

# **Sistemas Multiagente Tema 2.3**

## **Coordinación, cooperación, formación de grupos**

*Authors: Vicent Botti, Vicente Julián*

# Coordinación: ¿Qué es?



# Coordinación: ¿Qué es?



# Coordinación: ¿Qué es?

“The organization of the different elements of a complex body or activity so as to enable them to work together effectively”

“The process of organizing people or groups so that they work together properly and well”

**“The harmonious functioning of parts for effective results”**

“The ability to use different parts of the body together smoothly and efficiently”

...

En sistemas multiagente (MAS):

- Coordinación es una característica clave de sistemas multiagentes
- Es la clave para obtener un funcionamiento efectivo y eficiente en sistemas compuestos por agentes individuales.

# Cooperación en SMAs

## Definición

La cooperación en un SMA consiste en la actuación coordinada entre agentes de tal manera que unos colaboran en la resolución de tareas de otros interesada o desinteresadamente.

Normalmente la cooperación es entre agentes benevolentes

Los agentes cooperan de forma natural al resolver un problema global, cada uno dedicado a su parcela

Distribución de Tareas → Task sharing, Task Allocation, Role Allocation, ...

# El problema del “Task Allocation”

Problema de **asignar** una serie de **tareas** a un conjunto de **agentes** para lograr el máximo número de tareas realizadas con éxito donde tanto los agentes como la tarea pueden salir y entrar en el sistema a lo largo del tiempo.

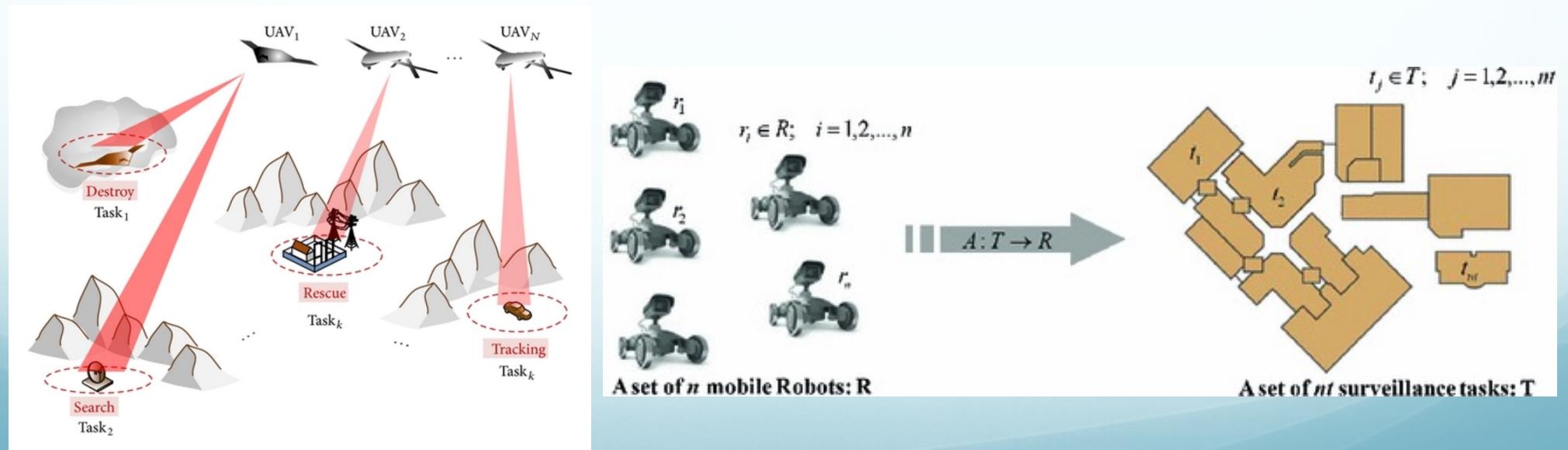
Puede haber más tareas que los agentes por lo que los agentes necesitan **planificarse (scheduling)** para intentar cada tarea a su vez

En el caso de un sistema heterogéneo, donde cada agente puede tener diferentes capacidades (roles) y no puede realizar la tarea individualmente, los agentes necesitan **comunicarse y negociar** con otros agentes.

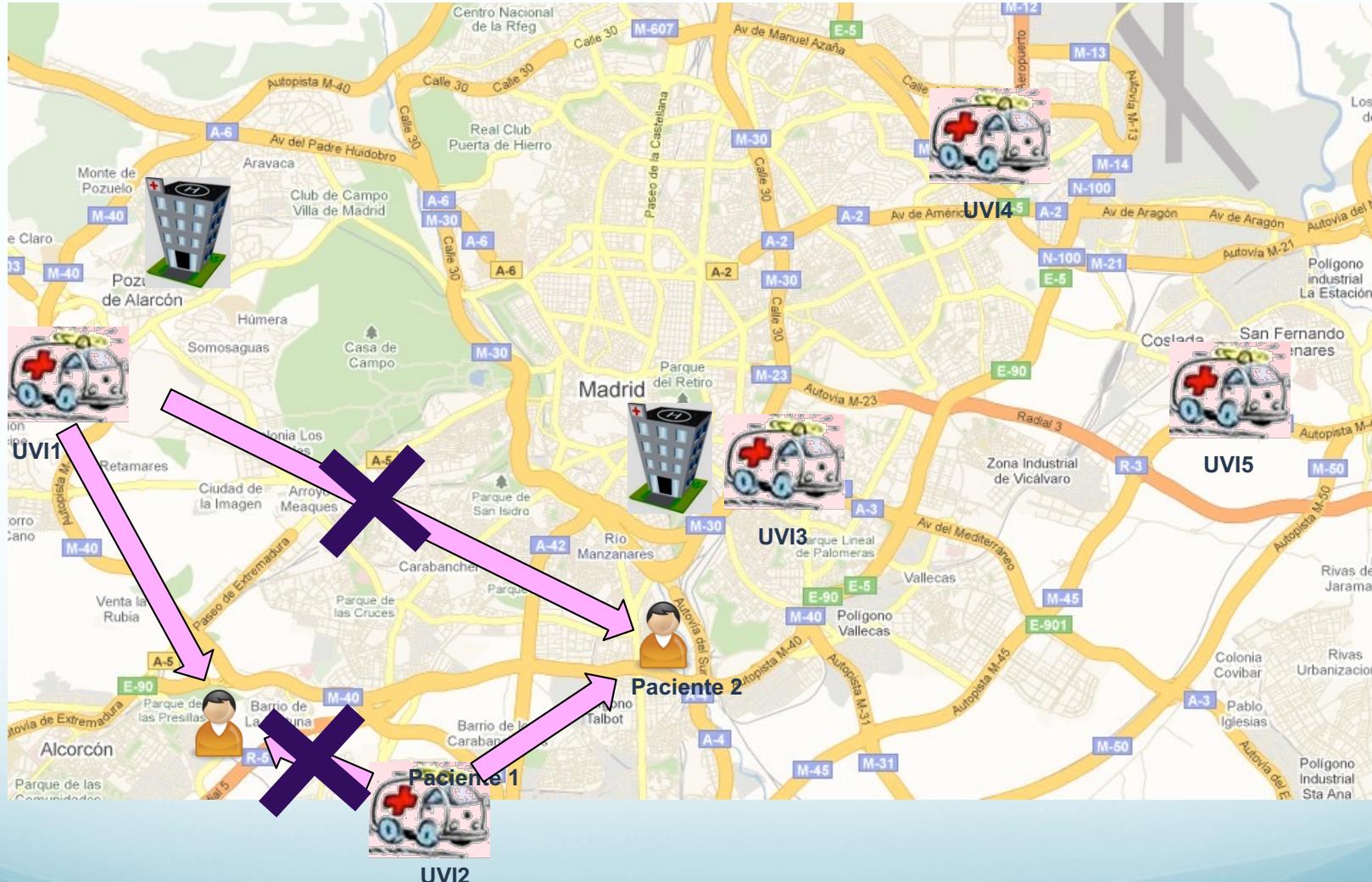
# El problema del “Task Allocation”

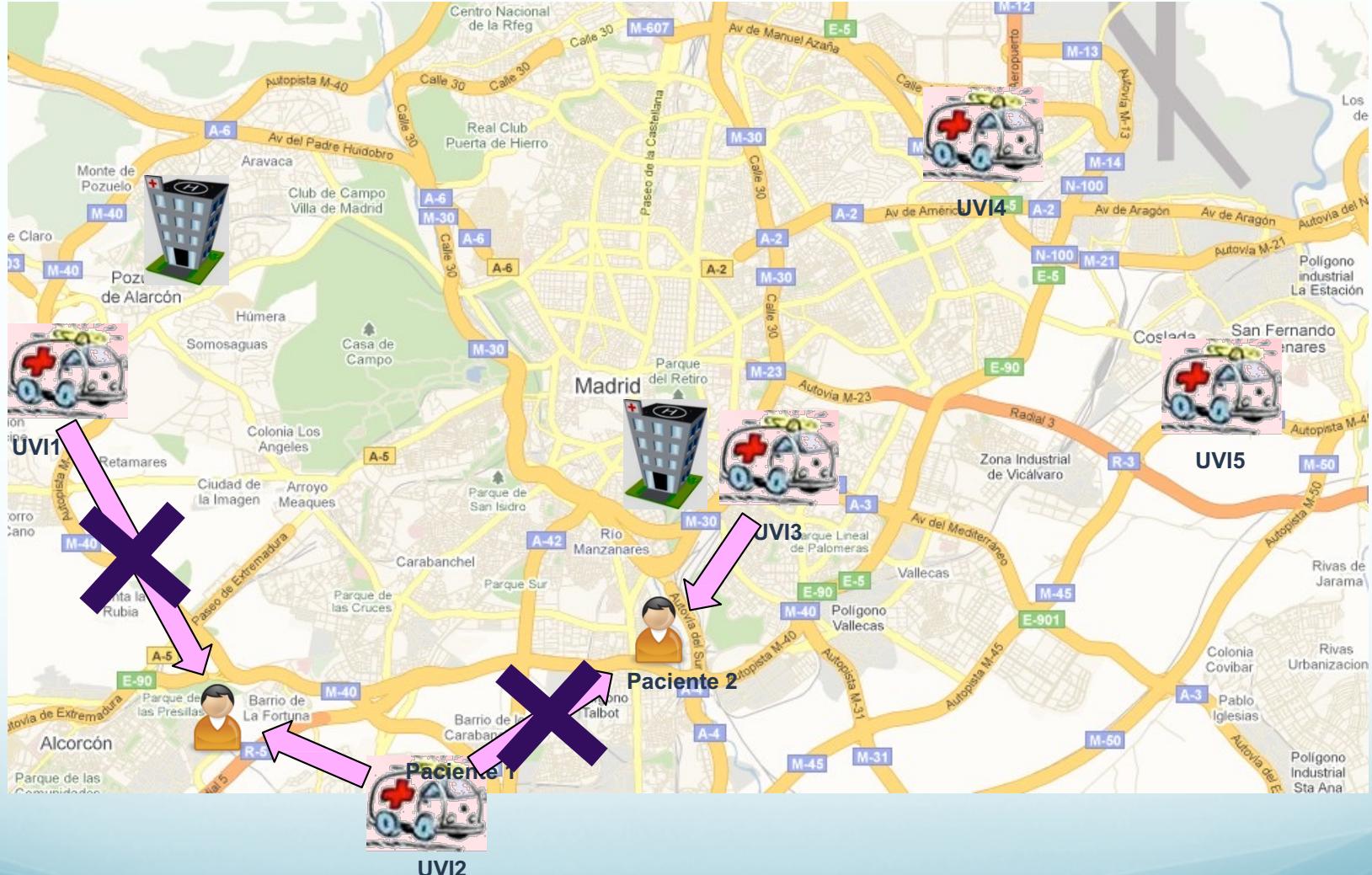
La asignación de tareas puede hacerse de dos maneras:

