10 COOPERACIÓN DISTRIBUIDA

¿Cómo podemos diseñar sistemas donde nuestros agentes trabajen juntos de manera efectiva? La idea de sistemas que trabajan juntos pareciera no ser una novedad, en el área de los sistemas distribuidos y concurrentes es común hablar de cooperación. Sin embargo, existen dos **diferencias** importantes entre los Sistemas Multi-Agentes (SMA) y los sistemas distribuidos tradicionales:

SMA vs Sistemas Distribuidos

1. Los agentes en un SMA pueden estar diseñados e implementados por diferentes individuos, con metas diferentes. Por lo tanto, estos agentes no comparten metas comunes. Sus encuentros están más cercanos de un juego o **competencia**, en donde actúan estratégicamente para alcanzar sus resultados preferidos.

   Competencia

2. Debido a que asumimos que los agentes actúan de manera autónoma tomando decisiones en tiempo de ejecución, estos deben ser capaces de **coordinar** sus actividades y cooperar con los demás agentes en tiempo de ejecución. En los sistemas distribuidos tradicionales, los mecanismos de coordinación y cooperación se establecen estáticamente durante su diseño.

   Coordinación dinámica

Trabajar conjuntamente involucra diferentes actividades, que revisaremos en detalle en este capítulo, En particular, el hecho de compartir tareas e información, así como la dinámica de coordinación en tiempo de ejecución.

10.1 SOLUCIONES COOPERATIVAS Y DISTRIBUIDAS

La resolución cooperativa y distribuida de problemas (*CDPS* por sus siglas en inglés) fue planteada originalmente en los trabajos de Durfee, Lesser y Corkill [56]. Su objetivo es estudiar como una red débilmente acoplada de programas capaces de resolver problemas, puede proponer soluciones más allá de las capacidades individuales de los miembros de la red. La cooperación es necesaria porque ninguno de estos programas es capaz, por si mismo, de solucionar los problemas que se le plantean al sistema. Esto se debe a su falta de experiencia, de recursos, o de información.

CDPS

La mayoría de los trabajos en CDPS asumen la **benevolencia** de los agentes: Los componentes del sistema comparten una meta común y, por lo tanto, no hay conflictos potenciales entre ellos. Esto supone que los agentes son diseñados para ayudar cuando es necesario. Lo que importa es resolver el problema planteado al sistema como un todo. La benevolencia se puede aceptar, cuando todos los agentes son diseñados y pertenecen a una misma organización o individuo. Asumir benevolencia simplifica enormemente el diseño de un SMA.

Benevolencia

En contraste a esta posición, los trabajos en SMA asumen que los agentes involucrados en el sistema tienen sus **propios intereses**. Los agentes en un SMA no asumen metas comunes y son diseñados normalmente por diferentes organizaciones o individuos. Por lo tanto, los intereses de un agente pueden entrar en conflicto con los intereses de otros agentes. A pesar de estos conflictos, al igual que en un sistema CDPS, los agentes de un SMA deberán cooperar para resolver los problemas que enfrentan.

Interés propio

Es importante también diferenciar un CDPS de los sistemas de **solución de problemas en paralelo** Bond y Gasser [12]. En estos sistemas los componentes son simples procesos y un solo componente es responsable de descomponer la tarea principal en subproblemas asignados a estos procesos. Los componentes del sistema se asumen homogéneos, en el sentido que no tienen experiencia diferenciada. Aunque los sistemas de solución paralela se consideraban sinónimos de los CDPS, actualmente suelen considerarse como dos áreas de estudio diferentes.

Solución de Problemas en Paralelo

La literatura de los SMA propone dos temas a considerar cuando se trata de resolver la siguiente cuestión: Una vez implementado un sistema de agentes artificiales para resolver un problema ¿Cómo puede uno **evaluar su éxito** o fracaso? [12].

