo hacia uno que tenga costos también...

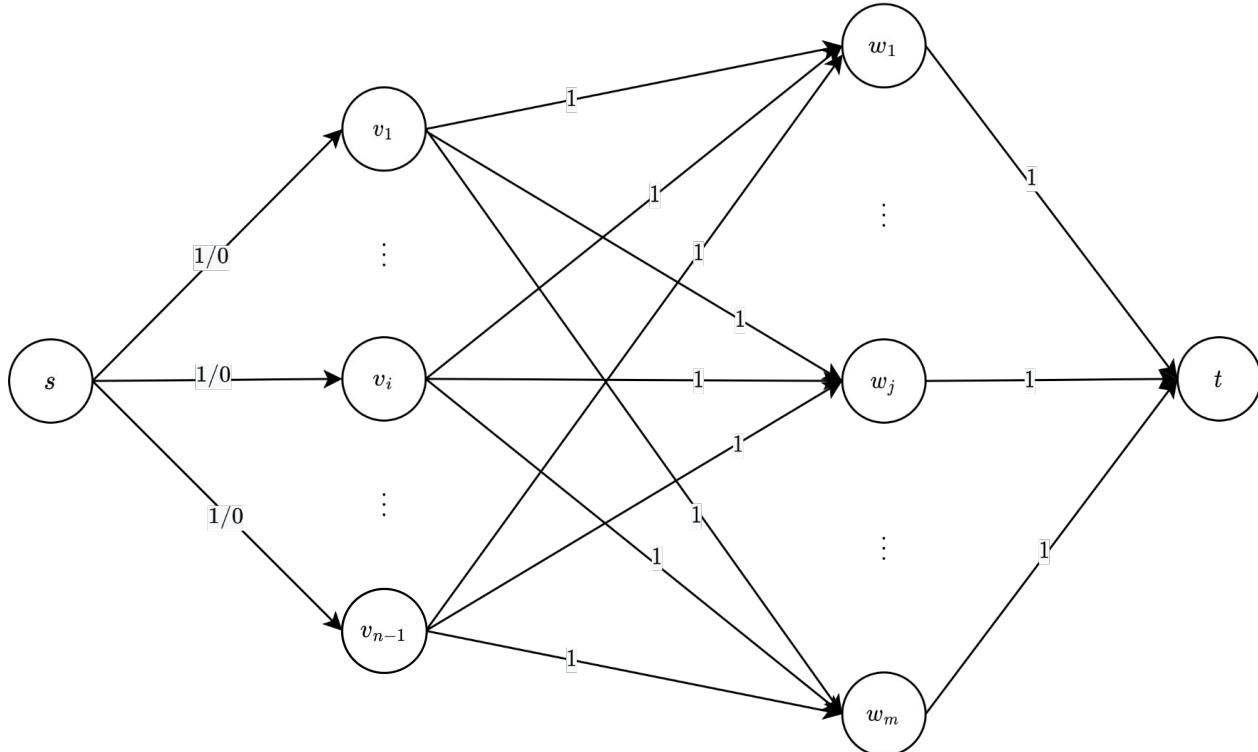
Modelado: agregando costos

Agregamos el costo ficticio de asignación de los v_i , que será el mismo para todos.