- Centralizada: se utiliza un planificador central para asignar las tareas a los agentes cooperativos. + fácil -robusta
- Distribuida: las tareas pueden llegar a cualquier agente y los agentes se comunican entre ellos para completar la asignación de tareas. -fácil +robusta

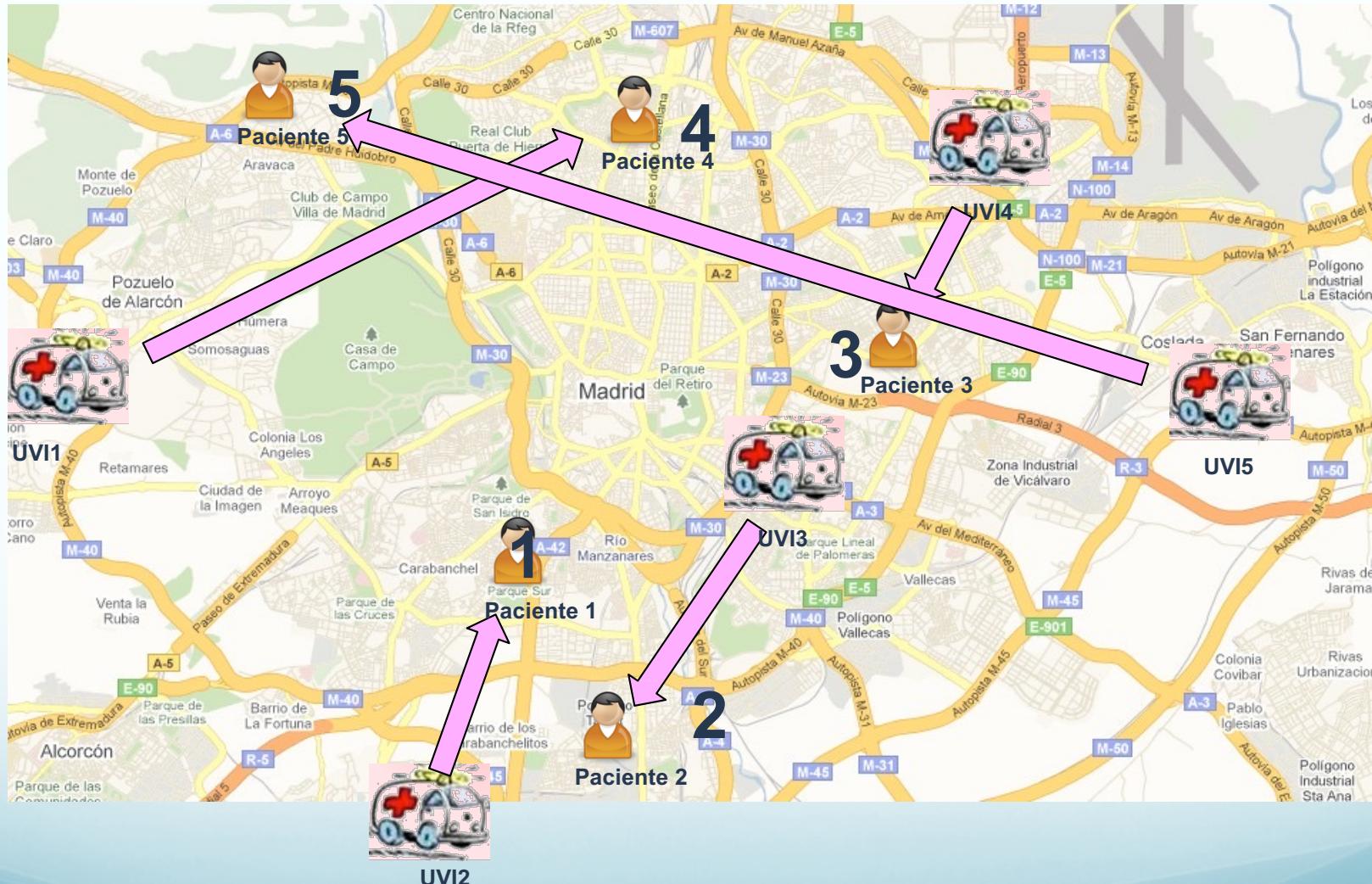


# Ej: Asignación de pacientes

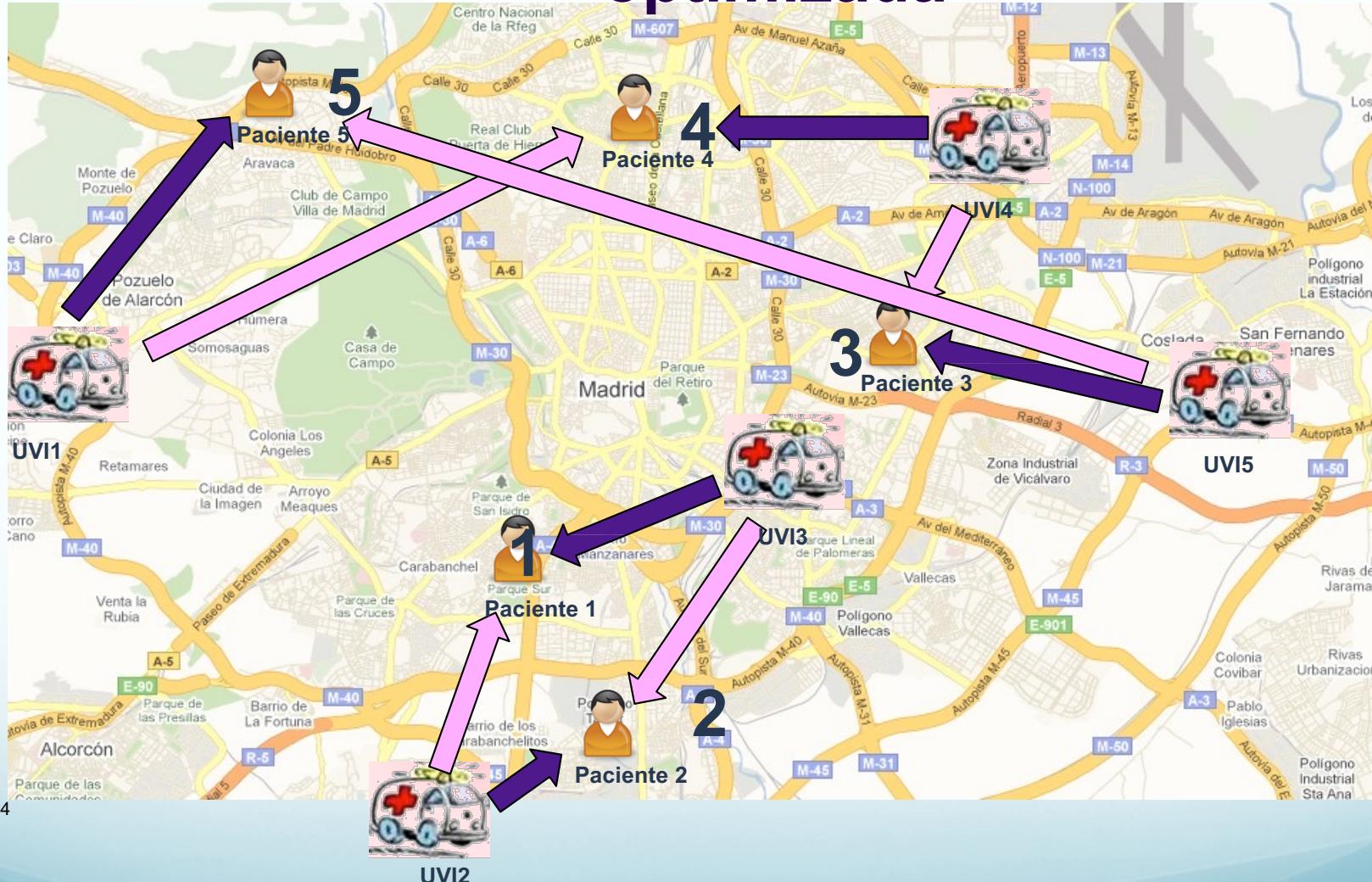




# Asignación de pacientes: situación inicio



# Asignación de pacientes: asignacion optimizada



# Task Allocation

Para tratar la asignación de tareas en un contexto de optimización, hay que decidir qué es exactamente lo que debe ser optimizado.

Lo ideal es que el objetivo sea directamente optimizar el rendimiento general del sistema → Difícil en un entorno distribuido

Concepto de utilidad: empleado en economía, teoría de juegos e investigación operativa, así como en la coordinación multiagente.

Se asume que cada agente es capaz de estimar su aptitud (fitness) para cada tarea que puede realizar. Esa estimación puede emplear dos factores: **beneficio esperado** y **coste esperado**

# Task Allocation

Dado un agente  $a$  de un conjunto  $A$

Dada una tarea  $t$  de un conjunto  $T$ ,

si  $a$  es capaz de ejecutar  $t$ , entonces se puede definir, en alguna escala estandarizada,  $Q_{a,t}$  y  $C_{a,t}$  como el beneficio y el coste, respectivamente, que se espera que resulte de la ejecución de  $t$  por  $a$ .

