Aproximación a los problemas Bidding y Procurement en TAC con estrategias Straightfoward Bidding

Juan Camilo Corena y William A. Romero R.
Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes

***Resumen*— Este documento presenta una implementación en JAVA de un simulador de mercado basado en las especificaciones de *Trading Agent Competition* (TAC). El simulador trabaja sobre los problemas de *Bidding* (decidir ofertas) y *Procurement* (decidir compras) en donde los agentes utilizan estrategias *Straightfoward Bidding*. Con este programa se quiere ilustrar el contexto de una modalidad competitiva a nivel académico como lo es TAC.**

***Palabras clave*— TAC, Diseño e implementación de subastas, Modelos de oferta, Aprendizaje basado en juegos.**

***Abstract*— This document presents a JAVA implementation of a market simulator based on the *Trading Agent Competition* (TAC) specifications. The simulator works on *Bidding* and *Procurement* problems where agents uses *Sraightfoward Bidding* strategies. With this program we want to illustrate the background from an academic competitive modality such as TAC.**

***Keywords*— TAC, Mechanism design and implementation, Sraightfoward Bidding, Game-based learning.**

I. Introducción

The Trading Agent Competition (TAC) es una competencia a nivel internacional que congrega grupos de investigación alrededor del problema de simular un mercado y desarrollar agentes que sean capaces de actuar en dicho espacio y en general alcanzar objetivos económicos y comerciales como maximizar utilidades, satisfacer peticiones de clientes, administrar una línea de producción, entre otros. "TAC provee un foro para evaluar la implementación (programación) de técnicas comerciales en un mercado, compitiendo con agentes de otros grupos de desarrollo"[1].

El juego consiste en la simulación de un mercado en donde un conjunto de ensambladores(agentes) compiten por las órdenes que varios clientes realizan para la adquisición de computadores. Cada día los clientes hacen peticiones de equipos y de las ofertas de los distintos ensambladores. En este mercado, al igual que en la realidad comercial, los ensambladores deben negociar con distintos proveedores para adquiririr los componentes necesarios para cumplir las ordenes de sus clientes. Los componentes necesarios para armar el producto (computador) son: procesador, tarjeta madre, memoria y discos; cada uno de estos componentes se consigue en distintas presentaciones (marcas), las cuales sirven para satisfacer los distintos tipos de configuración requeridos por los clientes. El objetivo de la competencia es el desarrollo y ejecución de programas que se enfrentan entre si en un mercado virtual.

La solución de este tipo de problemas permite la integración de distintas disciplinas (Matemáticas, Ciencias de la computación y Economía) con el fin de desarrollar un sistema capaz de responder a las exigencias de un mercado, validar modelos teóricos y simular comportamientos inferidos de la realidad. Con lo anterior, resulta interesante el contexto como plataforma de aprendizaje del diseño e implementación de mecanismos de asignación (Computer mechanism design). Motivo por el cual este trabajo propone, con base en las especificaciones de TAC, un juego sencillo que permita el estudio de elementos teóricos alrededor de mecanismos de asignación.

En este documento se presentan los resultados de la implementación de un juego tipo TAC. Primero, se describe y formaliza el juego y sus reglas; seguido, se expone el modelo teórico y los algoritmos implementados; finalmente se muestran y analizan los resultados de ejecución del juego.

II. Objetivos

El desarrollo de este trabajo tiene como objetivos:

1. Plantear un modelo de juego sencillo que permita estudiar las problemáticas implícitas en este tipo de planteamientos.
2. Diseñar una solución y desarrollar un programa que ilustre la interacción entre los participantes y los resultados obtenidos de esta.
3. Incentivar la conformación de grupos de interés, a nivel local, que impulsen el desarrollo de proyectos docentes y de investigación alrededor de estos temas.

III. Modelo de juego

En TAC se resuelven los problemas de:

- *Bidding*: Decidir ofertas a clientes.
- *Scheduling*: Decidir asignación de la producción.
- *Procurement*: Decidir compra de componentes.
- *Delivery*: Decidir envió de productos terminados a clientes.
- *Forecasting*: Predicción de precios, demanda, etc.

El juego propuesto sólo se enfoca de manera sencilla en los problemas de *Bidding* y *Procurement*.

Cada ensamblador (agente) tiene un listado de computadores por ensamblar y cada uno de estos computadores tiene una especificación técnica de acuerdo a 4 componentes:

CPU, Motherboard, Memory y *Disk drives*. Cada agente debe cumplir las órdenes de ensamblaje por día para un intevalo de tiempo deteminado(1 semana, 1 mes, etc.) al menor costo posible.