Éxito

- **Coherencia.** ¿Qué tan bien el SMA se comporta como una unidad, sobre alguna dimensión de evaluación? La coherencia debe medirse en términos de calidad de la solución, eficiencia en el uso de recursos, claridad conceptual de la operación, o qué tan bien el sistema se degrada en presencia de ruido y falta de información.
- **Coordinación.** ¿En qué grado los agentes pueden evitar procesos relacionados con la sincronización y alineamiento de sus actividades. La presencia de conflictos donde los agentes interfieren entre si, es un indicador de coordinación pobre.

Investigación en CDPS

Los principales **temas de investigación** en que se focaliza el trabajo en CDPS incluye:

- ¿Cómo puede un problema dividirse en tareas más pequeñas que puedan distribuirse entre un grupo de agentes?
- ¿Cómo puede una solución sintetizarse efectivamente a partir de los resultados obtenidos para los sub-problemas?
- ¿Cómo pueden ser optimizadas las todas actividades para obtener soluciones que maximicen la medida de coherencia?
- ¿Qué técnicas pueden usarse para coordinar la actividad de los agentes, de forma que se eviten las interacciones destructivas y se maximice la efectividad al explotar cualquier interacción positiva?

10.2 COMPARTIENDO TAREAS Y RESULTADOS

¿Cómo puede un grupo de agentes trabajar conjuntamente para resolver problemas? Smith [181] sugirió que el proceso de CDPS puede verse canónicamente como un proceso en tres pasos:

1. **Descomposición del problema.** La descomposición del problema global en sub-problemas es típicamente **jerárquico**. Los diferentes niveles de descomposición corresponden a diferentes **niveles de abstracción** Observen que el nivel granular de los problemas es importante: un enfoque extremo del CDPS diría que la descomposición debe continuarse hasta que los sub-problemas en cuestión representen acciones atómicas que no pueden descomponerse más. Esto es esencialmente lo que se hace en el paradigma conocido como ACTOR [1]. Otros problemas asociadas a la descomposición incluyen ¿Quién llevará a cabo la descomposición de tareas? En el caso **centralizado**, un sólo agente lleva a cabo esta tarea, necesitando tener experiencia sobre la estructura de la tarea en cuestión. Otra posibilidad es tratar la distribución como una tarea descentralizada, donde todos los agentes tienen cierto conocimiento de la estructura del problema.

   Jerárquico
   Nivel de abstracción
   Centralizado vs Descentralizado

2. **Solución de subproblemas.** Los problemas identificados en la fase de descomposición son resueltos individualmente por los agentes del SMA. La fase involucra normalmente **intercambio de información** entre los agentes.

   Intercambio de información

3. **Síntesis de la solución.** Las soluciones individuales a los sub-problemas identificados en la primera fase, son **integrados** en una solución global. Al igual que la descomposición, esta tarea puede ser jerarquizada: las soluciones parciales se ensamblan en diferentes niveles de abstracción.

   Integración

Existen por tanto, dado este marco de referencia para los CDSP, dos actividades específicas asociadas a la resolución distribuida de problemas: compartir tareas y compartir resultados.

10.2.1 Compartiendo tareas en una red de contratos

Smith [181] propuso un protocolo conocido como **red de contratos** (*CNP*, por sus siglas en inglés), orientado a lograr una cooperación eficiente a través de la distribución de tareas en una red de agentes comunicantes, usando la metáfora de una **contratación**: Cuando un agente que genera una tarea, debe anunciar a los demás agentes la existencia de esta tarea mediante un **anuncio de tarea**. En ausencia de información sobre las capacidades de los demás miembros de la red, el administrador utilizará una forma de *broadcasting* para anunciar la tarea. Si por el contrario, conoce la competencia de otros agentes en la red, puede hacer un *broadcasting* limitado a los agentes de su interés. Más aún, si tiene conocimientos suficientes, puede hacer convocatorias puntuales. El agente que anuncia una tarea debe comprometerse a actuar como **administrador** de la misma. Conforme la solución del problema

Contract Net Protocol
Contratación
Anuncio de tarea
Administrador

original progrese, diferentes administradores generarán diferentes anuncios de tarea.

