Coordinación PARTE II: Cooperation

1. Introduction

1.1 Advantages of Cooperation

- Completar tareas más rápido a través del esfuerzo compartido
- Al compartir recursos, lograr tareas que de otro modo no serían posibles
- Hacer uso de capacidades complementarias
- Evitar las interacciones dañinas

1.2 Cooperation challenges

- Evitar la duplicidad de esfuerzos
- Evitar interferencias, es decir, interacciones dañinas
- Evitar la sobrecarga de comunicación, es decir, la necesidad de compartir información con la menor cantidad de mensajes posible
- Necesidad de sincronizar comportamientos.

Todos estos requieren recursos computacionales y de memoria.

1.3 Modes of Cooperation

- **Accidental**: no intencionado
- **Con intención unilateral**: un agente ayuda intencionalmente a otro
- **Cooperación mutua**: dos o más agentes colaboran intencionalmente

2. Theories and models for Cooperation

Theories of Cooperation:

- Cooperative Problem Solving
- Joint Intentions
- Commitments
- Interaction Protocols
- Norms and Institutions
- Teamwork

Distributed Planning:

- PGP/GPGP
- MA-STRIPS/MA-PDDL
- MA-A*

Social Choice

Coordination by algorithm:

- DCOPs
- Distributed Consensus

2.1 Cooperation Mechanisms

Cooperating with message exchange

- Definido por primera vez por Cohen y Levesque, Wooldridge y Jennings
- Los agentes **se comunican entre sí** para compartir:
    - Tareas
    - Asignaciones de tareas
    - Información sobre el Estado del Mundo
    - Motivaciones
    - etc
- Estas comunicaciones forman la base para formar un acuerdo conjunto sobre qué hacer.
- Esto forma la base de un "**Proceso Cooperativo de Resolución de Problemas**"

2.2 Cooperative Problem Solving Process

Four steps to (cooperation) heaven

1. **Identificación del problema**: el proceso comienza cuando uno o más agentes identifican un problema para el cual se necesita cooperación.
2. **Formación del equipo**: el agente (o agentes) que reconocieron el problema solicitan asistencia y buscan a otros para ayudar con el problema. Si esta etapa tiene éxito, se forma un grupo con un compromiso conjunto para la acción.
3. **Formación del plan**: el equipo de agentes forma un plan de acción que utiliza las habilidades individuales del equipo. El resultado de esta etapa es una serie de compromisos individuales e interdependientes para actuar.
4. **Acción en equipo**: durante esta fase, los agentes realizan las acciones que tienen asignadas.

Seguido de limpieza

2.3 Joint Intentions

Descrito por primera vez por Cohen y Levesque:

Características comunes:

- **Realista**: los agentes deben creer que el estado de cosas deseado es alcanzable.
- **Temporalmente estable**: las intenciones deben ser persistentes en algún sentido (aunque no completamente inflexibles)

Algunos argumentan que las **Intenciones Conjuntas (Joint Intentions)** son **necesarias** para la **Acción Conjunta (Joint Action)**, es decir, si usted hace lo correcto, pero no tenía una intención conjunta, esto no era una Acción Conjunta.

Jennings veía a los **compromisos (Commitments)** como ejemplificaciones de intenciones conjuntas.

Jennings también introduce la **responsabilidad conjunta** como:

- Un **objetivo conjunto** (intención conjunta).
- Una **receta (plan)** para lograr ese objetivo.

Esto se basa en Intenciones Conjuntas para vincular una meta a acciones concretas ya que:

- Si tenemos el mismo objetivo no significa que necesariamente estemos de acuerdo en las acciones para lograrlo.
- Además, cuando empiezo a actuar, necesito estar seguro de que usted está comprometido a hacer su parte.

Hay una serie de **críticas** bien conocidas de las teorías basadas en las intenciones conjuntas (no es aplicable a todo):

- **No tener en cuenta la estructura social**: ¿qué pasa con la coerción? ¿Responsabilidad social?
- **Centrarse en las estructuras internas**: ¿a quién le importa lo que pretendamos mientras actuáramos de manera coherente?
- **Aplicabilidad limitada**: la teoría no funciona para (p. ej.) casos de coordinación implícita.

Sin embargo, la teoría proporciona un fuerte punto de enlace con enfoques como la confianza y la reputación.

2.4 Commitments about actions

- Otra forma de verlo es crear formas en que los agentes establezcan compromisos/compromisos sobre las acciones.
- Los compromisos traen beneficios, limitan (posibles) acciones y/u opciones futuras al modificar las (posibles) acciones de otros agentes en nuestro propio beneficio.
- Los agentes pueden beneficiarse de poder limitar sus acciones futuras (posibles) y realizar aquellas a las que se han comprometido.
- Un agente (solo) se compromete a realizar una acción futura, que restringe sus opciones/incentivos futuros, si recibe una ganancia mayor.

Mechanisms to acquire commitments

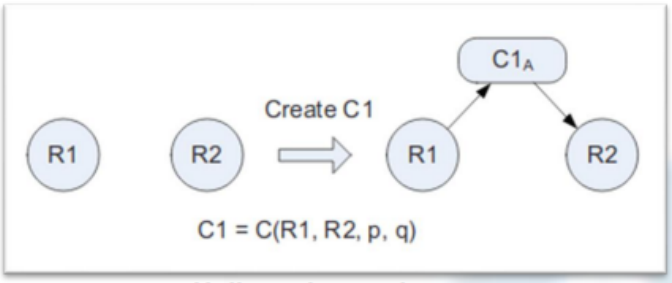
- Comprometerse con acciones/opciones futuras es siempre una decisión difícil.
- La ley, las reglas sociales, **las reglas de encuentro**, las promesas y las reglas de honor sí contribuyen a apoyar a un agente a comprometerse.

Example:

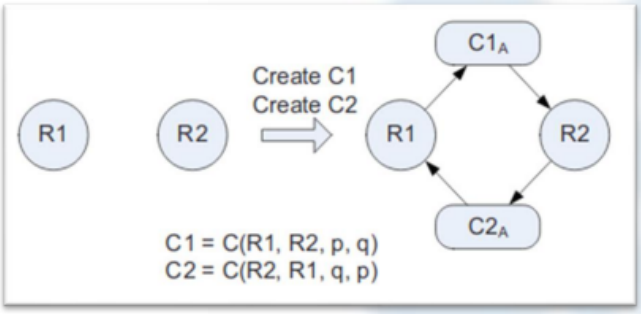
Commitments are defined as C(R1, R2, p, q), where:

- R1 is the debtor
- R2 is the creditor
- p is the antecedent
- q is the consequent

