ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería Eléctrica y Computación

"POLISIMEM, UN SIMULADOR DE MERCADOS ELÉCTRICOS MAYORISTAS A BASE DE AGENTES INTELIGENTES"

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

Presentado por

Juan Francisco Arteaga Mendoza

Guayaquil – Ecuador

2008

TRIBUNAL DE GRADUACIÓN

Presidente: Sub-decano de la FIEC.

Director: Ing. Carlos Jordán V.

Miembros Principales: Ing. Galo Valverde L.

Dr. Cristóbal Mera G.

Miembros Suplentes: Ing. Xavier Ochoa CH.

Ing. Otto Alvarado M.

DECLARACIÓN EXPRESA

La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral.

RESUMEN

En la presente tesis se expone el diseño de un programa que simule varios tipos de mercados de compra y venta de energía eléctrica basado en un sistema de múltiples agentes inteligentes.

En un mercado eléctrico mayorista intervienen varias entidades como compradores, vendedores y entidades estatales. Aparte, existen varias maneras en que el mercado puede funcionar. La presente tesis propone simular los diferentes tipos de mercados haciendo que los agentes inteligentes tomen los lugares de los compradores y los vendedores para estudiar y comprender el comportamiento de estos mercados en diferentes situaciones.

Índice General.

Introducción.

1. Análisis del Problema.

1.1 El Mercado Eléctrico Mayorista: una definición.

1.2 Modelos de un mercado eléctrico mayorista.

1.3 Los Simuladores de Mercados Eléctricos.

1.4 Especificación del simulador. (Objetivos, función, entradas y salidas)

2. Análisis de las Herramientas / Conocimientos Disponibles.

2.1 Introducción a los Agentes Inteligentes.

2.2 Análisis de los Algoritmos de aprendizaje.

2.2.1 Holland's LCS

2.2.2 Wilson's ZCS

2.2.3 Wilson's XCS

3. Diseño de la Solución

3.1 Revisión panorámica del diseño.

3.1.1 Revisión panorámica de los Agentes.

3.1.2 Revisión panorámica del algoritmo de aprendizaje.

3.1.3 Revisión panorámica del simulador.

3.2 Diseño estructural

3.2.1 Clases creadas para el programa.

3.2.2 Relaciones entre Clases.

3.3 Las Especificaciones del Diseño.

4. Implementación.

5.1 El lenguaje de programación.

5.2 Problemas con el uso de la base de datos.

5.3 Requerimientos de hardware y software.

5.4 Productos.

5. Pruebas

5.1 El entorno para probar el simulador

5.2 Resultados

5.3 Discusión de los resultados

Conclusiones y Recomendaciones

Bibliografía.

Abreviaturas.

CENACE: Centro Nacional de Control de Energía. Regula los Contratos entre Compradores y Vendedores.

CONELEC: Consejo Nacional de Electricidad. Define el valor de la demanda eléctrica en el país.

LCS: Learning Classifying System. Sistema de Aprendizaje por Clasificación. Método de aprendizaje utilizado en la inteligencia artificial.

POLISIMEM: Politécnica Simulador de Mercado Eléctrico Mayorista. Es el nombre del programa.

SIN: Sistema Nacional Interconectado. Es el nombre del sistema nacional eléctrico.

XCS: Extended Classifying System. Sistema de Clasificación Extendido. Es una variación del LCS.

ZCS: Zeroth Level Classifying System. Sistema de Clasificación de nivel Cero. Es una variación del LCS.

Introducción.

Existen problemas cuya complejidad demanda realizar una serie de pruebas para poder comprender su comportamiento. Estas pruebas pueden resultar muy costosas para ser realizadas en un entorno real. Es aquí donde entran los programas simuladores, que imitan el entorno real para poder estudiar estos problemas sin necesidad de realizar grandes gastos en el mundo real.

El problema escogido es el de los mercados eléctricos mayoristas. Estos mercados pueden ser implementados de varias maneras y dar diferentes resultados bajo diferentes circunstancias. Por lo tanto es necesario realizar pruebas para elegir la mejor opción.

Nuestra hipótesis es que es posible simular los diferentes mercados eléctricos con un programa basado en el paradigma de los Agentes Inteligentes, capaces de aprender a comportarse y adaptarse a diferentes entornos y circunstancias.

Para probar esto se ha creado un programa en Java capaz de simular diferentes tipos de mercados eléctricos donde los compradores y vendedores son reemplazados por Agentes Inteligentes. Se ejecutarán varias pruebas para comprobar que los agentes son capaces de aprender y adaptarse, y luego una vez comprobado esto se ejecutarán pruebas para examinar los diferentes tipos de mercados y los resultados que se obtienen bajo diferentes parámetros.

1. Análisis del Problema.

1.1 El Mercado Eléctrico Mayorista: una definición.

1.2 Modelos de un mercado eléctrico mayorista.

1.3 Los Simuladores de Mercados Eléctricos.

1.4 Especificación del simulador. (Objetivos, función, entradas y salidas)

En este capítulo se explicará que es el mercado eléctrico mayorista y las diferentes formas en que puede funcionar (por subasta, contratos a largo plazo, precio, o costos); se describirán diferentes simuladores de mercados eléctricos mayoristas que ya se han creado; y se especificarán los objetivos que se intentarán lograr con este proyecto.

1.1 El Mercado Eléctrico Mayorista: una definición.

Para el beneficio de aquellos que no son especialistas en el tema, a continuación describimos en que consiste un sistema eléctrico de potencia y el mercado eléctrico mayorista. En particular, vamos a considerar dos aspectos: la infraestructura física de un sistema de potencia y el gobierno o administración del mismo.

El propósito de un sistema eléctrico de potencia es suministrar a un conjunto de consumidores la energía eléctrica generada por un conjunto de plantas generadoras. Puesto que generalmente productores y consumidores no se encuentran localizados en un mismo sitio, Para el efecto se utilizan líneas de transmisión, transformadores, etc.

Las plantas de generación pueden ser de muy variado tipo: hidroeléctricas, termoeléctricas, nucleares, a gas, eólicas, fotovoltaicas, etc.; esta clasificación corresponde esencialmente al combustible o medio que se utiliza para generar la energía eléctrica. Esto conlleva a que diferentes plantas produzcan diferentes cantidades de energía eléctrica a diferentes costos.

Los grandes consumidores de energía eléctrica pueden ser clasificados según el uso que den a la energía eléctrica, en tres tipos: residencial, industrial o de servicio. Estos consumidores tienen diferentes necesidades energéticas y estar preparados a pagar diferentes precios para satisfacer estas necesidades.

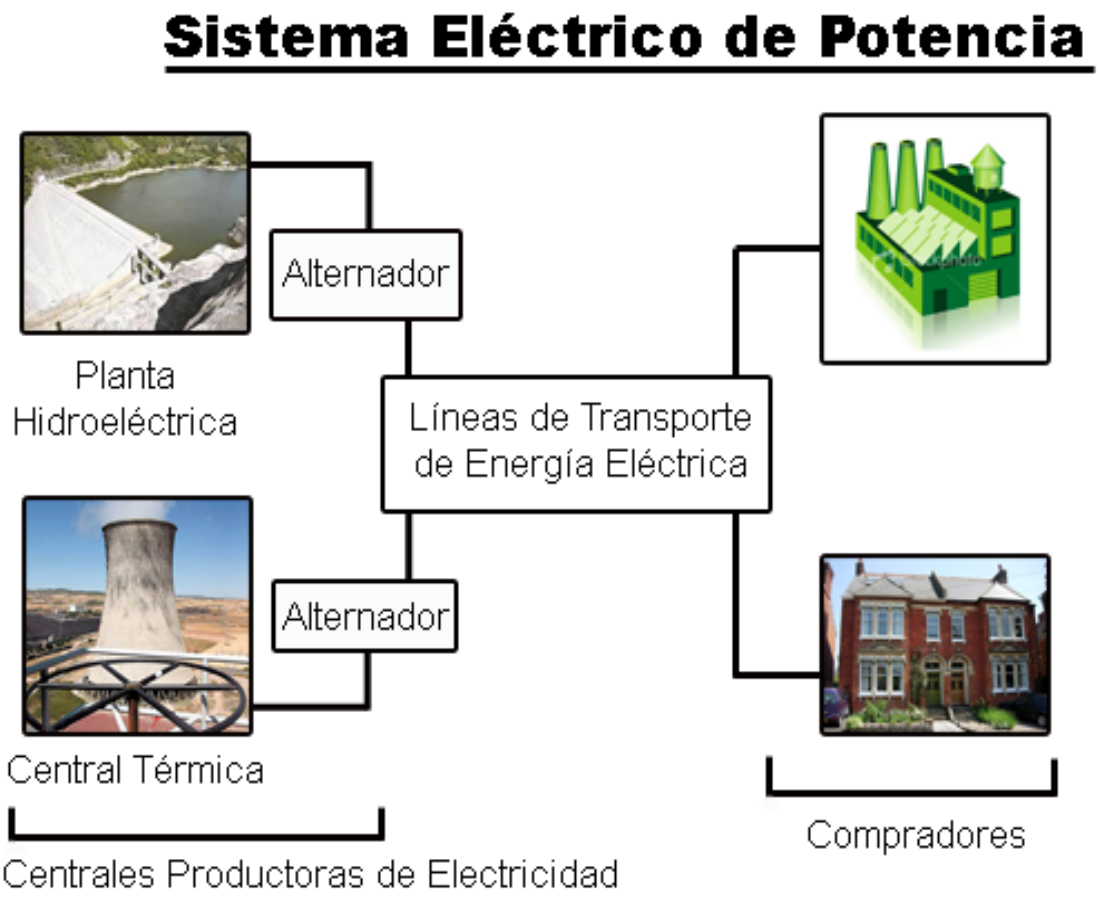


Ilustración 1.Gráfico de un sistema eléctrico de potencia.

El Mercado Eléctrico Mayorista es el lugar donde las plantas generadoras venden su producción y los grandes consumidores de energía la compran. Existen varias maneras en que este mercado puede operar y se las explicará detalladamente en la sección 1.2.

Actualmente en el Ecuadores mercado eléctrico está muy regulado. En el Mercado Eléctrico Mayorista ecuatoriano participan los generadores, los distribuidores y los grandes consumidores incorporados al Sistema Nacional Interconectado, SNI. En este mercado se realizan transferencias de energía y potencia entre los participantes.

El Consejo Nacional de Electricidad (CONELEC) es el que define periódicamente el valor de la demanda, los límites de energía consumida anualmente y los demás parámetros que caractericen a los grandes consumidores. Estas definiciones no tienen en ningún caso efecto retroactivo.

A pesar de que el valor de la demanda es definido por un ente del estado, los compradores y vendedores pueden crear contratos a largo plazo a su conveniencia que deben ser registrados en el Centro Nacional de Control de Energía (CENACE) (1)

En la mayoría de los países, las políticas en cuanto a planificación de la energía eléctrica eran establecidas por un centro regulador, pero esto cambio en los anos 80, con la ola de la desregularización, que empezó en Inglaterra, y que luego se propago por el mundo entero.

La desregularización consiste en fraccionar el negocio eléctrico mediante una separación de intereses, en tres sectores: generación, transmisión y distribución, y permitir la competencia del sector privado. Actualmente se busca desregular el mercado eléctrico ecuatoriano. Para ayudar en este propósito, se ha diseñado este simulador con la finalidad de poder observar como se comportan el mercado eléctrico con diferentes reglas y bajo diferentes condiciones.

_____

1 - José Layana Ch. Características del Mercado Eléctrico del Ecuador. ESPOL.

1.2 Modelos de un mercado eléctrico mayorista.

Existen dos maneras o modelos generales por la cuales el mercado eléctrico mayorista puede funcionar. A continuación se explicarán brevemente estas formas:

- Modelo de contratos bilaterales – El mercado eléctrico puede permitir que los productores vendan la energía que producen directamente a los consumidores, o que los consumidores la compren directamente de los productores por medio de un contrato bilateral donde se especifica la cantidad de energía que se está comprando, el precio y el tiempo en que este contrato estará vigente.

- Modelo de subastas – En una subasta, conocida como Mercado Spot, los consumidores y productores envían ofertas, se establece un precio y se aceptan o rechazan las ofertas. La subasta puede funcionar de varias formas.

    - Por Costo de Producción (Solo productores participan) – En este tipo de subasta los productores no pueden realizar ofertas basadas en el precio que desean que les sea pagado por la energía, si no que su oferta simplemente establece el costo de producción y la cantidad de energía que pueden producir. Las ofertas se ordenan por el costo de menor a mayor, y se aceptan las ofertas que sean necesarias para satisfacer la demanda.

