UIMP
Universidad Internacional Menéndez Pelayo

AEPIA
ASOCIACIÓN ESPAÑOLA PARA LA INTELIGENCIA ARTIFICIAL

Negociación: Análisis del Algoritmo NB^3

Máster Universitario en Investigación en Inteligencia Artificial

Sistemas Multiagente

Javier Vela Tambo
100012966@alumnos.uimp.es

25 de enero de 2025

Introducción

Este documento presenta el resultado de los ejercicios de la práctica de "Negociación" de la asignatura. Para el desarrollo de este trabajo se ha seguido el manual[1], primero estudiando el funcionamiento del algoritmo NB^3[2] y posteriormente realizando las modificaciones propuestas.

Dado que los enunciados no especifican detalladamente las expectativas para cada ejercicio más allá de los cambios en el código, se ha decidido enfocar el análisis en la realización de experimentos que permitan observar cómo afectan las modificaciones al algoritmo.

En primer lugar, se analizarán las ejecuciones independientes del agente original (Sección 1), prestando especial atención a las propuestas realizadas, si estas han sido aceptadas y confirmadas. Además, se evaluará la ganancia obtenida por los agentes en dichas ejecuciones. Posteriormente, se examinará cómo los distintos cambios introducidos influyen en estos aspectos (Secciones 2 y 3).

La práctica se ha realizado utilizando el código fuente base, proporcionado por los profesores de la asignatura, y se han realizado las modificaciones propuestas. El código fuente modificado se encuentra en el repositorio de GitHub[1].

[1] https://github.com/javiervela/nb3_negotiation_agent

1. Análisis de Propuestas y Ganancia de Utilidad de los Agentes

Para analizar las mejoras propuestas en la práctica, se ha utilizado el agente original como referencia para realizar comparaciones. Se han ejecutado 31 simulaciones de 30 segundos cada una, con 5 agentes en cada simulación.

Se ha desarrollado un script en Python que automatiza la lectura de los resultados de las simulaciones y calcula el número total de propuestas realizadas, aceptadas y confirmadas, la ganancia de utilidad obtenida por cada agente y el tiempo de simulación en el que se realiza la primera propuesta. Además, se incluye el número de simulaciones en las que se ha obtenido al menos una propuesta realizada, aceptada y confirmada, y el porcentaje sobre el total de simulaciones con al menos una propuesta.

Para obtener esta información, se han completado las funciones de registro de los resultados en el fichero `server.log` con la ganancia de los agentes y el tiempo transcurrido en la simulación.

Las Tablas 1 y 2 muestran los resultados obtenidos en las simulaciones del agente original. Estos resultados muestran que, aunque los agentes realizan un número considerable de propuestas, la tasa de aceptación y confirmación es relativamente baja. Además, la utilidad obtenida varía significativamente entre las simulaciones, debido a que en la mayoría de ellas los agentes no consiguen realizar propuestas aceptadas.

Métrica	**No. Ejecuciones**	**%**
Propuestas Realizadas ≥ 1	22	-
Propuestas Aceptadas ≥ 1	19	86.36 %
Propuestas Confirmadas ≥ 1	14	63.64 %

Cuadro 1: Resumen de las ejecuciones del agente original

Métrica	**Media**	**Mediana**	**Máx**	**Mín**	**SD**
Propuestas Realizadas	32.29	14	147	0	37.10
Propuestas Aceptadas	2.39	1	15	0	3.75
Propuestas Confirmadas	0.45	0	1	0	0.51
Tiempo Primera Propuesta	14	15	28	3	8.24
Ganancia de Utilidad	1.72	0	53	0	6.86

Cuadro 2: Resultados de las simulaciones del agente original

2. Mejora del Cálculo de la Cota Superior

Para implementar las mejoras propuestas se ha duplicado el módulo `agent` y se ha renombrado a `improvedAgent`. En esta copia del agente original se ha modificado la función `calculateUpperBound` para que no se realicen propuestas en las que el agente ya ha aparecido como proveedor en la misma rama de la negociación. El Listado 1 muestra la implementación de esta función.

Las Tablas 3 y 4 muestran los resultados obtenidos en las simulaciones del agente mejorado. Estos resultados indican que, aunque el número de ejecuciones con al menos una propuesta realizada ha disminuido, el porcentaje de ejecuciones donde se han aceptado y confirmado estas propuestas ha aumentado. Esto sugiere que la modificación realizada ha mejorado la eficacia del agente a la hora de realizar propuestas en la negociación.

El cambio más significativo se observa en el tiempo de simulación en el que se realiza la primera propuesta, que ha aumentado considerablemente. Esto es posible que se deba a que el agente realiza menos propuestas, pero estas son más efectivas. El resto de los resultados no sugieren ninguna mejora significativa frente al agente original.

Métrica	**No. Ejecuciones**	**%**
Propuestas Realizadas ≥ 1	16	-
Propuestas Aceptadas ≥ 1	15	93.75 %
Propuestas Confirmadas ≥ 1	13	81.25 %

Cuadro 3: Resumen de las ejecuciones del agente mejorado

Métrica	**Media**	**Mediana**	**Máx**	**Mín**	**SD**
Propuestas Realizadas	38.71	6	188	0	51.52
Propuestas Aceptadas	1.97	0	15	0	3.48
Propuestas Confirmadas	0.45	0	2	0	0.57
Tiempo Primera Propuesta	18.19	19	28	8	5.62
Ganancia de Utilidad	1.51	0	34	0	5.56

Cuadro 4: Resultados de las simulaciones del agente mejorado

3. Experimentación con Grados de Concesión

Con el objetivo de observar las diferencias en el comportamiento de los agentes en función del grado de concesión, se han realizado simulaciones con distintas estrategias de negociación caracterizadas por estos valores. Se han ejecutado 10 simulaciones de 30 segundos cada una, con 5 agentes en cada simulación, por cada estrategia de valores de concesión.