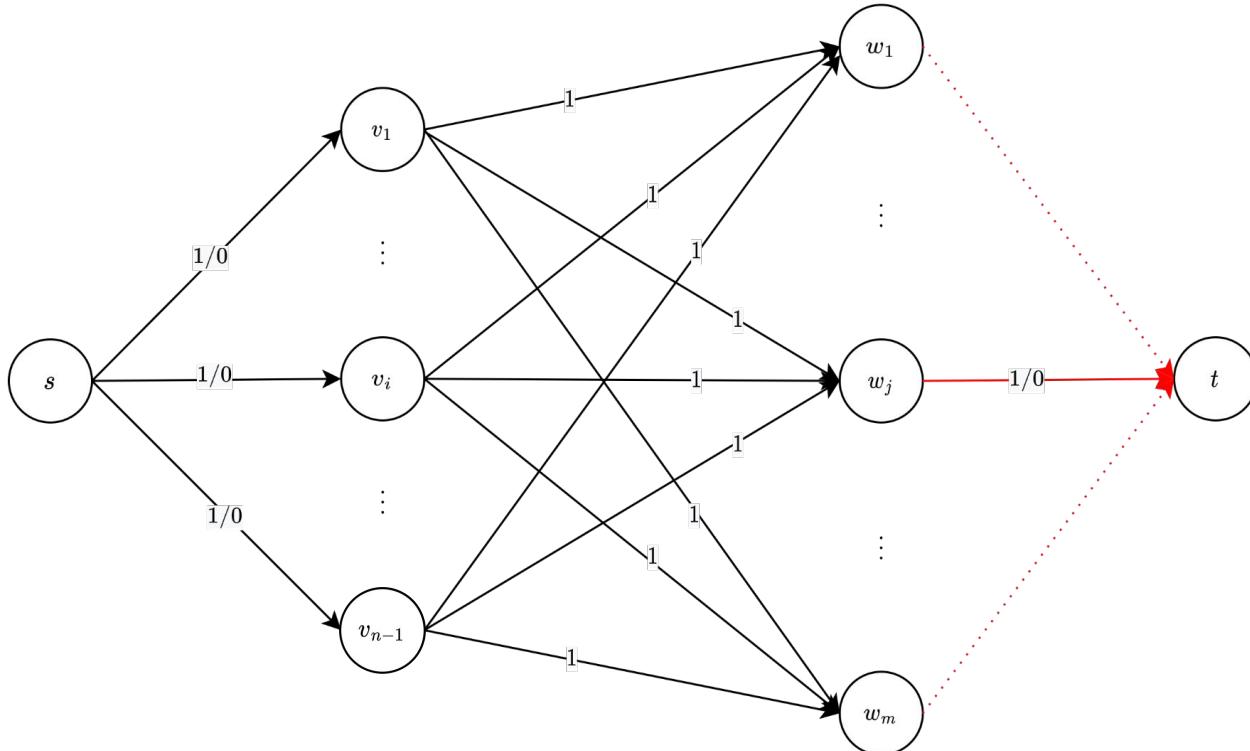
Modelado: agregando costos

Agregamos el costo ficticio de asignación de los v_i , que será el mismo para todos.