Esto resulta en una medida de utilidad no negativa:

$$U_{a,t} = \begin{cases} Q_{a,t} - C_{a,t} & \text{if } a \text{ es capaz de ejecutar } t \text{ y } Q_{a,t} > C_{a,t} \\ 0 & \text{otherwise} \end{cases}$$

Para calcular la utilidad del sistema, habría que encontrar una asignación  $X_n$  (de agentes a tareas) que maximice la utilidad

$$U(X_n) = \max \sum_{X_i \in (A,T)} U(X_i^{a,t})$$

# Taxonomía de problemas de TA

## Características a considerar

- **Agentes de una sola tarea (ST) vs agentes de múltiples tareas (MT):** ST significa que cada agente es capaz de ejecutar a lo sumo una tarea a la vez, mientras que MT significa que algunos agentes pueden ejecutar múltiples tareas simultáneamente.
- **Tareas de un solo agente (SA) vs tareas de múltiples agentes (MA):** SA significa que cada tarea requiere exactamente un agente para lograrlo, mientras que la MA significa que algunas tareas pueden requerir múltiples agentes.
- **Asignación instantánea (IA) vs asignación extendida en el tiempo (TA):** IA significa que se permite sólo una asignación instantánea de tareas, TA significa que está disponible el conjunto de todas las tareas que deberán ser asignadas, o un modelo de cómo se espera que las tareas lleguen con el tiempo.

# Taxonomía de problemas de TA

ST-SA-IA: Single-Task Agents, Single-Agent Tasks, Instantaneous Assignment

Es la combinación más simple

Dados A agentes, N tareas y las estimaciones de utilidad para cada uno de los posibles pares agente-tarea, se debe asignar a lo sumo una tarea para cada agente.

Si las utilidades de los agentes pueden ser conocidas entonces se puede usar un enfoque de programación lineal **centralizada**, o soluciones alternativas.

Alternativamente, se puede usar un enfoque **distribuido** basado en subastas, como el algoritmo de subasta de (Bertsekas 1990), encuentra el óptimo.

# Taxonomía de problemas de TA

ST-SA-IA: Single-Task Agents, Single-Agent Tasks, Instantaneous Assignment

Variante: Asignación Iterada

Pocos problemas muestran exactamente la asignación en un único instante

Muchos problemas pueden ser enmarcados como instancias iteradas del ST-SR-IA: cada tarea llega en un momento

Algoritmo:

1. If any agent remains unassigned, find the agent-task pair  $(i, j)$  with the highest utility. Otherwise, quit.
2. Assign agent  $i$  to task  $j$  and remove them from consideration.
3. Go to step 1.

En distribuido se puede emplear el **Contract-Net Protocol**

# Taxonomía de problemas de TA

ST-SA-TA: Single-Task Agents, Single-Agent Tasks,  
Time-Extended Assignment

Hay más tareas que agentes, o hay un modelo de cómo llegarán las tareas,

Es posible predecir las utilidades de cada agente para las tareas con cierta precisión

Este problema es un ejemplo de problemas de scheduling

Los agentes ejecutan tareas en paralelo y el criterio de optimización es a partir de la suma ponderada de los costes/utilidades de ejecución

Algoritmo simple: given  $m$  agents and  $n$  tasks, with  $n > m$

1. Optimally solve the initial  $m \times n$  assignment problem.
2. Use the Greedy algorithm to assign the remaining tasks as the agents become available.

# Taxonomía de problemas de TA

MT-SA-IA and MT-SA-TA: Multi-Task Agents, Single-Agent Tasks

El agente es capaz de realizar varias tareas en paralelo.

Se pueden aplicar algoritmos de las variantes anteriores teniendo en cuenta:

- Un agente tendrá una capacidad máxima de tareas
- A un mismo agente se le pueden seguir asignando tareas sin superar ese máximo
- Si la utilidad del sistema tiene en cuenta el reparto de tareas → se tenderá a asignar al mayor número de agentes posibles.

# Taxonomía de problemas de TA

ST–MA–IA: Single-Task Agents, Multi-Agent Tasks,  
Instantaneous Assignment

Problemas que implican tareas que requieren la combinación de esfuerzo de múltiples agentes.

Considerar utilidades combinadas de grupos de agentes, que no son en general sumas sobre utilidades individuales

En la comunidad multiagente, el problema del ST-MA-IA se conoce como formación de coaliciones y ha sido ampliamente estudiado

Es necesario **crear un equipo** de agentes **capaz** de resolver la tarea

# Taxonomía de problemas de TA

ST–MA–TA: Single-Task Agents, Multi-Agent Tasks,  
Time-Extended Assignment

Estos problemas incluyen una combinación de formación de coaliciones y de algoritmos de scheduling

Ejemplo: entregar un número de paquetes de varios tamaños de un solo centro de distribución a diferentes destinos.

El número de los paquetes y sus destinos se conocen de antemano

El tamaño de cada paquete determina el número de los agentes-robots necesarios para llevarlo.

Dada la existencia de un grupo de agentes-robots, el problema es construir una planificación de entrega de los paquetes, asegurando que un equipo del tamaño adecuado se forma para cada paquete.

# Taxonomía de problemas de TA

MT–MA–IA: Multi-Task Agents, Multi-Agent Tasks,  
Instantaneous Assignment

Ejemplo: asignación de tareas de vigilancia a un equipo de robots en un edificio de oficinas.

Cada robot continuamente patrulla una parte fija del edificio.

Debido a los cálculos y/o limitaciones sensoriales, cada robot puede simultáneamente detectar sólo un número limitado de eventos ambientales (por ejemplo, persona sospechosa, humo, puerta abierta).

Dado un conjunto de eventos a buscar, y el conocimiento acerca de dónde es probable que ocurra cada evento en el edificio :

¿Qué robots deben ser asignados para buscar cada evento?

# Taxonomía de problemas de TA

MT–MA–IA: Multi-Task Agents, Multi-Agent Tasks,  
Instantaneous Assignment

En comparación con una partición, los subconjuntos no tienen por qué ser disjuntos.

Es un problema bien conocido denominado **Set Covering Problem**

MT–MR–TA: Multi-Task Agents, Multi-Agent Tasks,  
Time-Extended Assignment

Se puede ampliar el dominio de vigilancia anterior especificando que no es necesario vigilar ciertos eventos inmediatamente o continuamente, si no siguiendo un programa predefinido.

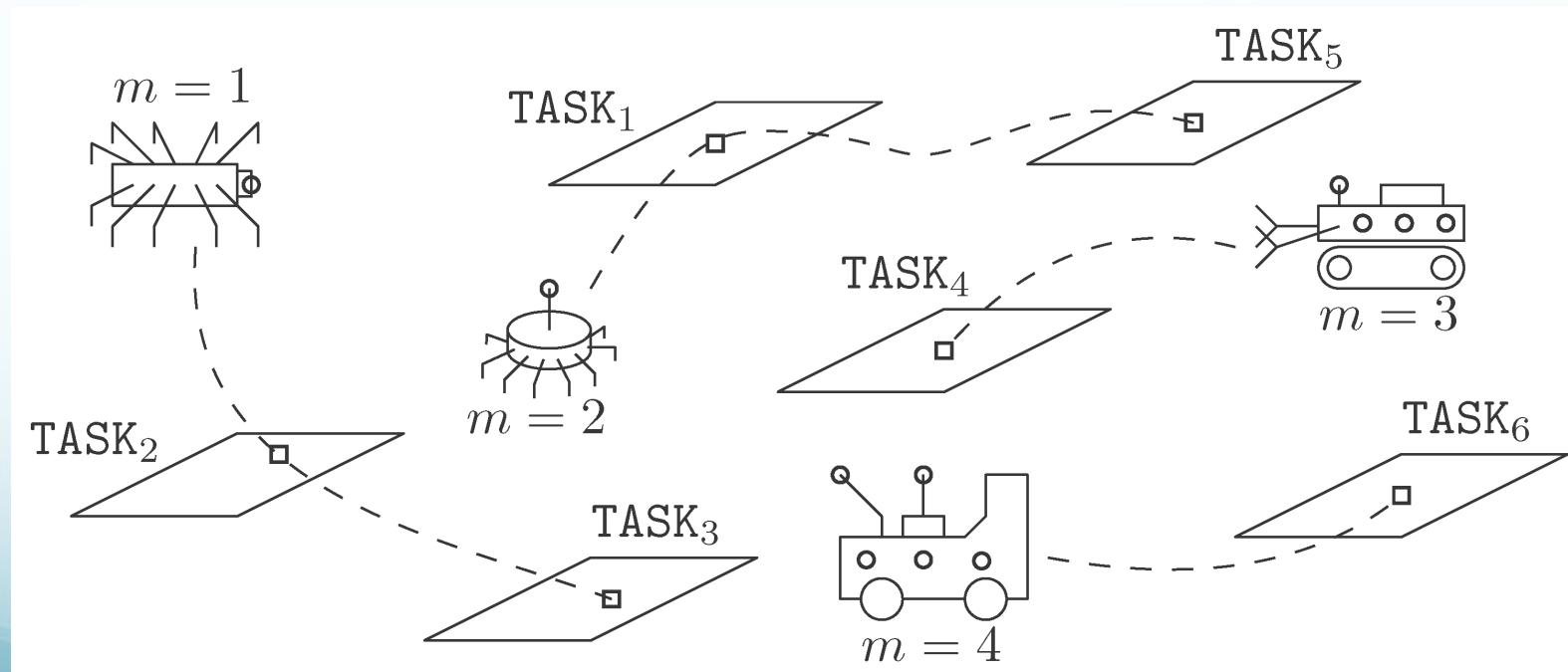
Por ejemplo, "*el ala izquierda del edificio deben ser revisada cada hora*"

El resultado es un problema como al anterior pero que requiere de una solución empleando algoritmos de scheduling

# Taxonomía de problemas de TA

Debido a que es el caso más simple, el ST-SA-IA ha recibido la mayor atención a nivel de investigación

Ya comentado: en distribuido se puede emplear el **Contract-Net Protocol**



# Protocolo de red de contratos: Contract Net

PRC (Smith, 1988).

Desarrollado para DPSs, sigue usandose extensivamente en los SMAs

Sirve para que un agente contrate tareas a otros agentes

Suposiciones

- 1 la negociación es un proceso local que no implica control centralizado,
- 2 existe un medio bidireccional para intercambiar información,
- 3 cada parte en la negociación evalúa la información desde su propia perspectiva
- 4 el acuerdo final se alcanza mediante selección mútua.