Los agentes en la red escuchan los anuncios de las tareas y los evalúan con respecto a sus capacidades y recursos. Cuando un agente se reconoce capaz de llevar a cabo una tarea anunciada, emite una **oferta**, como en las subastas. La oferta indica las capacidades del agente que la somete, con respecto a la tarea a resolver. El administrador de la tarea en cuestión recibe diversas ofertas y con base en la información recibida, selecciona a los agentes más apropiados. La selección es comunicada a los agentes que tuvieron éxito, mediante un mensaje de **reconocimiento**. Estos agentes asumen la responsabilidad de ejecutar la tarea y se asumen como **contratistas** para la tarea que les ha sido asignada. Una vez que la tarea ha sido completada, el contratista envía un **reporte** al administrador.

Ofertas

Reconocimiento

Contratistas

Reporte

Esta negociación puede **simplificarse** en algunos casos, mejorando la eficiencia del protocolo: Cuando el administrador sabe cual es el agente apropiado para la ejecución de una tarea dada, puede efectuar una **contratación directa**, donde no hay anuncio público de la tarea. En ese caso, los contratistas tienen la posibilidad de **rechazar** la tarea propuesta.

Simplificaciones

Contratación directa

Rechazo

Observen que para aquellas tareas que simplemente solicitan información, un contrato no es lo más apropiado. Una secuencia de actos de habla *ask-tell* es más **adecuada** en esos casos. Un contrato debe usarse para distribuir el control en la ejecución de una tarea, no simplemente datos.

Contratos vs Actos de Habla

Además de describir los diversos mensajes enviados por los agentes, Smith describe los procedimientos que se ejecutan al **recibir mensajes**:

Recepción de Mensajes

- **Procesamiento de anuncio de tarea.** Al recibir un anuncio de tarea, un agente decide si es elegible para tal tarea. Esto lo hace revisando la especificación de elegibilidad que el mensaje incluye. Si es elegible, entonces almacena los detalles de la oferta y lanza una oferta.
- **Procesamiento de propuesta.** Los detalles de las ofertas de los posibles contratistas son almacenados por los administradores hasta que un plazo limite (*dead-line*) se cumple. Entonces el administrador premia al ganador.
- **Procesamiento de reconocimiento.** Los agentes que presentaron ofertas pero no ganaron, simplemente borran los detalles de la tarea en cuestión. El agente reconocido tratará de llevar a cabo la tarea, posiblemente generando nuevas sub-tareas.
- **Procesamiento de solicitudes y informes.** Estos mensajes son actos de habla *ask* y *tell*.

¿Como **decide** un agente si debe someter una oferta? Sandholm [167] propone la siguiente estrategia: Supongamos que al tiempo t, un potencial contratista i tiene asignado un conjunto de tareas τ_i^t y que tiene recursos totales r_i. Entonces i recibe un anuncio de tarea con la especificación ts. Denotaremos con $\tau(ts)$ el conjunto de tareas especificado por ts. Sea $c_i^t(\tau)$ el **costo** para el agente i de llevar a cabo las tareas especificadas en τ al tiempo t. Entonces, el **costo marginal** de llevar a cabo las tareas, denotado por $\mu_i(\tau(ts)|\tau_i^t)$,

Decidir oferta

Costo

Costo marginal

es:

$$\mu_i(\tau(ts)|\tau_i^t) \quad = \quad c_i(\tau(ts)) \cup \tau_i^t - c_i(\tau_i^t) \tag{10.1}$$

Si $\mu_i(\tau(ts)|\tau_i^t) = 0$, entonces el costo marginal (extra) de llevar a cabo $\tau(ts)$ es 0. Estas tareas pueden hacerse "gratis". Si $\mu_i(\tau(ts)|\tau_i^t) < r_i$, el costo marginal es menor a los recursos a disposición del agente i y sería **racional** proponer una oferta; en cualquier otro caso no lo es. Observen que si la especificación de la tarea ts incluye un posible pago $r(ts)$, entonces la decisión dependerá de si $\mu_i(\tau(ts)|\tau_i^t) < (r(ts) + r_i)$.