If p holds, then R1 is obliged to R2 to bring about q



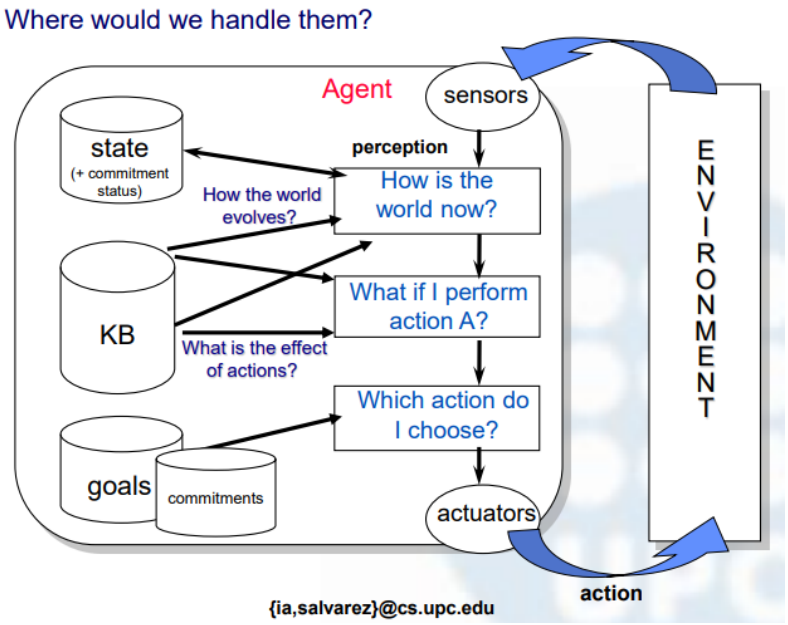
Unilateral commitment



Commercial transaction

Algorithm 1: verifyInteractions(m, i): Verify agent interaction model i with respect to business model m

```
1  C = m.C; // Model Commitments
2  CS = (); // Satisfied commitments
3  CV = (); // Violated commitments
4  T = i.T; // Tasks completed in the interaction model
5  foreach c ∈ C do
6      if (eval(c.consequent, T) = true) then
7          CS.add(c);
8  foreach ((c ∈ C) ∧ (c ∉ CS)) do
9      if (eval(c.antecedent, T) = true) then
10         CV.add(c)
11 return CV;
```



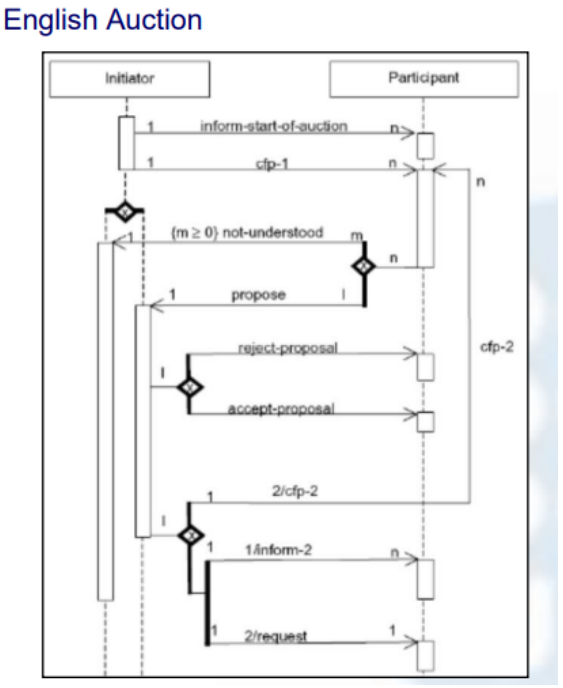
2.5 Cooperation protocols

- La estrategia básica es descomponer y luego distribuir tareas
- Descomposición realizada por el diseñador del sistema o por agentes

Mecanismos de distribución:

- **Mecanismo de mercado (Market mechanism)**: acuerdos generalizados o selección mutua
- **Neto de contratos (Contract net)**: ciclos de anuncio, licitación y adjudicación
- **Estructura organizativa (Organizational structure)**: los agentes tienen responsabilidades fijas
- **Planificación multiagente (Multiagent planning)**: los agentes de planificación realizan la asignación de tareas

2.5.1 Market mechanisms



subasta inglesa (English Auction)

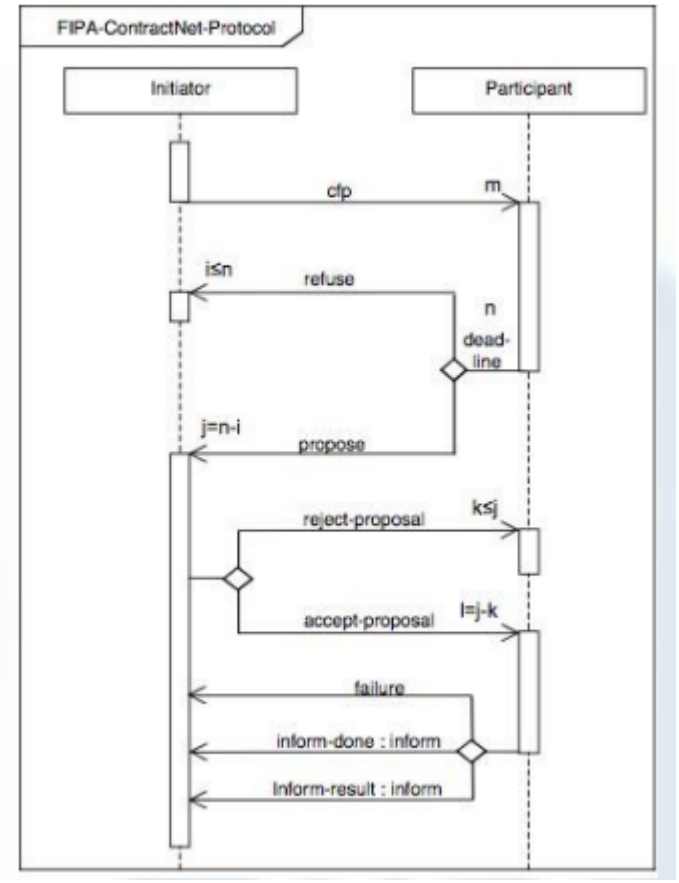
2.5.2 Contract-Net

Task-sharing protocol for task allocation

A contract net is a network of agents that can act as **managers** or **contractors** (or even both).

5 steps:

1. Recognition
2. Announcement
3. Bidding
4. Awarding
5. Expediting



2.6 Norms and institutions

Las **normas** son patrones de comportamiento o juicio compartidos por los miembros de un grupo social.

Las **instituciones** son contextos normativos:

- Mecanismo descentralizado de cooperación
- Bloque de construcción para organizaciones: asignación de normas a roles

¿Quién crea y hace cumplir las normas?

- **Normas legales**: aplicadas por especialistas a cambio de algo, generalmente descritas usando lógica deóntica
  Está prohibido pasarse un semáforo en rojo: is_red_light(x) → O(¬run(x))
- **Normas sociales**: generalmente emergentes, a menudo no orientadas a los resultados y aplicadas por un colectivo social
  Si el vagón está casi vacío, no se siente al lado de alguien
- **Normas morales**: consecuencialistas basadas en alguna concepción de bueno/malo (generalmente valores)
  Trata a los demás como quieres que te traten

on adoptados por los agentes como motivaciones externas (a diferencia de los deseos):

- La estabilidad del sistema dependerá de la aplicación

¿Cuáles son los efectos de las normas?