# Performativas en Contract Net

Los pasos del Contract Net se expresarían como:

- *cfp* (call for proposals):

Se usa para anunciar una tarea;

- *propose, refuse*:

Usados para hacer una propuesta o declinar el hacer una propuesta

- *accept, reject*:

Usados para indicar la aceptación o rechazo de una propuesta

- *inform, failure*:

Usados para indicar la finalización de una tarea o su fallo

# Contract Net

Un ejemplo ...

## Gestión distribuida de un Radio Taxi

- Cada taxi y cada usuario son agentes
- Los taxis se ofrecen a llevar a una persona
- Criterio de elección: Se elige por distancia

# Ejemplo en Cooperativa de Taxis



Usuario de Taxi



Taxi 4



Taxi 1

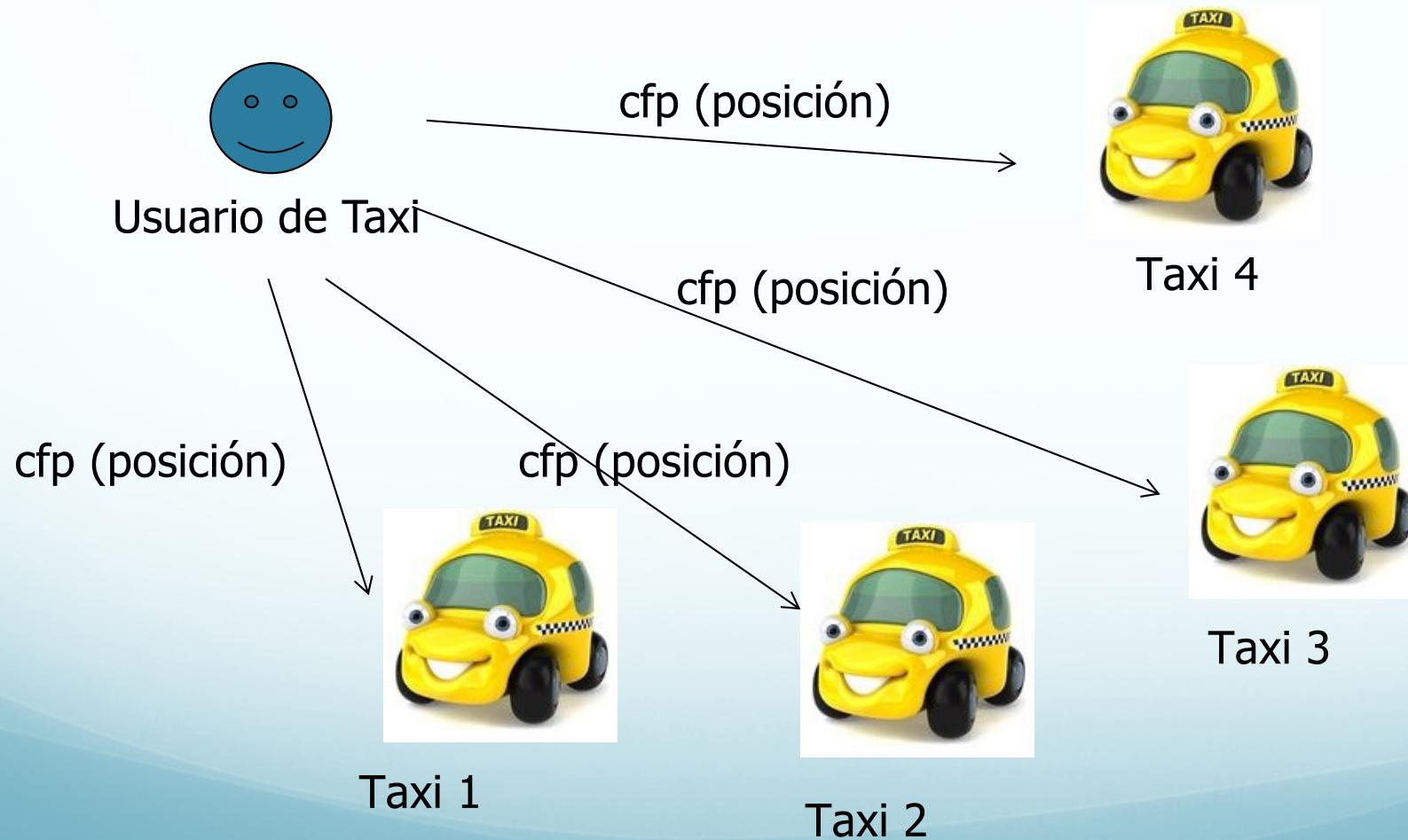


Taxi 2

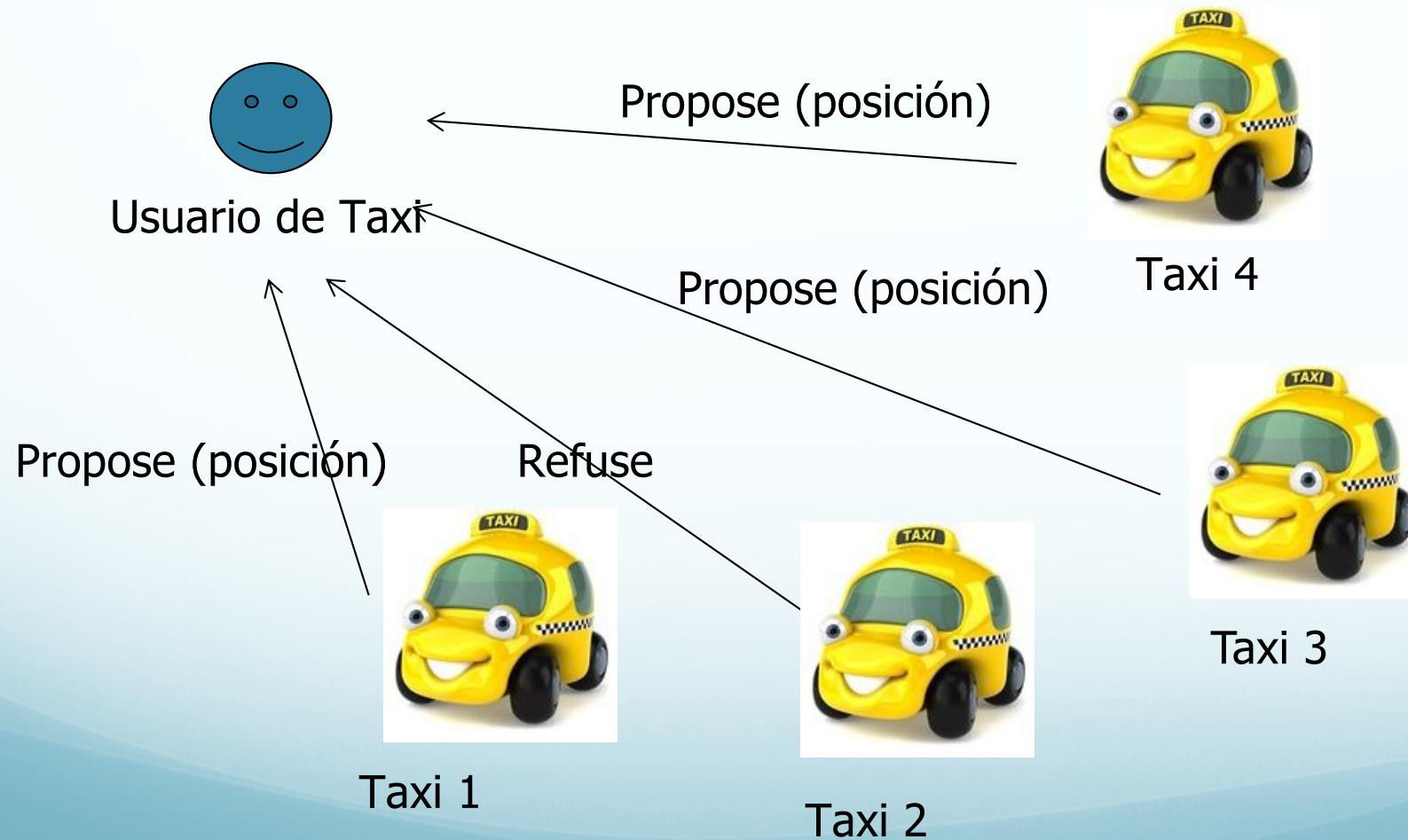


Taxi 3

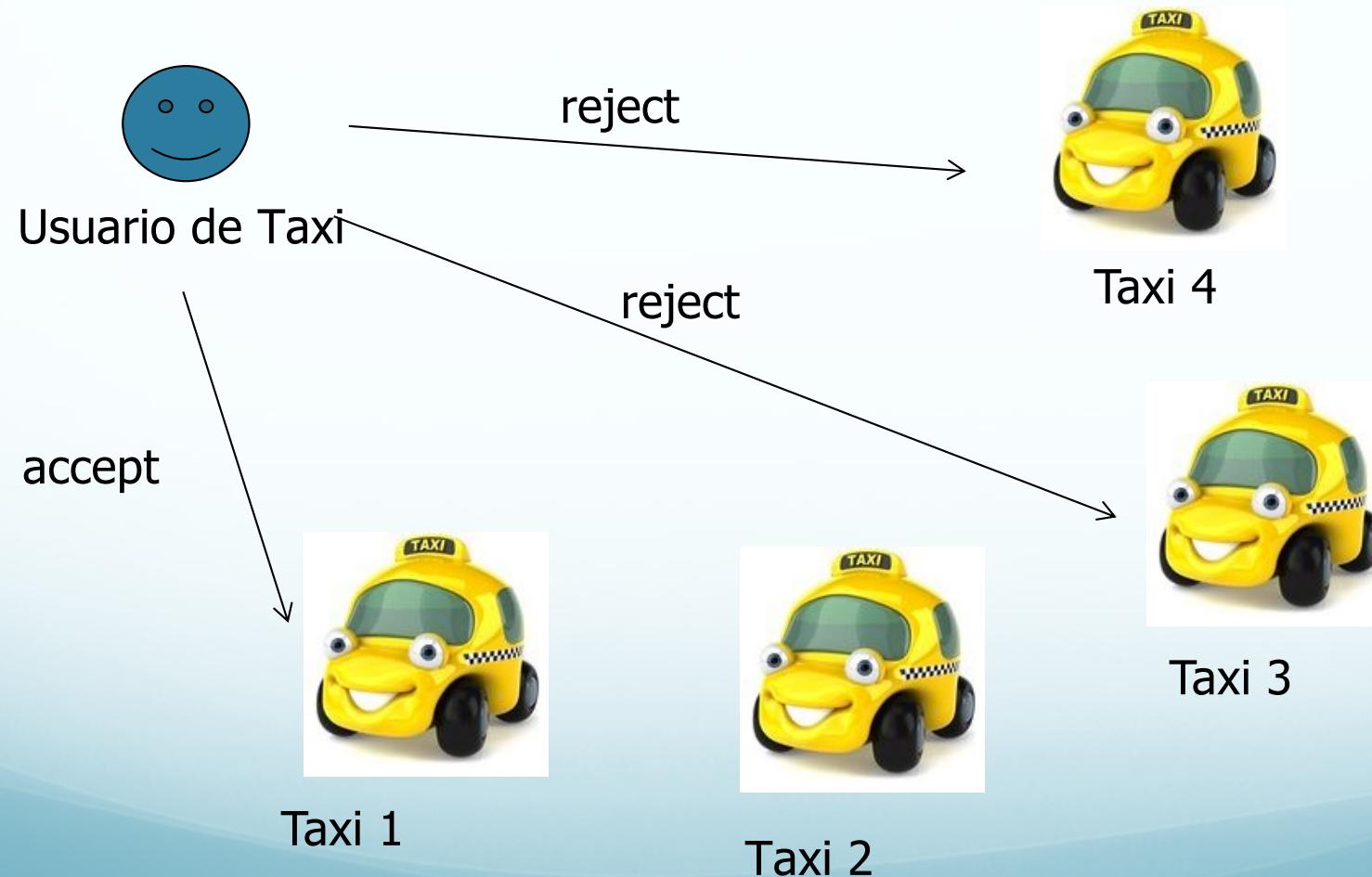
# Ejemplo en Cooperativa de Taxis



# Ejemplo en Cooperativa de Taxis



# Ejemplo en Cooperativa de Taxis

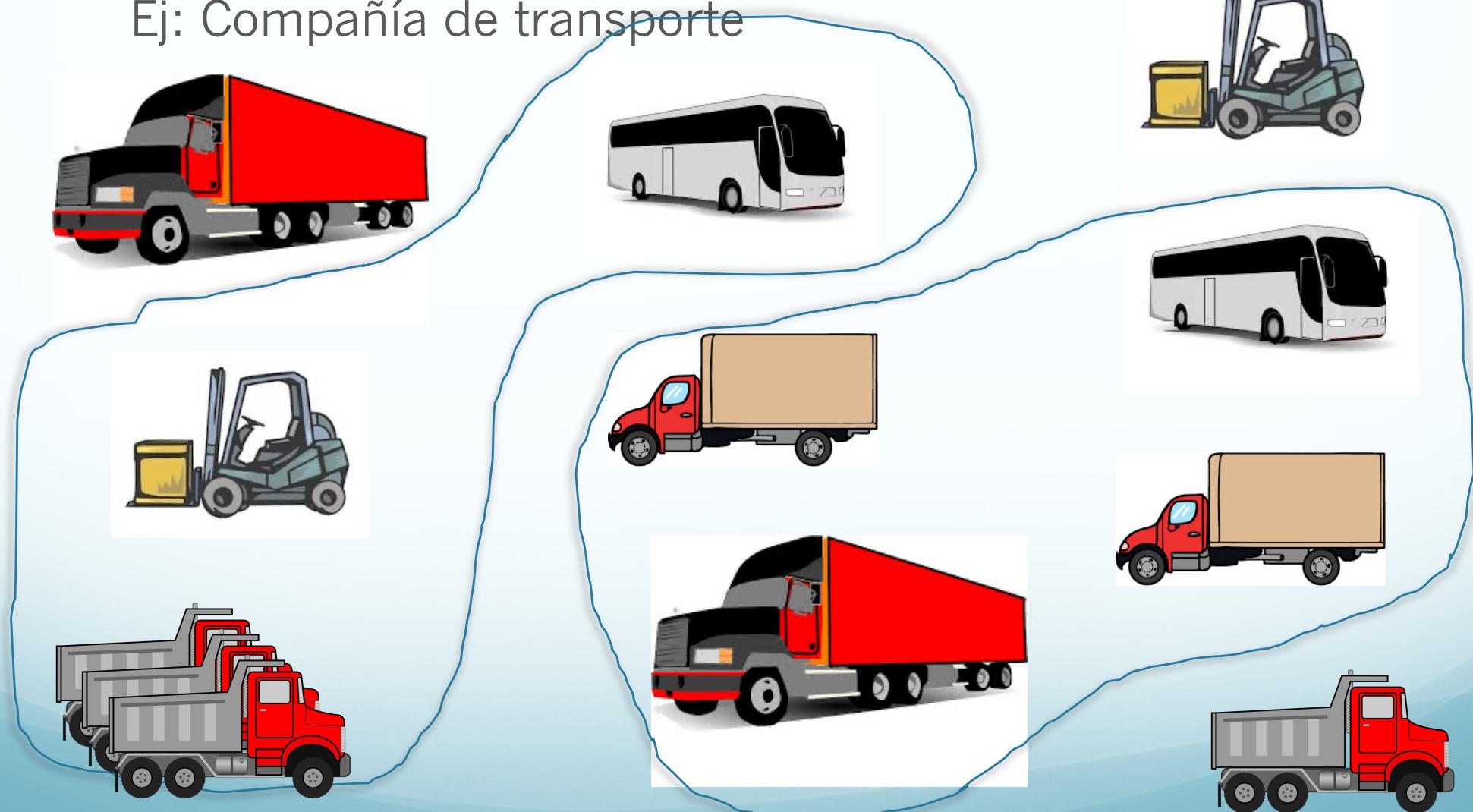


# Formación de coaliciones

- Problema estudiado en:
  - Microeconomía
  - Teoría de juegos
  - Informática
    - particularmente en ... Sistemas Multiagente
- Formación de grupos de agentes para conseguir de forma eficiente los objetivos individuales y colectivos

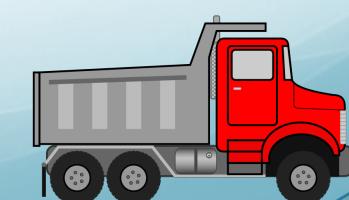
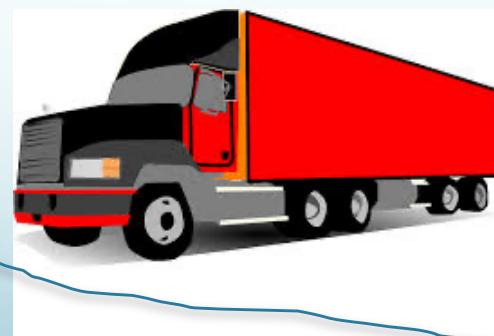
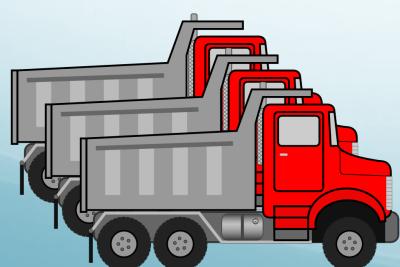
