Diseño de mecanismos

Sistemas Inteligentes Distribuidos

Sergio Alvarez
Javier Vázquez

Bibliografía

- *Artificial intelligence: a modern approach* (Russell & Norvig), cap. 17
- *Multiagent systems: algorithmic, game-theoretic, and logical foundations* (Shoham & Leyton-Brown), cap. 10
- *Game Theory, Alive* (Karlin & Peres), cap. 10, 14, 15, 16

Definición

Diseño de mecanismos

The cake problem

- Dos agentes, sin predisposición a cooperar, tienen que repartirse un pastel
- Colectivamente, ¿qué debería suceder?
- ¿Podemos influir de manera que los agentes, actuando en su propio interés, se comporten de manera que suceda lo que queremos que suceda?

Teoría de Juegos, al revés

- En Teoría de Juegos hemos visto cómo analizar un sistema multiagente definido como un juego
  - ¿Qué estrategias deben seguir los agentes racionales, dado un juego específico (una estructura de recompensas)?
- Le damos la vuelta: Diseño de mecanismos (Maskin, Myerson, Hurwicz)
  - ¿Es posible diseñar **un sistema multiagente que cumpla nuestros objetivos** como diseñadores, teniendo en cuenta que **los agentes del sistema podrían ser racionales y con incentivos asimétricos**?
  - En caso de existir tal sistema multiagente, **¿qué forma tendría y cómo se podría diseñar?**
  - ¿Es posible saber cuándo un sistema con estas características es **teóricamente imposible** que exista?

Teoría de Juegos, al revés

- El diseño de un mecanismo puede ser **adoptado voluntariamente** por un grupo que busca un comportamiento colectivo cooperativo
- También puede **imponer/promover la cooperación** en presencia de agentes poco fiables o impredecibles
  - Problema *principal - agent*: ¿puedo confiar en mi abogado/corredor de seguros/banquero...?
  - ¿Puedo crear una estructura de incentivos que garantice que el agente que me representa actúe en mi interés?

Definición

- Dados los siguientes elementos:
  - Un **conjunto de agentes** $N = \{1, 2, \dots, n\}$
  - Cada agente i tiene un **tipo** $\theta_i \in \Theta$ que representa sus **preferencias privadas**
  - Un conjunto de **resultados** O (e.g., asignaciones de tareas o candidatos de líder)
  - Una **función de utilidad** para cada agente: $u_i: O \to \mathbb{R}$
  - Una **función objetivo de elección social** $f: \theta_1 \times \theta_2 \times \dots \times \theta_n \to O$
  - Un **perfil estratégico** para cada agente: $S = (S_1, \dots, S_n)$
  - Una **función de resultado** $g: S_1 \times S_2 \times \dots \times S_n \to O$
- Los tipos de agentes y/o sus funciones de utilidad **pueden ser privados**
- Definimos un **mecanismo** (también llamado **institución**) como una tupla:

$$M = \langle N, \Theta, S, O, f, g \rangle$$

Teoría de la implementación

Diseño de mecanismos

Ejemplo (Maskin, 2007)

- Estamos diseñando un mecanismo para que una autoridad energética seleccione una fuente de energía para una sociedad compuesta por dos agentes consumidores, A y B
- Hay cuatro opciones energéticas: gas, petróleo, nuclear y carbón
- Hay dos estados posibles en el mundo:
  - Estado 1: los consumidores ponen poca importancia en el futuro
  - Estado 2: le dan mucha importancia (descuentan poco) al futuro
- A y B tienen incentivos conflictivos

Ejemplo (Maskin, 2007)

Estado 1

(preferencia por el presente)

A	B
gas	nuclear
petróleo	petróleo
carbón	carbón
nuclear	gas

Estado 2

(el futuro importa)

A	B
nuclear	petróleo
gas	gas
carbón	carbón
petróleo	nuclear

- La autoridad energética está interesada en seleccionar una fuente de energía con la que los consumidores estén *razonablemente* contentos
- ¿Cuál sería un resultado razonable para la autoridad energética?
  - ¿Cuál sería una función sobre las preferencias que cumpla los objetivos de diseño?

Ejemplo (Maskin, 2007)

Estado 1

A	B
gas	nuclear
petróleo	petróleo
carbón	carbón
nuclear	gas

Estado 2

A	B
nuclear	petróleo
gas	gas
carbón	carbón
petróleo	nuclear

- Una función que asigna resultados a conjuntos de preferencias es una función de elección social
- Ejemplo: $f(estado_1, estado_1) = petroleo$ y $f(estado_2, estado_2) = gas$ (Borda), $petroleo$ o gas al azar si no coinciden

Ejemplo (Maskin, 2007)

Estado 1

A	B
gas	nuclear
petróleo	petróleo
carbón	carbón
nuclear	gas

Estado 2

A	B
nuclear	petróleo
gas	gas
carbón	carbón
petróleo	nuclear

- ¿Y si el estado es desconocido (e.g. es información privada)?
- El agente A tiene un incentivo para decir que está en el estado 2, independientemente de su preferencia real, porque **siempre** prefiere el gas al petróleo (y B tiene un incentivo similar: estado 1)

Ejemplo (Maskin, 2007)

Estado 1

A	B
gas	nuclear
petróleo	petróleo
carbón	carbón
nuclear	gas

Estado 2

A	B
nuclear	petróleo
gas	gas
carbón	carbón
petróleo	nuclear

- La probabilidad de un resultado verdaderamente óptimo es de un 50%
- ¿Podemos obtener un mecanismo donde los agentes estén incentivados a revelar sus preferencias reales?

Ejemplo (Maskin, 2007)

Estado 1

A	B
gas	nuclear
petróleo	petróleo
carbón	carbón
nuclear	gas

Estado 2

A	B
nuclear	petróleo
gas	gas
carbón	carbón
petróleo	nuclear

A \ B	L	R
T	petróleo	carbón
D	nuclear	gas

- B en estado 1: equilibrio **(L,T)**
  - B tiene estrategia dominante **L**
  - A revela su tipo: **T** si estado 1, **D** si estado 2
- A en estado 2: equilibrio **(R,D)**
  - A tiene estrategia dominante **D**
  - B revela su tipo: **L** si estado 1, **R** si estado 2
- **No hace falta conocer el estado real y sin embargo conseguimos el óptimo verdadero**

Teoría de la implementación

- Hay solución a este juego, por lo que es un mecanismo que **implementa** la función de elección social f formulada anteriormente sin necesitar información privada:

$$f(estado_1, estado_1) = petroleo,\ f(estado_2, estado_2) = gas$$

- Una función de elección social f es **implementable** si, para un estado (perfil de tipos) θ y un resultado óptimo a: $f(\theta) = a$, **si cuando para todo estado θ' donde a no cae en el ranking de preferencias de ningún agente respecto de las alternativas, se mantiene el resultado óptimo**:

$$f(\theta) = a \rightarrow \forall \theta' : f(\theta') = a$$

Teoría de la implementación

- **Monotonía**: para todo a que es un resultado óptimo en algún estado, si no cae en el ranking de preferencias de ningún agente respecto de las alternativas en otro estado, es resultado óptimo en ese estado
- **Teorema de Maskin (1977):** si una función de elección social es implementable, entonces debe ser **monótona**

$$\forall a, \theta, \theta' :$$

$$a \in f(\theta) \wedge [\forall b, i : u_i(a, \theta) \geq u_i(b, \theta) \rightarrow u_i(a, \theta') \geq u_i(b, \theta')]$$

$$\rightarrow$$

$$a \in f(\theta')$$

Teoría de la implementación

Estado 1

A	B
gas	nuclear
petróleo	petróleo
carbón	carbón
nuclear	gas

Estado 2

A	B
nuclear	petróleo
gas	gas
carbón	carbón
petróleo	nuclear

- $f(estado_1) = petroleo$, y $petroleo$ cae en el ranking de A en el estado 2 (no se viola la monotonía)
- $f(estado_2) = gas$, y gas cae en el ranking de B en el estado 2 (no se viola la monotonía)
- Por lo tanto f es monótona (*implementable* → *monótona*)