- **Reglas constitutivas** (count-as): dan forma a la realidad social al definir eventos institucionales a partir de hechos brutos
- **Reglas reglamentadas**: definen restricciones duras en el diseño e implementación del razonamiento de los agentes
- **Reglas regulativas**: definen restricciones blandas que deben ser aplicadas por otros: Sanciones, Acciones de reparación, Ostracismo, …

Los agentes pueden interpretar la **realidad social** utilizando las normas:

- Las normas actúan como restricciones para la planificación basada en objetivos
- Autonomía vs conformidad

2.7 Teamwork

Another view on CPS

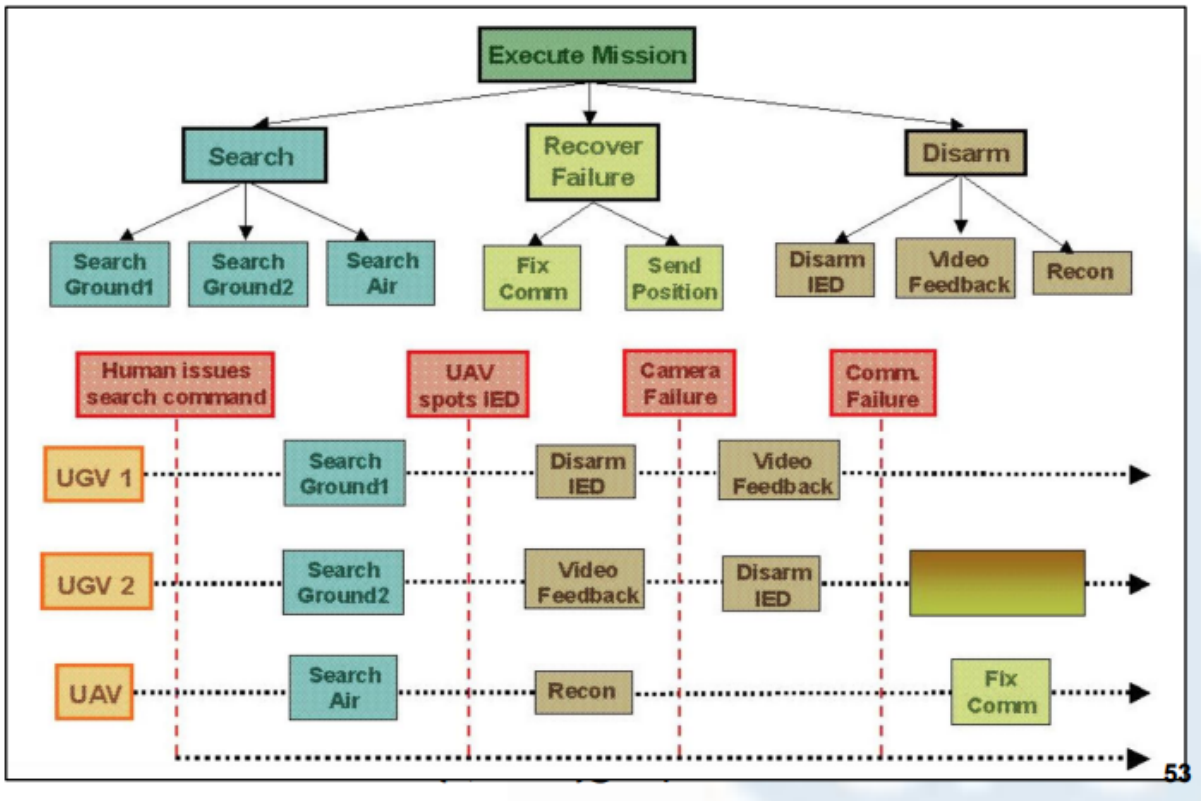
Un sabor particular de la resolución cooperativa de problemas que enfatiza el modelo del equipo (y las actitudes hacia el equipo) en lugar de las actitudes mentales individuales.

La teoría enfatiza:

- **Detección de interacciones**: detección de interacciones positivas y negativas entre subplanes
- **Plan de seguimiento y progreso del equipo**: ¿se alcanzan los objetivos? ¿Siguen siendo accesibles los miembros del equipo, etc.?
- **Planificación y resolución de conflictos dentro del equipo**: red de contratos y otros mecanismos de resolución de conflictos

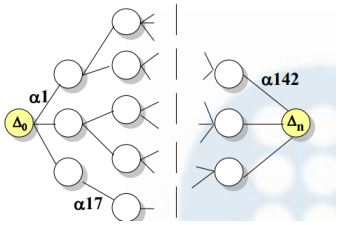
Los sistemas incluyen: STEAM, GRATE, COLLAGEN

Example



3. Distributed Planning

3.1 Traditional AI Planning



Un plan es una secuencia (lista) de acciones, con variables reemplazadas por constantes.

$Ac = \{\alpha_1, \ldots, \alpha_n\}$: a fixed set of actions.

$< P_\alpha, D_\alpha, A_\alpha >$ a descriptor for an action $\alpha \in Ac$

- P_α is a set of formulae of first-order logic that characterise the *precondition* of action α
- D_α is a set of formulae of first-order logic that characterise those *facts* made false by the performance of α (the delete list)
- A_α is a set of formulae of first-order that characterise those facts made *true* by the performance of α (the add list)

A **planning problem** is a triple $<\Delta, O, \gamma>$

$\pi=(\alpha_1, \ldots, \alpha_n)$: a plan with respect to a planning problem $<\Delta, O, \gamma>$ determines a sequence of $n+1$ models:

$$\Delta_0, \Delta_1, \ldots, \Delta_n$$

where $\Delta_0 = \Delta$ and

$$\Delta_i = (\Delta_{i-1} \setminus D_{\alpha_i}) \cup A_{\alpha_i} \quad \text{for} 1 \leq i \leq n$$

A plan π is acceptable iff $\Delta_{i-1} \vdash P_{\alpha_i}$,for all $1 \leq i \leq n$

A plan π is correct iff

- π is acceptable, and
- $\Delta_n \vdash \gamma$

Multiple Agents make planning difficult

Planificación de Inteligencia Artificial Tradicional:

- Se centra en la planificación de una sola Acción

  ¿Qué debo hacer?
- A menudo asume que el agente es el único actor en el mundo

  ¡¿Quién cerró la puerta?!
- Planner es omnisciente:

  Conoce toda la información relevante sobre el estado actual del mundo

  Conoce todas las acciones posibles que se pueden aplicar
- Las acciones son deterministas e instantáneas.
- Las metas son fijas y categóricas.

No es trivial generalizar a casos de múltiples agentes

Planning variations

Hay tres variaciones clave:

- Planificación en situaciones en las que se supone que **varios agentes amigos** deben trabajar juntos: ¿quién hace qué y cuándo? Sin embargo, los agentes son los únicos actores en el entorno.
- Planificación en situaciones donde hay **otros agentes (neutrales) presentes**
- Planificación en situaciones donde hay otros **agentes hostiles presentes**

Incluso los casos de agentes amistosos son complejos y requieren:

- Conocer las capacidades de otros agentes
- Compartir fragmentos de planes
- Coordinación de acciones individuales

3.2 Distributed Planning

Una combinación de planificación de IA tradicional y resolución de problemas distribuidos.

Hay tres sabores:

- la creación del plan está centralizada pero la ejecución del plan se puede distribuir
- la creación del plan está distribuida pero la ejecución del plan está centralizada
- se distribuye la creación del plan y se distribuye la ejecución del plan

"Autonomous agents in dynamic, multiagent environments also need to be able to manage the plans they generate.

They need to determine which planning problems and opportunities to consider in the first place.

They need to be able to weigh alternative incomplete plans and decide among competing alternatives.