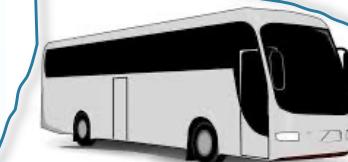
# Formación de coaliciones

Ej: Compañía de transporte



# Formación de coaliciones

Ej: Compañía de transporte



# Formación de Coaliciones

## Definición Formal

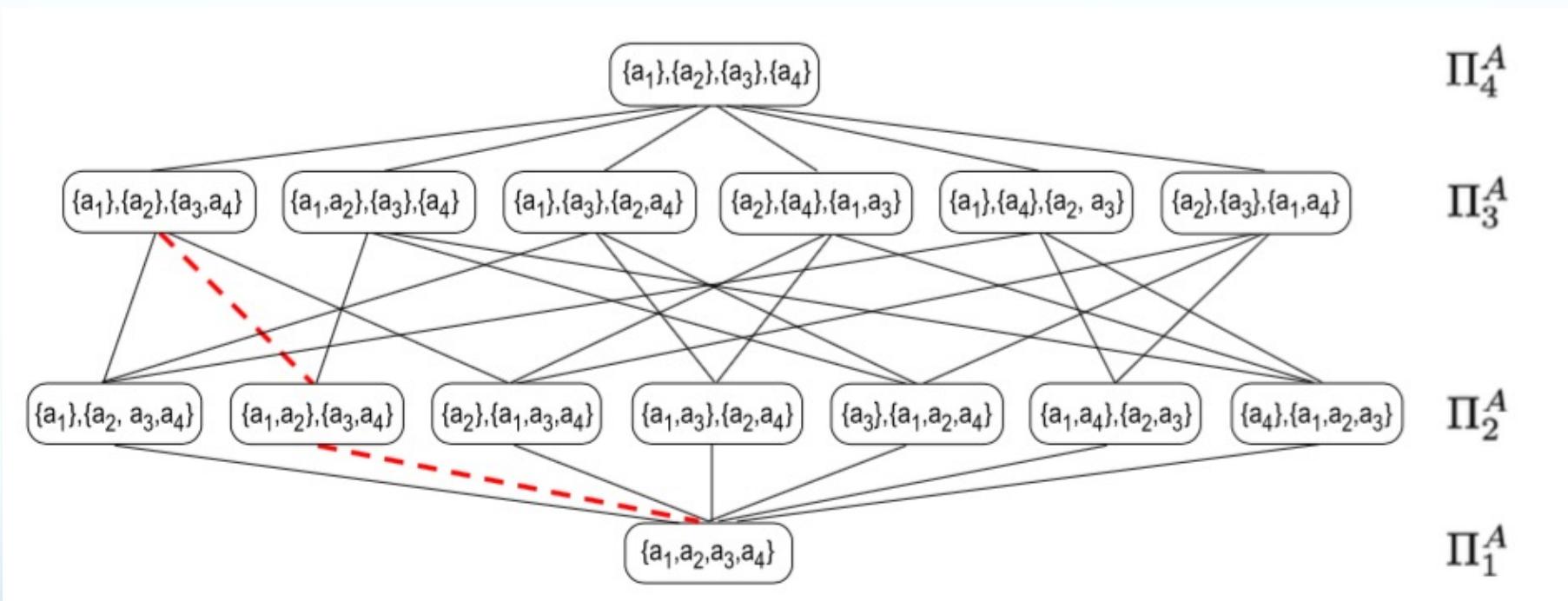
- Un conjunto  $A$  de  $n$  agentes  $A=\{A_1, \dots, A_n\}$
- Una **coalición**  $C$  es cualquier subconjunto no vacío de  $A$
- Una **estructura de coaliciones** es una colección de coaliciones disjuntas

Para una coalición  $C$ , una estructura de coaliciones sobre  $C$  es un conjunto de coaliciones  $CS=\{C_1, \dots, C_k\}$  tal que  $\cup CS = C$ , y  $C_i \cap C_j = \emptyset$  para cualquier  $i, j \in \{1, \dots, k\}: i \neq j$

El conjunto de estructuras de coalición sobre  $C$  se denota como  $\mathbb{M}^C$

# Formación de Coaliciones

Un ejemplo con 4 agentes:  $a_1, a_2, a_3, a_4$



# Formación de Coaliciones

El valor de una coalición se suele expresar en términos “monetarios”

$v(C)$  definida como  $v: 2^A \rightarrow \mathbb{R}$

Para una estructura de coaliciones, se suele calcular el valor como la suma de los valores de las coaliciones. De esta forma:

Dado  $CS \in \mathbb{I}^A$  se calcula  $V(CS) = \sum_{C_i \in CS} v(C_i)$

Se define el problema de la **generación de estructuras de coalición**, como el problema de encontrar la estructura sobre el conjunto A cuyo valor sea máximo.