En el modelo utilizado se realizan las siguientes suposiciones:

Proveedor:

- Existe un único proveedor (Subastador).
- Este proveedor tiene una cantidad inicial finita de componentes que se incrementa diariamente de acuerdo con una función de producción.
- Cada componente tiene un valor básico (precio de reserva) conocido por todos los ensambladores.

Ensambladores:

- Cada ensamblador tiene cantidades infinitas de dinero. En TAC existe una entidad denominada banco la cual realiza prestamos, asignación de multas etc. Para eliminar la presencia de este actor se asume capacidad adquisitiva estable.
- Cada ensamblador tiene un listado de computadores por ensamblar diario. Si no puede cumplir el número de computadores por día, estos se sumarán a la orden del día siguiente.

III-A. Definiciones básicas

Formalizando el juego se tiene:

- Existe un conjunto I de ensambladores, i representa un ensamblador de I
- $j \in J$: denota el tipo de componente, $j \in \{0, 1, 2, 3\}$
- $d \in D$: es un día del conjunto de días que determinan el intervalo de tiempo de interacción o de negociación.
- C_j: representa la capacidad de producción que tiene el proveedor del componente j
- Cada ensamblador tiene una reputación de compra r_i
- Una oferta se define como un conjunto de tuplas de cantidad y precio:

$$\begin{aligned} Oferta_d = \quad & \{(wCPU, valor_w), \\ & (xMotherboard, valor_x), \\ & (yMemory, valor_y), \\ & (zDiskdrive, valor_z) \end{aligned}$$

III-B. Dinámica del juego

La dinámica del juego inicia con una $oferta$ inicial que hace el ensamblador i al proveedor para el día d, el proveedor, responde con una contraoferta $Oferta_d^*$, Esta contraoferta es determinada por la cantidad actual del componente, el precio y la reputación del ensamblador; la reputación del ensamblador es la relación entre los días transcurridos y los días en que ha comprado. A manera de ejemplo, suponga que en 4 días un ensamblador no ha realizado compra alguna, esto implica una reputación no muy buena y en consecuencia el valor a pagar es mayor al que este pagaría si su reputación fuera mejor.

Con base en la contraoferta, el ensamblador, decide si compra o no. Cada día se realizan 5 subastas, que se denominan rondas, esto con el ánimo de simular una situación de negociación, es decir que por día se realizan 5 rondas.

Terminado el día, se deducen los computadores que pueden ser ensamblados para la orden del día d y los que no se ensamblaron se le adicionan al día $d+1$. Para el siguiente día, el proveedor aumenta la cantidad de componentes con relación a la cantidad en inventario y la capacidad de producción para cada componente (C_j). Entonces la cantidad de componentes del día $d + 1$ = cantidad no vendida del día d + producción para el día $d + 1$.

III-C. Comportamiento de los participantes

Básicamente, los comportamientos están definidos de acuerdo los objetivos del subastador y los ensambladores.

Proveedor:

- Maximizar utilidad, la utilidad del proveedor es una funcion de las ofertas, el inventario y la reputación de los ensambladores.
- Capacidad de producción:

$$\begin{aligned} Produccion_{d+1,j} = \quad & max(1, C_{d,j} + \\ & random(-0{,}05, 0{,}05) C_{nom} + \\ & 0{,}01(C_{d,j} - C_{nom})) \end{aligned}$$

Donde C_{nom} es la capacidad nominal de producción utilizada por el proveedor para propositos de planeación ([2]). Este valor se tomo como una constante(0.35) de acuerdo a lo propuesto en [2].

Ensambladores:

- Minimizar el costo de ensamblaje.
- Maximizar el número de computadores ensamblados por día.
- Estrategia con base en *Straightforward bidding* [7].
- La valoración del ensamblador por un componente está dada por:

$$v_j = p_j + h$$

  Para p_j el precio de reserva del componente y h un número aleatorio entre 0 y $2 \times p_j$. Esto quiere decir que un ensamblador máximo pagaría el doble del precio del componente.
- Es averso al riesgo, lo que implica que no va a ofrecer más de la valoración que tenga hacia un componente.
- No compra más de lo necesario.
- Del costo total de ensamble de un computador, el porcentaje de costo de cada componente esta dado de la siguiente manera: CPU 39 %, Motherboard 33 %, Memory 12 % y Disk drive 16 %. Esta relación se utiliza para el cálculo oferta.