They need to be able to form incomplete plans now, adding detail later, and thus, they need to be able to decide how much detail to include now and when to add more detail.

They need to be able to integrate plans with one another and to decide when to treat an existing plan as an inflexible commitment and when, instead, to consider modifications of it.

They also need to be able to do this in a way that comports with the inherent bounds on their computational resources."

3.2.1 Partial Global Planning

La Planificación Global Parcial (**PGP y GPGP**) es la metodología más representativa en este campo. Generación, coordinación y ejecución de planes intercalados:

- Los agentes crean fragmentos de planes
- Compártelos utilizando un protocolo de estilo de convocatoria de propuestas
- Los agentes modifican su comportamiento w.r.t. lo que creen que otros están haciendo.

Proporciona una capacidad dinámica para revisar los planes de manera rentable

- Definición de tareas flexible (lenguaje propuesto: TÆMS)

Supone comunicación a lo largo del tiempo.

Etapas:

- Descomposición de tareas
- Formación del Plan Local
- Abstracción del plano local
- Comunicación
- Identificación de objetivos globales parciales
- Construcción y Modificación del Plan Global Parcial
- Planificación de la comunicación
- Actuación en Planes Globales Parciales
- Modificación en curso
- Reasignación de tareas

3.2.2 Current state of the art

MA-PDDL/MA-STRIPS son otra alternativa para la planificación distribuida.

Diferencias sobre GPGP/TAEMS:

- Los agentes no identifican automáticamente las tareas
- La comunicación no es parte del marco
- Mayor énfasis en la privacidad individual
- Las metas no se construyen colectivamente

Son versiones multiagente de los lenguajes de planificación clásicos **PDDL/STRIPS**

A MA-STRIPS problem for a system of agents $\Phi = \{\varphi_i\}_{i=1}^k$ is given by a quadruple

$$\Pi = \langle P, \{A_i\}_{i=1}^k, I, G \rangle$$

where:

- P is a finite set of atoms (also called propositions),
- $I \subseteq P$ encodes the initial situation,
- $G \subseteq P$ encodes the goal conditions,
- For $1 \leq i \leq k$, A_i is the set of actions that the agent φ_i is capable of performing.
- Each action $a \in A$ has the standard STRIPS syntax and semantics.

A MA-STRIPS problem reduces to STRIPS when $k = 1$

Agents have their local KBs and capabilities

Definitions:

- $\text{eff}(a) = \text{add}(a) \cup \text{del}(a)$
- $P_i = \bigcup_{a \in Ai} (\text{pre}(a) \cup \text{eff}(a))$

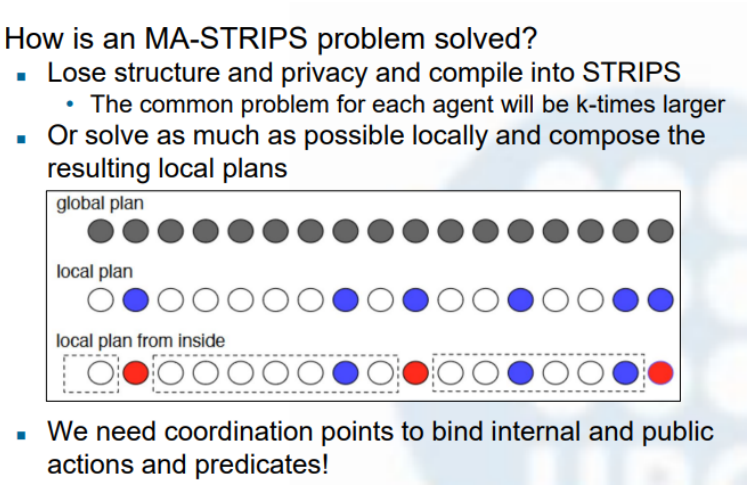
Internal and public atoms and actions:

- $P_i^{\text{int}} = P_i \setminus \bigcup_{\varphi(j) \in \Phi \setminus \{\varphi(i)\}} P_j$
- $P_i^{\text{pub}} = P_i \setminus P_i^{\text{int}}$

- P_i^{int} contains all atoms only the agent i can see
- P_i^{pub} contains all atoms every agent can see

- $A_i^{\text{int}} = \{a \mid a \in A_i, \text{pre}(a) \cup \text{eff}(a) \subseteq P_i^{\text{int}}\}$

Algorithm



Para cada valor x de un rango 1..N (profundización iterativa):

- Buscamos un conjunto de x acciones públicas que nos lleve a un objetivo:

  Estos son los puntos de coordinación

  No se puede utilizar la planificación anticipada tradicional (porque se deben ignorar las condiciones previas)

  Técnicas posibles: CSP (problema de satisfacción de restricciones), CSP distribuido, búsqueda local/heurística
- Planificación local:

  Cada agente genera sus propios planes locales utilizando un planificador STRIPS tradicional para ir de un punto de coordinación al siguiente
- Si se encuentran planes locales válidos para todos los puntos de coordinación, el algoritmo finaliza

Example

Ejemplo: dominio logístico

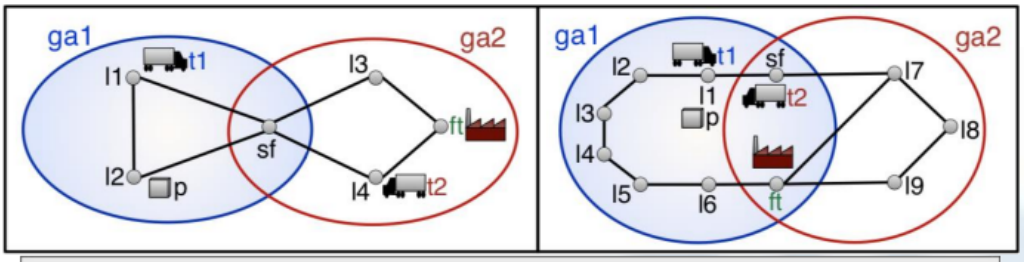
- Los agentes son camiones

  Tienen ubicación específica L en cada momento

  Pueden llevar una carga P
- Las acciones posibles son:

  conducir (T, L) que es privado

  cargar (P, T, L), descargar (P, T, L) que son públicos




```
(define (domain transport-agency)
  (:requirements :factored-privacy :typing :equality :fluents)
  (:types transport-agency area location package product - object
          truck place - location
          factory - place)
  (:predicates
    (manufactured ?p - product) (at ?p - package ?l - location)
    (:private
      (area ?ag - transport-agency ?a - area) (in-area ?p - place ?a - area)
      (owner ?a - transport-agency ?t - truck) (pos ?t - truck ?l - location)
      (link ?p1 - place ?p2 - place)
    )
  )
  (:action drive
    :parameters   (?ag - transport-agency ?a - area ?t - truck ?p1 - place ?p2 - place)
    :precondition (and (area ?ag ?a) (in-area ?p1 ?a) (in-area ?p2 ?a)
                  (owner ?a ?t) (pos ?t ?p1) (link ?p1 ?p2))
    :effect       (and (not (pos ?t ?p1)) (pos ?t ?p2))
  )
[...]
)
```