Dicha estructura se considera como “óptima” y se denota como  $CS^*$

$CS^* \in \operatorname{argmax}_{CS \in \mathbb{I}^A} V(CS)$

# Formación de Coaliciones

Existen múltiples formas de abordar este problema, el cual es NP-completo.

Puede aplicarse:

- Programación dinámica
- Algoritmos Anytime
- Metaheurísticas
- Teoría de juegos
- ....

# Formación de Coaliciones

## Ejemplo de Programación dinámica

Dada una coalición  $C \subseteq A$ , sea  $f(C)$  el valor de una estructura coalicional óptima sobre  $C$

$$f(C) = \max_{CS \in \mathbb{M}^C} V(CS)$$

Entonces:

$$f(C) = \begin{cases} v(C) & \text{if } |C| = 1 \\ \max\{v(C), \max_{\{C', C''\} \in \mathbb{M}^C} (f(C') + f(C''))\} & \text{otherwise} \end{cases}$$

El algoritmo itera analizando primero coaliciones de talla 1, luego de talla 2, ...

Durante el proceso, el cálculo de  $\max\{v(C), \max_{\{C', C''\} \in \mathbb{M}^C} (f(C') + f(C''))\}$  se almacena en una variable  $t(C)$  de forma que:

- Si  $v(C)$  es mayor, entonces  $t(C) = \{C\}$  (no conviene dividir  $C$ )
- Si  $v(C)$  es menor, entonces  $t(C) = \operatorname{argmax}_{\{C', C''\} \in \mathbb{M}^C} (f(C') + f(C''))$  (es mejor dividir  $C$  en  $C'$  y  $C''$ )

# Formación de Coaliciones

characteristic function	$v(\{a_1\}) = 30$	$v(\{a_1, a_2\}) = 50$	$v(\{a_2, a_4\}) = 70$	$v(\{a_1, a_3, a_4\}) = 100$
	$v(\{a_2\}) = 40$	$v(\{a_1, a_3\}) = 60$	$v(\{a_3, a_4\}) = 80$	$v(\{a_2, a_3, a_4\}) = 115$
	$v(\{a_3\}) = 25$	$v(\{a_1, a_4\}) = 80$	$v(\{a_1, a_2, a_3\}) = 90$	$v(\{a_1, a_2, a_3, a_4\}) = 140$
	$v(\{a_4\}) = 45$	$v(\{a_2, a_3\}) = 55$	$v(\{a_1, a_2, a_4\}) = 120$	

<b>C</b>	The values that must be compared before setting $t(C)$ and $f(C)$		<b><math>t(C)</math></b>	<b><math>f(C)</math></b>
{a <sub>1</sub> }		$v(\{a_1\}) = 30$	{a <sub>1</sub> }	30
{a <sub>2</sub> }		$v(\{a_2\}) = 40$	{a <sub>2</sub> }	40
{a <sub>3</sub> }		$v(\{a_3\}) = 25$	{a <sub>3</sub> }	25
{a <sub>4</sub> }		$v(\{a_4\}) = 45$	{a <sub>4</sub> }	45
{a <sub>1</sub> , a <sub>2</sub> }	$v(\{a_1, a_2\}) = 50$	$f(\{a_1\}) + f(\{a_2\}) = 70$	{a <sub>1</sub> } {a <sub>2</sub> }	70
{a <sub>1</sub> , a <sub>3</sub> }	$v(\{a_1, a_3\}) = 60$	$f(\{a_1\}) + f(\{a_3\}) = 55$	{a <sub>1</sub> , a <sub>3</sub> }	60
{a <sub>1</sub> , a <sub>4</sub> }	$v(\{a_1, a_4\}) = 80$	$f(\{a_1\}) + f(\{a_4\}) = 75$	{a <sub>1</sub> , a <sub>4</sub> }	80
{a <sub>2</sub> , a <sub>3</sub> }	$v(\{a_2, a_3\}) = 55$	$f(\{a_2\}) + f(\{a_3\}) = 65$	{a <sub>2</sub> } {a <sub>3</sub> }	65
{a <sub>2</sub> , a <sub>4</sub> }	$v(\{a_2, a_4\}) = 70$	$f(\{a_2\}) + f(\{a_4\}) = 85$	{a <sub>2</sub> } {a <sub>4</sub> }	85
{a <sub>3</sub> , a <sub>4</sub> }	$v(\{a_3, a_4\}) = 80$	$f(\{a_3\}) + f(\{a_4\}) = 70$	{a <sub>3</sub> , a <sub>4</sub> }	80
{a <sub>1</sub> , a <sub>2</sub> , a <sub>3</sub> }	$v(\{a_1, a_2, a_3\}) = 90$	$f(\{a_1\}) + f(\{a_2, a_3\}) = 95$	{a <sub>2</sub> } {a <sub>1</sub> , a <sub>3</sub> }	100
{a <sub>1</sub> , a <sub>2</sub> , a <sub>4</sub> }	$f(\{a_2\}) + f(\{a_1, a_3\}) = 100$	$f(\{a_3\}) + f(\{a_1, a_2\}) = 95$	{a <sub>1</sub> , a <sub>2</sub> , a <sub>4</sub> }	120
{a <sub>1</sub> , a <sub>3</sub> , a <sub>4</sub> }	$v(\{a_1, a_2, a_4\}) = 120$	$f(\{a_1\}) + f(\{a_2, a_4\}) = 115$	{a <sub>1</sub> } {a <sub>3</sub> , a <sub>4</sub> }	110
{a <sub>2</sub> , a <sub>3</sub> , a <sub>4</sub> }	$f(\{a_2\}) + f(\{a_1, a_4\}) = 120$	$f(\{a_4\}) + f(\{a_1, a_2\}) = 115$	{a <sub>2</sub> } {a <sub>3</sub> , a <sub>4</sub> }	120
{a <sub>1</sub> , a <sub>2</sub> , a <sub>3</sub> , a <sub>4</sub> }	$v(\{a_1, a_2, a_3, a_4\}) = 140$	$f(\{a_4\}) + f(\{a_1, a_2, a_3\}) = 145$	{a <sub>1</sub> , a <sub>2</sub> } {a <sub>3</sub> , a <sub>4</sub> }	150
	$f(\{a_1, a_2\}) + f(\{a_3, a_4\}) = 150$	$f(\{a_3\}) + f(\{a_1, a_2, a_4\}) = 145$		
	$f(\{a_1, a_3\}) + f(\{a_2, a_4\}) = 145$	$f(\{a_2\}) + f(\{a_1, a_3, a_4\}) = 150$		
	$f(\{a_1, a_4\}) + f(\{a_2, a_3\}) = 145$	$f(\{a_1\}) + f(\{a_2, a_3, a_4\}) = 150$	step 5	

# Formación de coaliciones

- La formación de coaliciones también puede ser entendida como un problema de asignación de tareas
- En este caso, la definición del problema debe ser extendida incluyendo conceptos como:
  - Tareas a realizar
  - Capacidades de los agentes
- Los agentes forman coaliciones para desarrollar tareas y mejorar la eficiencia en su ejecución.
- Problema complejo
  - No todos saben hacer de todo
  - Las tareas pueden tener precedencias
  - Distribuido vs Centralizado

# Formación de coaliciones

- Definición del problema:

Un conjunto de m tareas  $T=\{t_1, \dots, t_m\}$  con sus capacidades requeridas

$B_l = \{b_1^l, \dots, b_r^l\}$  (habilidades necesarias para realizar la tarea)

Un conjunto de n agentes  $N=\{A_1, \dots, A_n\}$  con sus capacidades

$B_i = \{b_1^i, \dots, b_r^i\}$  (habilidad para realizar una acción)

→ Una coalición es un grupo de agentes que deciden cooperar para alcanzar una tarea  $t_i$

Una coalición tiene un vector de capacidades  $B_c$  que es la suma de las capacidades de cada agente

Una coalición puede desarrollar la tarea  $t_i$  si  $B_t$  satisface  $\forall 0 \leq i \leq r, b_i^t \leq b_i^c$

# Formación de coaliciones

Para cada coalición  $C$  se puede calcular una utilidad conjunta  $V$  (dependiente del problema)

La **solución** consiste en asignar tareas  $t_i$  a coaliciones  $C_i \subseteq N$ , tal que,

$\sum_i V_i$  es máximo

Si en lugar de utilidad se quisiese hablar de coste  $c$  se podría calcular como el inverso de  $V$

# Formación de coaliciones

- En un principio, para simplificar, asumimos:
  - Los agentes forman parte de una coalición sólo si el beneficio es mayor que actuando en solitario
  - Los agentes tratan de maximizar la utilidad común
  - No siempre es posible un entorno super-aditivo
  - La población no cambia durante el proceso de formación
  - La lista de tareas no cambia durante el proceso de formación
  - No todos los agentes conocen o pueden conocer las habilidades del resto

# Formación de coaliciones

## Solución centralizada

- Se puede ver como un problema de “set covering” o “set partitioning”
  - E. Balas and M. Padberg, On the set covering problem: An algorithm for set partitioning, Oper. Res. 23 (1975) 74-90.
- Es un problema NP-completo
- Existen multitud de variantes para encontrar sub-optimos
- En nuestro caso tenemos que:
  - Es posible que las coaliciones sean o no sean disjuntas
  - La información no está disponible para todos
  - Buscamos un algoritmo **distribuido**