Racionalidad

10.2.2 Compartiendo resultados

Los resultados se comparten cooperativamente, por intercambio de información conforme la solución se va desarrollando. Lo normal es que estos resultados progresen de soluciones a pequeños problemas, a soluciones más abstractas, por refinamiento. Durfee, Lesser y Corkill [56] sugieren que los agentes pueden mejorar su desempeño al compartir resultados, de la siguientes maneras:

- **Confianza.** Soluciones derivadas de manera independiente, puede validarse por cruce, señalando posibles errores e incrementando la confianza de la solución global.
- **Compleción.** Los agentes pueden compartir su perspectiva local para lograr una mejor visión global.
- **Precisión.** Los agentes pueden compartir resultados para incrementar la precisión de sus soluciones globales.
- **Tiempo.** Aunque un agente pueda resolver sólo una tarea, al compartirse resultados parciales, podría terminar su tarea más rápido.

10.3 CNP EN JASON

Aunque Jason no provee una versión predefinida del CNP, es posible implementar este protocolo fácilmente, utilizando las facilidades propuestas por este lenguaje de programación orientado a agentes, en particular los actos de habla.

La red de contratos que implementaremos está compuesta por un agente `administrador`, tres agentes que proponen una `ofertaAleatoria`, un agente que `nuncaConstesta` y otro agente que `siempreRechaza`. La definición del SMA, `cnp.mas2j` se muestra en el Cuadro 10.1.

El agente administrador `admin`, se muestra en el Cuadro 10.2, es quien inicia la red de contratos, subastando la tarea *componer*(*computadora*) con identificador 1. Los participantes en la red se computan con la acción interna `.findall` y el anuncio de la tarea se les comunica con una acción interna `.send` de fuerza ilocutoria `tell`. El administrador incluye una regla en sus creencias para determinar si ha recibido todas las ofertas sobre una tarea anunciada, incluyendo los casos donde el contratista rechaza la tarea; para,

```
MAS cnp {

  infrastructure: Centralised

  agents:
    nuncaContesta;
    siempreRechaza;
    ofertaAleatoria #3;
    admin;

  aslSourcePath: "src/asl";
}
```

Cuadro 10.1: El Sistema MultiAgente del CNP.

si ese es el caso, tratar de asignar un contrato, con un **plazo de vencimiento** en cuatro segundos. El ganador es quien ofrece el costo mínimo para la tarea anunciada, en este caso, reparar la computadora. Al menos una oferta debe recibirse para poder computar un ganador. Una vez computado el ganador, el `administrador` anuncia el resultado al ganador y a los perdedores.

Vencimiento

Los agentes de tipo `ofertaAleatoria` comparten el código que se muestra en el Cuadro 10.3. Estos agentes tienen una regla para computar aleatoriamente el precio de sus ofertas, con valores entre 100 y 110. Al reconocer que agente `admin` es quien inicia el CNP, estos agentes se presentan ante él como participantes de la red. Estos agentes incluyen planes para responder a un anuncio de tareas `cfp`, y a los anuncios de que sus ofertan han sido aceptadas o rechazadas.

El agente que `siempreRechaza` se muestra en el Cuadro 10.4. Este agente se registra como participante ante el agente `admin` y, sin importar la tarea anunciada, siempre rechaza el `cfp`.

El agente que nunca contesta se muestra en el cuadro 10.5. Este agente se registra como participante, pero esto todo lo que hace. No responde a ningún anuncio de tareas.