- Por Costo de Producción (Consumidores y productores participan) – Los productores solo pueden ofertar un precio, el costo de producción, pero los compradores pueden ofertar diferentes precios. El precio se fija por medio de las curva de oferta y demanda. Las ofertas de productores que tengan un precio mayor son rechazadas, y las ofertas de consumidores con precios menores al establecido también.

- Por Precio (Solo productores participan) – Con este método cada productor de energía puede elegir el precio con el que van a ofertar la energía producida. Los consumidores en cambio no pueden enviar ofertas. Las ofertas de los productores se ordenan por precio de menos a mayor, y se aceptan las ofertas necesarias para satisfacer la demanda. El precio de la última oferta aceptada es tomada como precio a pagar por todos los consumidores.

- Por Precio (Consumidores y productores participan) – En esta versión ambos lados pueden ofertar. Cada productor elige el precio que desea que le sea pagado y la cantidad de energía que puede generar. Cada consumidor elige el precio que desea pagar y la cantidad de energía que necesita. El precio se fija por medio de las curva de oferta y demanda. Las ofertas de productores que tengan un precio mayor son rechazadas, y las ofertas de consumidores con precios menores al establecido también.

Todo mercado eléctrico funciona por lo menos con la opción de subasta, y puede permitir la creación de contratos bilaterales entre compradores y vendedores.

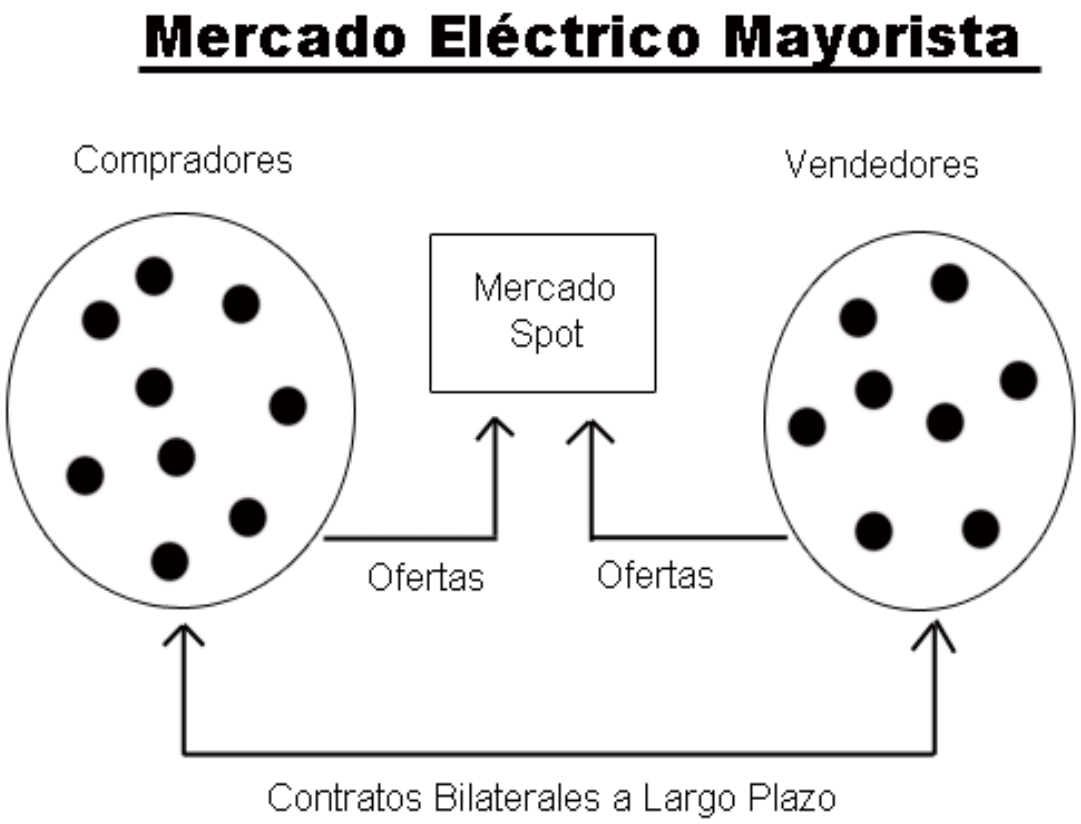


Ilustración 2. Gráfico de un Mercado Eléctrico Mayorista.

1.3 Los Simuladores de Mercados Eléctricos.

Se han creado algunos simuladores de mercados eléctricos basados en el paradigma de Agentes Inteligentes, a continuación se hablará brevemente de algunos ejemplos y sus diferencias:

A Multi-Agent Model of the UK Market in Electricity Generation

Este simulador con sistema multi-agentes fue creado en el Reino Unido por Anthony J. Bagnall y George D. Smith para estudiar los efectos de estructuras y mecanismos alternativos sobre el comportamiento del mercado eléctrico. Se utilizaron agentes autónomos adaptables, que utilizaban sistemas de aprendizaje y clasificación basados en jerarquías mezclado con otros métodos de aprendizaje y que aprendían a través de la competición en un modelo simulado del mercado de generación eléctrica inglés.

Se creó una compleja estructura de agentes para comprobar si el simulador cumplía con los tres siguientes puntos:

1 - Los agentes adaptables aprenden estrategias óptimas compitiendo contra agentes no-adaptables.

2 - Los agentes son capaces de aprender estrategias observables en el mundo real cuando compiten contra agentes adaptables.

3 - Los agentes son capaces de cooperar sin comunicación explícita. Es decir, que la cooperación evolucione espontáneamente en ciertas situaciones de mercado.

Este simulador logró tener buenos resultados en los dos primeros puntos. El tercer punto es más difícil de analizar y no se espera que los agentes encuentren un equilibrio de cooperación sin poder comunicarse. (2)

MASPOWER.

Este simulador fue creado en el departamento de computación de Iowa State por V. Vishwanathan, J. McCalley y V. Honavar. Es un programa desarrollado en JAVA para ayudar en la toma de decisiones con múltiples objetivos a través de un sistema de agentes interesados en sus ganancias económicas.

En este simulador los agentes pueden comunicarse entre ellos y llegar a acuerdos para tomar acciones que maximicen sus beneficios. (3)

---

2 - Anthony J. Bagnall, George D. Smith. A Multi-Agent Model of the UK Market in Electricity Generation. 2005.

3 - V. Vishwanathan, J. McCalley, V. Honavar. A Multiagent System Infrastructure and Negotiation Framework for Electric Power Systems. Iowa State University. 2001.

MASCEM

Este simulador fue creado por Isabel Praca, Carlos Ramos, y Zita Vale del instituto politécnico de Porto y por Manuel Codeiro de la Universidad de Tras-os-Montes e Alto Douro. Su propósito es para servir de herramienta en el estudio del comportamiento y evolución de los mercados eléctricos. Los agentes representan a los compradores, generadores e inclusive revendedores que aprenden y se adaptan a cambios. El simulador incluye varios mecanismos de negociación encontrados en mercados eléctricos para que el usuario los pruebe y aprenda la mejor manera de negociar en cada uno de ellos. El simulador es flexible y permite definir el modelo que se desea simular, incluyendo el número de agentes, su tipo y sus estrategias y el tipo de mercado en el que competirán. (4)

Diferencias entre POLISIMEM y los otros simuladores mencionados.

La mayor diferencia entre POLISIMEM y los otros simuladores, excepto por MASCEM, es la flexibilidad del entorno que se desea simular. En los otros simuladores, no se menciona la posibilidad de cambiar el entorno y las reglas del mercado. Uno de los propósitos de nuestro simulador es ver los comportamientos de los diferentes tipos de mercados eléctricos que se pueden implementar en el país, y no solamente comprobar que los agentes inteligentes se pueden utilizar para simular mercados específicos.

---

4 - Isabel Praca, Carlos Ramos, Zita Vale, Manuel Cordeiro. MASCEM: A Multiagent System That Simulates Competitive Electricity Markets. 2003

Otra gran diferencia es que los agentes de los otros simuladores tienen una inteligencia artificial más específica con estrategias previamente programadas mientras que la inteligencia artificial de los agentes de POLISIMEM trabaja puramente con el algoritmo de aprendizaje de Holland. Es decir que no contienen ninguna estrategia programada previamente y no tienen ningún conocimiento al empezar la simulación.

1.4 Especificación del simulador. (Objetivos, función, entradas y salidas)

Los objetivos que se esperan alcanzar con este simulador son los siguientes:

- Estudiar la aplicación de sistemas multiagentes para la implementación de simuladores.

- Estudiar los diferentes modelos de mercados eléctricos que se encuentran en uso actualmente.

Función:

El programa será capaz de simular diferentes tipos de mercados eléctricos con un sistema de agentes inteligentes con capacidad de aprender las mejores respuestas a los cambios del entorno. Después de la simulación el programa le detallará al usuario las decisiones tomadas por los agentes y los resultados que estos obtuvieron en una base de datos Access transportable.

Entradas:

Las únicas entradas que necesita el programa son la información sobre el tipo de simulación que se desea realizar y la localización de la base de datos por parte del usuario. Cómo ingresar esta información se encuentra detallada en el manual de usuario.

Salidas:

La salida del programa es los resultados de la simulación los cuales se guardan en una base de datos Access transportable. Estos datos también pueden ser revisados a través del programa por medio de la ventana de reportes.

2. Análisis de las Herramientas / Conocimientos Disponibles.

2.1 Introducción a los Agentes Inteligentes.

2.2 Análisis de los Algoritmos de aprendizaje.

2.2.1 Holland's LCS

2.2.2 Wilson's ZCS

2.2.3 Wilson's XCS

En este capítulo se explicarán los Agentes Inteligentes. (que son, características, tipos, como funcionan, como se usan para resolver el problema: simulación) y se justificará porque son convenientes los agentes inteligentes en problemas de simulación. También se describirán los algoritmos de aprendizaje del tipo Sistemas de Aprendizaje por clasificación (Holland LCS, Wilson ZCS, Wilson XCS) y se justificará porque se eligió el Holland LCS de entre los tres mencionados.

2.1 Introducción a los Agentes Inteligentes.

Los sistemas de agentes inteligentes, y los agentes inteligentes en sí, son un nuevo paradigma de la inteligencia artificial que facilitan el diseño de ciertos sistemas y programas al permitirnos ver ciertas partes del sistema como si fueran entidades que actúan, interactúan, compiten y cooperan entre sí para lograr sus propios objetivos en un entorno.

Para resolver ciertos problemas se necesita que un sistema pueda tomar decisiones sobre las acciones que debe realizar por si mismo para así poder satisfacer los objetivos para los cuales fue diseñado. Estos sistemas son conocidos como Agentes. Un agente que debe operar sobre un entorno impredecible de rápido cambio, o abierto, donde las acciones que se toman pueden fallar, son conocidos como agentes inteligentes.

Un ejemplo sencillo de un sistema de agentes inteligentes podría ser un programa que simula un juego de fútbol con varios jugadores en la cancha. Cada jugador sería un agente, y cada uno de ellos tener un objetivo diferente, o serie de objetivos diferentes a realizar, dependiendo de en que equipo juegue o en que posición juega.

Existen muchas definiciones de Agentes que han sido propuestas por diferentes investigadores de la Inteligencia Artificial. La definición más sencilla es:

Un Agente es una entidad que percibe y actúa sobre un entorno. *(Russell 1996)* (5)

---

5 - Stuart Russell, Peter Norvig. Inteligencia Artificial: Un enfoque moderno. 1996.

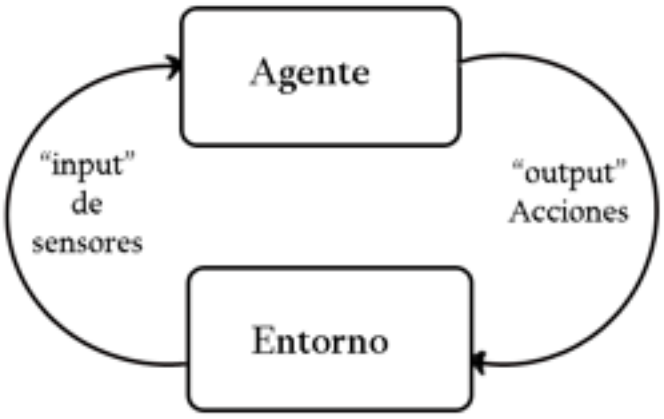


Ilustración 3. Agente Inteligente.

En esta ilustración se muestra un diagrama básico que contiene un agente realizando acciones sobre un entorno, y a su vez retroalimentándose de información sobre este.