# Formación de coaliciones

## Solución distribuida

- IMPORTANTE: es una posible forma de resolverlo
- Algoritmo *anytime*
- Se mira por el beneficio del grupo
- Dispone de 2 etapas
  - 1<sup>a</sup> FASE: Se calculan de forma distribuida todas las posibles coaliciones y sus valores iniciales
  - 2<sup>a</sup> FASE: De forma iterativa y distribuida:
    - Los valores de cada coalición se recalculan
    - Los agentes deciden entre las coaliciones preferidas y las forman

# Formación de coaliciones

## Solución distribuida

- Se introduce la siguiente heurística:
  - Grupos pequeños son preferibles a grandes coaliciones
    - Menos comunicación, organización, ...
    - Se introduce un factor  $k$  que denota el tamaño máximo de una coalición
    - Permite limitar el número de coaliciones a  $O(n^k)$  siendo  $n$  el número de agentes
- El proceso parte de  $n$  agentes que interactúan y forman coaliciones.
  - Si las coaliciones son disjuntas, el agente que forma parte de una coalición se sale del proceso

# Formación de coaliciones

## Solución distribuida (FASE 1)

- Cada agente  $A_i$  :
  - Calcula todas las combinaciones hasta  $k$  agentes incluyendo  $A_i$
  - Crea una lista  $L_i$  con las combinaciones (lista de potenciales coaliciones)
  - Contactar con agentes  $A_j$  miembros potenciales de una coalición en  $L_i$  y que no hayan contactado con  $A_i$
  - Para cada agente  $A_j$  contactado
    - Si es el primer contacto, solicitar sus capacidades (obtener  $B_j$ )
    - Comprometerse a calcular los valores de las potenciales coaliciones conjuntas, subconjunto  $S_{ij}$  de  $L_i$ , donde  $A_i$  y  $A_j$  están presentes.
  - Para cada agente  $A_k$  que nos haya contactado, eliminar de  $L_i$  todas las potenciales coaliciones (el agente  $A_k$  se ha comprometido a calcular el valor de esas potenciales coaliciones conjuntas)
  - Para evitar redundancias crear una lista  $LC_i$  ya contactados
  - Repetir hasta que  $LC_i = N - \{A_i\}$  (no hay más agentes que contactar)

# Formación de coaliciones

## Solución distribuida (FASE 2a)

Cálculo repetido de valores de cada coalición

- Cada agente  $A_i$ 
  - $L_i$  lista de coaliciones para las que **debe** calcular su valor
  - Para cada coalición  $C$  en la lista  $L_i$ 
    - Calcula el vector de potenciales capacidades  $B_c^{pc}$  sumando las capacidades *no usadas* de los miembros de la coalición  $B_c^{pc} = \sum_{A_i \in C} B_i$
    - Crea la lista  $E_c$  con los resultados esperados de las tareas de  $T$  si la coalición  $C$  las ejecutase.
    - Para cada tarea  $t_j \in T$ 
      - Comprueba qué capacidades  $B_j$  se necesitan para satisfacer  $t_j$
      - Compara  $B_j$  con la suma de las capacidades no usadas de los miembros  $B_c^{pc}$  encontrando que tareas satisface la coalición  $C$
      - Si  $\forall i, b_j^i \in B_j \leq b_{pc}^i \in B_c^{pc}$  ( $t_j$  se satisface con  $C$ ) calcula el resultado esperado  $e_j$  respecto a  $C$ . Añadir  $e_j$  en  $E_c$
    - Seleccionar el valor máximo de  $E_c$  (ese será el valor de la coalición  $V_c$ )
    - Calcular el coste de la coalición  $c_c = \frac{1}{V_c}$

# Formación de coaliciones

## Solución distribuida (FASE 2b)

Elección de coaliciones (coaliciones disjuntas)

- Se denota el ratio  $w_i = \frac{c_i}{|C_i|}$
- Cada agente  $A_i$  repite de forma iterativa
  - Busca en  $L_i$  la coalición  $C_j$  con el valor  $w_j$  más pequeño
  - Anuncia el peso  $w_j$  al resto de agentes
  - Selecciona el peso más pequeño entre todos los anunciados ( $w_{low}$ ). Será elegido por todos los agentes.
    - Seleccionar  $C_{low}$  y  $t_{low}$
    - Borrar los miembros de  $C_{low}$  de la lista de candidatos a nuevas coaliciones
    - Si  $A_i$  es miembro de  $C_{low}$  → **Formar** junto con el resto de miembros  $C_{low}$
    - Borrar de  $L_i$  las posibles coaliciones que incluyan a los agentes borrados
    - Borrar de  $T$  la tarea elegida  $t_{low}$
  - (después del proceso en  $L_i$  puede haber coaliciones para las que se debe recalcular sus valores)
- Este proceso se repetiría hasta que todos los agentes son borrados

# Formación de coaliciones

## Solución distribuida (FASE 2b)

Elección de coaliciones (coaliciones superpuestas)

- Cada agente  $A_i$  repite de forma iterativa
  - Busca en  $L_i$  la coalición  $C_j$  con el coste  $c_j$  más pequeño
  - Anuncia el coste  $c_j$  al resto de agentes
  - Selecciona el peso más pequeño entre todos los anunciados ( $c_{low}$ ). Será elegido por todos los agentes.
    - Seleccionar  $C_{low}$  y  $t_{low}$
  - Si  $A_i$  es miembro de  $C_{low}$  → **Formar** junto con el resto de miembros  $C_{low}$
  - Borrar de  $T$  la tarea elegida  $t_{low}$
  - Actualizar los vectores de capacidades de todos los miembros de  $C_{low}$  (capacidades que ya no deben estar)
- (después del proceso en  $L_i$  puede haber coaliciones para las que se debe recalcular sus valores)
- Este proceso se repetiría hasta que todos los agentes son borrados

# Formación de coaliciones

- Aspectos no tratados en el algoritmo:
  - Puede exigirse precedencia entre tareas
    - La elección de una tarea implica la elección de todas sus predecesoras
  - Recursos compartidos limitados
    - Los recursos se van agotando a medida que se crean coaliciones
  - Cambios en los agentes: número y capacidades
  - Cambios en las tareas
    - Reconsiderar coaliciones ya formadas

# Formación de coaliciones

- Otras consideraciones

- Pueden existir restricciones añadidas que modificarían el algoritmo
  - Se exige un número mínimo de coaliciones
  - El agente  $Ai$  es incompatible con el agente  $Aj$
  - El agente  $Ai$  tiene una reputación baja en alguna de sus capacidades
  - El agente  $Aj$  tiene una reputación alta en alguna de sus capacidades
  - ...
- Gestión del posible beneficio entre los miembros
- Formación de la coalición con información incompleta

# Formación de coaliciones

- Por último ...

Un ejemplo real: Formación de grupos heterogéneos de alumnos para la realización de trabajos



# An artificial intelligence tool for heterogeneous team formation in the classroom

**Teamwork** is now considered as a prominent general competence in the European Higher Education Area and is a common practice in business projects.

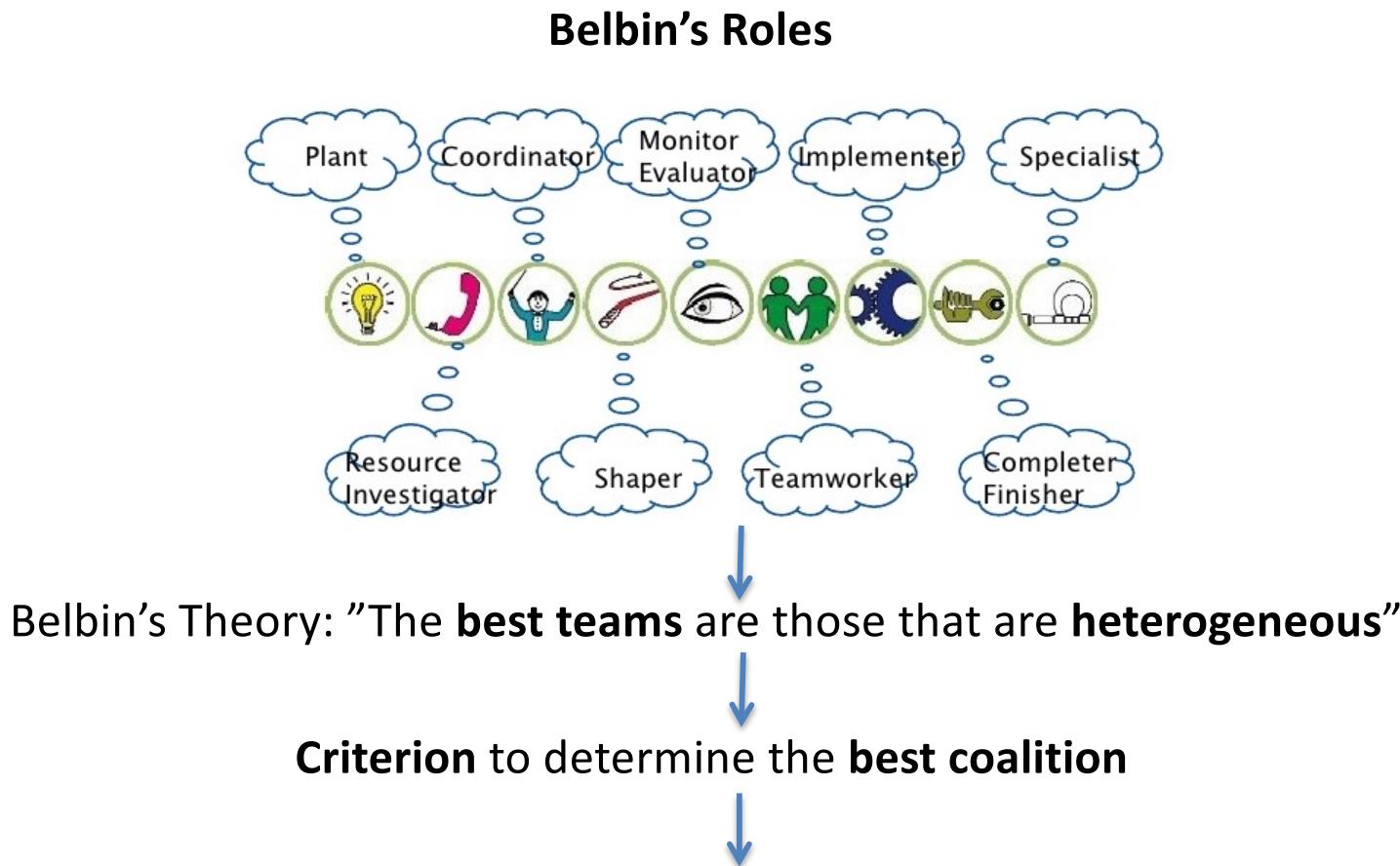
## **Problems:**

- Which is the best **pattern** that drives **successful teams**?
- The total **number** of **team configurations** for the classroom explodes exponentially with this quantity (Even a small classroom with 30 students has 142506 different teams of five individuals )