La ejecución del SMA muestra al agente `admin` esperando a que los participantes en su red se registren y luego enviándoles en anuncio de la tarea `cfp` de interés, componer la computadora. A continuación despliega las ofertas recibidas y anuncia que la oferta ganadora es la de menor precio (el agente ganador puede variar pués los agentes generan sus precios aleatoriamente). En respuesta al anuncio de resultados, los agentes que perdieron lo reconocen y el ganador también.

```
[admin] Esperando a los participantes...
[admin] Enviando CFP a [ofertaAleatoria1, nuncaContesta
    ↪ , ofertaAleatoria2, siempreRechaza,
    ↪ ofertaAleatoria3]
[admin] Las ofertas son [oferta(103.77618324256807,
    ↪ ofertaAleatoria3),
oferta(108.985744452349, ofertaAleatoria2), oferta
    ↪ (106.33970211579945,ofertaAleatoria1)]
[admin] El ganador es ofertaAleatoria3 con
    ↪ 103.77618324256807
[ofertaAleatoria3] Mi oferta 103.77618324256807 ganó el
    ↪  CNP 1 para la tarea componer(computadora).
```

```
// Agente admin en el proyecto cnp

/* Initial beliefs and rules */

todasPropuestasRecibidas(CNPId) :-
    .count(intro(participante,_),NP) &  // No participantes
    .count(oferta(CNPId,_), NO) &       // No ofertas recibidas
    .count(rechazo(CNPId), NR) &        // No rechazos recibidos
    NP = NO + NR.

/* Initial goals */

!iniciarCNP(1,componer(computadora)).

/* Plans */

+!iniciarCNP(Id,Tarea) <-
    .print("Esperando a los participantes...");
    .wait(2000);
    +estadoCNP(Id,proponer);   // recordar estado CNP
    .findall(Nombre,intro(participante,Nombre),LP);
    .print("Enviando CFP a ",LP);
    .send(LP,tell,cfp(Id,Tarea));
     // el plazo del cfp es dentro de 4 segs.
     .at("now +4 seconds", { +!contrato(Id) }).

+oferta(CNPId,_)
   :  estadoCNP(CNPId,proponer) & todasPropuestasRecibidas(CNPId)
   <- !contrato(CNPId).

+rechazo(CNPId)
   :  estadoCNP(CNPId,proponer) & todasPropuestasRecibidas(CNPId)
   <- !contrato(CNPId).

@lc1[atomic]
+!contrato(CNPId)
   :  estadoCNP(CNPId,proponer)
   <- -+estadoCNP(CNPId,contratar);
      .findall(oferta(O,A),oferta(CNPId,O)[source(A)],L);
      .print("Las ofertas son ",L);
      L \== []; // constraint the plan execution to at least one offer
      .min(L,oferta(WOf,WAg)); // sort offers, the first is the best
      .print("El ganador es ",WAg," con ",WOf);
      !anunciarResultado(CNPId,L,WAg);
      -+estadoCNP(CNPId,fin).

+!contrato(_).

-!contrato(CNPId)
   <- .print("CNP ",CNPId," ha fallado.").

+!anunciarResultado(_,[],_).

+!anunciarResultado(CNPId,[oferta(_,WAg)|T],WAg)
   <- .send(WAg,tell,ofertaAceptada(CNPId));
      !anunciarResultado(CNPId,T,WAg).

+!anunciarResultado(CNPId,[oferta(_,LAg)|T],WAg)
   <- .send(LAg,tell,ofertaRechazada(CNPId));
      !anunciarResultado(CNPId,T,WAg).
```

Cuadro 10.2: El agente administrador del CNP.

```
// Agente ofertaAleatoria en proyecto cnp

// Obtener un precio para el producto como un valor aleatorio entre 100 y 110
precio(_,X) :- .random(R) & X = (10*R)+100.
rol(inicia,admin).

// Presentarse ante el agente que inicia en CNP
+rol(inicia,Ag)
   :  .my_name(Yo)
   <- .send(Ag,tell,intro(participante,Yo)).

// responder a un cfp
+cfp(CNPId,Tarea)[source(Ag)]
   :  rol(inicia,Ag) & precio(Tarea,Oferta)
   <- +oferta(CNPId,Tarea,Oferta); // recordar mi oferta
      .send(Ag,tell,oferta(CNPId,Oferta)).

+ofertaAceptada(CNPId)
   :  oferta(CNPId,Tarea,Oferta)
   <- .print("Mi oferta ",Oferta," ganó el CNP ",CNPId,
             " para la tarea ",Tarea,").
      // Hacer la tarea y reportar los resultados a admin

+ofertaRechazada(CNPId)
   <- .print("Perdí el CNP ", CNPId, ".");
      -oferta(CNPId,_,_).
```