Esta definición, aunque correcta, no logra explicar en su totalidad lo que es y lo que hace un agente. Note por ejemplo que la definición tampoco menciona algo sobre la inteligencia o la autonomía. Un agente no solo percibe y actúa sobre un entorno, sino que es parte del entorno. El agente también solo actúa para lograr sus propios objetivos. Es decir, que el agente actúa en su beneficio, o, después de evaluar la acción que va a tomar, decide que la acción que va a realizar no le causa ningún perjuicio. Del mismo modo, el agente no solo debe percibir el entorno, sino también cambiar sus planes de acción dependiendo de los cambios que ha percibido. En otras palabras, el agente debe poder ser flexible. Un agente también debe ser autónomo. Esto significa que el agente, una vez iniciado el programa del que es parte, no necesita ninguna interacción con el usuario del programa para funcionar o tomar decisiones.

De este modo podemos extender la definición original y llegar a la siguiente:

Un agente inteligente es una entidad situada en y parte de un entorno que siente ese entorno y actúa de forma flexible sobre él, a través del tiempo, persiguiendo sus propios objetivos de forma que afecte lo que siente en el futuro.

De este modo, siguiendo con el ejemplo del simulador de fútbol, un agente solo hará jugadas que en su opinión sean beneficiosas para lograr su objetivo de ganar el partido para su equipo, y tomará cada decisión sin intervención alguna del usuario.

Aparte de esta definición a la que hemos llegado, muchos expertos consideran que para que una entidad del sistema sea considerada como un agente inteligente, debe poseer las siguientes tres características:

Ser Reactivo – El agente debe ser capaz de responder a cambios en el entorno en el que se encuentra situado.

Ser Pro-Activo – El agente debe ser capaz de intentar cumplir con sus propios planes y objetivos.

Ser Social – El agente debe ser capaz de comunicarse con otros agentes.(6)

_____

6 - Michael Wooldridge. Intelligent Agents. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.

Al agregar estas tres características a nuestra definición, y siguiendo con el ejemplo del simulador de juego de fútbol, podemos determinar que si se puede programar de tal forma que cumpla todos los requisitos. Cada jugador en la cancha debe poder responder a cambios de posición de jugadores en su equipo, del equipo contrario, y del balón. Debe poder responder a cambios de estrategia del equipo contrario y de su propio equipo. En otras palabras, debe ser Reactivo. Cada jugador debe ser capaz de intentar evitar que el equipo contrario meta goles, y debe ser capaz de intentar ayudar a que su equipo meta goles. Y así cumple con la característica de ser Pro Activo. Cada jugador también debe ser capaz de comunicarse a los otros jugadores para decirles que no está marcado y que desea que le pasen el balón, o comunicar un cambio de estrategia si es el capitán o el director técnico del equipo, cumpliendo así con la característica de ser Social.

Hay que recalcar que la tercera característica, por lo menos en mi opinión, no es necesaria para que construir un agente inteligente. Un agente que trabaja solo para resolver un problema no es social, ya que no tiene otros agentes en su entorno para comunicarse con ellos. También es posible tener un grupo de agentes que compitan entre sí en lugar de trabajar entre ellos, y por lo tanto puede que no sea necesario que se comuniquen entre ellos para negociar.

En POLISIMEM los agentes poseen las tres características. Son reactivos ya que sus estrategias cambian dependiendo de los resultados del mercado spot. Son pro-activos porque siempre están pensando en minimizar sus gastos si son compradores, o maximizar sus ganancias si son vendedores. Pero, son sociales solo cuando se les está permitido

crear contratos a largo plazo entre ellos. En simulaciones sin contratos los agentes no se comunican entre sí.

Otra cosa que un agente normalmente debe poseer es un repertorio de acciones que pueda realizar para operar sobre el entorno en el que se encuentra. Este conjunto de posibles acciones es conocido como ’capacidad efectórica.’ Estas acciones usualmente tienen precondiciones asociadas a ellas, que definen en que situaciones posibles estas acciones pueden ser aplicadas.

Al hablar de acciones, y las situaciones en que los agentes deben aplicarlas, es importante definir los tipos de entornos en los que un agente puede operar. En bastantes entornos de relativa complejidad, un agente no va a tener completo control sobre este. En el mejor de los casos se puede decir que el agente tiene un control parcial sobre el entorno, o que es capaz de influenciar el entorno. Esto puede llevar a que una misma acción, ejecutada por el agente en situaciones que aparentan ser similares, conduzca a diferentes resultados en instancias diferentes. Debido a esta falta de predeterminación de los resultados que se van a obtener, un agente debe estar preparado para la posibilidad de fallar.

La toma de decisiones es el problema clave al que se enfrentan los agentes operando en un entorno. Las arquitecturas de agentes, de las cuales hablaremos mas adelante, son en verdad arquitecturas para sistemas de toma de decisiones. La complejidad del proceso de tomar una decisión es afectada por las diferentes propiedades del entorno. Russell y Norvig nos sugieren la siguiente clasificación de propiedades de entorno:

- **Accesible vs. Inaccesible.**

  Un entorno accesible es aquel donde el agente puede obtener información completa, actualizada y correcta sobre el estado del entorno. Mientras más accesible sea un entorno, mas simple es la toma de decisión.

  Un ejemplo de un entorno muy accesible sería un agente que juega ajedrez contra un humano o contra otro agente, ya que este siempre sabe en que casilla está cada ficha del juego, lo cual es toda la información que necesita para tomar decisiones. Y un ejemplo de un agente en un entorno menos accesible sería un agente robot que limpia pisos, como el Roomba, el cual tiene un sensor limitado y solo puede detectar polvo y suciedad en un rango limitado; por lo tanto el robot solo sabe por donde ya ha limpiado y que lugares sucios tiene cerca, pero no sabe nada sobre las partes del piso por las cuales todavía no ha pasado.

- **Determinístico vs. No-determinístico.**

  Un entorno determinístico es aquel donde cada acción tiene un solo efecto garantizado. Es decir, que cada vez que el agente tome esa acción, el resultado sobre el entorno siempre será el mismo. Para ilustrar esto, volvemos a tomar el ejemplo del agente jugador de ajedrez. Cada movida que hace el agente es una acción, y cada acción afecta al entorno solo de una manera, y siempre es de la misma manera. Si el agente jugador de ajedrez mueve un peón una casilla hacia delante, el entorno será igual que antes de la movida, excepto que un peón ahora

está una casilla mas adelante y ningún otro resultado es posible. En cambio, un agente que compra artículos en un simulador de subasta, y tiene que competir con otros agentes por los artículos que compra, se encuentra en un entorno No-determinístico, ya que puede, por ejemplo, tomar la acción 'Ofertar mil dólares' y lograr ganar el artículo que deseaba ganar, como también puede realizar la misma acción y no ganar. Esto se debe a que el entorno depende de las acciones de todos los agentes – lo cual va más allá del control del agente del ejemplo- y no solo del agente en cuestión.

- **Episódico vs. No-episódico.**

  En un entorno episódico, el agente no debe preocuparse de sus acciones pasadas o sus acciones futuras, y solo en el episodio en el que se encuentra actualmente, ya que su desempeño en cada episodio no está enlazado de ninguna manera con su desempeño en cualquiera de los otros episodios.

Un ejemplo de un agente en un entorno episódico puede ser un agente encargado de controlar la temperatura en un edificio. Si necesita enfriar una habitación, prenderá el aire acondicionado, y si necesita calentarlo, entonces prenderá el calentador. A este agente no le importa las acciones que realizó hace una hora o hace un día, o las que realizará en el futuro, ya que no afectan en nada sus acciones actuales. Por lo general, los agentes en un entorno episódico son agentes reactivos. Un agente que juega ajedrez, por otra parte, se encuentra en un entorno No-Episódico, ya que su meta es ganar, y para llegar a esa meta debe tomar una serie de acciones, y no solo una

acción. Este agente no solo debe tomar una acción que lo beneficie a corto plazo, sino que debe pensar en el futuro cada vez que mueve una pieza.

- **Estático vs. Dinámico.**

  Un entorno estático es aquel que se puede asumir se mantendrá igual excepto por la realización de acciones del agente. Es decir, que en un ambiente estático, solo el agente tendrá la capacidad de modificar el estado del entorno. En un entorno dinámico existen otros procesos operando sobre este, y están más allá del control del agente, lo cual significa que el estado del entorno puede cambiar sin que el agente actúe. Estos procesos que operan sobre el entorno bien podrían ser otros agentes. Encontrar un ambiente estático puro es muy difícil, ya que la descripción descarta completamente cualquier ambiente del mundo real, donde cualquier cosa puede afectar al entorno de maneras difíciles de prever, y cualquier ambiente con más de un agente.

- **Discreto vs. Continuo.**

  Un entorno es discreto si existe un número finito y fijado de estados y Continuo si el número de estados es infinito o prácticamente infinito. Un juego de damas es discreto, ya que el número de posibles combinaciones de posicionamiento de las fichas tiene un límite. Un juego de ajedrez es más complicado, pero también es discreto, aunque en la práctica son tantas las posibilidades que es posible considerarlo continuo. Un ambiente netamente continuo es el mundo real, donde es casi imposible determinar el número de variables. Es posible simplificar un

ambiente continuo y convertirlo en discreto simplificando las variantes e ignorando las variables que no afectan al funcionamiento del agente.

Diferencias entre Agentes, Objetos y Sistemas Expertos.

Es muy posible crear un objeto, en un lenguaje orientado a objetos, que a la vez sea un Agente Inteligente, pero esto no significa que todos los agentes son objetos o que todos los objetos son agentes. Se puede crear un Agente que en si contenga varios objetos interactuando entre sí para que el sistema tenga el comportamiento de un agente, como también se puede crear un agente sin necesidad de un lenguaje orientado a objetos. Aunque los objetos y los agentes tengan ciertas similitudes, tienen tres grandes e importantes diferencias conceptuales:

1. La primera diferencia es el grado de autonomía que tienen los objetos y los agentes. Conceptualmente, un objeto tiene control sobre su estado, pero no tiene control sobre su comportamiento, mientras que todo el concepto de agente se basa en que este controla su comportamiento. Un objeto puede llamar a una subrutina o función de otro objeto y obligarle a este a que ejecute esta subrutina. En cambio, los agentes solo pueden enviarse pedidos de ejecutar acciones, y cada agente decide si ejecuta la acción o no. Esto lleva a que los objetos realicen cualquier trabajo que se

les pida, mientras que los agentes solo realizan acciones que los benefician a ellos.

2. La segunda diferencia es respecto a la flexibilidad del comportamiento de estos. El modelo de objetos estándar no dice nada sobre incluir **reactividad**, **pro-actividad** o **sociabilidad** en el comportamiento de los objetos, los cuales son conceptos claves en el comportamiento de un agente inteligente.

3. Otra gran diferencia es que se asume que cada agente tiene su propio hilo de control. En el modelo estándar de objetos, solo existe un hilo de control, aunque es posible aumentar la cantidad de hilos, pero esto no se considera necesario para tener un modelo de objetos.

Entre Agentes y sistemas Expertos, la diferencia es mas sencilla. Los Sistemas Expertos, a diferencia de los Agentes, no interactúan directamente con el entorno, sino que aconsejan o responden a un tercero.

Arquitecturas para agentes inteligentes.

A continuación explicaré algunas arquitecturas de toma de decisiones que se pueden aplicar a los agentes. Esto ayudará a aclarar las diferentes formas en que un agente puede razonar y actuar dependiendo de la arquitectura que se le ha aplicado.

- **Agentes puramente reactivos.**

  Estos son los agentes que deciden que hacer sin hacer referencia a su historia. Basan sus decisiones completamente en el presente, sin ninguna referencia al pasado. Se llaman puramente reactivos, ya que simplemente responden al entorno. Funcionan del siguiente modo:

  Si se da tal situación, entonces se deberá realizar la siguiente acción.

- **Agentes basados en lógica.**

  Son agentes que toman decisiones mediante una deducción lógica.

  Estos agentes tienen una serie de reglas de deducción programadas por el programador de la forma Si (tal condición, o combinación de condiciones, se cumple) entonces hacer lo siguiente. De esta forma el comportamiento del agente es determinado por las reglas de deducción que posee, es decir su programa, y la información que el agente tenga sobre su entorno, que puede incluir las acciones del agente en el pasado.

- **Agentes Creencia-Deseo-Intención.**

  Son agentes donde la decisión depende de la manipulación de estructuras de datos que representan las creencias, deseos e intenciones del agente.

  Las creencias del agente, que son la información que el agente tiene acerca del entorno actual en que se encuentra, sirven para generar opciones de metas que el agente se puede proponer. Los deseos del agente sirven para filtrar estas

opciones y así escoger la meta o metas que el agente tiene que proponerse para satisfacer las necesidades para las cuales fue programado. Estas metas se convierten en intenciones las cuales llevan al agente a tomar acciones para cumplir con lo que se propone.