Para realizar estas simulaciones se ha modificado la clase `ImprovedNegotiator` para que acepte un parámetro para cada valor de concesión α_1 y α_2. Además, se ha modificado la clase `RunMarketAndFiveAgents`, que ejecuta las simulaciones, para definir 4 estrategias negociación:

- **Greedy** (Codicioso): $\alpha_1 = -4$, $\alpha_2 = -4$. Solo propone planes muy egoístas.
- **Lazy** (Perezoso): $\alpha_1 = 4$, $\alpha_2 = -4$. Propone planes egoístas, pero cede si no encuentra mejores.
- **Picky** (Exigente): $\alpha_1 = -4$, $\alpha_2 = 4$. Busca planes que sean tanto egoístas como altruistas.
- **Desperate** (Desesperado): $\alpha_1 = 4$, $\alpha_2 = 4$. Cede rápidamente, incluso con baja utilidad.

Los resultados obtenidos en las simulaciones con diferentes estrategias de negociación se muestran en las Tablas 5 y 6. Estos resultados indican que la estrategia *Picky* es la más efectiva, ya que ha obtenido el mayor número de propuestas aceptadas y confirmadas. Por otro lado, la estrategia *Greedy* y *Lazy* han sido la menoss efectivas, ya que han obtenido el menor número de propuestas aceptadas y confirmadas.

En cuanto a las métricas de tiempo y ganancia de utilidad, la estrategia *Greedy* ha sido la más rápida en realizar la primera propuesta, mientras que la estrategia *Desperate* ha obtenido la mayor ganancia de utilidad. Estos resultados sugieren que la estrategia *Desperate* es la más efectiva en términos de ganancia de utilidad, pero también la más arriesgada, ya que cede rápidamente en la negociación.

Métrica	**Greedy**	**Lazy**	**Picky**	**Desperate**
Propuestas Realizada ≥ 1	5	5	7	4
Propuestas Aceptada ≥ 1	1 (10.00 %)	1 (10.00 %)	6 (85.71 %)	4 (100.00 %)
Propuestas Confirmada ≥ 1	0 (0.00 %)	0 (0.00 %)	6 (85.71 %)	4 (100.00 %)

Cuadro 5: Resumen de las ejecuciones con diferentes estrategias de negociación

Métrica (Media)	Greedy	Lazy	Picky	Desperate
Propuestas Realizadas	4.00	6.60	54.10	33.60
Propuestas Aceptadas	0.10	0.10	3.90	3.80
Propuestas Confirmadas	0.00	0.00	0.60	0.60
Tiempo Primera Propuesta	6.60	24.00	22.86	23.25
Ganancia de Utilidad	0.00	0.00	2.42	2.60

Cuadro 6: Resultados de las simulaciones con diferentes estrategias de negociación

Para comparar las estrategias de los agentes, se realizaron 20 simulaciones con 5 agentes, cada uno con una estrategia diferente. La Tabla 7 muestra la ganancia de utilidad media por agente. Los resultados confirman que la estrategia *Desperate* es la más efectiva en términos de ganancia de utilidad.

Agente	Ganancia de Utilidad (Media)
Alice (Greedy)	0.00
Bob (Lazy)	0.86
Charles (Picky)	0.67
David (Desperate)	1.90

Cuadro 7: Valores de utilidad ganada media para los agentes con diferentes estrategias

Conclusiones

En este documento se ha analizado las modificaciones al algoritmo NB^3 para mejorar la negociación de los agentes. Los resultados del Ejercicio 2 indican que la modificación ha incrementado la eficacia del agente en propuestas aceptadas y confirmadas.

En el Ejercicio 3, la estrategia *Desperate* resultó ser la más efectiva en términos de ganancia de utilidad, aunque es la más arriesgada. La estrategia *Picky* fue la más efectiva en propuestas aceptadas y confirmadas.

Referencias

[1] Dave De Jonge. Implementing an nb3 agent. Technical report, December 2015.

[2] Dave De Jonge and Carles Sierra. Nb 3: a multilateral negotiation algorithm for large, non-linear agreement spaces with limited time. *Autonomous Agents and Multi-Agent Systems*, 29(5):896–942, 2015.

```java
@Override
public float calculateUpperBound(int agentID, List<NB3Node
    > branch, NB3WorldState ws) {

    CommodityAssets maximalAssets = ((CmWorldState) ws).
       commodityAssets.copy();

    for (int otherAgentID = 0; otherAgentID <
       maximalAssets.NUM_AGENTS; otherAgentID++) {

        if (otherAgentID == agentID) {
            continue;
        }

        for (int commodity = 0; commodity < maximalAssets.
           NUM_COMMODITIES; commodity++) {

            if (!hasSupplied(agentID, commodity, branch)
               && !hasConsumed(otherAgentID, commodity,
               branch)) { // CHANGED

                int quantity = maximalAssets.getAssets(
                   agentID, commodity)
                        + maximalAssets.getAssets(
                           otherAgentID, commodity);

                maximalAssets.setAssets(agentID, commodity
                   , quantity);
            }
        }
    }

    int val = this.theAgent.preferenceProfile.
       calculateValue(agentID, maximalAssets);
    return val;
}
```

Listing 1: Método para calcular la cota superior