Cuadro 10.3: Agente que genera ofertas aleatorias.

```
[ofertaAleatoria1] Perdí el CNP 1.
[ofertaAleatoria2] Perdí el CNP 1.
```

10.4 MANEJO DE INCONSISTENCIA

Formas de inconsistencia

Uno de los problemas que surgen al cooperar es la inconsistencia entre los diferentes agentes de un sistema. Los agentes pueden ser **inconsistentes** con respecto a sus creencias y a sus metas e intenciones. La inconsistencia entre las metas, normalmente se debe a que se asume que los agentes son autónomos, y por lo tanto no comparten objetivos comunes. La inconsistencia entre las creencias tiene fuentes diversas. Primero, el punto de vista de un agente es limitado, es de suponer que ningún agente tiene información completa sobre su ambiente. Luego, los sensores de los agentes suelen ser ruidosos. Finalmente, las fuentes de información de los agentes pueden ser poco confiables.

En un sistema de talla moderada, la inconsistencia es inevitable. La pregunta es ¿Cómo podemos enfrentarla? Durfee, Lesser y Corkill [56] sugieren posibles aproximaciones a este problema:

- **Evitar que esto suceda**, o al menos ignorar las inconsistencias. Este es el enfoque que adopta CNP. La compartición de tareas es siempre controlada por un agente administrador, quien tiene autoridad sobre el problema en cuestión.

```
// Agente siempreRechaza en proyecto cnp

rol(inicia,admin).

+rol(inicia,Ag)
   :  .my_name(Yo)
   <- .send(Ag,tell,intro(participante,Yo)).

+cfp(CNPId,_)[source(Ag)]
   :   rol(inicia,Ag)
   <- .send(Ag,tell,rechazar(CNPId)).
```

Cuadro 10.4: Agente que siempre rechaza los anuncios de tarea.

```
// Agente nuncaContesta in project cnp

// El agente que tiene el rol de iniciar el CNP
rol(inicia,admin).

// Enviar un mensaje a ese agente anunciándose como participante
+rol(inicia,Ag)
   :  .my_name(Yo)
   <- .send(Ag,tell,intro(participante,Yo)).
```

Cuadro 10.5: Agente que nunca contesta los anuncios de tarea.

- **Resolver las inconsistencias** mediante negociación, por ejemplo, usando subastas. Mientras que esto es al menos teóricamente deseable, los costos de comunicación y computación sugieren que raramente es realizable en la práctica.
- Construir sistemas que se **degradan con gracia** ante la presencia de inconsistencia. Lesser y Corkill [120] se refieren a estos sistemas como funcionalmente precisos y cooperativos (FA/C).

Los sistemas FA/C se caracterizan por:

- La resolución de problemas no está restringida estrictamente a una secuencia particular de eventos. Por el contrario, este proceso progresa de manera **oportunista** e **incremental**.
- Los agentes comunican **resultados intermedios de alto nivel**, en lugar de comunicar datos simples.
- La incertidumbre y la inconsistencia se resuelven implícitamente cuando las soluciones parciales son intercambiadas para su **comparación**. De esta forma, estos problemas se resuelven conforme el problema se va resolviendo, ni antes, ni después.
- La solución no se restringe a una sola ruta posible. Existen **diversas formas de llegar a una solución**, de manera que si una de ellas falla, existen alternativas para satisfacer el mismo fin. Esto hace que el sistema sea robusto ante fallas locales y cuellos de botella en el proceso de resolución de problemas.

10.5 LECTURAS Y EJERCICIOS SUGERIDOS

Una introducción accesible a la solución cooperativa y distribuida de problemas se encuentra en el capítulo 8 del libro de Wooldridge [192].