Teoría de la implementación

Estado 1

A	B
gas	nuclear
petróleo	petróleo
carbón	carbón
nuclear	gas

Estado 2

A	B
gas	**nuclear**
petróleo	**petróleo**
nuclear	**carbón**
carbón	**gas**

- Supongamos una f' definida como una cuenta de Borda con desempate por pluralidad
  - $f'(estado_1) = petroleo, f'(estado_2) = nuclear$
- $f'(estado_1) = petroleo$, y $petroleo$ **no cae** en el ranking de A en el estado 2, donde **el óptimo ya no es $petroleo$ sino $nuclear$**
- Por lo tanto f' **no** es monótona (y por lo tanto **no implementable**)

Teoría de la implementación

- Condiciones para que una función de elección social tenga una implementación con equilibrio dominante garantizado (DSIC):
  - **Monotonicidad**
  - **Compatibilidad de incentivos (incentive-compatible, IC):** cada agente maximiza su utilidad declarando su tipo verdadero
- Condiciones para que una función de elección social tenga una implementación con equilibrio de Nash garantizado (BNIC):
  - Compatibilidad de incentivos (incentive-compatible, IC)
  - Monotonicidad
  - **Inexistencia de agentes con veto**

Compatibilidad de incentivos

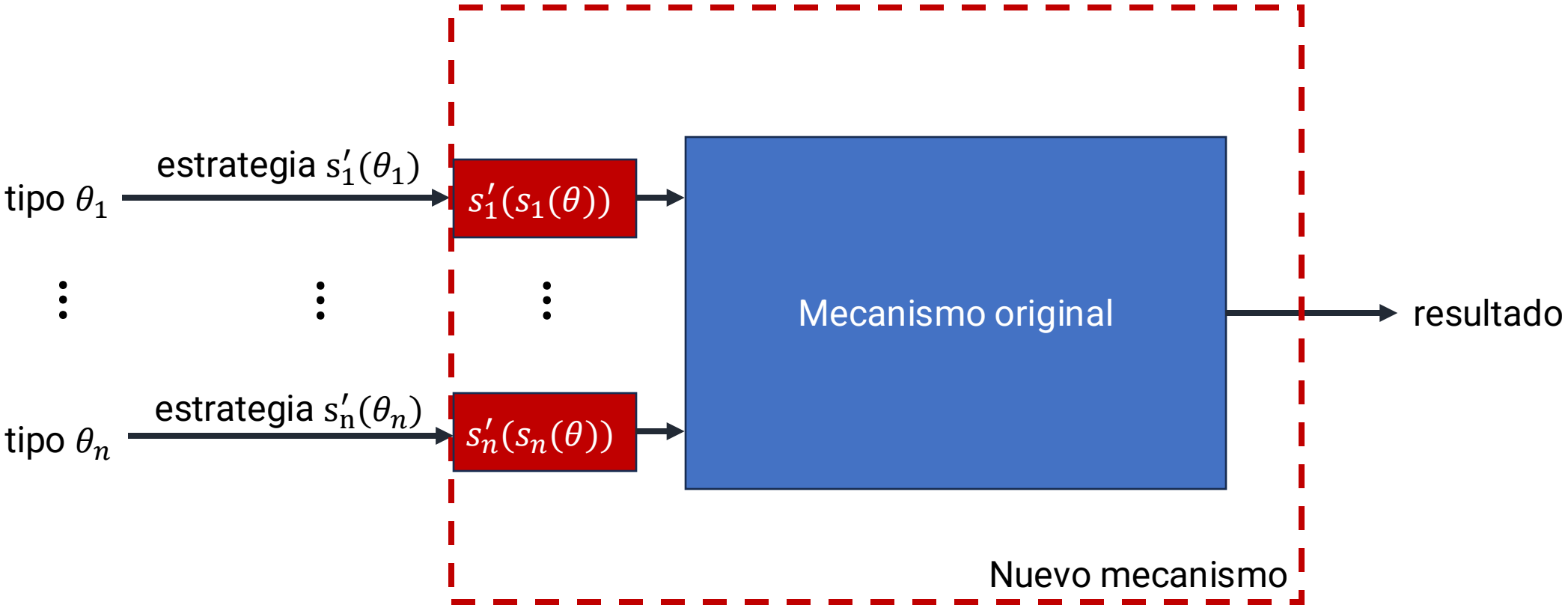
- Este es un resultado muy valioso, ya que podemos diseñar funciones de elección social que se ajusten a nuestros objetivos y **verificar si existe alguna implementación posible**
  - Por ejemplo, podemos diseñar y validar funciones que sean Pareto-eficientes
- Una **condición necesaria** para que una función de elección social sea implementable es la **compatibilidad de incentivos**: cada agente maximiza su utilidad declarando su tipo verdadero
  - Ya sea con una estrategia dominante o llegando a un equilibrio de Nash
- Sin embargo, ¿cómo garantizamos que existe un mecanismo con compatibilidad de incentivos?
  - **¿Qué pasa cuando los agentes no tienen incentivo para decir la verdad?**

Principio de revelación (Myerson, 1979)

- Para cualquier **mecanismo arbitrario**…
  - Sea **directo o indirecto**
  - **Sea o no compatible con los incentivos** de los agentes
- …existe un **mecanismo directo equivalente** en el que los participantes están **incentivados a declarar sus tipos verdaderos**
- Intuición: a partir de un mecanismo arbitrario M, podemos dejar que los agentes escojan sus tipos θ_i (sin importar si mienten o no) y simulamos M para encontrar las estrategias s_i en equilibrio, obteniendo resultado O
  - Creamos nuevos tipos θ_i' con estrategias s_i' creadas a partir del perfil s (e.g. si una estrategia s_i es indirecta, se puede reducir a una estrategia s_i' indirecta)
  - El nuevo mecanismo M' se define por los nuevos tipos θ' que generarán un resultado $O' = O$ por lo que los agentes estarán incentivados a declarar sus tipos verdaderos

Principio de revelación (Myerson, 1979)

Principio de revelación (Myerson, 1979)



Principio de revelación (Myerson, 1979)

- El conjunto de equilibrios posibles no es el mismo entre M y M'
  - **Sí se garantiza que el equilibrio original (simulado) de M se mantiene en M'**
  - Especialmente, en mecanismos procedentes de mecanismos indirectos es difícil garantizar que los nuevos equilibrios sean tan buenos o más que el original, tanto en utilidad individual como colectiva
- **El principio de revelación reduce la complejidad del estudio de mecanismos**
  - **No es imprescindible tener agentes que hagan públicas sus preferencias para diseñar un mecanismo que maximice el bienestar**
  - **Los mecanismos indirectos no son más *expresivos* que los directos**
  - **Podemos reducir el estudio de mecanismos al análisis de los mecanismos directos y con compatibilidad de incentivos**