III-D. Mecanismo

El modelo descrito hasta el momento corresponde a un problema de asignación combinatorio (The combinatorial allocation problem, CAP). El mecanismo se define a partir de:

- Un conjunto G de objetos a ser asignados a I agentes.
- Sea K el conjunto de escogencias tal que: $K = (S_1, ..., S_I) : S_i \subseteq G$ y S_i son los objetos asociados a un agente i.
- De acuerdo a las preferencias de cada agente, la utilidad se define como:

$$u_i(S, v_j, p_j) = v_j - p_j$$

- Se selecciona un asignación que:

$$S = max \sum_{S=(S_1,,S_I)} v_i(S_i)$$

Queda fuera del alcance de este documento presentar detalladamente las propiedades del mecanismo, para una mayor explicación puede remitirse a [6].

IV. Implementación

Para explicar la implementación es necesario hacer las siguientes definiciones:

- S: Conjunto de ofertas
- C: Conjunto de participantes
- $p(x_i)$: Pago hecho por la oferta del agente i
- $p(x_{ij})$: Pago hecho por el componente j de la oferta del agente i
- B: Vector de precios de reserva para cada componente
- R: Vector de reputaciones de los participantes
- L: Vector de inventario para cada tipo de componente
- V: Vector de valoraciones por tipo de componente
- N_{ij}: Necesidad de compra del participante i por el componente j
- $|S| = |C| = |R|$
- $p(x_i) \geq B(x_i)$

IV-A. Algoritmo del subastador para una ronda

Cada oferta debe hacerse en forma explicita por cada unidad de componente que se desea comprar. El algoritmo se compone de 3 pasos:

1. Recibir las ofertas de los clientes:

   $X := \emptyset$
   $X := X \cup S_i, \forall i \in C$

2. Seleccionar el subconjunto O de ofertas con mayor utilidad:

   $O := X \subseteq S \mid$
   $max \sum_{(i \in X)} p(x_i) \wedge \sum_{(i \in X)} x_{ij} \leq L_j$
   $\forall j \in J$

3. Generar contra ofertas para aquellos cuya oferta no está en O, las nuevas ofertas tienen como precio base el valor máximo pagado por cada uno de los componentes en el conjunto O, las ofertas se realizan de manera descendente respecto a la reputación que se tiene de ese cliente. El número de objetos de la contra oferta esta dado por la relación entre número de componentes solicitados y el puesto ocupado en la escala de reputacion: $\frac{num\_componentes}{escalafón}$.

   Do $j := \forall x_i \notin O, R_j \geq R_{j+1}, \forall j$
   $|x_i| = \left\lfloor \frac{|x_i|}{j} \right\rfloor$
   $p(x_{ij}) := maxp(o_{ij}) \mid o_i \in O$
   $O := O \cup x_i$
   od

4. Preguntar a los participantes si aceptan la oferta y actualizar precios base:

   if $c_i.decidir(o_i)$, $\forall o_i \in O$ then
   $B_{c_i} := p(c_i)$
   fi

IV-B. Algoritmo de oferta del ensamblador

Se seleccionan los componentes de menor costo que satisfacen las necesidades del ensamblador, en caso de que la sumatoria de estos componentes sea mayor que su valoración, no se realiza oferta. Para decidir cuanto dinero se va a ofertar por los componentes mas económicos se usa la estrategia *sunk awareness*, representada mediante el parámetro k.

Hacer *Straightforward Bidding* con *Sunk awareness*:

$x := \emptyset$
Do $\forall i, j \in N$
$k := random(1, 20)$
min $p(B_l) | B_l$ Sea de tipo$j, l \in B$

$$x := \begin{cases} x \cup B_l + \epsilon & \text{Si no tiene lo que quiere} \\ \\ x\frac{1}{k} \times N_l & \text{Si tiene lo que quiere} \end{cases}$$

od

IV-C. Algoritmo de aceptación de la contra oferta

Decidir si se acepta o no la contra oferta del subastador, esta se acepta si la sumatoria del precio de los productos no excede la sumatoria de las valoraciones de cada uno de los objetos en la oferta:

if $\sum_{j \in X_i} V(x_{ij} \geq \sum_{j \in X_i} P(x_{ij})$ then
return **true**
fi
return **false**

IV-D. Complejidad

La complejidad del algoritmo utilizado por el proveedor es exponencial con respecto al número de ofertas, para lo cual se tiene: **O**($2^n \times m$) para n número de ofertas y m número de

Fig. 1. Interfaz simulador

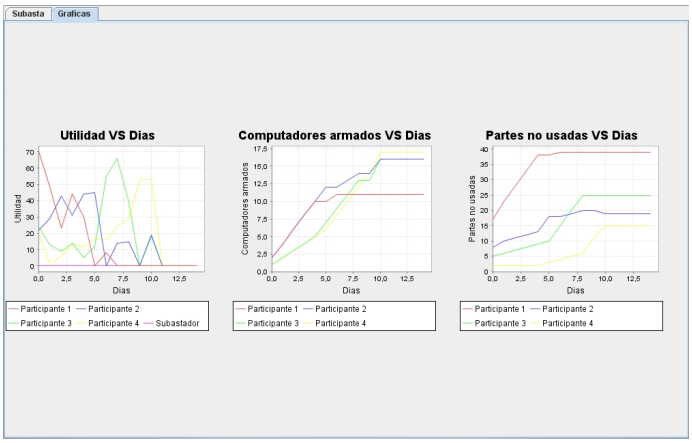

Fig. 2. Análisis de resultados simulador

productos del subastador. Por parte del ensamblador la relación es lineal con relación al número de elementos subastados m: **O**(m).

