Práctica Tema 7: Agentes JADE

Para la realización de esta práctica me he basado en el comportamiento base del agente *BookBuyerAgent*, intentando mantener su comportamiento base y exclusivamente añadiendo funcionalidades adicionales al ya algoritmo presentado en el ejemplo.

Descripción de la estrategia

La estrategia de compra seguida se basa casi totalmente en la estrategia seguida con el *Book Trading* del ejemplo de JADE. A continuación se describe la estrategia seguida:

- Los vendedores envían ofertas de un libro con un precio establecido (esto se mantiene).
- El comprador quiere comprar un libro determinado (parámetro modificable) a un precio máximo establecido (parámetro modificable), recibiendo las ofertas de los diferentes vendedores para elegir la mejore oferta.
- El comprador cada minuto consulta el precio mínimo de venta del libro que quiere comprar a cada uno de los y memoriza dicho valor.
- Para que el comprador decida comprar un libro, necesita recibir al menos un número mínimo de buenas ofertas (parámetro modificable) que no superen el precio máximo establecido de compra (no tendría sentido considerar precios superiores al máximo permitido).
- De las buenas ofertas recibidas, se queda siempre con la mejor de ellas (como es lógico, cuando compras algo siempre quieres el menor de los precios).
- Antes de decidir comprar o no, es necesario comparar el mejor precio de las ofertas recibidas y comprobar si es menor o igual al precio histórico de venta más bajo del libro (obtenido preguntando a cada vendedor el menor precio de venta del libro).
    - Si es menor o igual, obviamente se compra (normalmente se tiende siempre a vender al mismo precio excepto ofertas puntuales).
    - En caso de que la mejor de las ofertas recibidas supere ese precio mínimo de mercado, solo lo comprará si la diferencia de precio es menor o igual al 35%[1]. Otro escenario que se podría dar en el mundo real seria esperar hasta encontrar un precio menor, o al menos esperar una serie de ofertas más (por ejemplo, esperar a recibir otras 3 ofertas desde este punto), y ver si la mejor oferta cumple o no.

De forma resumida, el vendedor realiza ofertas de libros por un precio marcado y el comprador está interesado en un libro determinado marcando un precio máximo de compra y necesitando un

1 Otra alternativa a este caso de uso podría ser decidir comprar de forma aleatoria o establecer una interfaz gráfica para preguntar al usuario; pero se quería disponer de un caso más real y que tomara la decisión de igual forma que haría un comprador normal.

mínimo de ofertas recibidas por parte de los vendedores para poder considerar el hecho de comprar o no un libro. A mayores, los compradores están interesados en saber el precio mínimo de venta de ese libro por cada uno de los vendedores y poder establecer un precio mínimo histórico de venta para poder tomar la decisión final de si comprar o no el libro con el menor precio de los recibidos:

- Cuando se realiza una compra es importante marcar un precio máximo de compra, o es lo que suele pasar en la vida real cuando vas a comprar un producto o te presentas a una subasta. El hecho de marcar el precio máximo te permite seguir una estrategia más restrictiva a la hora de elegir proveedor, obteniendo normalmente precios de venta más bajos.
- Es importante disponer de la información, si es posible, relativa a los históricos de venta para poder saber los precios los productos de cada proveedor. Esto permite al comprador saber el precio mínimo alcanzando en el mercado.
- Normalmente cuando vas a comprar algo al supermercado no te sueles marcar un tiempo máximo de compra por el modo de funcionamiento de esta acción. En cambio, cuando se participa en un entorno basado en múltiples ofertas de diferentes proveedores, suele ser Comporrecomendable marcar un tiempo límite de tiempo (bien porque el límite superior de gasto es muy bajo para las ofertas recibidas o bien porque no se emiten ofertas).

Los puntos anteriores muestran de forma muy abstracta el comportamiento de los compradores y vendedores. En el siguiente apartado se entrará más en detalle a nivel de implementación.

Descripción y motivación del comportamiento y otros cambios introducidos

Parte del comportamiento de los 2 tipos de agentes y de la motivación fueron presentados en el apartado Descripción de la estrategia. A continuación se listarán los nuevos comportamientos añadidos y los diferentes cambios necesarios para poder implementar la estrategia descrita en el apartado anterior.

Nuevos comportamientos

Se han añadido un par de nuevos comportamientos (en algún caso utilizando el mismo tipo de comportamiento) tanto en el *BookBuyerAgent* y el *BookSellerAgent*:

BookBuyerAgent

- Se añade un comportamiento de tipo *CyclicBehaviour* para poder obtener la información acerca del menor precio de venta del libro especificado de cada uno de los proveedores (en caso de que no hayan vendido ese libro, no se recibe información al respecto).

- Se añade un comportamiento de tipo *WakerBehaviour* en caso de especificar un tiempo máximo de compra para cerrar el periodo de compra y dar por finalizado el Agente. En caso de haber finalizado el tiempo máximo establecido, se finaliza el agente excepto si se encuentra en medio de una negociación con un Vendedor. Es este caso no se realiza ninguna acción (se activa el modo *tiempoCompraFinalizado*) y se espera a finalizar la negociación con el comprador:
    - en caso de ser aceptado el mensaje de negociación y haber comprado el libro, el Agente finaliza de forma normal (este comportamiento era el definido en el ejemplo) y,
    - en caso de haber sido rechazada la negociación, se finaliza el Agente.