Diseño de mecanismos como problema de optimización

- El diseño de mecanismos se puede ver como el problema de encontrar el mejor mecanismo posible, dado un **conjunto de restricciones impuestas por las necesidades del diseñador**: es un **problema de búsqueda**
  - Sobre un espacio de juegos en forma normal o repetida/extensiva
  - Sobre un espacio de funciones de elección social
- Estas restricciones pueden ser utilizadas como acotaciones sobre el espacio de búsqueda o como sistemas de ecuaciones lineales a optimizar
  - CSP/COP
  - Aprendizaje por refuerzo
  - Métodos de investigación operativa, e.g. Simplex, optimización lineal/convexa
  - …

Ejemplos de restricciones típicas

- **Veracidad**
  - El mecanismo incentiva a los agentes a expresar sus preferencias verdaderas ($\hat{v}$)
  - Se cumple cuando el equilibrio del mecanismo es una decisión x tal que:

$$\forall i: \hat{v_i}(x) = v_i(x)$$

- **Eficiencia**
  - El mecanismo es estrictamente Pareto-eficiente
  - Se cumple cuando el equilibrio del mecanismo es una decisión x tal que:

$$\forall v \forall x': \sum_i v_i(x) \geq \sum_i v_i(x')$$

Ejemplos de restricciones típicas

- **Equilibrio presupuestario**
  - El mecanismo, en el equilibrio, reparte lo que tiene/recibe/recolecta
  - Si el perfil de estrategias en el equilibrio es s:

$$\forall v: \sum_i p_i\big(s(v)\big) = 0$$

- **Racionalidad individual**
  - Ningún agente pierde por participar en el mecanismo
- **Tratabilidad**
  - Se puede ejecutar en tiempo polinómico
- **Maximización/minimización de ingresos**
  - En equilibrio, el mecanismo maximiza/minimiza la suma de los precios pagados por los agentes
- **Justicia maxmin**
  - En equilibrio, el mecanismo maximiza la recompensa del agente que tiene la menor recompensa entre todos los agentes
- **Minimización del precio de anarquía**
  - En equilibrio, el mecanismo minimiza la diferencia entre el óptimo bienestar y el bienestar en dicho equilibrio

Subastas

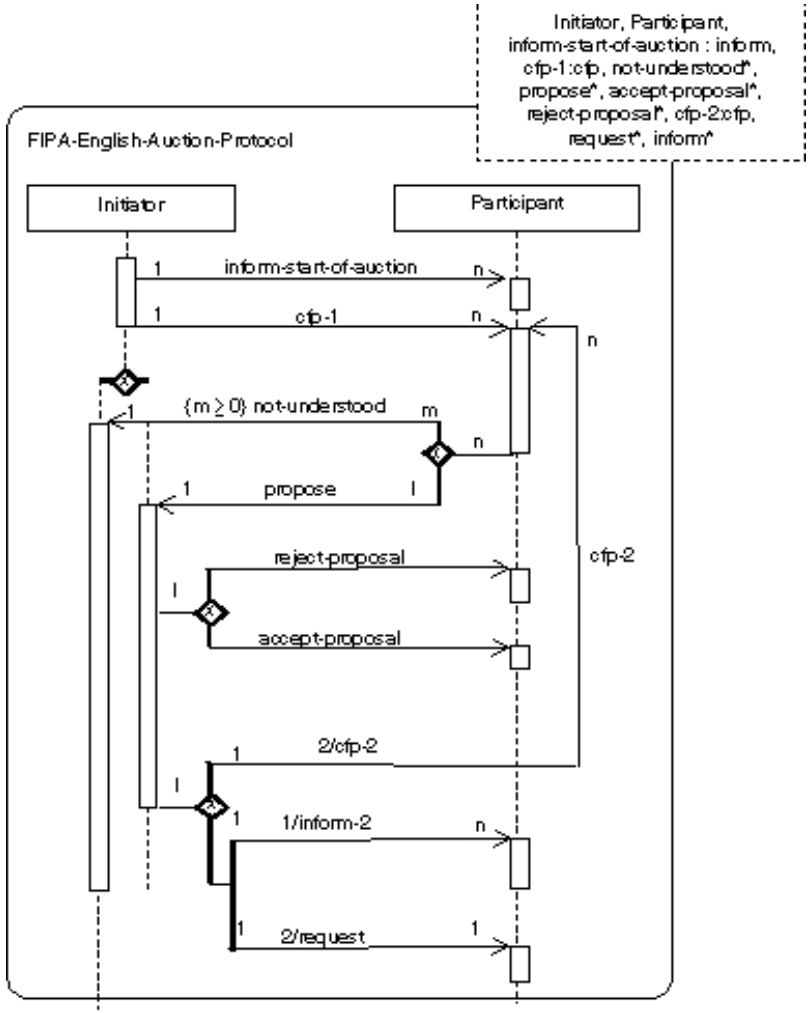
Diseño de mecanismos

Mecanismos de mercado

- Forma básica de mecanismo de mercado: vendedor y comprador (agentes egoístas) negocian un precio p para una transacción por un bien g
  - El vendedor tiene un precio de reserva p_s: $p \geq p_s$
  - El comprador tiene un precio de reserva p_b: $p \leq p_b$
- Cualquier precio propuesto p tal que $p_s \leq p \leq p_b$ es una solución válida
  - Como vimos, podemos hablar de utilidades en lugar de precios: $u = p - v$ (vendedor), $u = v - p$ (comprador)
  - Los precios de reserva se suelen definir en el punto en el que la utilidad (u) es 0, en este caso $p_s = v_s$ y $p_b = v_b$
- Los problemas de regateo vistos en el tema de Competición son formas simples de mecanismos de mercado
  - El juego del ultimátum para N turnos es un mecanismo indirecto e incentivo-compatible

Modelos de subasta

- Otra forma de mecanismo de mercado
- Mecanismos directos e indirectos
  - Protocolo estricto basado en comunicación (e.g., FIPA)
- **No necesariamente incentivo-compatibles**
  - Las políticas de licitación son privadas de cada agente
- Forma general (**problema de asignación**):
  - Tenemos n agentes y n objetos
  - Para cada agente i y objeto j, u_{ij} es la utilidad para i
  - Queremos encontrar una asignación uno a uno $(1, j_1), \ldots, (n, j_n)$ tal que maximicemos: $\sum_{i=1}^{n} u_{i j_i}$



Naive auction (Bertzekas, 1990)

- Una manera ingenua (*naive*) de intentar esta maximización es definiendo que un agente está **contento** si su asignación maximiza el valor v_i donde

$$v_i = (u_i - p_i)$$

- El precio p_i es público, pero la utilidad u_i no
- El equilibrio de esta manera no está garantizado, por lo que aplicamos una relajación:
  - Hay equilibrio cuando todos los agentes están **quasi-contentos**

$$\forall i\colon u_{ij_i} - p_{j_i} \geq \max_{j=1,\ldots,n} \{u_{ij} - p_j\} - \epsilon$$