## **Solution: Coalition generation algorithm that integrates the following areas:**

- One of the most important theories regarding successful team dynamics is Belbin's role taxonomy
- Collective intelligence
- Bayesian learning
- Coalition formation algorithm

# An artificial intelligence tool for heterogeneous team formation in the classroom



## Problems:

How to determine the most predominant role in students → Bayesian Learning

How to generate the coalitions based on Belbin's Theory → Coalition formation problem

# An artificial intelligence tool for heterogeneous team formation in the classroom

## Coalition structure generation problem

- $A = \{a_1, \dots, a_n\}$ , set of students,
- $R = \{r_1, r_2, \dots, r_m\}$  set of roles that the student can play (in our case it is the set of Belbin's roles),
- $\text{role}_i$  denote the true predominant role of  $a_i$ ,
- $T \in A$  is called a *team*
- *team structure*  $S = \{T_1, T_2, \dots, T_k\}$  is a partition of disjoint teams such that  $\bigcup_{\forall T_j \in S} T_j = A$  and  $S \in 2^A$

# An artificial intelligence tool for heterogeneous team formation in the classroom

## Goal: optimal team structure

- $v(S)$  is an evaluation function for the team structure
  - $\underset{S \in 2^A}{\operatorname{argmax}} v(S)$ , where the value of the team structure is
$$v(S) = \sum_{T_j \in S} v(T_j)$$
- $v(T_j)$  can be calculated attending to the predominant role that each student  $a_i \in T_j$  has ( $\operatorname{role}(a_i)$ )
- Let  $|T_j| = k$  denote the size of the team and  $\pi_j = \{r'_1, \dots, r'_k\}$  with  $\forall r'_i \in R$  be a vector with the true predominant role of each team member. In that case,  $v(T_j) = v(\pi_j)$

# An artificial intelligence tool for heterogeneous team formation in the classroom

After every activity...

- Students evaluate their peers by stating the most predominant role of each of his/her teammates
- Update the probability for an agent  $a_i$  to have  $r'_i$  as his/her most predominant role given the evaluation history  
 $p(role_i = r'_i | H)$
- Bayesian Learning



## Start course

Initially, in the 1st round, the groups are randomly generated

Teams are generated following the criterion:  
“teams with heterogeneous roles are better”

Using the information about the most predominant role of each student, the algorithm generates the next set of teams

### Coalition Structure Generation Algorithm

Re-execute policy

Teams formed

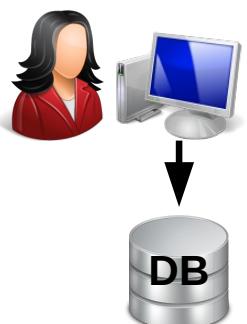


### Execute activity

Activity finished

Each student associates a role to his/her team members

### Evaluate teammates



Evaluated

### Update role information by Bayesian learning

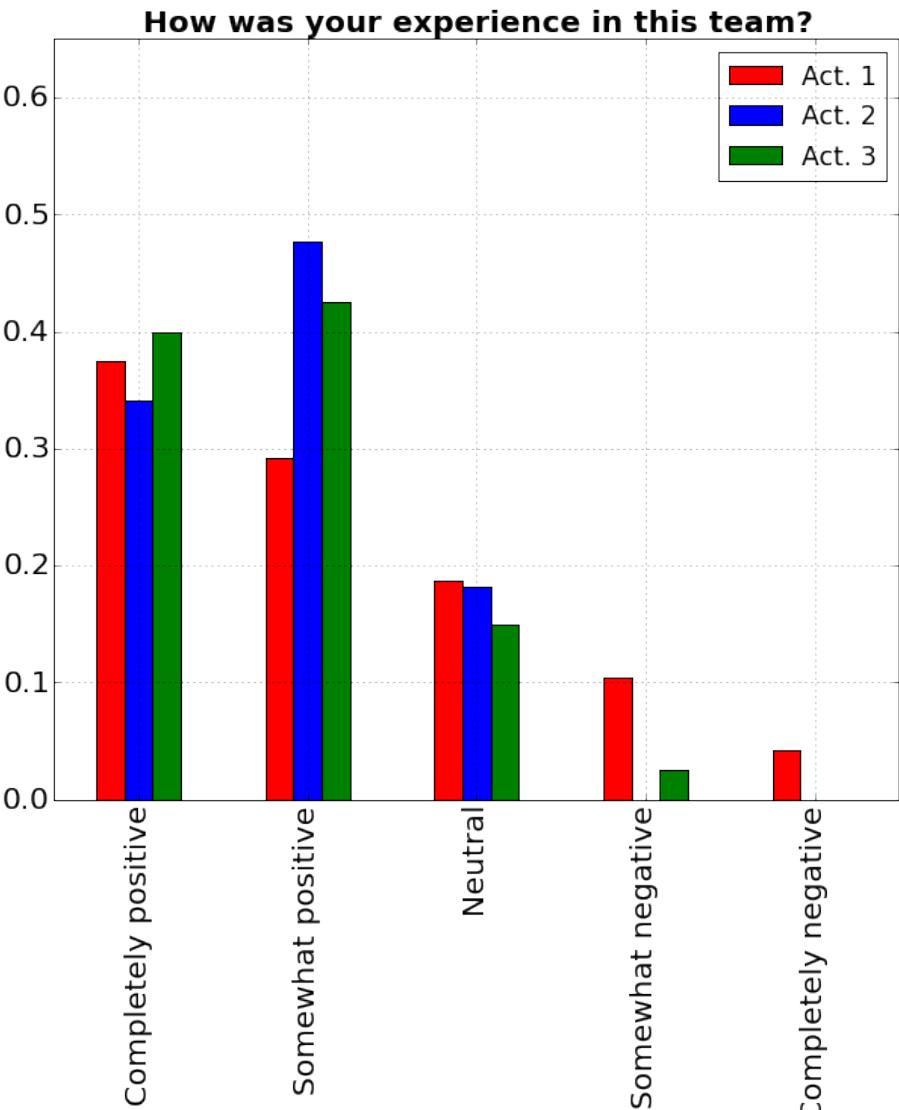
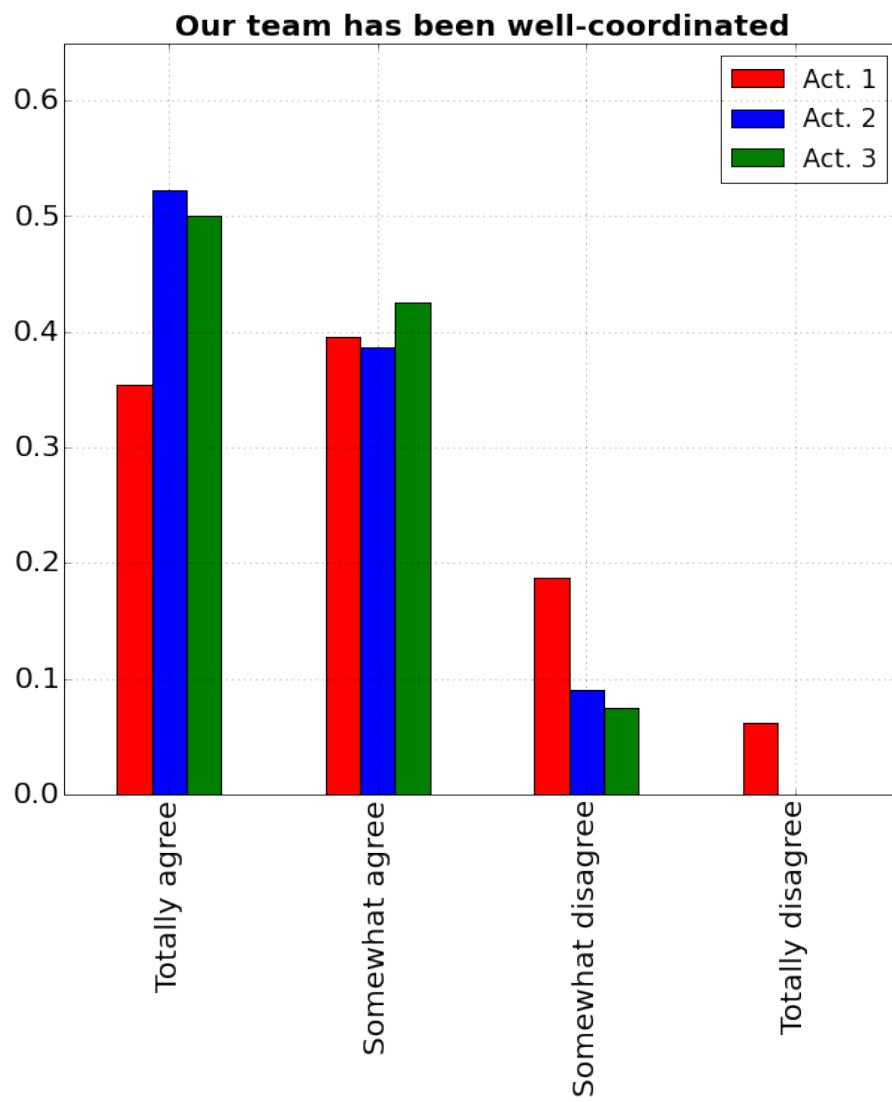
Bayesian learning is used to estimate the most predominant role of each student.  
In each iteration, there is more data available about each student. Therefore, the role estimation will be more accurate.

### Start new activity



Updated

# An artificial intelligence tool for heterogeneous team formation in the classroom



# Formación de coaliciones

Onn Shehory and Sarit Kraus. 1998. Methods for task allocation via agent coalition formation. *Artif. Intell.* 101, 1-2 (May 1998),

Shehory, O., & Kraus, S. (1996, December). Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. In Proc. of ICMAS-96 (pp. 330-337).

Radu-Casian Mihailescu, [Matteo Vasirani](#), [Sascha Ossowski](#): An Organizational Approach to Agent-Based Virtual Power Stations via Coalitional Games. [PAAMS \(Special Sessions\) 2011](#): 125-134

Dignum, F., Dunin-Keplicz, B., & Verbrugge, R. (2001). Agent theory for team formation by dialogue. In Intelligent Agents VII Agent Theories Architectures and Languages (pp. 150-166). Springer Berlin Heidelberg.