BookSellerAgent

- Se añade un comportamiento de tipo *CyclicBehaviour* para gestionar las peticiones de información de precios de los libros por parte de los compradores. La acción desencadenada es simplemente coger el precio mínimo de venta del libro especificado en el mensaje y notificar al vendedor mediante un mensaje de tipo *INFORM*. En caso de no haber vendido nunca ese libro, no se le notificada.

Para el correcto funcionamiento de la mayoría de estos comportamientos fue necesario realizar ciertas modificaciones en los comportamientos ya existentes en los dos Agentes. Estos cambios se describirán en el aparatado Otros cambios introducidos, pero en los archivos *BookBuyerAgent.java* y *BookSellerAgent.java* se puede encontrar el código fuente con toda la información y cambios aplicados.

Otros cambios introducidos

A continuación se describen los cambios realizados en cada una de las clases.

BookBuyerAgent

1. Se modifican los parámetros necesarios para inicializar a los compradores.
    - **Número** mínimo **de ofertas** recibidas por los vendedores: indica el número necesario de ofertas por parte de los compradores para poder tomar la decisión de si comprar o no el libro y a qué vendedor comprárselo.
    - **Precio máximo** del libro que se quiere pagar: indica el precio máximo para considerar las ofertas recibidas por los vendedores.
    - **Tiempo máximo** para realizar la compra (parámetro opcional): indica el tiempo máximo para poder recibir ofertas y empezar las negociaciones de compra, teniendo cierta utilidad por si durante mucho tiempo no se reciben ofertas. En caso de que no se especifique este tiempo, el agente podrá recibir ofertas de forma indefinida.

2. Se establecen las variables necesarias para almacenar los nuevos parámetros de entrada así como dos variables que nos indican modos o estados del Agent:
    - Una variable (*comprandoLibro*) para saber si el Agente está negociando con un vendedor la compra de un libro (necesario para el comportamiento de tipo *WakerBehaviour*).
    - Una variable para saber si se ha agotado el tiempo de compra (pensado por si fuera necesario modificar el comportamiento *RequestPerformer*).
3. Se añade el nuevo comportamiento de tipo *CyclicBehaviour* para gestionar los mensajes de información de precio enviado por los vendedores.
4. Se añade el nuevo comportamiento de tipo *WakerBehaviour* para gestionar la finalización de la compra (solo en caso de haber especificado el tiempo al iniciar el Agente).
5. Se modifica primer comportamiento lanzado por el Agente (el de tipo *TickerBehaviour*) para, adicionalmente a la búsqueda de vendedores, emitir un mensaje de tipo REQUEST para obtener la información de precio mínimo de venta del libro.
6. Se realizan ciertas modificaciones en el comportamiento *RequestPerformer* (aunque base mi enfoque de la práctica a modificarlo lo menos posible)
    - Al recibir las ofertas de libros por parte de los vendedores se realizan varios cambios:
        - solo se consideran ofertas las que tienen un precio menor al máximo establecido,
        - se pasa al siguiente paso (proceso de negociación para comprar un libro) cuando recibamos el número de ofertas marcado y el precio mínimo de las ofertas es menor que el mínimo de venta (recibido por los vendedores) o la diferencia de precio sea menor o igual al 35%.
    - Cuando se inicia la negociación se cambia la variable *comprandoLibro.*
    - Si la negociación con el vendedor falla, en caso de que el tiempo de compra haya finalizado se finaliza el Agente (sin haber comprado ningún libro).

BookSellerAgent

1. Se define una nueva Hashtable (*bestPrices*) para gestionar los precios mínimo de venta por libro vendido por el vendedor.
2. Se añade el nuevo comportamiento de tipo *CyclicBehaviour* para gestionar los mensajes de petición.
3. Se modifica el comportamiento *PurchaseOrdersServer* para permitir actualizar el precio mínimo de venta del libro vendido en caso de mejorar el precio mínimo histórico.

Alternativas en los nuevos comportamientos/estrategia

Añado este nuevo apartado como punto de reflexión para posibles mejoras a realizar sobre la solución de la práctica propuesta; principalmente debido a falta de tiempo por motivos de laborales a la hora de compaginar trabajo y universidad.

- En el Agente *BookBuyerAgent*, el nuevo comportamiento de tipo *WakerBehaviour*, actualmente simplemente para comprobar si está negociando una compra y terminar la ejecución del Agente en caso de que la negociación falle. Pero gracias a este modo se podría implementar/mejorar la estrategia planteada anteriormente:
  - En caso de no encontrarse negociando la compra de un libro con un vendedor, en lugar de finalizar la ejecución del agente, se podría coger el mejor precio obtenido y ejecutar una última acción de negociación para la compra de éste. En principio este cambio no debería de suponer demasiado trabajo.
- En el comportamiento *RequestPerformer* del Agente *BookBuyerAgente*, cuando se han recibido todas las ofertas y y el mejor precio obtenido sigue siendo superior al mínimo histórico, en lugar de decidir comprar o no si entra dentro de un rango, se podría intentar recibir una serie de ofertas más. Si todavía así no se mejora el tiempo, realizar la misma comprobación.