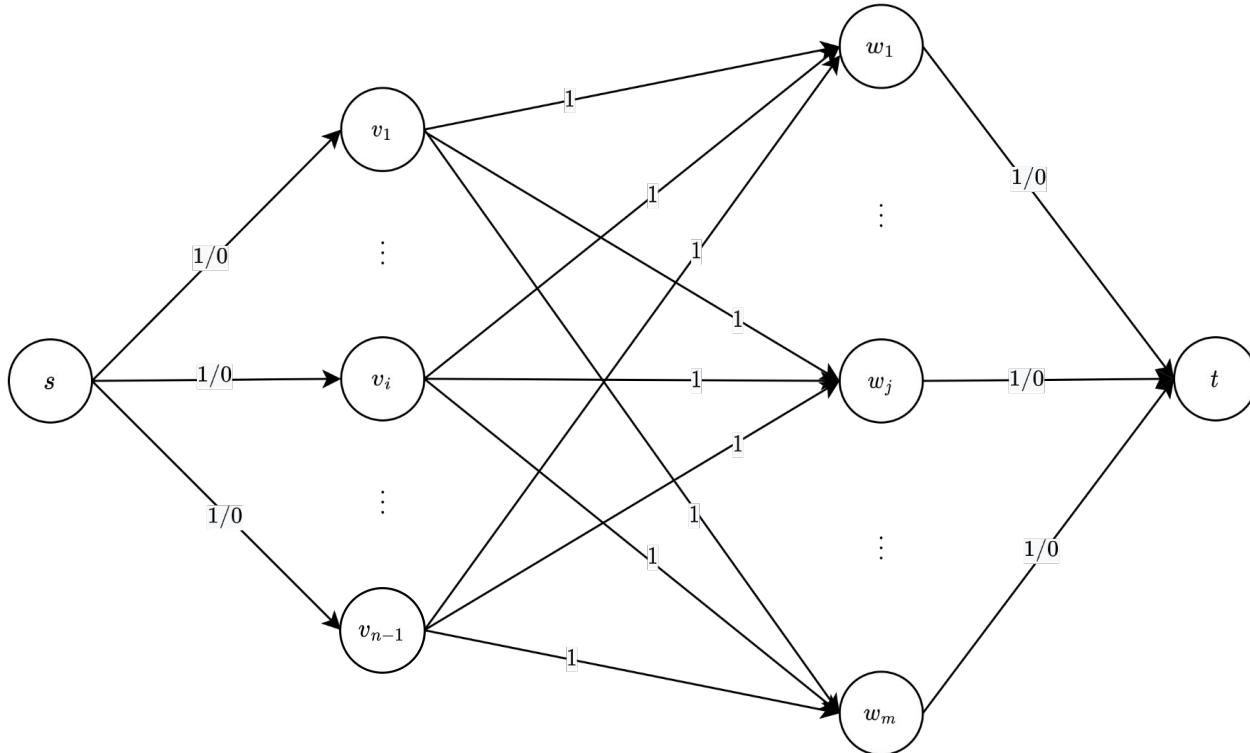
Modelado: agregando costos

Hacemos lo mismo para los $w \square$.



Modelado: agregando costos

Hacemos lo mismo para los $w \square$.



Modelado: agregando costos

Ahora pensemos en el resto de las características del problema para