73

Task	$\mathcal{T}_1$			$\mathcal{T}_2$		
$\mathcal{AG}$	$ta1$	$ta2$	ft	$ta1$	$ta2$	ft
$\mathcal{P}^i$	$(pos\ t1\ *)$	$(pos\ t2\ *)$	$(pending\ fp)$	$(pos\ t1\ *)$	$(pos\ t2\ *)$	$(pending\ fp)$
	$(at\ p\ *)$		$(at\ p\ ft)$	$(at\ p\ *)$		$(at\ p\ ft)$
	$(manufactured\ fp)$			$(manufactured\ fp)$		
$\mathcal{A}^i$	drive, load, unload		manufacture	drive, load, unload		manufacture
$\mathcal{I}^i$	$(pos\ t1\ l1)$ $(at\ p\ l2)$	$(pos\ t2\ l4)$	$(pending\ fp)$	$(pos\ t1\ l1)$ $(at\ p\ l1)$ $(at\ p3\ ft)$	$(pos\ t2\ sf)$	
$\mathcal{G}$	$(manufactured\ fp)$			$(at\ p\ ft)$		

```
(define (problem ta1)
  (:domain transport-agency)
  (:objects
    ta1 - transport-agency
    ga1 - area                    l1 l2 sf - place
    p - package                   fp - product
    (:private t1 - truck)
  )
  (:init
    (area ta1 ga1) (pos t1 l1) (owner t1 ta1) (at p l1)
    (link l1 l2) (link l2 l1) (link l1 sf)
    (link sf l1) (link l2 sf) (link sf l2)
    (in-area l1 ga1) (in-area l2 ga1) (in-area sf ga1)
  )
  (:goal (manufactured fp))
)
```

3.2.3 MA-A*

Una alternativa a MA-STRIPS/MA-PDDL es MA-A*. Similar a A*, en cada iteración:

- Recuperar el primer nodo n de la lista abierta
- Si n es una solución, realice una verificación de optimización distribuida
- Expanda n usando sólo las propias acciones del agente
- Calcule el valor h y agregue todos los niños n0 a la lista abierta
- Si n0 se obtuvo mediante la aplicación de una acción pública con condiciones previas públicas que tienen:

  Enviar n0 a todos los agentes para los que n0 sea relevante
- Si se recibe un mensaje de otro agente con un nodo n0, se agrega n0 a la lista abierta

4. Social Choice

Teoría de la elección social (Social Choice Theory):

- ¿Cómo tomar una decisión cuando los agentes que tienen incentivos asimétricos acceden a votar?
- El resultado de un mecanismo (o función) de elección social es una agregación de las preferencias de todos

Modelo formal para una función de elección social:

- Dado:

  Un conjunto de agentes N = {1, 2, …, n}

  Un conjunto de preferencias V = {v1, v2, ..., vn} para cada agente

  Un conjunto de resultados O (p. ej., conjuntos de recompensas, asignaciones de tareas potenciales o candidatos para el liderazgo)
- Una función de elección social es cualquier función f: V → O

1	2	3
A	B	C
B	C	A
C	A	B

→ ***A, B or C?***

4.1 Condorcet Paradox

Una función básica de elección social es la mayoría (N/2 + 1)

Paradoja de Condorcet:

- Suponga tres candidatos potenciales para la elección: A, B, C
- Si se eligiera A sobre B y B sobre C, ¿podemos asumir que A sería elegido sobre C?

- ¡No! Las diferentes mayorías pueden estar compuestas por diferentes personas.
- Transitividad sobre las preferencias individuales no implica transitividad sobre las preferencias sociales

4.2 Desirable properties

Propiedades deseables de una función de elección social:

- Siempre un ganador:

  Siempre hay al menos un ganador, independientemente de la entrada
- Condorcet condición de ganador:

  Por cada alternativa y al ganador x, una mayoría siempre prefiere x sobre y
- Condición de Pareto:

  Si todos prefieren x sobre y, y nunca puede ser el ganador
- Monotonicidad:

  Si y es el ganador en una configuración específica y, después de eso, un agente clasifica a y aún más alto, y aún debe ser el ganador
- Independencia de alternativas irrelevantes:

  Si x es un ganador e y no lo es, y los agentes cambian sus preferencias sobre otras alternativas pero no sobre x e y, y no debería convertirse en un ganador

Some Social Choice functions

Método de Condorcet:

- x está entre los ganadores si para cada alternativa y, la mayoría prefiere x sobre y

Voto por pluralidad (Plurality voting):

- Los ganadores son las opciones con el mayor número de rankings de primeros lugares

Cuenta Borda (Borda Count):

- Cada opción (de un conjunto de n) obtiene (n – r) para cada agente, siendo r el rango de ese agente

Hare:

- Eliminar la opción menos clasificada y repetir la votación hasta que solo quede una opción

Votación secuencial por parejas con agenda fija:

- Decidir un orden de opciones y votar entre la primera y la segunda, luego votar entre la ganadora y la tercera, etc.

Dictadura:

- Decidir un votante como dictador y el primer clasificado para el dictador será la elección social

4.3 Mechanism Design

Teoría de juego:

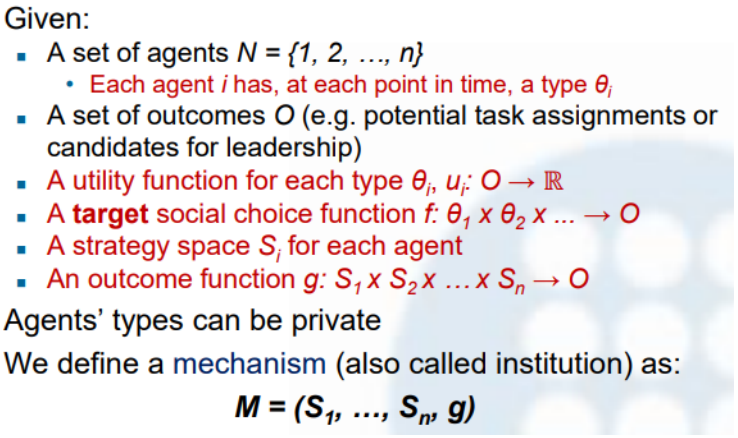
- ¿Qué estrategias deberían seguir los agentes racionales, dado un juego específico (estructura de pago)?

Vamos a darle la vuelta: **Diseño de mecanismos** (Maskin, Myerson, Hurwicz):

- ¿Qué juego (estructura de pago) deberíamos diseñar para nuestros objetivos, dado un conjunto de agentes racionales con incentivos asimétricos?
- ¿Qué garantías podemos tener sobre los resultados de este juego diseñado?

El diseño del mecanismo puede ser adoptado voluntariamente por un grupo que busca cooperación. También puede hacer cumplir/promover la cooperación en presencia de agentes no confiables o impredecibles.

- Problema principal – agente: ¿puedo confiar en mi abogado/corredor/banquero…? ¿Puedo crear una estructura de incentivos que garantice que el agente actúe en mi mejor interés?



Algoritmo básico:

- Un director (o diseñador de mecanismos) diseña y se compromete con un mecanismo M.
- Cada agente recibe en privado información interesada (¿incompleta?) sobre M.
- Cada agente se compromete con un tipo θ (podría ser mentira).
- Luego, cada agente juega el juego definido por M.