V. Resultados

El programa está implementado en JAVA. Se hace uso de la librería *JFreeChart* [8] para la incorporación de gráficos de análisis de los resultados de la simulación. Como se puede observar en la figura 1, la interfaz presenta el estado del subastador a nivel de inventario, producción del día y total vendido hasta el momento. De igual manera se visualiza la información de los ensambladores y la relación de componentes faltantes para el cumplimiento de las ordenes. En los ensambladores se puede seleccionar la estrategia para analizar el comportamiento de la estrategia *Straightforward Bidding* con *Sunk awareness*. Las cantidades de inventario inicial del proveedor y las ordenes de los ensambladores son generadas aleatoriamente al iniciar el programa.

El programa al finalizar la simulación presenta 3 tipos de gráficas(ver figura 2): Utilidad vs Días, Comptadores ensamblados vs Días y Partes no usadas vs Días. Con esta información se quiere ilustrar el comportamiento del mercado simulado.

La gráfica Utilidad vs Días no es acumulada, es día a día; la ganancia del subastador se obtiene con respecto al precio base que que él determina y la utilidad es con respecto a la valoración. Tambien es importante resaltar que no se toman en cuenta los componentes que no se pudieron utilizar para el ensamblaje de un computador y que representan costo muerto. Las demás graficas son acumuladas.

VI. Conclusiones y trabajo futuro

En el modelo planteado, se realizan 5 rondas con el fin de obligar a los ensambladores a participar (tienen que seguir ofertando en las siguientes rondas para poder adquirir el conjunto mínimo de componentes para cumplir con la orden del día). Como los agentes siguen una estrategia *straightforward*, estos sólo ofertan cuando pueden maximizar su *surplus*. Así, los agentes descartan conjuntos de componentes útiles y se retiran de la subasta prematuramente [7]. Para que los agentes puedan adquirir estos conjuntos se implementó la estrategia *sunk aware*; cuando el parámetro $k = 1$ se tiene una estrategia de oferta *straightforward* simple.

El comportamiento del mercado se ilustra en las gráficas, de estas se puede inferir:

- Con este modelo el incremento del precio de venta del proveedor con respecto a los precios de reserva es casi nulo.
- Para un número significativo de simulaciones se observa que los ensambladores ocacionalmente logran suplir sus demandas.
- Las curvas de componentes no utilizdos son siempre crecientes, esto corresponde a que a los ensambladores no les conviene aceptar ofertas diferentes (con relacion a la cantidad de componentes) a las propuestas por estos. En este caso incluso con la estrategia *sunk aware* se están adquiriendo componentes innecesarios.

Este desarrollo se puede utilizar como herramienta de apoyo en la enseñanza de diseño e implementación de mecanismos (Computer mechanism design) y se puede expandir para simular otros escenarios y condiciones de mercado.

Referencias

[1] *The Trading Agent Competition*, [En línea]. Disponible: http://tac.eecs.umich.edu/

[2] John Collins, Raghu Arunachalam, Norman Sadeh, Joakim Eriksson, Nicolas Finne, Sverker Janson. *The supply Chain Management Game for the 2007 Trading Agent Competition*, [En línea]. Disponible: http://www.sics.se/tac/tac07scmspec.pdf

[3] Michael P. Wellman and Peter R. Wurman. *A Trading Agent Competition for the Research Community*, University of Michigan.

[4] Enrico Gerding, Peter McBurney, Jinzhong Niu, Simon Parsons and Steve Phelps. *Overview of CAT: A Market Design Competition*, 2007.

[5] Steven Skiena and Miguel Revilla. *Programming Challenges*, Springer, 2003.

[6] Mattew O. Jackson. *Mechanism Theory*. California Institute of Technology, 2005.

[7] Daniel M. Reeves, Michael P. Wellman and others. *Exploring bidding strategies for market-based scheduling*. University of Michigan Ann Arbor, 2005.

[8] *JFreeChart*. [En línea] Disponible: http://www.jfree.org/jfreechart