- **Arquitectura por capas.**

  Son agentes donde las decisiones se toman en varias capas de software, cada una razonando más o menos explícitamente acerca del entorno a diferentes niveles de abstracción. Esta arquitectura es el resultado de necesitar agentes que sean capaces de tener comportamiento reactivo y comportamiento pro-activo. Esto hace necesario agentes que tengan subsistemas separados para manejar estos dos diferentes tipos de comportamiento. Esto lleva a una arquitectura con varios subsistemas puestos en una jerarquía de capas que interactúan entre sí.

  Dos formas de poner estas capas son:

  - Horizontal.

    Cada capa está directamente conectada a los sensores para recibir input del entorno, y a la parte que ejecuta la acción. De esta manera, cada capa funciona como un agente y estas capas discuten que acción deben tomar.

  - Vertical

En esta forma, solo una capa está conectada a los sensores, y solo una capa está conectada a la parte que ejecuta las acciones.

Sistemas multiagentes y sociedades de agentes.

Teniendo ya en claro una definición adecuada para un Agente Inteligente, podemos definir un Sistema de Agentes Inteligentes como un programa o sistema que contiene agentes inteligentes como parte de su estructura. No todos los componentes del sistema deben ser Agentes Inteligentes, pero para cumplir con todas las características, debe de tener por lo menos dos para satisfacer el requisito de sociabilidad.

Los sistemas multiagentes o sociedades de agentes surgieron de la necesidad de resolver problemas demasiado complejos o grandes como para ser resueltos por un solo agente. La capacidad de un agente está limitada por su conocimiento, recursos y perspectiva. Para enfrentar estos problemas demasiado complejos, se decidió que la mejor ruta era la modularidad. La única forma razonable de atacar un problema que es particularmente grande, complejo e impredecible es con un número de componentes modulares, cada uno diseñado para resolver una parte del problema. En otras palabras, mediante agentes. Esta modularidad permite que cada agente solucione su parte del problema de la forma que le parezca más conveniente. De esta forma, el sistema multiagente se convierte en una red de módulos que resuelven problemas que están más allá de sus capacidades individuales y que interactúan entre sí para evitar conflictos entre ellos.

Los sistemas multiagentes son muy buenos realizando los siguientes tipos de trabajos:

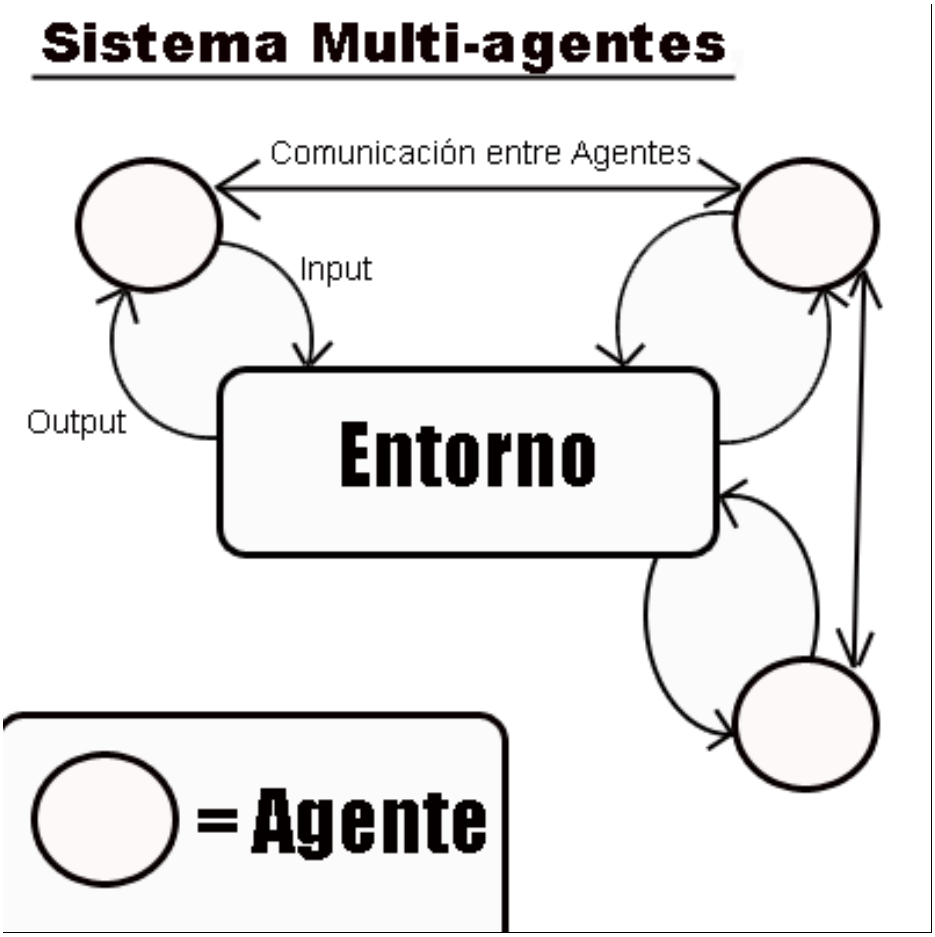


Ilustración 4. Sistema Multi.Agentes.

1- Resolver problemas demasiado complejos para sistemas centralizados debido a limitaciones de recursos, o para evitar que este sistema centralizado se vuelva en un cuello de botella capaz de fallar en un momento crítico.

2- La tecnología de sistemas multiagentes permitiría que software ya anticuado pueda seguir siendo útil al incorporarlos a redes de agentes donde este software puede ser utilizado por otro software. Esto se puede lograr programando una

envoltura que permita al viejo software trabajar como un agente. Este es una solución a mediano plazo que ahorraría tiempo y dinero en vez de hacer programas completamente nuevos.

3- Resolver problemas donde existen varios componentes autónomos interactuando. Por ejemplo un simulador de mercado como esta.

4- Resolver problemas donde las fuentes de información se encuentran distribuidas en varias computadoras.

5- Mejorar el rendimiento en las siguientes áreas:

   a. Eficiencia de Procesamiento, ya que al tener varios agentes trabajando, se explota al máximo la habilidad de tener varios procesos corriendo en paralelo.

   b. Seguridad, ya que si un componente llega a fallar, otros agentes con capacidad redundante, es decir, que pueden realizar la misma tarea, pueden tomar su lugar.

   c. Extensibilidad, ya que si llegaran a faltar, siempre se pueden poner más agentes a trabajar en el problema. O, en casos extremos, poner nuevos agentes con nuevas y diferentes capacidades.

d. Robustez, ya que el sistema tiene la habilidad de tolerar incertidumbre, porque la información pertinente es intercambiada entre los agentes.

e. Mantenimiento, ya que un sistema modular, como lo son los sistemas multiagentes, es más fácil de mantener que uno centralizado.

f. Respuesta, ya que la modularidad permite que el sistema resuelva problemas locales de forma local, en vez de propagarlo por todo el sistema.

g. Flexibilidad, debido a que agentes con diferentes habilidades pueden adaptarse de mejor manera para resolver diferentes problemas.

h. Reutilización, ya que un agente con una funcionalidad específica puede ser utilizado en diferentes grupos de agentes para resolver diferentes problemas.

Un sistema multiagente posee las siguientes características:

1. Un entorno multiagentes provee una infraestructura especificando los protocolos de comunicación e interacción.

2. Normalmente son entornos abiertos y no poseen un diseño centralizado.

3. Deben contener agentes autónomos y distribuidos, que pueden ser egoístas o cooperativos.

4. Cada agente posee información o capacidad incompleta para resolver el problema por si solo.

5. El proceso computacional es asincrónico. Es decir, que todos los agentes están activos todo el tiempo y no tienen que esperar su turno para empezar a actuar. Esto se lo logra con un programa de varios hilos (threads).

Los protocolos de comunicación permiten a los agentes intercambiar y entender mensajes. Los protocolos de interacción permiten a los agentes tener conversaciones, que para nuestros propósitos son intercambios de mensajes estructurados. Los agentes pueden intercambiar los siguientes tipos de mensajes.

- Proponer un curso de acción
- Aceptar un curso de acción.
- Rechazar un curso de acción.
- Retractar un curso de acción.
- No estar de acuerdo con un curso de acción propuesto.

- Contraproponer un curso de acción.

  De esta forma un agente es capaz de comunicarles a los otros agentes lo que él cree que se debe hacer, y los otros agentes son capaces de discutir con él, y proponer acciones diferentes.

Los agentes inteligentes no necesariamente deben ser capaces de aprender por medio de la experiencia que ganan al intentar resolver un problema, pero los agentes que han sido creados para POLISIMEM si utilizan un algoritmo de aprendizaje que les permite aprender como lograr sus objetivos y a la vez les da una gran flexibilidad para enfrentarse a cambios en el entorno. En la siguiente sección se explicará la teoría detrás del algoritmo de aprendizaje escogido para los agentes.

2.2 Análisis de los Algoritmos de aprendizaje.

Los sistemas de aprendizaje y clasificación (Learning Classifying Systems o L.C.S.) son un paradigma de la inteligencia artificial enfocados a resolver problemas mediante inteligencias capaces de aprender a través de la experiencia y adaptarse a diferentes situaciones. Son particularmente útiles para resolver problemas complejos difíciles de entender, como procesos de control o data-mining.

Una agente que utiliza L.C.S. aprende por medio de técnicas de computación evolutivas. Esto significa que aplican el principio de Darwin conocido como ‘La supervivencia del

más apto’. El agente comienza con un grupo de soluciones creadas al azar para resolver el problema con el que se enfrenta. Con el tiempo se generan nuevas soluciones, y, mediante la experiencia ganada con el tiempo y la utilización de las soluciones disponibles, se van guardando las soluciones más aptas y descartando las soluciones menos aptas. Después de algún tiempo, el agente contará con un grupo de soluciones aceptablemente aptas y será capaz de resolver el problema de manera satisfactoria. Para lograr esto se utiliza un sistema de aprendizaje por recompensa, donde se asigna una recompensa numérica dependiendo de que tan correcta fuera la respuesta.

Los L.C.S. están basados en reglas. Las soluciones que se generan en estos sistemas deben tener la forma

“SI estado ENTONCES acción”.

Una forma más clara de presentar este formato es de la siguiente forma: “SI el sistema sobre el cual está trabajando se encuentra en este estado, ENTONCES se deberá tomar la siguiente acción.” Por ejemplo una regla para un robot inteligente encargado de aspirar el polvo del piso podría tener la siguiente forma: “SI el piso se encuentra limpio de polvo, ENTONCES no aspirar y pasar a la siguiente sección de piso.” El programa simplemente debe de alguna forma saber en qué estado se encuentra el sistema y compararlo con la sección ‘SI estado’ de la regla para encontrar reglas que pueda utilizar.

Cada regla tiene un puntaje, el cual se puede implementar de diversas maneras y sirve para indicar que tan buenos han sido los resultados para el agente que ha utilizado la regla en su intento de resolver el problema que se le presenta. Utilizando la filosofía darvinista de estos algoritmos, las reglas con peores puntajes son las que se reemplazan con nuevas reglas mientras que las reglas con buenos puntajes se quedan en el grupo de soluciones.

Existen varias formas de L.C.S. y para este trabajo se tomó como base la forma original conocida como los L.C.S. de Holland.

2.2.1 LCS de Holland.

En este tipo de L.C.S. el programa recibe el estado del entorno como una cadena de caracteres, donde cada carácter representa una condición diferente. Un ejemplo simple para ilustrar de lo que hablamos puede ser el mismo programa de un robot aspiradora, done el estado del sistema se muestra con dos caracteres.

El primer carácter muestra si la sección del suelo sobre la que se encuentra el robot aspiradora está sucia, y puede tomar, por ejemplo, tres valores: “0” si está limpia, “1” si está un poco sucio, y “2” si está muy sucio. El segundo carácter indica si todavía falta por limpiar otras secciones del suelo, en cuyo caso los valores serían “0” si faltan, y “1” si no faltan.

La base de reglas contiene una población de N reglas tipo SI-ENTONCES. Las condiciones y acciones especificadas en la regla son una cadena de caracteres como esta {0,1,#}. El carácter # se utiliza como un comodín, lo que permite la generalización de la condición de regla, de tal forma que una condición como esta {0,#,1} haga juego con los estados del sistema representados por 011 y 001. Los comodines también son permitidos en la sección que especifica la acción a tomar, pero solo si la parte condicional de la regla también lleva un comodín. Por ejemplo, si el estado del sistema es 101, la regla SI 1#1, ENTONCES 0#0 produciría la acción 000.