Naive auction (Bertzekas, 1990)

```
función naive_auction(agentes, objetos)
    precios :- random_prices()
    asignación :- asignación_aleatoria(agentes, objetos, precios)
    mientras no todos_contentos(asignación) hacer
        i :- seleccionar_agente_no_contento(asignación)
        j_i :- seleccionar_mejor_objeto(i, objetos, precios)
        i_prima :- máximo_pujador(asignación, j_i)
        asignación :- intercambiar_asignación(asignación, i, i_prima)
        asignación :- pujar(asignación, j_i, ?)
    retornar asignación
```

Por lo general, el incremento es por una cantidad entre:

- 0, y
- La diferencia entre el valor del mejor objeto para i (j_i) y el valor del segundo mejor objeto para i (de lo contrario, ya no tiene sentido pujar por j_i)

Naive auction (Bertzekas, 1990)

- Si v_i es el valor del mejor objeto y w_i es el valor del segundo mejor objeto para i en una determinada iteración, y los precios están en un espacio discreto ($\forall p: p \in \mathbb{N}$), el siguiente incremento en la puja (el interrogante que hemos dejado en el algoritmo):

$$\gamma_i = v_i - w_i + \epsilon$$

- Garantiza:
  - Que el algoritmo termina en tiempo finito
  - Que se alcanza un equilibrio siempre y cuando $\epsilon < \frac{1}{|A|}$ donde A es el conjunto de agentes
- Limitaciones:
  - No es generalmente aplicable (no actúa sobre $\mathbb{R}$)
  - El equilibrio no es un equilibrio dominante ni equilibrio de Nash teórico (depende de ϵ)
  - **No es incentivo-compatible**: para que haya convergencia, las pujas no pueden ser libres
  - **Es un mecanismo indirecto**

Subasta de primer precio

- En una subasta de primer precio, los agentes presentan ofertas de precio privadas, y **el agente con la oferta más alta gana y paga el total del precio que propuso**
- Es un **mecanismo directo** que es equivalente al mecanismo indirecto de subasta holandesa (descendente)
- La subasta de primer precio **no es incentivo-compatible**
  - Si un agente puja su precio de reserva, en caso de ganar la subasta su utilidad es 0 (debido a que, en este caso, $p = v$) por lo que le es indiferente ganar o no
  - Por lo tanto, todos los agentes tienen el incentivo de encontrar un precio más bajo que su precio de reserva pero más alto que el de la puja más alta de los demás agentes (desconocida)
  - En consecuencia, en el equilibrio es posible que el agente con el precio de reserva más alto no se lleve el objeto

Subasta Vickrey (de segundo precio)

- En una subasta de segundo precio (Vickrey-Clarke-Groves o VCG), los agentes presentan ofertas de precio privadas, y **el agente con la oferta más alta gana y paga el total del precio de la segunda oferta más alta**
- Es un mecanismo directo que es equivalente a los mecanismos indirectos de subasta inglesa (ascendente) de un solo objeto o la subasta japonesa
- Este mecanismo es incentivo-compatible
  - **Los agentes tienen el incentivo de ofrecer su precio de reserva real**

Demostración de incentivo-compatibilidad en VCG

- El agente i gana, y el agente i' quedó en segunda posición
- i hizo una oferta por p_i pero pagará $p_{i'}$
- Supongamos ahora que i tenía como precio de reserva real $q_i \neq p_i$
  - Si $q_i < p_i \wedge q_i < p_{i'}$: i se queda sin el objeto, cuando podría haber pujado más
  - Si $q_i < p_i \wedge q_i > p_{i'}$: a i le era indiferente ofrecer p_i en vez de q_i, porque el precio a pagar en caso de ganar depende de $p_{i'}$, no de p_i
  - Si $q_i > p_i$: i corre el riesgo de que $p_{i'} > p_i$ y acabe pagando más que su precio de reserva, perdiendo utilidad
- No ganar representa una pérdida en utilidad, porque al ganar se paga $p_{i'}$ y sabemos que $p_i > p_{i'}$, por lo que

$$u_i = v_i - p_{i'} > 0$$

- Por lo tanto, **en VCG, todos los agentes están incentivados a mostrar su preferencia real** y, además:
  - Es Pareto-eficiente
  - Minimiza el precio de anarquía

Ejemplo: subasta + consenso

Diseño de mecanismos

Modelo de subasta descentralizada

- Source: Choi, H. L., Brunet, L., & How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. IEEE transactions on robotics, 25(4), 912-926. [pdf]
- CBAA: Consensus-based bundle algorithm
  - Objetivo: coordinar (de forma descentralizada) una flota de vehículos autónomos
- Combinación de dos algoritmos descentralizados:
  - Subasta de mercado
  - Consenso para redes de comunicación no completas
- Garantías (bajo ciertas restricciones de modelización):
  - Convergencia
  - Asignación libre de conflictos
  - $recompensa\_global \geq (0{,}5 * óptimo\_teórico)$

Auction Model: CBAA

Auction Model: CBAA

recompensa - coste

valor > puja_actual

```
function select_task(a_i, assignment, bid_list)
    local_value_function = get_local_value_function(a_i)
    if not have_assignment(assignment, a_i)
        good_tasks = filter_tasks_by_positive_difference(bid_list, local_value_function)
        if not empty(good_tasks)
            best_task = best_task_by_value(good_tasks, local_value_function)
            assignment = assign_task(assignment, best_task, a_i)
            current_bid = get_current_bid(bid_list, best_task)
            new_bid = choose_value_between(current_bid, local_value_function(best_task))
            bid_list = add_bid(bid_list, best_task, a_i, new_bid)
        end_if
    end_if
    return assignment, bid_list
end_function
```

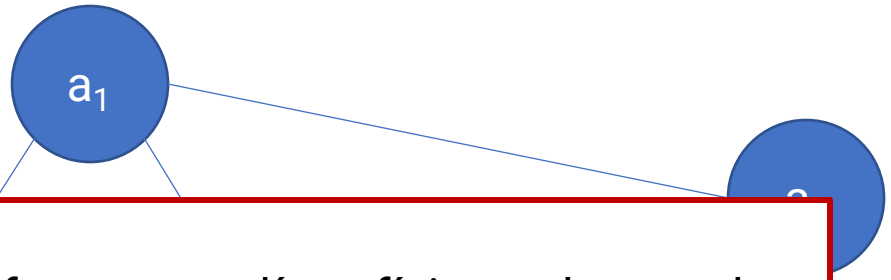
Algoritmo CBAA

recompensa - coste

valor > puja_actual

```
function select_task(a_i, assignment, bid_list)
    local_value_function = get_local_value_function(a_i)
    if not have_assignment(assignment, a_i)
        good_tasks = filter_tasks_by_positive_difference(bid_list, local_value_function)
        if not empty(good_tasks)
            best_task = best_task_by_value(good_tasks, local_value_function)
            assignment = assign_task(assignment, best_task, a_i)
            current_bid = get_current_bid(bid_list, best_task)
            new_bid = choose_value_between(current_bid, local_value_function(best_task))
            bid_list = add_bid(bid_list, best_task, a_i, new_bid)
        end_if
    end_if
    return assignment, bid_list
end_function
```