El objetivo del diseñador de mecanismos es encontrar una función g tal que:

- Los agentes están motivados para compartir información veraz
- Los agentes alcanzan un equilibrio

- Las estrategias dominantes de cada tipo equiparan g (función de resultado) a f (función de elección social objetivo)

$$g(s1^*(\theta 1), s2^*(\theta 2), \ldots) = f(\theta 1, \theta 2, \ldots)$$

El campo de Diseño de Mecanismos propone algoritmos y métodos para lograr esto: “La parte ingenieril de la economía”.

Ejemplos:

- Quieres que dos niños se dividan un pastel de manera justa solos. ¿Cómo se aseguraría de que lo lograran?
- ¿Cómo modificaría el dilema del prisionero para que ambos agentes deserten? ¿Y para que ambos agentes cooperen?

Las subastas son ejemplos de mecanismos. Cada tipo de subasta está diseñado para un tipo de objetivo diferente

- Maximizar los ingresos del vendedor: subasta en inglés
- Maximizar el bienestar – Subasta Vickrey

5. Coordination by algorithm

La cooperación por "Algoritmo" es algo controvertida ya que algunos enfoques no permiten una Autonomía significativa del Agente en el proceso.

Dos enfoques principales:

- **Satisfacción/Optimización de Restricciones Distribuidas (DCSPs/DCOPs)**: una extensión de las técnicas de resolución de CSP que capturan varias variables en cada agente. Los agentes propagan opciones para las "variables de borde" que afectan a otros.
- **Algoritmos de consenso**: una familia de métodos para alcanzar, automáticamente, un acuerdo unificado de manera descentralizada, dado que todas las partes aceptarán, implementarán y apoyarán el resultado.

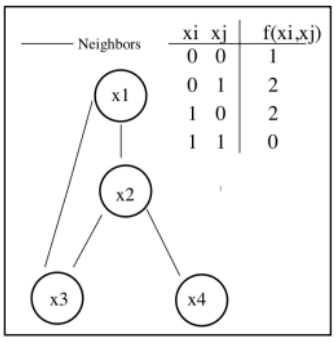
5.1 Distributed Constraint Optimisation Problems (DCOPs)

DCOP is a type of problem in which a set of agents distributedly assign values to a set of variables, optimising the cost attributed to those values

All algorithms that solve DCOP can solve all problems of this family

DCOP can be defined as a tuple $\langle A, V, \mathfrak{D}, f, \alpha, F \rangle$, where

- A is a set of agents
- V is a set of variables
- $\mathfrak{D}$ is a set of domains for the values in V
- f is a function mapping every possible variable assignment to a cost, which can be infinite
- α is a function mapping variables to each agent
- F is an operator aggregating individual costs for all possible variable assignments, usually a summation

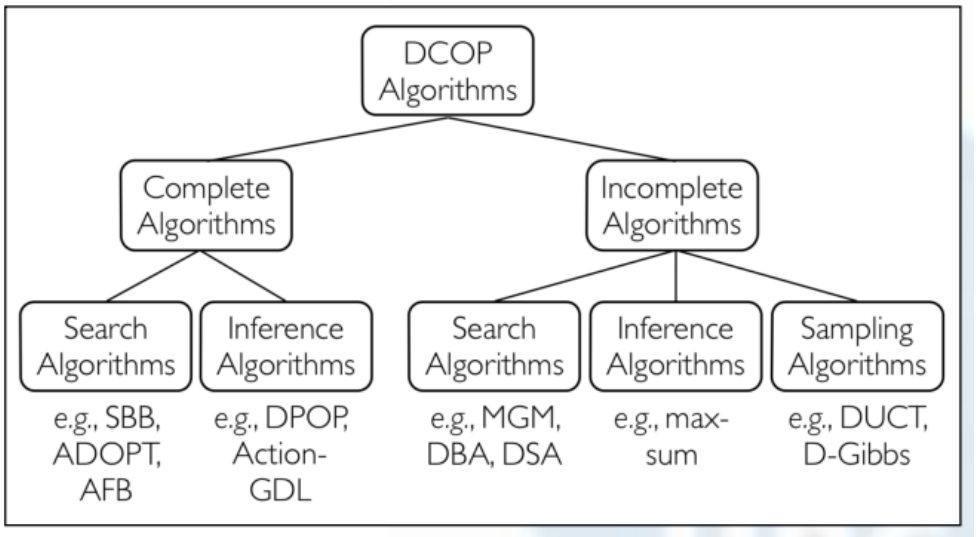


- Assuming minimization
- We prefer to have neighbors with the same value, with a preference for (1, 1)
- An example of algorithm would be to let x_1 assign a value and continue with x_2, and so on
- In any case, the system will end up optimising to all 1's

$$F(\mathcal{A}) = \sum_{x_i, x_j \in X} f_{ij}(d_i, d_j) \text{ where } x_i \leftarrow d_i \text{ and } x_i \leftarrow d_i \text{ in } \mathcal{A}$$

$$F(\{(x_1, 0), (x_2, 0), (x_3, 0), (x_4, 0)\}) = 4$$
$$F(\{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 1)\}) = 0$$

DCOP algorithms



Recap

Utilizado en muchos dominios:

- Redes IoT, Wireless y P2P
- Programación de reuniones
- Control de tráfico

Los DCOP son NP-Hard. Los algoritmos completos son óptimos pero no escalan bien.

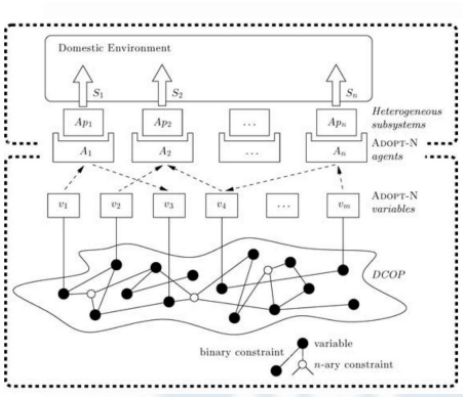
El rendimiento de los algoritmos incompletos depende mucho del dominio de la aplicación:

- Abordan la sobrecarga de comunicación o la complejidad computacional

No se deja mucha autonomía a los agentes, esp. si las variables representan la asignación de tareas. No hay mucha tolerancia a fallas, los agentes deben ser confiables

Example from *Pecora, Federico, and Amedeo Cesta. "DCOP for smart homes: A case study." Computational Intelligence 23.4 (2007): 395-419.*

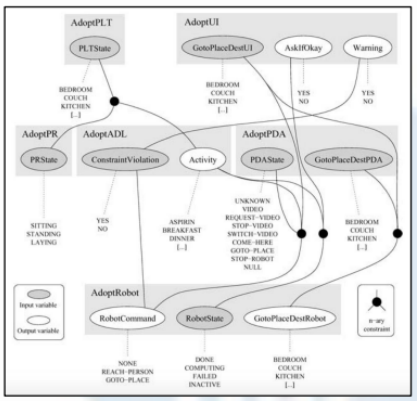
Variables v_i can be inputs or outputs, e.g. person/robot location or service triggers



The desired behavior of the whole *smart home* is defined as a set of constraints relating variables

Examples of cost assignments:

- *f(Activity == EMERGENCY && RobotCommand == REACH-PERSON) = 0*
- *f(Object == COFFEE && Activity == BREAKFAST) = 10*