Al recibir un estado, se busca en toda la lista de reglas y cualquier regla que haga juego con el estado se convierte en un miembro de la lista de reglas que hace juegos, llamado "match set" [M]. La regla, o reglas, que se utilizarán son escogidas al comparar dos componentes: que tan específicas son – es decir, que número de caracteres no-comodines contiene -, y que tan buenos resultados han conseguido. Por lo general se da más importancia primero al nivel de especificación, y luego al puntaje que haya recibido la regla.

La regla ganadora es puesta en un "balde". Se guarda un registro de todas las reglas escogidas anteriormente y todas reciben una parte igual del puntaje que ha ganado la nueva regla escogida.

El LCS para crear nuevas reglas muta reglas existentes o las mezcla y estas reemplazan a las reglas existentes. Es importante recalcar que la intención es crear un grupo de reglas

cooperativas que juntas puedan resolver el problema; a diferencia de un escenario de optimización tradicional, donde se busca encontrar una sola regla óptima.

2.2.2 Wilson's ZCS

El ZCS, Zeroth Level Classifying System, es una versión simplificada del LCS de Holland. Zeroth significa ser el número cero de una serie, y por lo tanto una traducción del nombre de este algoritmo sería Sistema de Clasificación de nivel Cero. En esta versión no se permiten comodines. De esta manera se selecciona una acción a tomar solamente por su puntaje. Al igual que en el LCS, las acciones tomadas reciben un puntaje, pero en cambio, en esta versión las reglas que entraron al "match set" pero que no fueron escogidas son penalizadas en su puntuación.

ZCS utiliza dos métodos para crear nuevas reglas. La primera forma es la siguiente: Hay una probabilidad p de que se cree una nueva regla en cada turno. Se escogen dos reglas basado en puntajes y se crea una nueva regla por mutación o combinación. Las dos reglas padre donan la mitad de su puntaje al puntaje de la nueva regla. La otra forma de crear una regla que tiene el ZCS funciona de la siguiente manera: Se crea una regla para la condición exacta del estado en que se encuentra el ambiente, y una acción creada al azar. Esta segunda forma por lo general solo se invoca las reglas para esa condición tienen un puntaje muy bajo, o si no existen reglas para esa condición.

2.2.3 Wilson's XCS

La más grande diferencia entre el XCS, Extended Classifying System (Sistema de Clasificación Extendido), y los otros sistemas es que el puntaje de la regla no está basado en los puntos recibidos por un buen trabajo, si no en su capacidad de predecir los puntos que va a recibir. La intención es formar un mapa detallado de todo el problema y no solo enfocarse en el mejor resultado o mejor paga.

En cada turno se crea un "match set". Se crea una predicción para cada acción en el match set de acuerdo al promedio de puntaje. La acción es escogida de manera determinística o al azar. Después de cada turno se actualiza la exactitud de la regla escogida.

Nuevas reglas son creadas de la siguiente manera: Se elijen dos reglas por puntaje, y basados en la última vez en que se usaron para crear una nueva regla. Con estas dos reglas, por mutación o combinación se crea una nueva.(7)

---

7 - Larry Bull. Learning Classifying Systems: A Brief Introduction.

CAPITULO 3. Diseño de la Solución

En este capítulo se explicará, sin entrar en muchos detalles el funcionamiento interno de las clases y objetos del programa, la forma en que trabajan los agentes inteligentes de este simulador, la forma en que trabaja el algoritmo de aprendizaje de los agentes, y el funcionamiento del simulador de mercado eléctrico mayorista. Finalizado esto, so procederá a detallar las clases que utiliza el programa para su funcionamiento, enumerando sus variables, métodos y relaciones con otras clases.

3.1 Revisión panorámica del diseño.

3.1.1 De los Agentes.

Los agentes inteligentes que toman el lugar de los compradores y vendedores en esta simulación de compra y venta de energía eléctrica están compuestos por dos clases de objetos: La clase Agente, la cual es una clase padre de la cual heredan las clases comprador y vendedor, y la clase Lista de Reglas.

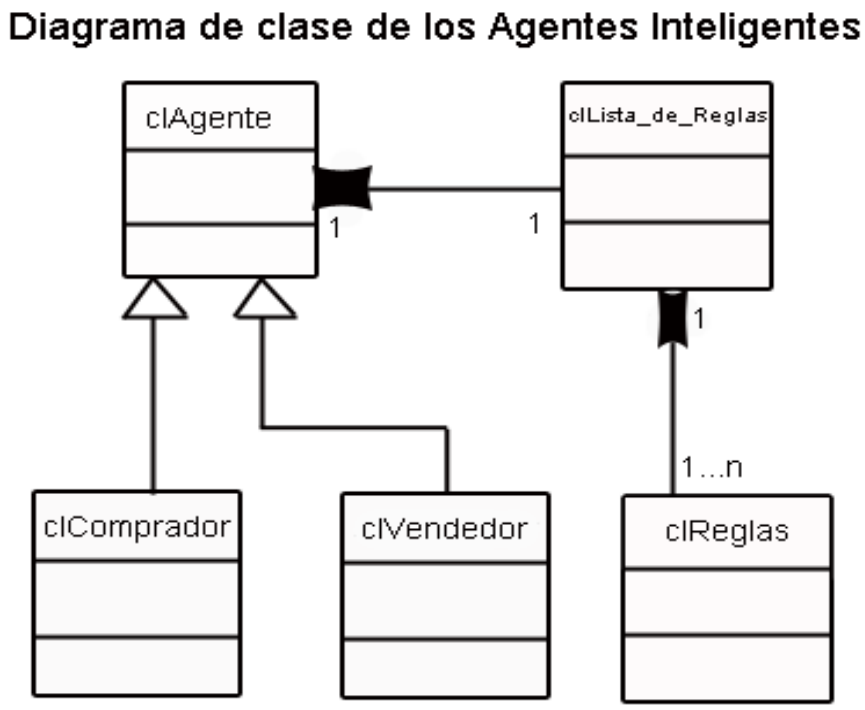


Ilustración 5. Diagrama de clase de los Agentes Inteligentes.

La clase Agente es la que se ocupa de las actividades más mecánicas del agente y envuelve a la inteligencia artificial para que esta pueda interactuar con el resto del programa. Esta parte se ocupa de recibir la información del pool, tales como la información sobre el entorno, el permiso de ofertar en el pool, y los resultados después de cada turno. También se encarga de construir la oferta según las decisiones de la Lista de Reglas, de construir las ofertas de contratos a largo plazo, de crear las nuevas reglas que poblarán la Lista de Reglas y asignarles su puntaje después de cada turno. Aunque no es necesario, desde el punto de vista de diseño, separar la Lista de Reglas del Agente y hacer dos objetos en lugar de uno, se decidió que sería más ordenado crear un objeto que se encargue del manejo y selección de reglas separado del objeto encargado de las otras tareas del agente inteligente.

La Lista de Reglas por su parte recibe la información del entorno del agente que la envuelve, escoge la mejor regla para la situación que se ha presentado y se la presenta al agente para que construya la oferta al pool. La Lista de Reglas contiene todas las reglas que se han creado y estas conforman la base de conocimiento que posee el agente.

La forma como La Lista de Regla funciona se explicará más detalladamente en la siguiente sección.

3.1.2 Del algoritmo de aprendizaje.

El algoritmo de aprendizaje de los agentes de este simulador se basa en el algoritmo de aprendizaje y clasificación de Holland, que previamente fue descrito, con unas pequeñas modificaciones para apresurar el proceso de aprendizaje.

Las reglas del algoritmo de aprendizaje, como las de todos los LCS, tienen una estructura de SI-ENTONCES.

La cláusula SI, que es la que sirve para identificar en que situaciones de debe utilizar esta regla, está compuesta por tres variables ‘char’ de la siguiente forma:

charSubioElPrecio

Esta variable sirve para indicar si subió o bajó el precio con los siguientes valores:

- "0" = Comodín.
- "1" = El precio subió en el último turno.
- "2" = El precio bajó en el último turno.
- "3" = El precio se mantuvo igual.

char charEstadoPrecio

Esta variable indica en que nivel se encuentra el precio. Es decir, si el precio actual se encuentra cercano al precio máximo, al precio mínimo o en un nivel intermedio. Esta variable puede tomar los siguientes valores:

* - "0" = Comodín.
* - "1" = El precio está más cerca del mínimo.
* - "2" = El precio se encuentra en una zona media.
* - "3" = El precio se encuentra más cerca del máximo.

char charVenta

Esta variable indica cuanto ha logrado vender o comprar el agente el turno pasado con los siguientes valores:

- "0" = Comodín.
- "1" = No se vendió nada en el turno pasado.
- "2" = Solo se vendió parte de la producción en el turno pasado.
- "3" = Se vendió toda la producción en el turno pasado.

Se decidió que estas tres variables representarían al entorno dentro de la mente de los agentes inteligentes, ya que son una parte del entorno que puede variar y le sirven al agente para ver la tendencia del mercado, si el precio está subiendo o bajando, o si ha logrado vender toda su producción.

Con estas tres variables los agentes inteligentes miran al entorno. Es posible y bastante fácil hacer que estas variables tomen más valores y hacer que los agentes experimenten el entorno de una manera más compleja, pero esto causaría que la cantidad de reglas posibles sea mayor y por lo tanto que el proceso de aprendizaje sea más lento.

La cláusula de ENTONCES, la cual indica la acción que se debe tomar en caso de que el entorno actual coincida con la cláusula SI, y está compuesto de dos variables char de la siguiente forma:

char charPorcentajeCambio
Esta variable indica el precio que el agente debe ofertar en el pool y siempre es en base a la diferencia entre el precio máximo y el mínimo. El precio mínimo es el mínimo precio que acepta un vendedor para vender su energía, y el precio máximo es el máximo precio que está dispuesto a pagar un comprador por su energía.

- "0 - 9" = del 10% hasta el 100% de la diferencia entre Precio maximo y minimo

char charCambio

Esta variable sirve para complementar la anterior y ayuda a indicar el precio que el agente debe ofertar en el pool.

- "0" = Precio es igual al mínimo.
- "1" = Se utiliza el charPorcentajeCambio para definir el precio.
- "2" = Elegir el precio al azar.

Debido a este sistema solo existe un máximo de once precios que pueden ofertar los agentes que va desde el precio mínimo, el precio mínimo sumado al diez por ciento de la diferencia entre el precio mínimo y el máximo, hasta el precio mínimo sumado al cien por ciento de la diferencia entre el precio mínimo y el máximo que vendría a ser igual al precio máximo. La cláusula de ENTONCES de una regla debe especificar una acción a tomar que se pueda representar como una cadena de caracteres. Por esta razón, el número de acciones que se pueden tomar debe ser matemáticamente discreto, ya que no sería posible representar un número infinito de acciones con un número finito de caracteres. Ya que el simulador está diseñado para ser una herramienta para simular tendencias y no una de precisión, se decidió que once precios eran suficientes.

El precio mínimo y el máximo sirven como límites. No tendría sentido para un comprador ofertar menos dinero de lo que está dispuesto a recibir un vendedor, ya que el vendedor estaría perdiendo dinero y nunca aceptaría. Del mismo modo, no tendría sentido para un vendedor ofertar su producción a un precio mayor al que están dispuestos a pagar

los compradores. Es posible que el precio mínimo sea igual al precio máximo, lo que causaría que la diferencia entre ellos sea cero y que solo exista un precio posible durante toda la simulación. Aunque es posible realizar esta simulación que daría resultados bastante predecibles, es responsabilidad del usuario no ejecutar simulaciones como esta.

Todos los agentes empiezan con una sola regla que nunca es reemplazada y que contiene la siguiente información:

SI:

charSubioElPrecio = 0.

charEstadoPrecio = 0.

charVenta = 0.

ENTONCES:

charPorcentajeCambio = 0.

charCambio = 2.

Como se puede apreciar, todos los valores de la cláusula de SI son comodines. De esta forma esta regla puede ser utilizada en cualquier ocasión, haciendo posible que todos los agentes por lo menos siempre tengan una regla que puedan utilizar en todo momento. Naturalmente, esta regla siempre es evitada si se tiene una regla más específica; es decir, con menos comodines.

La cláusula de ENTONCES le indica al agente que escoja un precio al azar para enviarlo al pool. El precio escogido es solo parcialmente escogido al azar, ya que los precios posibles están limitados por la personalidad del agente de la siguiente forma:

Conservador: Solo pone precios desde el mínimo hasta mínimo más 20% de la diferencia.
Moderado: Solo pone precios desde el mínimo más 30% hasta mínimo más 60% de la diferencia.
Arriesgado: Solo pone precios desde el mínimo más 70% de la diferencia hasta el precio máximo.

La personalidad solo afecta la toma de decisiones cuando se le pide al agente que escoja un precio al azar. En las reglas específicas, el precio que debe ofertar siempre está indicado explícitamente.