Ejemplo CBAA



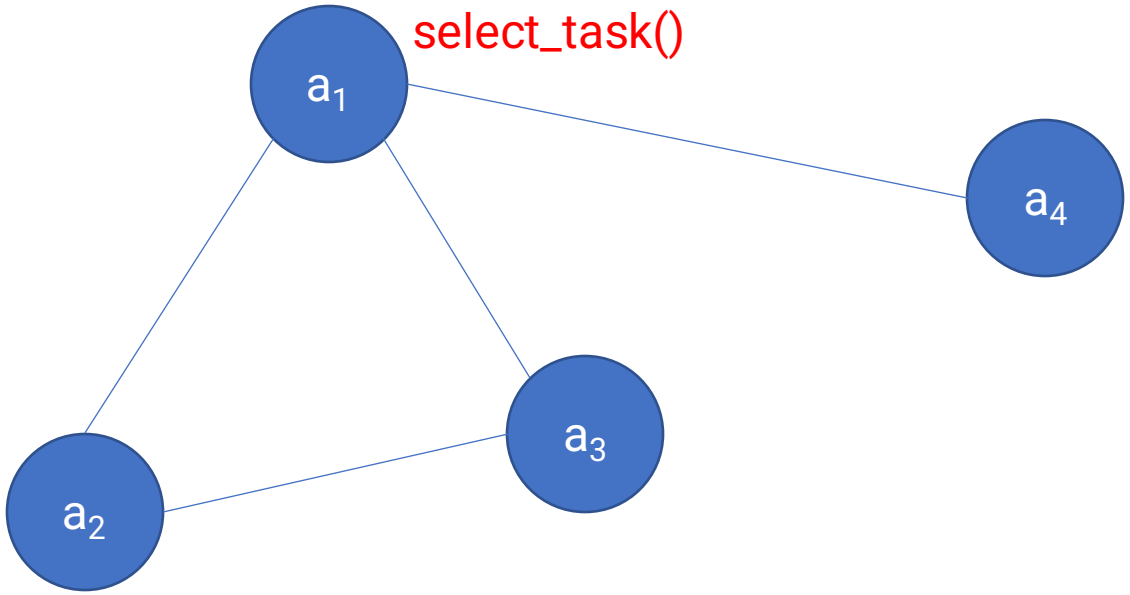
Suposiciones:
$s_1 - s_2 - s_3 - s_4$ forman una línea física y el coste de pasar de s_i a s_{i+1} es 1. Inicialmente, para todo i, el agente a_i está físicamente en s_i
Los agentes distribuyen sus preferencias de modo que la suma de sus recompensas es 10. Los agentes siempre pujan el máximo posible sin perder utilidad.
El algoritmo de consenso elude la topología y lo ejecutamos después de cada modificación de la lista de pujas.

r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
nt				
bids	0	0	0	0

Ejemplo CBAA



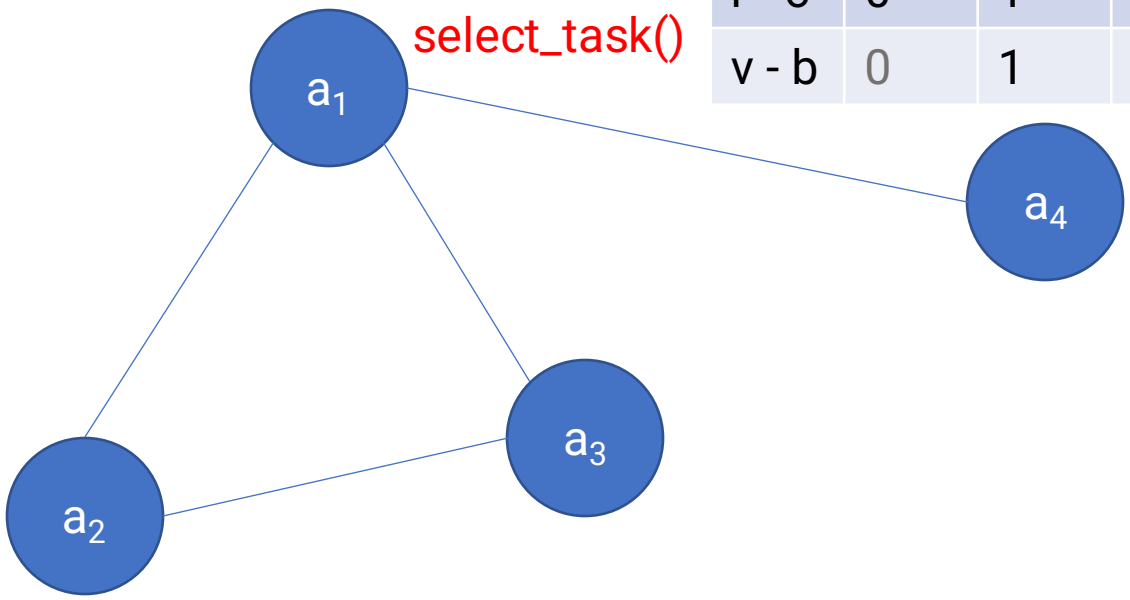
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment				
bids	0	0	0	0

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	0	1	1	2
v - b	0	1	1	2



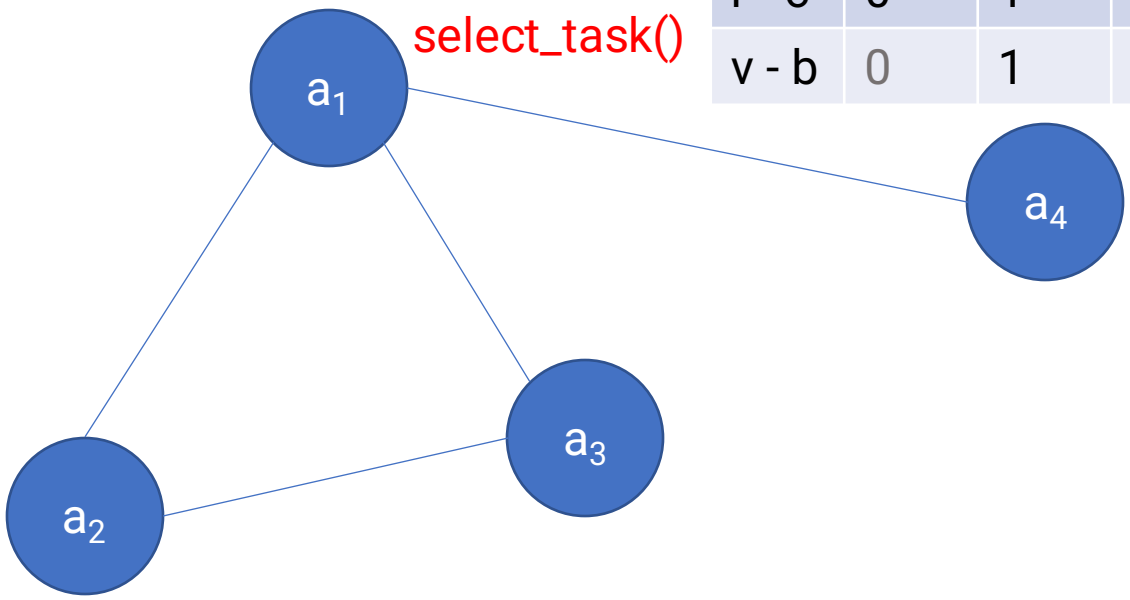
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment				
bids	0	0	0	0