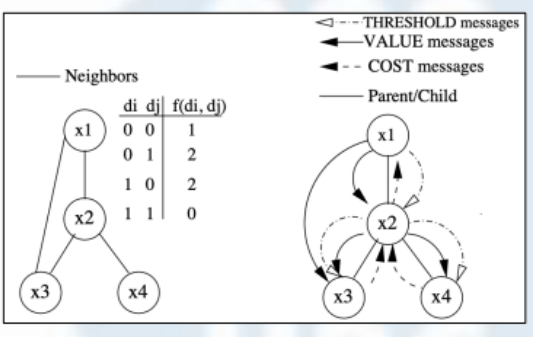


ADOPT

Es el algoritmo DCOP más popular

Los agentes están estructurados en un árbol:

- Los padres-hijos están determinados por una optimización de las relaciones
- Restricciones propagadas entre padres e hijos
- No se propagan restricciones entre hermanos



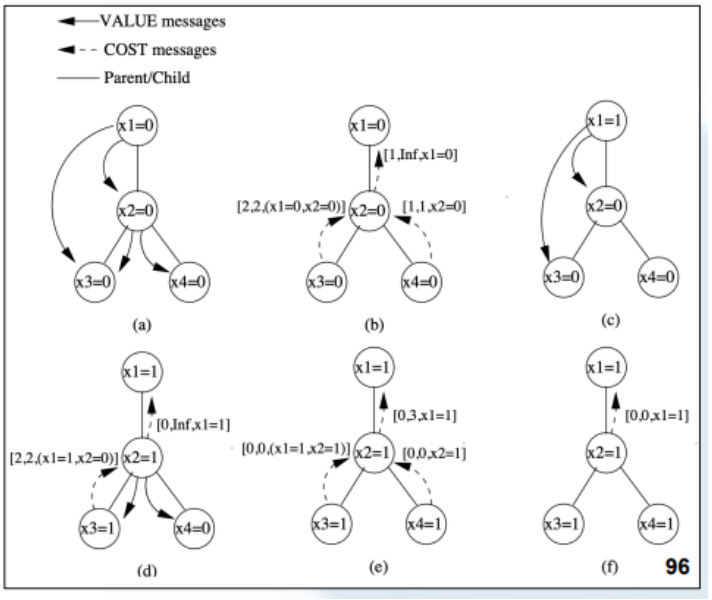
Para cualquier nodo, de forma asíncrona:

- Elija VALOR y envíelo a {vecinos} - {padres}
- Enviar COSTO = [límite inferior del árbol, límite superior del árbol, contexto] al padre
- Cuando se recibe cualquier VALOR/COSTO

    Actualizar UMBRAL con valores mínimos para niños

    Vuelva a calcular el VALOR, manteniéndose dentro del UMBRAL

        Si no es posible optimizar, TERMINAR



5.2 Distributed Consensus algorithms

- **Problema de consenso**: cómo lograr la confiabilidad general del sistema en presencia de agentes defectuosos
- Un protocolo de consenso es un protocolo de interacción que es tolerante a fallas o resiliente w.r.t. un número (limitado) de agentes defectuosos
- En general, un protocolo de consenso describe cómo mantener un valor único en un sistema multiagente, a través de la elección del líder o la votación por mayoría.
- El valor único puede ser muy complejo, por ejemplo, el libro mayor de Bitcoin
- Un protocolo de consenso define los procesos que debe ejecutar cada agente para mantener este valor

Propiedades deseadas:

La tolerancia a fallos en consenso se define como el cumplimiento de las siguientes propiedades:

- **Terminación**

    Eventualmente, cada proceso correcto decide algún valor

- **Integridad (Vigencia)**

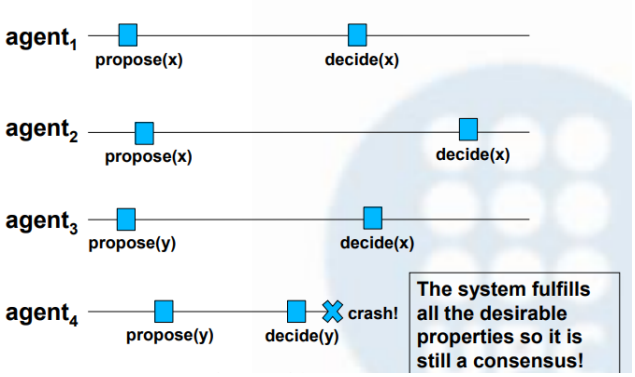
  Si todos los procesos correctos propusieron el mismo valor, entonces cualquier proceso correcto debe decidir ese valor

  Un nodo decide como máximo una vez

  Cualquier valor decidido es un valor propuesto

- **Acuerdo**

  Todo proceso correcto debe coincidir en el mismo valor

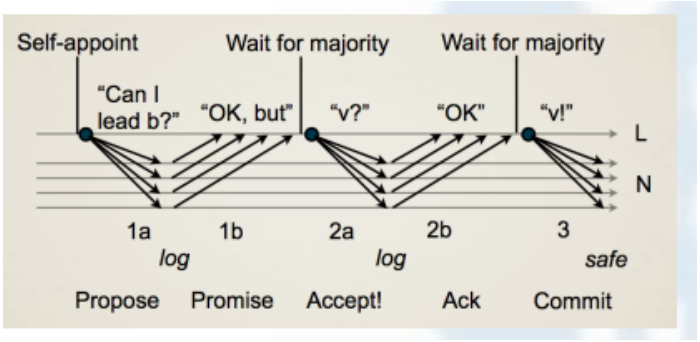


5.2.1 Paxos

Paxos es actualmente el algoritmo de consenso distribuido más popular:

- Raft es una propuesta similar, (posiblemente) más simple formalmente

Una máquina de estado específica del dominio se replica en todo el sistema.



En Paxos, hay tres roles de agente:

- Proponente
- Aceptador
- Aprendiz

Cualquier agente puede convertirse en proponente en cualquier momento

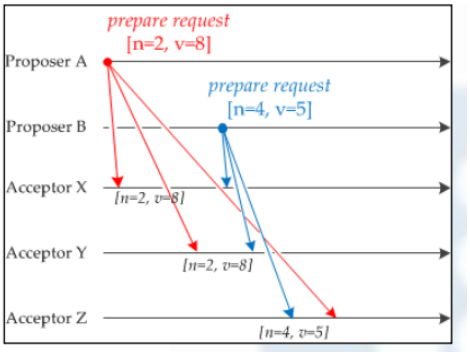
Un proponente crea un Quórum de Aceptantes para su propuesta

- Un quórum es un subconjunto del conjunto de agentes que es mayoría, es decir, su tamaño es mayor que la mitad del tamaño del conjunto completo
- {A,C,D} es un quórum posible para {A,B,C,D}