incorporar los costos que nos faltan al modelo.

Tenemos algunos costos por los viajes:

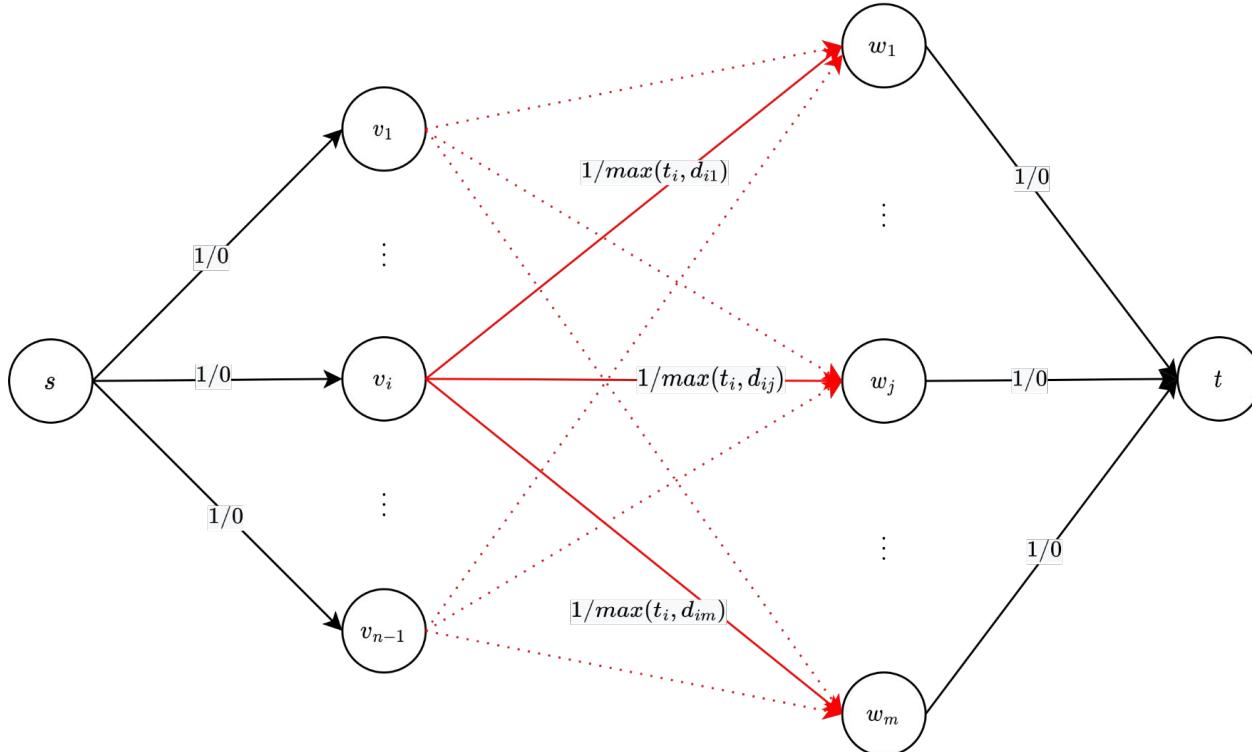
- El costo del camión para viajar entre 2 clientes consecutivos es t_i .
- El costo del drone para viajar de un v_i a un $w \square$ es de $d_i \square$.