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	0	1	1	2
v - b	0	1	1	2



r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

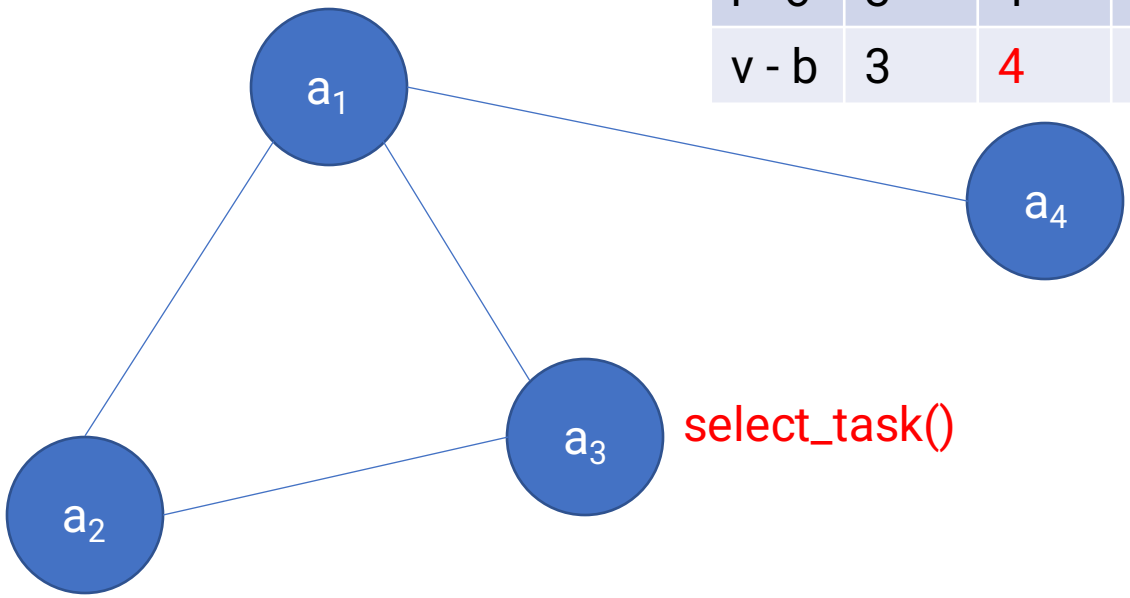
c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment				a_1
bids	0	0	0	2

Could have been any value in (0, 2]

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	3	4	0	-1
v - b	3	4	0	-3



select_task()

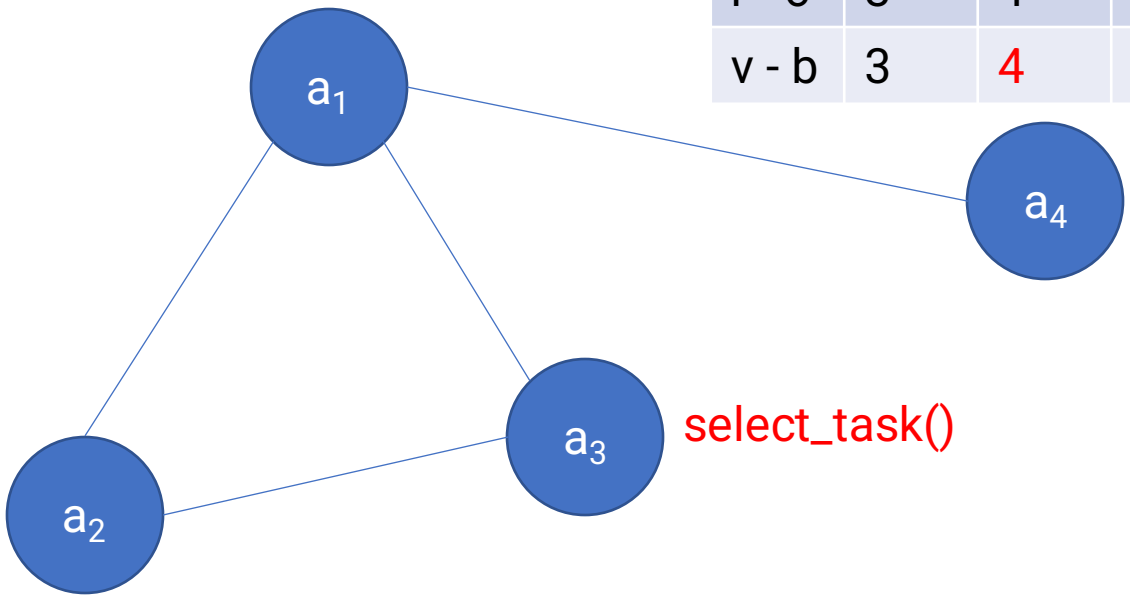
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment				a_1
bids	0	0	0	2

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	3	4	0	-1
v - b	3	4	0	-3



select_task()

r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

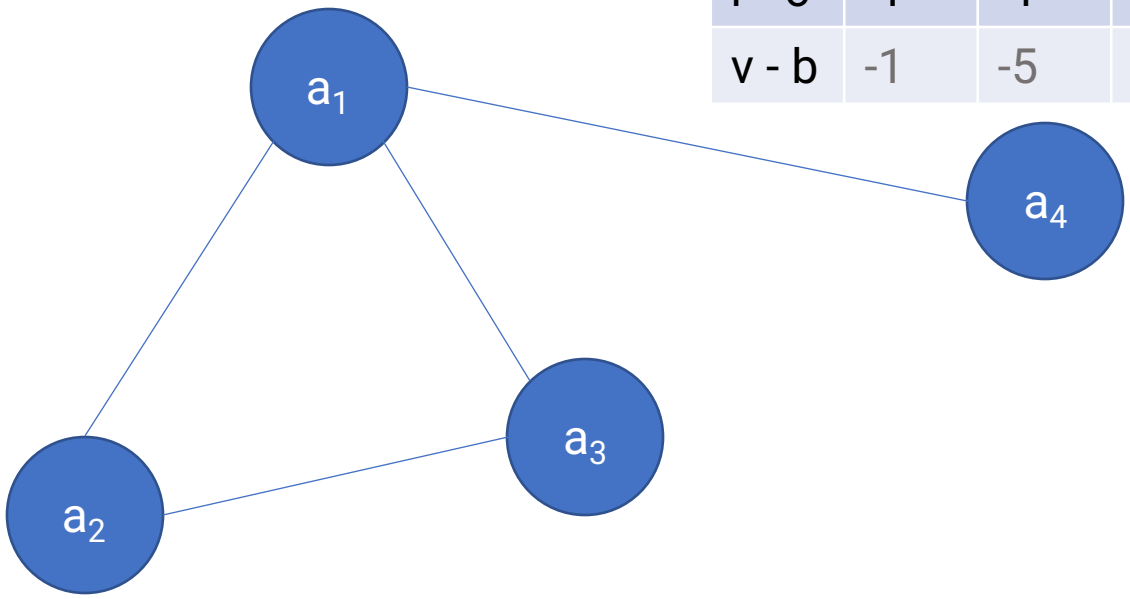
c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment		a_3		a_1
bids	0	4	0	2

Could have been any value in (0, 4]

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	-1	-1	1	5
v - b	-1	-5	1	3



select_task()