Esta regla con que comienzan todos los agentes es la única regla con comodines. El algoritmo original de Holland permitía que se creen nuevas reglas con comodines, y se intentó aplicar esa idea a este simulador. Después de varias pruebas se pudo notar que las reglas con comodines casi nunca eran utilizadas, ya que el mismo algoritmo daba prioridad a las reglas sin comodines. Ya que no aportaban mucha utilidad a la simulación, se prefirió evitar que se creen nuevas reglas con comodines dejando a la regla original como la única regla que se puede utilizar en más de una situación. La regla original también es la única que indica tomar una acción al azar, como ya se dijo previamente.
Uno de los fines de este simulador es verificar que los agentes inteligente que hemos

diseñado son capaces de aprender a actuar según la situación que se les presente. Por lo tanto, crear reglas cuyos resultados dependan completamente de la suerte no ayuda a lograr este propósito.

Al final de cada turno se crean dos reglas; estas reglas son:

- Una regla basada en lo que aconteció en el turno anterior. La cláusula de SI es el estado del entorno en el turno que acaba de terminar. La cláusula de ENTONCES indica al agente que debe ofertar el precio del pool del turno que acaba de terminar. De este modo si el estado del entorno del turno pasado se vuelve a repetir en algún momento de la simulación, el agente ofertará el precio del pool de ese turno.

- Una regla creada al azar para que los agentes experimenten nuevas posibilidades. En este caso todos los valores de SI y ENTONCES son escogidos al azar.

La Lista de Reglas solo puede tener un número limitado de reglas. Este número se ha fijado en cien. Cuando se llega a cien reglas la Lista de Reglas escoge la peor regla, y la reemplaza por la nueva regla. La peor regla se escoge basado en dos puntos: Primero se verifica si la regla se ha utilizado. Es posible que existan reglas para una situación del entorno que nunca se da por algún motivo. Si han pasado varios turnos desde que se creó esta regla y no se ha utilizado, entonces será reemplazada. En segundo lugar, si no se ha

encontrado una regla que nunca se ha utilizado a pesar de llevar bastante tiempo en la lista, se escogerá la regla con el peor puntaje y se la reemplazará con una nueva.

A continuación se explicará como los agentes elijen le regla que utilizarán en cada ocasión.

En cada turno los agentes reciben la información del estado del entorno. Esta información viene en la forma de las tres variables previamente mencionadas: charSubioElPrecio, charEstadoPrecio, y charVenta. El agente compara esta información con todas las reglas que tenga en su lista de reglas y separa las reglas que hagan juego en otra lista. El agente revisa la nueva lista y separa las reglas que tengan menos comodines. En nuestro caso, solo la regla original tiene comodines. Una vez que el agente solo tiene reglas específicas elige la regla con mejor puntaje de las que queden. La regla elegida le indicará al agente que precio debe ofertar en el pool para comprar o vender.

Al final de cada turno los agentes serán avisados del resultado del pool y si lograron comprar o vender energía exitosamente. En este punto los agentes asignarán puntuación a las reglas. La puntuación difiere si el agente es comprador o vendedor, ya que no tienen las mismas metas.

Para los agentes compradores, sus reglas tienen dos puntuaciones. La primera y más importante es la puntuación basada en que tan buena es esta regla para conseguir energía eléctrica. Si la regla logra comprar toda la energía que necesita el comprador, recibe una

puntuación de 100, si logra comprar solo una parte entonces recibe una puntuación equivalente porcentaje de energía que esa regla ha logrado conseguir, y si la regla ha conseguido absolutamente nada, entonces recibe una puntuación de cero. El puntaje recibido se promedia con el puntaje que tenía anteriormente la regla para crear el nuevo puntaje. Le segunda puntuación es la puntuación basada en el precio que la regla indica se debe ofertar y básicamente el puntaje es igual al precio. Esta puntuación solo se mide una vez cuando se crea la regla y nunca cambia. El principal interés de los compradores es conseguir toda la energía que necesitan, por lo tanto lo primero en que se fijan es en el primer puntaje, el que está basado en la cantidad de energía que la regla logra conseguir. En el caso de que el comprador tenga dos reglas con el primer puntaje idéntico, el comprador se fijará en el segundo puntaje y escogerá la regla que le pida ofertar un precio menor, ya que a los compradores también les interesa ahorrar dinero, pero no a costa de no poder conseguir la energía que necesitan.

Para los agentes vendedores, sus reglas también tienen dos puntajes. El primer puntaje indica cuanta energía logra vender esta regla. Si la regla logra vender toda la producción obtiene una puntuación de 100, si solo logra vender una parte entonces recibe equivalente al porcentaje vendido, y si ha vendido nada entonces el puntaje es cero. Cada turno se promedia el puntaje obtenido en ese turno con el puntaje total para sacar el nuevo puntaje total. El segundo puntaje está basado en el precio. En este caso los productores, a diferencia de los compradores, prefieren precios altos. Este puntaje nunca varía ya que está basado en la cláusula ENTONCES que nunca cambia. Al igual que los compradores, los vendedores se fijarán primero en el primer puntaje, y en el caso de tener dos reglas

con el mismo primer puntaje, compararán los segundos puntajes y escogerán la regla con mayor precio.

Para terminar con esta sección ahora se hablará sobre como la inteligencia artificial maneja los contratos a largo plazo. En sí, la parte de contratos a largo plazo no es manejada por la inteligencia artificial, ya que esta solo se limita a los precios que se ofertan en el pool. Pero, sí se puede decir que las decisiones de la inteligencia artificial afectan a los contratos a largo plazo. Cada agente automáticamente, si es que en la simulación se permiten contratos a largo plazo, le pondrá un precio a su oferta de contrato a largo plazo basado en el precio de la oferta al pool que decidió por medio de su inteligencia artificial. Los compradores le pondrán a su oferta de contrato a largo plazo un precio menor que al precio del pool, ya que la única ventaja que tendrían con un contrato a largo plazo es comprar energía por menos dinero. Los vendedores en cambio intentarán vender energía eléctrica a un mayor precio, y por lo tanto le pondrá un precio mayor que el precio del pool a sus contratos a largo plazo. Para hacer el proceso más sencillo, el precio de las ofertas de contratos a largo plazo son 10% de la diferencia sumado o restado según sea el caso del precio que el agente espera obtener del pool.

Debido a que el manejo de contratos se tornó complicado, se decidió que los agentes tomen turnos para ofertar contratos. En un turno solo los compradores ofrecen contratos, en el siguiente solo los vendedores, y así sucesivamente. Cada agente escoge al azar un agente del otro tipo para hacer una oferta en el turno que le corresponde. El otro agente comparará el precio que se le oferta con el precio que espera obtener en el pool. Si el

precio también le favorece, aceptará la propuesta. Esta oferta de contrato a largo plazo, aunque ha sido aceptada por ambas partes, todavía no se puede poner en vigencia. Existe un filtro que examina todos los contratos aceptados y verifica que el vendedor no venda más energía que la que produce y que el comprador no compre más de lo que necesite. Una vez que el nuevo contrato haya pasado por este filtro se lo considera en vigencia.

3.1.3 Revisión panorámica del simulador.

En esta sección no nos vamos a adentrar dentro del funcionamiento de las formas y ventanas del simulador. Estos componentes son una parte importante del programa, pero su importancia es solo relevante a la interacción con el usuario y no al fondo de la tesis. Basta con decir que las formas crean las instancias de los objetos más importantes de la simulación tales como el objeto Pool, la lista de Agentes, la lista de Contratos y los agentes.

El objeto pool es sin lugar a duda la clase y el objeto más importante de todo el simulador. Este objeto contiene la lista de agentes y la lista de contratos, que a su vez contienen a los agentes y a los contratos. El objeto pool realiza las siguientes acciones:

- Contiene la información sobre el tipo de simulación que se está ejecutando.
- Inicia cada turno, le indica a los agentes que envíen sus ofertas y finaliza cada turno.

- Guarda el número de turnos que se han realizado y finaliza la simulación cuando se han ejecutado el número de turnos deseados.
- Calcula el precio de la energía eléctrica en el pool cada turno usando los mecanismos de subasta que se explicarán más adelante.
- Acepta o rechaza las ofertas hechas al pool por parte de los agentes.
- Escribe los resultados y la información de la simulación en la base de datos.

El objeto Lista de Agentes guarda dos listas, una de agentes compradores y otra de vendedores. Este objeto ayuda al objeto Pool a comunicarse con los agentes y también calcula los precios máximo y mínimo de la simulación.

El objeto lista de contratos contiene todos los contratos vigentes que se han creado durante la simulación y se encarga de actualizar la información de estos y sirve de filtro para evitar que por medio de los contratos los agentes compren o vendan más de lo que puedan o necesiten.

Mecanismos de Subasta.

Una de las partes importantes para que el simulador funcione correctamente el la programación de los diferentes métodos de subastas con las que puede funcionar el mercado eléctrico. Los dos modos a tomar en cuenta son las subastas donde los compradores participan, y las subastas donde no participan los compradores, ya que la

subasta funcionando por costo o por precio solo afecta a la toma de decisiones de vendedores y no al mecanismo por el cual se resuelve el precio de mercado.

- **Sin Participación de los Compradores (Mercado Asimétrico).**

En un mercado asimétrico, los surtidores presentan sus ofertas, y el operador de mercado las organiza empezando desde la que tiene el precio mas bajo hacia el precio mas alto. Los consumidores revelan sus necesidades para establecer la demanda. Una vez que el operador de mercado sabe la demanda, acepta las ofertas de los surtidores empezando por la más baja y acepta todas las que sean necesarias para satisfacer la demanda. El precio de mercado – que se le pagará a todos los surtidores aceptados – es el precio de la última oferta aceptada (que tiene el precio más alto.)

La programación de este mecanismo de mercado no es nada complicado, ya que lo único que hay que hacer es poner en orden las ofertas de los vendedores, e ir sumando la producción de energía hasta llegar a un número igual o mayor que la necesidad total de todos los consumidores.

- **Con Participación de los Compradores (Mercado Simétrico)**

En un mercado simétrico, los vendedores y compradores presentan ofertas. El operador de mercado ordena las ofertas. Ofertas de vendedores empiezan con los precios más bajos y suben hasta los más altos. Las ofertas de compradores empiezan con los

precios más altos y bajan hasta los precios más bajos. Luego, las ofertas propuestas forman la curva escalonada de oferta y demanda, y el punto en que se interceptan determina el precio de mercado. Las ofertas de cada vendedor con precios más bajos que el precio de mercado establecido y las ofertas de cada comprador con precios más altos son aceptadas.

La programación de este mecanismo de mercado presenta ciertos retos, ya que los lenguajes de programación no vienen provistos con funciones automáticas para crear curvas (escalonadas o de cualquier otro tipo) basadas en puntos dados en un plano de dos dimensiones. Por lo tanto es necesario programar funciones para esto. Vamos a asumir que las ofertas ya han sido ordenadas de manera adecuada por una función anterior.

Lo primero que hay que determinar es el número de puntos que generará la demanda, y el número de puntos que generará la oferta para poder representar una curva escalonada en un plano de dos dimensiones. El número de puntos siempre es igual al número de ofertas multiplicado por dos más uno; es decir Numero de Puntos = (Numero de Ofertas x 2) + 1. Se debe sacar un número de puntos basado en el número de ofertas de los vendedores, y un número de puntos basado en el número de ofertas de compradores. Con estos dos números creamos dos arrays de puntos, uno para puntos de compradores y otros para los puntos de vendedores, cuyo tamaño sea el número de puntos calculado.

Ahora poblamos los arrays con puntos, para lo cual es necesario crear los puntos. Para crear los puntos y poblar el array de puntos basados en ofertas de vendedores. Cada

oferta genera dos puntos, y se necesita además un punto más para cerrar la curva escalonada. Por eso es que se necesita un array con capacidad para (Numero de Ofertas x 2) + 1 puntos.

Para crear los puntos de las ofertas de vendedores se utilizan las siguientes instrucciones: El primer punto generado por una oferta tiene por valor X siempre el precio de venta, y por valor Y el valor acumulado de cantidad ofertada empezando desde cero que se tenga sin sumarle la cantidad ofertada en la oferta actual. El segundo punto generado tiene el mismo valor 'X', y 'Y' es el valor acumulado pero ahora sumándole la cantidad ofertada en la oferta actual. El último punto tiene un valor X infinito y el valor de Y es el mismo del punto anterior. Esto es más sencillo de explicar con un ejemplo:

Digamos que tenemos una subasta donde solo se han presentado dos ofertas de parte de los vendedores.

Oferta 1: Precio 1 Cantidad 1.

Oferta 2: Precio 3 Cantidad 1.