¿Cuánto es el costo efectivo de ir de un cliente v_i al siguiente?

- Sabemos que el vehículo que llega primero espera al otro.
- El tiempo efectivo entonces depende del vehículo que tarde más.
- El costo entonces es el máximo entre t_i y $d_i \square$.

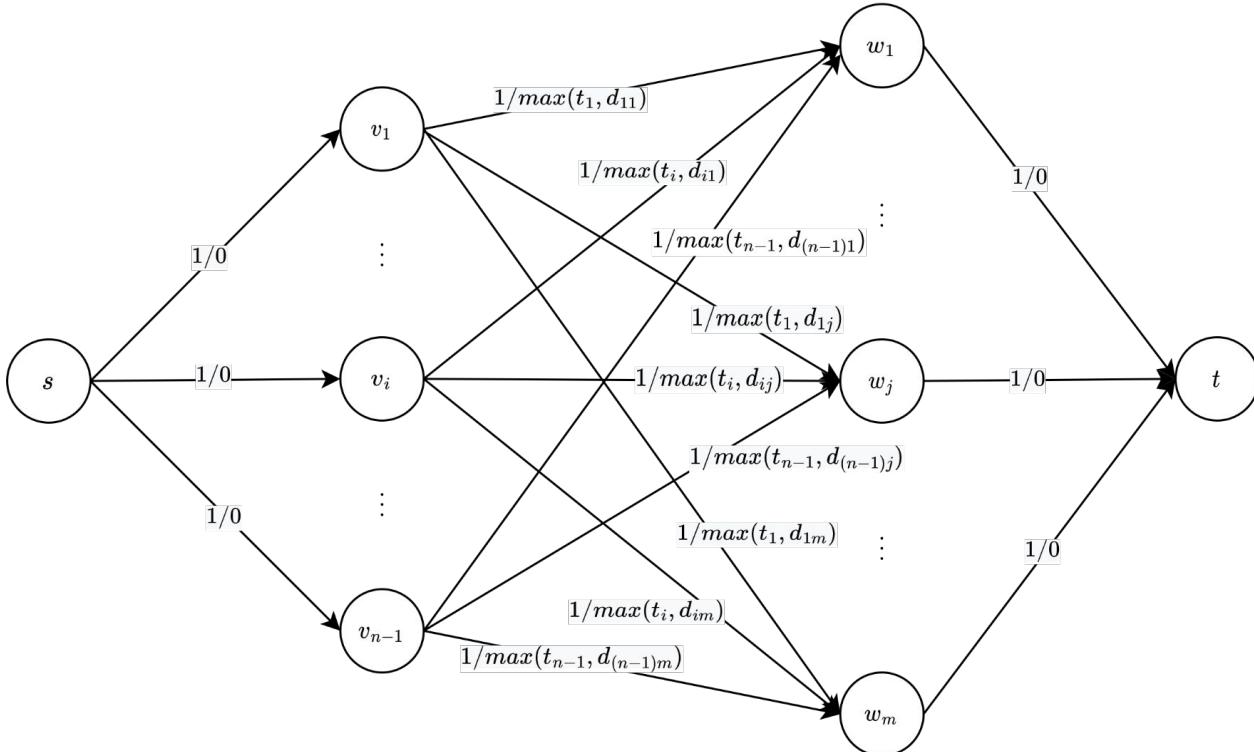
Modelado: agregando costos

Agregamos el costo de un v_i a un w_j con el $\max(t_i, d_{ij})$.



Modelado: agregando costos

Agregamos el costo de un v_i a un w_j con el $\max(t_i, d_{ij})$.



Modelado - justificación

Resta ver si este modelo encuentra el camino de menor costo, para lo cual vamos a extender la demostración anterior de la asignación así:

“Existe un camino válido de costo K \Leftrightarrow Existe un flujo factible máximo de costo K”

Modelado: asignación - justificación

“Existe un camino válido de costo K \Rightarrow Existe un flujo factible máximo de costo K”

Como el camino es válido, pasa por un $w \square$ diferente desde cada v_i , y podemos construir un flujo factible máximo con esto:

- Por cada cliente v_i del que sale el drone a $w \square$, asignamos una unidad de flujo que va de s a v_i , pasa por $w \square$, y termina en t .
- El flujo respeta capacidades porque son todas 1, y terminamos con $n - 1$ de flujo, que es el máximo posible.
- Por definición del peso de las aristas, para la unidad de flujo que hace el recorrido $s \rightarrow v_i \rightarrow w \square \rightarrow t$, el costo es $0 + \max(t_i, d_{ij}) + 0$.
- En total, el flujo cuesta:

$$\sum_{f(v_i \rightarrow w_j)} \max(t_i, d_{ij}) = \sum_{\text{drone pasa por } w_j \text{ desde } v_i} \max(t_i, d_{ij}) = K$$

Modelado: asignación - justificación

“Existe un camino válido de costo K \Leftarrow Existe un flujo factible máximo de costo K”

Como el flujo cada arista es entero, podemos armar el siguiente camino:

- Cada unidad de flujo pasa por un solo v_i y por un único $w\Box$.
 - Esto es así porque todas las capacidades son 1, entonces si ya pasa una unidad de s a v_i , esa arista está saturada y no puede pasar más, Lo mismo pasa con $w\Box$ a t .
- Nos construimos por cada unidad que pasa por un v_i y un $w\Box$, esa asignación.
- Como $m \geq n - 1$, siempre se pueden asignar todos los v_i a algún $w\Box$, entonces el flujo máximo es igual a $n - 1$, y tenemos una asignación completa.
- El costo de ese camino, es el del máximo entre lo que demora el camión en llegar desde cada v_i al siguiente y lo que tarda el drone pasando por $w\Box$.
- La elección del drone se basa en la arista con flujo positivo, así que tenemos:

$$\sum_{\text{drone pasa por } w_j \text{ desde } v_i} \max(t_i, d_{ij}) = \sum_{f(v_i \rightarrow w_j)} \max(t_i, d_{ij}) = K$$