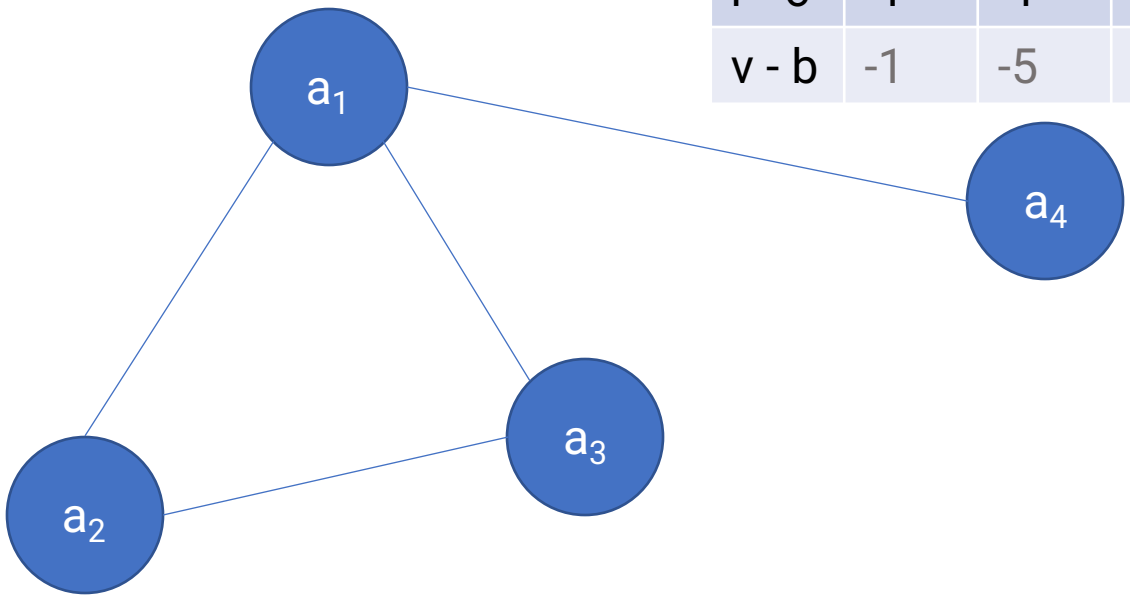
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment		a_3		a_1
bids	0	4	0	2

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	-1	-1	1	5
v - b	-1	-5	1	3



select_task()

r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

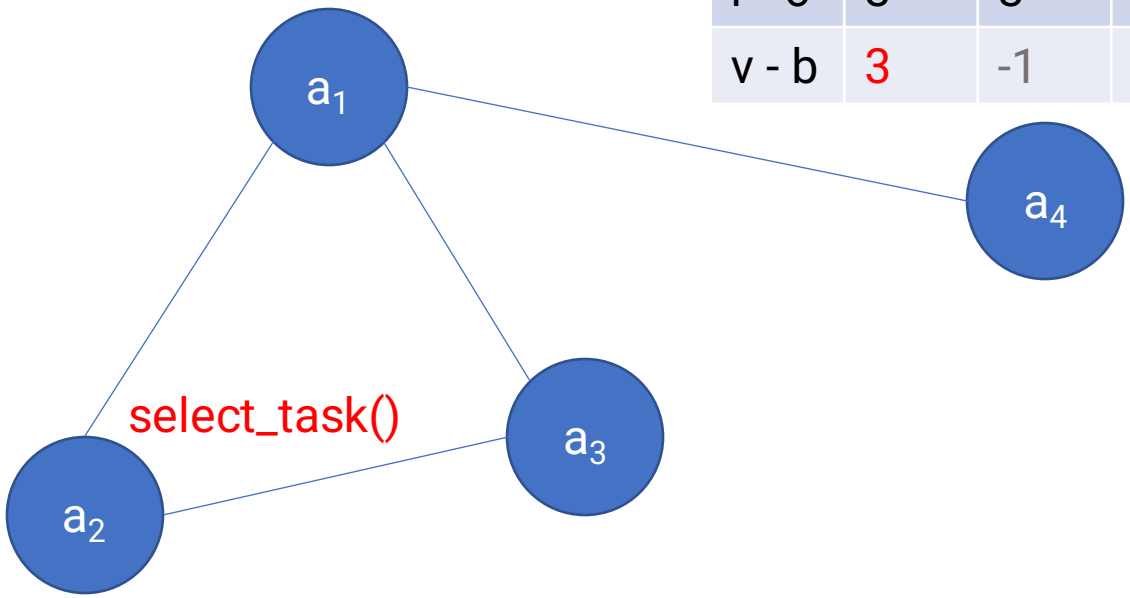
c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment		a_3		a_4
bids	0	4	0	5

Could have been any value in (2, 5]

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	3	3	1	-1
v - b	3	-1	1	-6



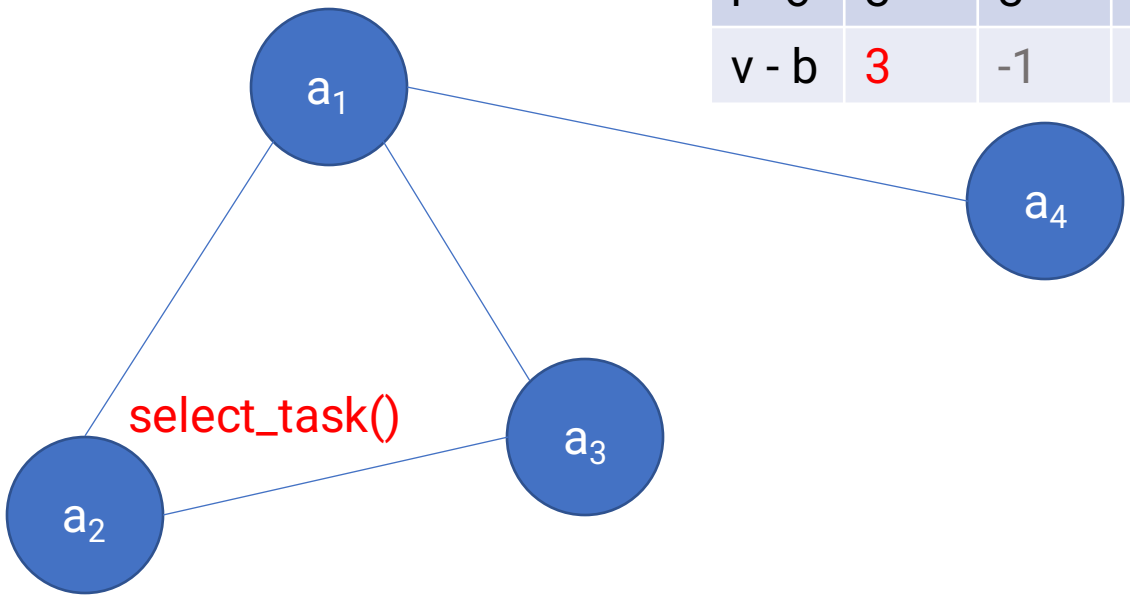
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment		a_3		a_4
bids	0	4	0	5