La primera oferta que tenemos vende 1 unidad a un precio de 1 dólar. Esto es solo un ejemplo para ilustrar como se crean los puntos, el precio y la cantidad no reflejan en nada los precios y producciones reales del mercado eléctrico. El primer punto (X, Y) creado sería (1, 0) donde 'X' es el precio y 'Y' la cantidad ofertada acumulada empezando desde cero. El segundo punto sería (1, 1) ya que 'X' siempre es igual al precio de venta y 'Y' es igual al 'Y' del primer punto creado basado en la oferta mas la

cantidad ofertada. Ahora tomemos la segunda oferta que vende 1 unidad a un precio de 3 dólares. El primer punto que sacamos de esta oferta – que vendría a ser el tercer punto en total – es (3, 1), ya que 3 es el precio ofertado y 1 es la cantidad acumulada que teníamos en el punto anterior. El segundo punto sería (3, 2) ya que ahora si podemos sumar la cantidad ofertada a la cantidad acumulada. El último punto que hay que crear no está basado en ninguna oferta; es un punto extra que se crea para terminar la curva. Este punto sería (∞, 2) ya que la curva se extiende hacia el infinito en la 'X'.

Al final obtenemos el siguiente array de puntos:

Punto #	X	Y
0	1	0
1	1	1
2	3	1
3	3	2
4	∞	2

NOTA: Son cinco puntos, pero en la mayoría de los lenguajes los arrays empiezan a contar desde cero.

Para crear los puntos de las ofertas de los agentes compradores las instrucciones son prácticamente las mismas, excepto que en el último punto el valor de 'X' siempre se debe poner como Cero. Como ejemplo utilizaremos dos sencillas ofertas:

Oferta 1: Precio 2 Cantidad 2.

Oferta 2: Precio 1 Cantidad 1.

NOTA: Recuerde que las ofertas de los compradores se ordenan de mayor a menor.

La primera oferta nos dice que el comprador necesita adquirir 2 unidades y propone pagar un precio de 2 dólares por unidad. La segunda que tiene una necesidad de 1 unidad y propone pagar 1 dólar por unidad.

El array de puntos resultante sería el siguiente:

Punto #	X	Y
0	2	0
1	2	2
2	1	2
3	1	3
4	0	3

Graficando las dos curvas obtenemos lo siguiente:

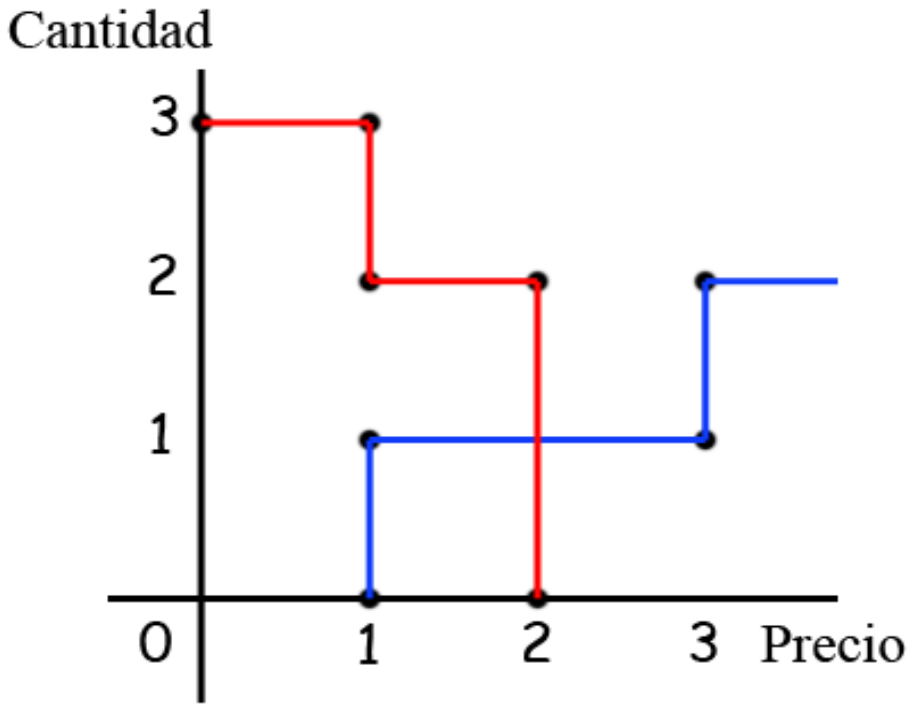


Ilustración 6. Curvas escalonadas de oferta y demanda.

Donde la curva escalonada roja es la curva de las ofertas de compradores y la curva azul la de los vendedores. Como podemos apreciar el precio del mercado es dos, ya que ahí se tocan.

Escribir el código que identifica donde se entrelazan las curvas escalonadas ofrece sus propios retos. Para el programa ambas curvas son solo una colección de puntos, y no es capaz de graficarlas y ver donde se tocan las líneas. Lo que el programa debe hacer es tomar los puntos en grupos de dos en dos de un grupo de puntos y compararlos con grupos de dos en dos del otro grupo. Esto se debe a que las curvas escalonadas están compuestas de líneas rectas verticales y horizontales. Para representar estas líneas solo necesitamos dos puntos. Para ahorrar tiempo solo se deben comparar líneas horizontales

con verticales y viceversa, ya que, aunque es posible que dos líneas verticales o dos líneas horizontales se entrelacen, en esos casos también se entrelazarían la vertical con la horizontal.

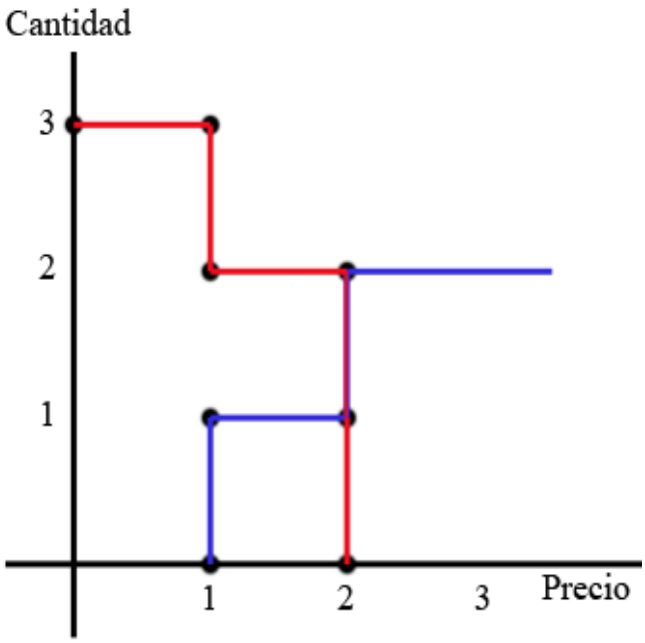


Ilustración 7. Curvas escalonadas de oferta y demanda entrelazadas verticalmente.

Como se puede apreciar, la línea entre los puntos B y C es parte de las dos curvas escalonadas, pero no es necesario comparar la línea B-C de ambas curvas, ya que B-C se entrelaza con A-B y con C-D que son horizontales.

Para comparar se deben tomar dos puntos de cada colección de puntos. Si el primer punto que se toma está en l posición cero o en un número par, entonces este y el punto que le sigue son parte de una recta vertical; por lo cual solo se debe comparar con líneas cuyo primer punto sea impar, ya que estas rectas son horizontales, y viceversa.

Ahora tenemos cuatro puntos. Los puntos de la primera línea (X1, Y1), (X2, Y2) y los puntos de la otra línea (X3, Y3) y (X4, Y4). En el caso de que la primera línea sea vertical, entonces X1 es igual a X2, y la segunda línea es horizontal, por lo cual Y3 es igual a Y4. Hay que revisar si X1 es igual o está entre X3 y X4, y si Y3 es igual o está entre Y1 y Y2. Si estas dos condiciones se cumplen, entonces X1 es el precio que dicta la oferta y la demanda. Si en cambio la primera línea fuera horizontal, entonces Y1 sería igual a Y2, y X3 sería igual a X4 para la segunda línea. Aquí entonces habría que revisar que Y1 sea igual o esté entre Y3 y Y4, y que X3 sea igual o esté entre X1 y X2. El precio es siempre el valor X de la recta vertical que se está comparando.

Utilicemos los puntos ya creados para el ejemplo:

Puntos de Vendedores.

Punto #	X	Y
0	1	0
1	1	1
2	3	1
3	3	2
4	∞	2

Puntos de Compradores

Punto #	X	Y
0	2	0

1	2	2
2	1	2
3	1	3
4	0	3

Empezamos tomando dos puntos de vendedores, el punto 0 y el punto 1 para obtener la recta (1, 0) (1, 1) y tomamos dos puntos de compradores, el 1 y el 2, para obtener la recta (2, 2) (1, 2). Ignoramos la recta de los puntos 0 y 1 de compradores porque es paralela a la primera recta de vendedores. Como la recta de vendedores es vertical y la recta de compradores es horizontal, verificamos si X1 está entre X3 y X4. X1 = 1, X3 = 2 y X4=1. Esta condición se cumple. Después verificamos si Y3 está entre Y1 y Y2. Y3 = 2, Y1 = 0 y Y2 = 1. Esta condición no se cumple, por lo tanto estas dos rectas no se entrelazan. Así vamos comparando todas las rectas hasta que llegamos a las rectas que se cruzan, que son los puntos 1 (1, 1) y 2 (3, 1) de vendedores y 0 (2, 0) y 1 (2, 2) de compradores ya que Y1 (1) está entre 0 y 2, y X3 (2) está entre 1 y 3. En este caso, como ya se comprobó en el gráfico, el valor a pagar por la energía eléctrica es igual 2, es decir el valor X de la recta vertical que se esté comparando en ese momento.

También podríamos tener el caso donde dos líneas horizontales se entrelazan perfectamente.

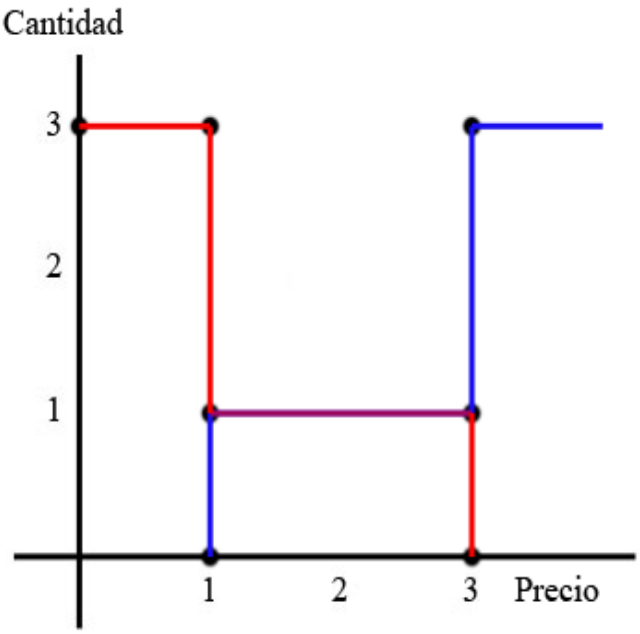


Ilustración 8. Curvas escalonadas de oferta y demanda entrelazadas horizontalmente.

Un caso como este se puede dar por ejemplo con las siguientes ofertas, que nos darían las curvas dibujadas:

Ofertas de Vendedores.

Oferta A: Precio 1, Cantidad 1.

Oferta B: Precio 3, Cantidad 2.

Ofertas de Compradores.

Oferta C: Precio 3, Cantidad 1.

Oferta D: Precio 1, Cantidad 2.

Como se puede apreciar en el gráfico, tenemos varias opciones para tomar como precio. Como la línea horizontal donde se tocan las curvas va desde uno a tres, cualquier punto dentro de esta recta se puede tomar como precio de mercado. Los dos únicos puntos que nos interesan son los extremos. El precio puede ser igual a uno o a tres.

Si fijamos el precio igual a uno, entonces solo se aceptarían las ofertas A, C, y D. Pero A no puede satisfacer toda la demanda de C y D.

Si fijamos el precio igual a 3, entonces solo se aceptan las ofertas A, B y C. Pero C no necesita tanta energía eléctrica y solo le basta con comprar lo que ofrece A.

Con esto podemos concluir que no debemos aceptar las ofertas ciegamente aunque el precio cumpla con las reglas establecidas, y que podemos tener un comprador con una demanda que no podemos satisfacer, o un vendedor con nadie que le compre su producción. Lo que se necesita es poner otro filtro después de este paso, un filtro que funcione parecido a como funciona el mercado Asimétrico. Es decir, que primero utilizamos las curvas escalonadas para establecer el precio de mercado, y luego seleccionamos las ofertas que cumplan con el precio establecido. Luego pasamos esas ofertas a un segundo filtro y tomamos todas las ofertas de vendedores que sean necesarias para satisfacer la demanda, o solo las ofertas de compradores que la oferta pueda satisfacer.