El resto de agentes son aprendices para el alcance de la propuesta

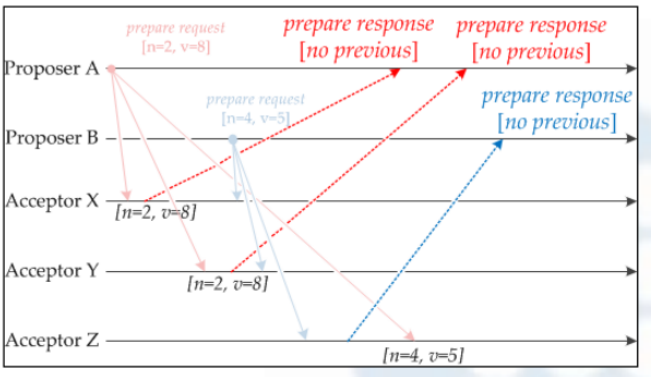
Fase 1a: Preparar

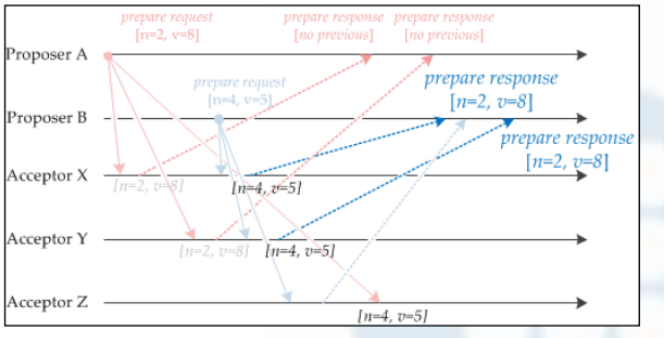
- Un proponente P crea una propuesta identificada con un número N
- N debe ser mayor que cualquier número de propuesta anterior utilizado o recibido por el proponente P
- P envía un mensaje de preparación que contiene esta propuesta a un quórum de aceptantes
- El Proponente P decide quién está en el Quórum



Fase 1b: Promesa

- Si el número de propuesta N es mayor que cualquier número de propuesta anterior recibido de cualquier proponente por un aceptador A, entonces A debe devolver una promesa de ignorar todas las propuestas futuras que tengan un número menor que N.
- Si A aceptó una propuesta en algún momento en el pasado, debe incluir el número de propuesta anterior N' y el valor anterior v en su respuesta a P.

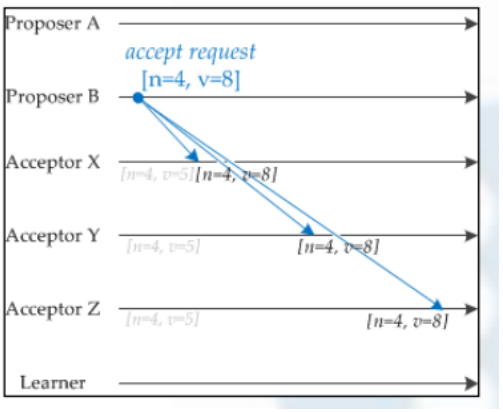




Fase 2a: Aceptar solicitud

- Si P recibe suficientes promesas de un Quórum de aceptantes, debe **establecer un valor para su propuesta**
- Si algún aceptador había aceptado previamente alguna propuesta, entonces P está recibiendo sus valores

  P debe entonces **establecer el valor de su propuesta en el valor v** asociado con el número de propuesta más alto informado por los aceptantes.
- Si ninguno de los aceptantes hubiera aceptado una propuesta hasta este punto, **P puede elegir cualquier valor** para su propuesta



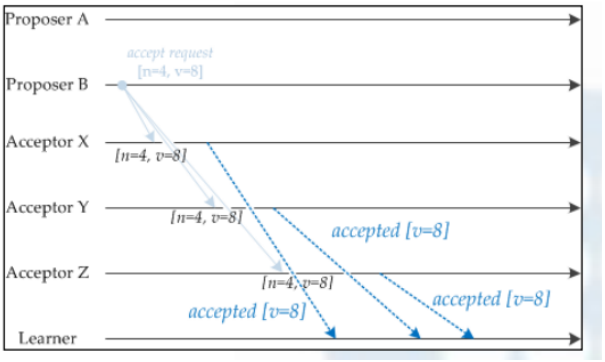
Fase 2b: Aceptar

- Si un aceptador A recibe un mensaje de solicitud de aceptación para una propuesta N.
  - Si ya se ha comprometido a considerar sólo las propuestas que tengan un identificador mayor que N

    A **ignora** la solicitud de aceptación
  - Si no se ha comprometido a considerar ninguna propuesta para un identificador mayor que N

    A **registra** el valor v correspondiente asociado a la propuesta N

    A **envía un mensaje de Aceptado** al proponente P y a todos **los aprendices**



Paxos permite:

- Agentes operando a velocidad arbitraria
- Agentes que experimentan fallas
- Los mensajes se pueden enviar de forma asíncrona
- Los mensajes se pueden perder, reordenar o duplicar
- Los agentes con almacenamiento persistente pueden volver a unirse después de fallas (siguiendo un procedimiento de recuperación de bloqueo específico)

La consistencia está garantizada siempre que:

- Si los agentes F fallan, los agentes 2F + 1 deben estar funcionando correctamente
- Los agentes pueden enviar mensajes a cualquier otro agente
- Los mensajes se entregan sin corrupción
- Las fallas bizantinas no ocurren

Aplicaciones y características de Paxos:

- Paxos, Raft y otros se utilizan en una amplia gama de aplicaciones, desde bases de datos hasta vehículos aéreos no tripulados.
- Muy confiable para escenarios asincrónicos distribuidos donde se supone una falla ocasional.
- Al igual que los DCOP, no permiten mucha autonomía del agente de forma predeterminada. Pero no requieren implementaciones de ciclo de razonamiento muy complejas.
- Las fallas bizantinas se pueden resolver con, por ejemplo, extensiones de Paxos o con Blockchain.

5.2.2 CBAA (Consensus-based bundle algorithm)

Objetivo: coordinar (de manera descentralizada) una flota de vehículos autónomos

Combinación de dos algoritmos descentralizados:

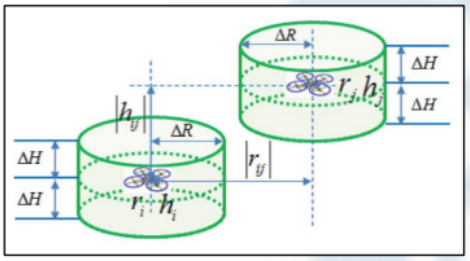
- Subasta de mercado
- Consenso

Garantías (bajo ciertas restricciones de modelado):

- Convergencia
- Asignación sin conflictos
- Pago global ≥ (0,5 * óptimo teórico)

Example from *Kuriki, Yasuhiro, and Toru Namerikawa. "Consensus-based cooperative formation control with collision avoidance for a multi-UAV system." 2014 American Control Conference. IEEE, 2014.*

In order to avoid collisions, the leader (which will compute the model) is elected via consensus



6. The future of cooperation in AI

La IA cooperativa es un campo en rápido crecimiento con muchas preguntas abiertas, como:

- ¿Cómo deberían los agentes y los humanos encontrar puntos en común en la comunicación, la comprensión del mundo y el trato con los compromisos?
- ¿Cómo definir y evaluar la inteligencia cooperativa?
- ¿Cómo diseñar instituciones para que haya cooperación real en lugar de coerción?
- Interpretabilidad de modelos de comportamiento y sesgos

Retos de MARL (aprendizaje reforzado con múltiples agentes):

- Entornos altamente dinámicos
- Objetivos asimétricos
- ¿Cuáles son los efectos de mis acciones y cuáles son los efectos de las acciones de los demás?
- RL no escala bien cuando N >> 2