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	3	3	1	-1
v - b	3	-1	1	-6



r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

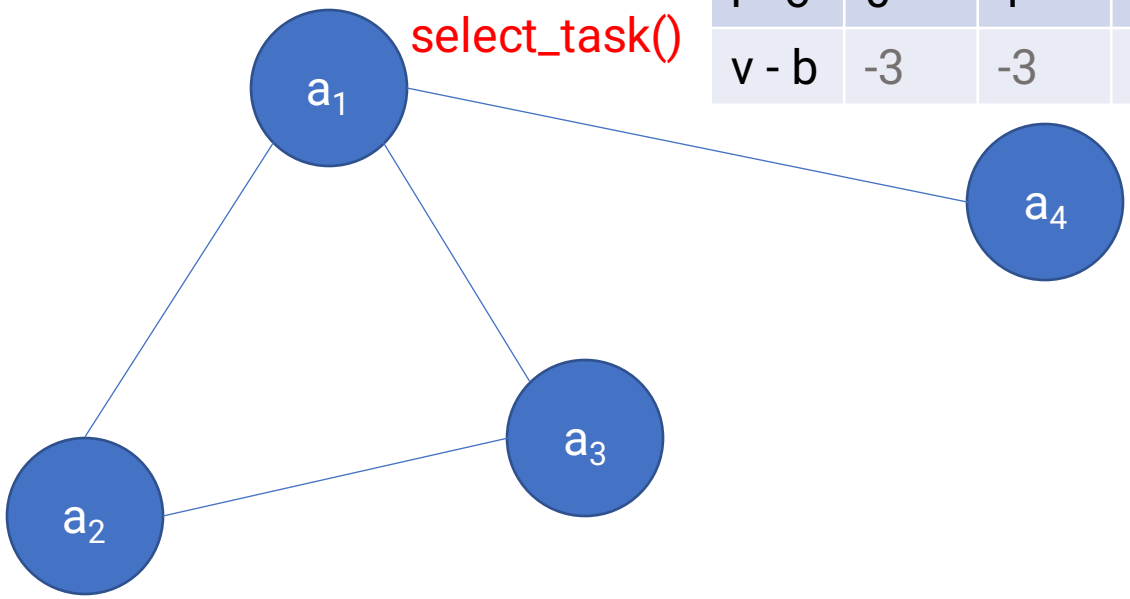
c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment	a_2	a_3		a_4
bids	3	4	0	5

Could have been any value in (0, 3]

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	0	1	1	2
v - b	-3	-3	1	-3



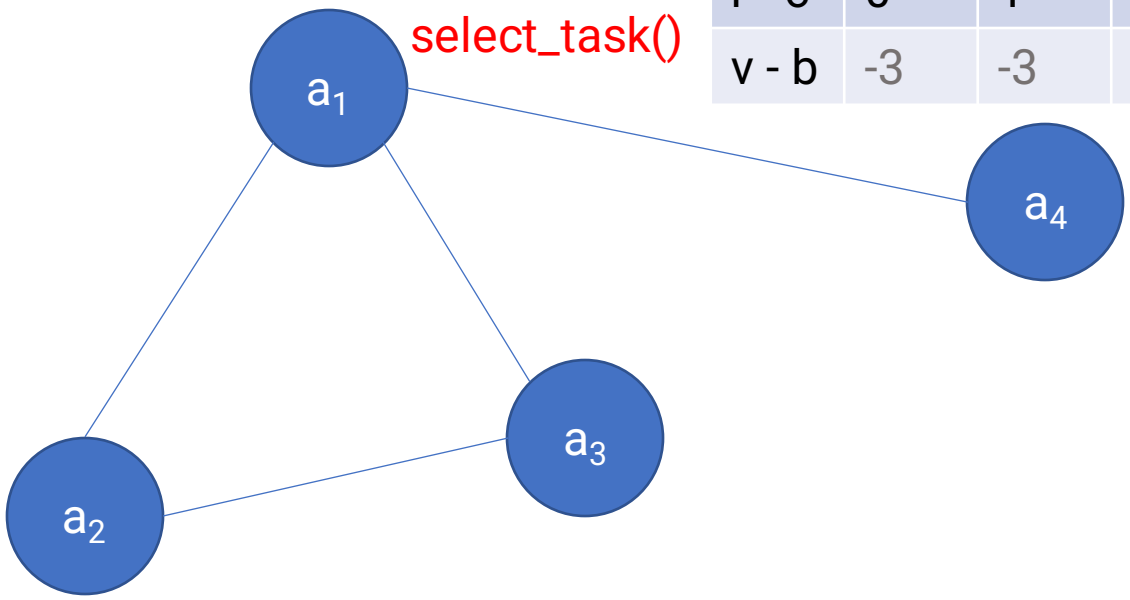
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment	a_2	a_3		a_4
bids	3	4	0	5

Ejemplo CBAA

	s_1	s_2	s_3	s_4
r - c	0	1	1	2
v - b	-3	-3	1	-3

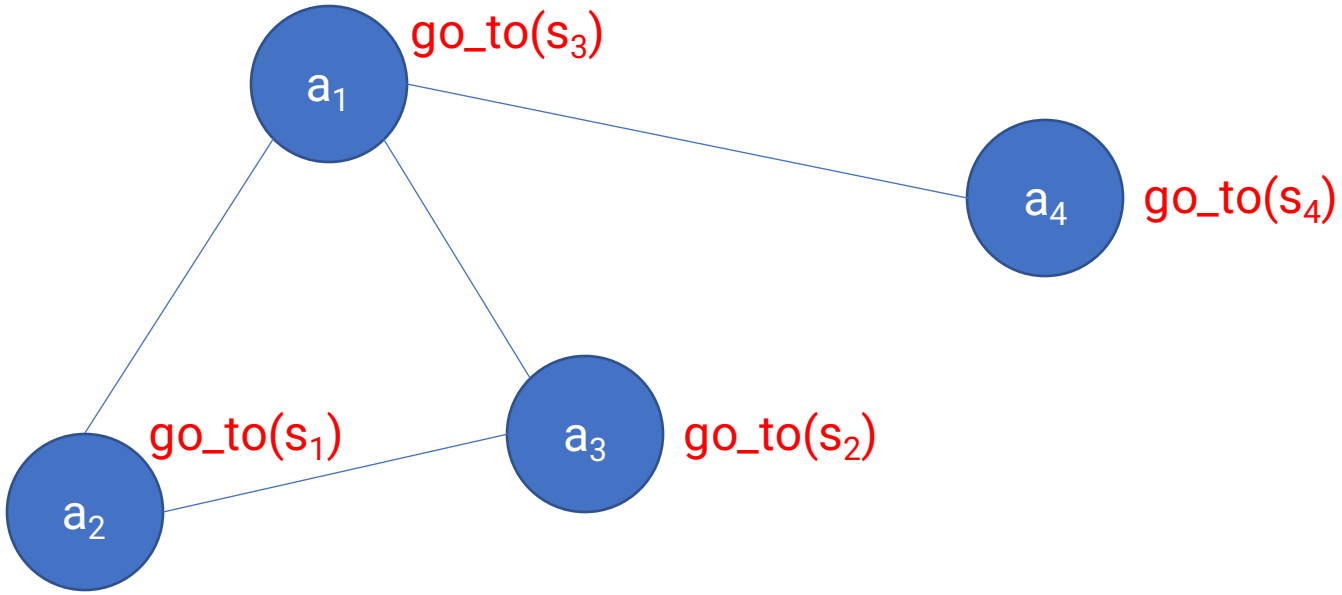


r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment	a_2	a_3	a_1	a_4
bids	3	4	1	5

Ejemplo CBAA



r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	2	1	0	1
a_2	0	1	2	3
a_3	1	0	1	2
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment	a_2	a_3	a_1	a_4
bids	3	4	1	5