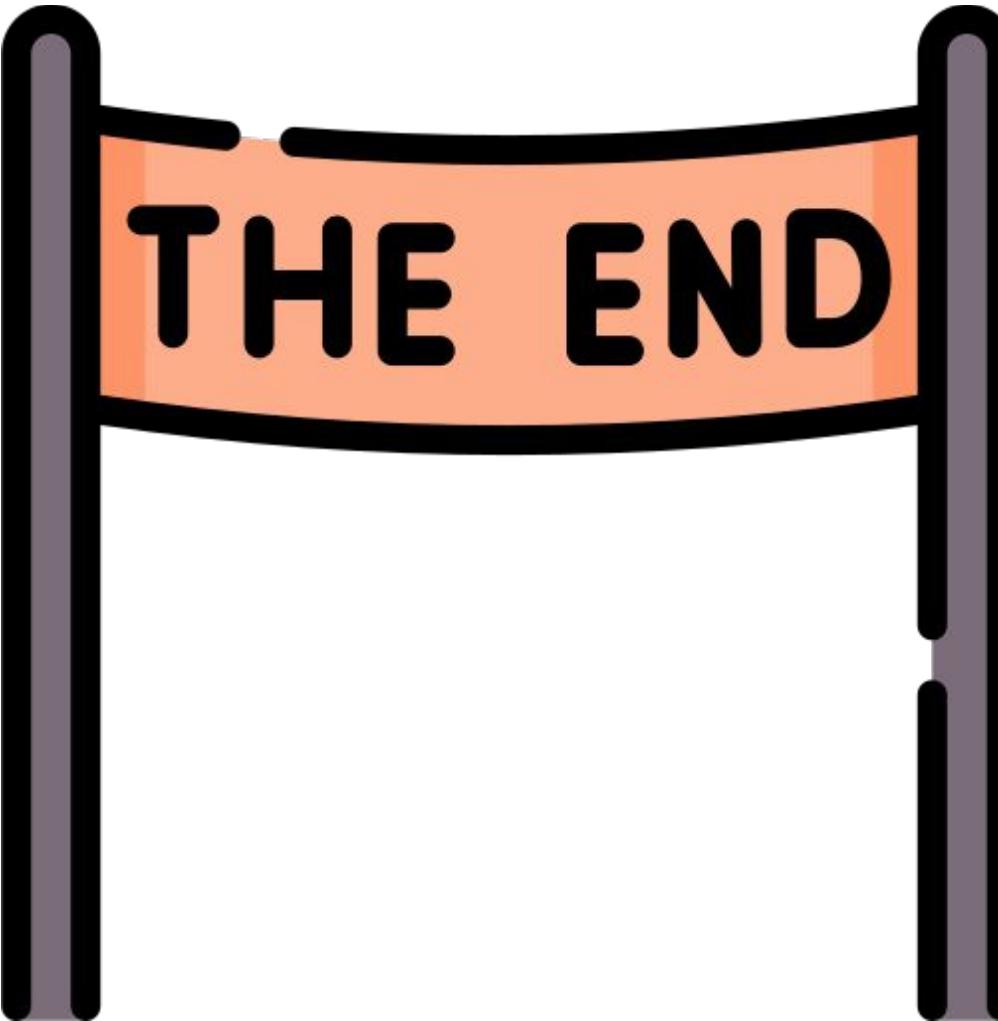
Resolución

Inciso (b)

¿Qué algoritmo usamos sobre esta red y que complejidad obtenemos?

Algoritmo + Complejidad

- Como estamos con una red que tiene capacidades y costos, estamos buscando el máximo flujo de costo mínimo (max cost - min flow).
- Podemos aplicar el algoritmo **successive shortest path**.
 - Aún más, como estamos en un problema de asignación, podemos aprovechar las optimizaciones y resolver esto en tiempo **$O(|V|^3)$** .
- Determinemos la cantidad de nodos:
 - Son $(n - 1 \text{ clientes } v_i) + (m \text{ clientes } w_j) = O(n + m)$.
 - Como $m \geq n - 1$, nos queda **$O(m)$** .
- Luego la complejidad final es **$O(m^3)$** , reemplazando $|V|$ por la cantidad de nodos del modelo.



A graphic design featuring a curved, orange rectangular banner with the words "THE END" written in large, bold, black capital letters. The banner is mounted on a white background by two thick, vertical black poles. The left pole has a horizontal bracket extending from its top to support the banner. The right pole is topped with a small, dark, vertical element.

THE END