El único caso en que las dos curvas escalonadas no se crucen es cuando todas las ofertas de los compradores tienen precios menores al precio mas bajo ofertado por los vendedores.

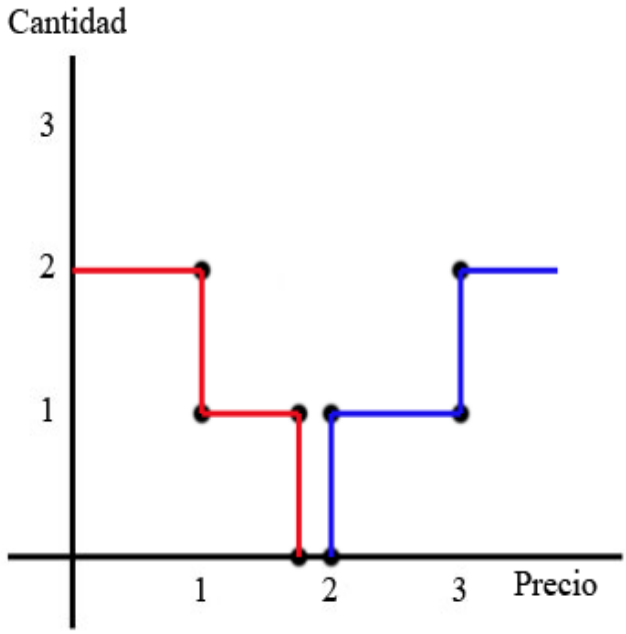


Ilustración 9. Curvas escalonadas que no se tocan.

En este caso todos los puntos de las ofertas de compradores están a la izquierda, y todos los puntos de ofertas de vendedores a la derecha. Esto se debe a que todas las ofertas de compradores tienen un precio menor a la oferta con menor precio de los vendedores.

3.2 Diseño estructural

En esta sección se presenta el diseño estructural de POLISIMEM. Se describirás las clases de objetos creadas para este programa, primero explicando en forma general la actividad que realizan en el programa, y luego se describirá brevemente su relación con otras clases por medio de dos funciones: La función ‘Contiene a’ que especifica las clases que son contenidas por la clase que se está explicando; y la función ‘Usa a’ que especifica la forma en que la clase utiliza a las otras clases para cumplir sus propósitos y cómo las utiliza.

3.2.1 Clases creadas para el programa.

Clases de Formas de Ventanas.

Clase: frmMain

La clase frmMain es la clase de la ventana principal del programa. En esta ventana el usuario elige las opciones para ejecutar el tipo de simulación que desea. Se encarga de instanciar al objeto Pool, la lista de agentes y la lista de contratos para que pueda funcionar la simulación. También llama momentáneamente a las ventanas para crear a los agentes y con esta ventana se puede elegir la base de datos a utilizar y borrar los datos de esta para poder realizar otra simulación.

Clase: frmAdvertencia

La clase frmAdvertencia es la clase de las ventanas de advertencia que aparecen en la pantalla cuando en el programa ha ocurrido un error, o el usuario ha realizado algo indebido. El mensaje que muestra esta ventana puede cambiar dependiendo de la situación, pero son todas instancias de una sola clase. Varias de las otras clases utilizan a esta para advertir sobre errores o informar al usuario.

Clase: frmComprador

La clase frmComprador es la clase de las ventanas donde se especifican los datos de los compradores que participarán en la simulación. Estas ventanas solo aparecen sí se ha escogido la opción de permitir a los compradores participar en el pool.

Clase: frmCompradorSimple

La clase frmComprador es la clase de las ventanas donde se especifican los datos de los compradores que participarán en la simulación. Estas ventanas solo aparecen si no se ha escogido la opción de permitir a los compradores participar en el pool. Es bastante parecida a la clase frmComprador, pero tiene menos opciones para la creación de compradores.

Clase: frmVendedor

La clase frmVendedor es la clase de las ventanas donde se especifican los datos de los vendedores que participarán en la simulación.

Clase: frmLector

Esta clase es la clase de la ventana que muestra el reporte con los datos generados por la simulación. Funciona con CrystalReports de Microsoft. Se pueden ver los datos del pool y de los contratos a largo plazo.

Clases.

Clase: clAgente

Esta clase es la clase padre de los compradores y los vendedores. Esta clase nunca se instancia directamente y solo sirve para que compradores y vendedores hereden ciertas variables y métodos en común.

Clase: clComprador

Esta clase es la pieza principal de los agentes compradores y maneja todo su funcionamiento, aunque relega ciertas funciones de inteligencia artificial a la clase clLista_de_Reglas. Hereda métodos y variables de la clase clAgente.

Clase: clContrato.

Esta clase representa los contratos a largo plazo que los compradores y vendedores realizan entre ellos.

Clase: clHilo

Esta clase sirve para crear hilos de ejecución, que en el lenguaje JAVA se manejan como si cada hilo fuera un objeto.

Clase: clLista_de_Agentes

Esta clase contiene la lista de compradores y de vendedores.

Clase: clLista_de_Contratos

Esta clase guarda los contratos generados durante la simulación.

Clase: clLista_de_Reglas

Esta clase guarda las reglas generadas por los agentes y realiza la mayoría del trabajo de inteligencia artificial.

Clase: clOferta

Esta clase representa las ofertas que los agentes envían al pool. Le reporta al agente que lo creó la cantidad de energía vendida o comprada en la subasta.

Clase: clPool

Esta clase fácilmente se podría considerar como la clase principal del programa. Esta clase controla no solo las subastas del pool; también se encarga de informar a los agentes turno por turno cuando tienen que ofertar y los resultados de las ofertas y se encarga de pasar los resultados a la base de datos.

Clase: clPunto

Esta clase representa un punto en el plano cartesiano que se utiliza para determinar el valor de la energía eléctrica en el pool cuando el mercado es simétrico. Esta clase no tiene métodos, por lo cual, de acuerdo con el paradigma de programación orientada a objetos, los punto no deberían ser una clase. La clase se creó de todas formas debido a que en el lenguaje JAVA la única información que se puede pasar a un método un objeto por referencia y no solamente por valor son los objetos. Como es necesario pasar el punto por referencia, no hubo otra opción más que crear esta clase.

Clase: clReglaAI

Esta clase representa las reglas que utiliza cada agente para decidir que acción tomar en cada situación. Contiene los métodos que clLista_de_Reglas llama para ajustar su propio puntaje y para definir si coincide con la situación del entorno.

Clase: clVendedor

Esta clase es la pieza principal de los agentes vendedores y maneja todo su funcionamiento, aunque relega ciertas funciones de inteligencia artificial a la clase clLista_de_Reglas. Hereda métodos y variables de la clase clAgente.

3.2.2 Relaciones entre Clases.

Clases de Formas de Ventanas.

Clase: frmMain

Contiene a:

objLista_Agentes: Es una instancia de la clase clLista_de_Agentes.

objLista_de_Contratos: Es una instancia de la clase clLista_de_Contratos.

objPool: Es una instancia de la clase clPool.

Usa a:

frmComprador, frmCompradorSimple y frmVendedor: Llama a las formas frmComprador, frmCompradorSimple y frmVendedor para que el usuario pueda crear a los agentes que participarán en la simulación.

frmLector: Llama a la forma frmLector para que el usuario pueda ver los resultados de una simulación.

clLista_Agentes, clLista_de_Contratos, clPool : Crea instancias de los objetos clLista_Agentes, clLista_de_Contratos, clPool.

clPool: Le pasa a clPool las instancias de clLista_de_Contratos y clLista_de_Agentes y le indica que empieze la simulación.

Clase: frmComprador

Contiene a:

objLista_Agentes: Es una instancia de la clase clLista_de_Agentes.

Usa a:

clComprador: Crea instancias de clComprador y las pone directamente en el objeto clLista_de_Agentes.

Clase: frmCompradorSimple

Contiene a:

objLista_Agentes: Es una instancia de la clase clLista_de_Agentes.

Usa a:

clComprador: Crea instancias de clComprador y las pone directamente en el objeto clLista_de_Agentes.

Clase: frmVendedor

Contiene a:

objLista_Agentes: Es una instancia de la clase clLista_de_Agentes.

Usa a:

clVendedor: Crea instancias de clVendedor y las pone directamente en el objeto clLista_de_Agentes.

Clases.

Clase: clAgente

Contiene a:

objOferta: Instancia de la clase clOferta.

objLista_de_Reglas: Instancia de clLista_de_Reglas que maneja las reglas para tomar decisiones.

objRegla: Instancia de clRegla que guarda la última regla utilizada.

arrayContratosRecibidos[]: Array que contiene los contratos no aprobados que han enviado los otros agentes.

arrayContratosPreAprobados[]: Array que contiene los contratos aprobados.

arrayContratosVigentes[]: Array que contiene los contratos vigentes.

Usa a:

clContratos : Inicializa los arrays de objetos clContratos que se utilizan para manejar los contratos a largo plazo. Esta clase también contiene varios métodos que sirven para el mantenimiento de los objetos contratos, como métodos para insertar los contratos a los diferentes arrays, actualización del estado de los contratos cada turno, limpieza de los arrays, verificación de que los contratos no vendan o compren más de lo posible, y cancelación de contratos no aceptados.

clRegla: Crea la regla con valores al azar.

Clase: clComprador

Contiene a:

Esta clase contiene lo que contiene clAgente.

Usa a:

clLista_de_Reglas, clOferta: Utilizando a clLista_de_Reglas, crea la oferta que es enviada al objeto clPool para la subasta cada turno y las ofertas de contratos que le hace y envía a otros agentes.

clRegla: Con los resultados que le envía clPool, esta clase crea una nueva regla basada en lo acontecido en la subasta del turno anterior y tambien crea una regla al azar utilizando

métodos que le pertenecen a la clase padre clAgente. También califica a las reglas después de utilizarlas de acuerdo al resultado obtenido con ellas.

clContrato: Recibe ofertas de contratos a largo plazo de otros agentes y verifica que ofertas debe aceptar y cuales no.

Clase: clContrato.

Contiene a:

objComprador: Instancia de clComprador que contiene al comprador que compra energía en este contrato.

objVendedor: Instancia de clVendedor que contiene al vendedor que vende energía en este contrato

Clase: clHilo

Contiene a:

objPool: Instancia de la clase clPool.

objAgente: Puede recibir una instancia de comprador o de vendedor. Dependiendo de que agente trabaja con el hilo.

Clase: clLista_de_Agentes

Contiene a:

arrayListaCompradores[]: Este array guarda la lista de Compradores.

arrayListaVendedores[]: Este array guarda la lista de Vendedores.

Usa a:

clComprador, clVendedor: Revisando los compradores y vendedores que contiene, encuentra los precios máximo y mínimo para la simulación.

Clase: clLista_de_Contratos

Contiene a:

arrayListaContratosPreAprobados[]: Este array guarda la lista de contratos preaprobados.

arrayListaContratosVigentes[]: Este array guarda la lista de contratos vigentes.

Usa a:

clContrato: Mantiene su propia lista de contratos aprobados de todos los agentes, a diferencia de la clase clAgente que solo contiene los contratos pertinentes a ese agente en particular. Sirve como última barrera para verificar que los contratos no compren o vendan más de lo posible.

Clase: clLista_de_Reglas

Contiene a:

arrayListaReglas[]: Lista de Reglas que guarda cada agente.

arrayListaReglasMatch[]: Lista de reglas que coinciden con la situación del entorno actual.

arrayListaReglasExactas[]: Lista de reglas que tienen menos comodines y coinciden con la situación del entorno actual.

Usa a:

clRegla: Recibe las reglas creadas por el agente que lo contiene y se encarga de eliminar las peores reglas y reemplazarlas por nuevas. Recibe la situación actual en que se encuentra el entorno y retorna la mejor regla para usarse en ese caso. Recibe los puntajes asignados por el agente para la regla que utilizó y lo promedia con el puntaje actual de la regla para obtener el nuevo puntaje.

Clase: clOferta

Esta clase representa las ofertas que los agentes envían al pool. Le reporta al agente que lo creó la cantidad de energía vendida o comprada en la subasta.

Clase: clPool

Contiene a:

objLista_Agentes: Instancia de clLista_de_Agentes que guarda las listas de agentes.

objLista_de_Contratos: Instancia de clLista_de_Contratos que guarda la lista de contratos vigentes.

arrayOfertasCompradores[]: array que guarda las ofertas generadas por los Compradores.

arrayOfertasVendedores[]: array que guarda las ofertas generadas por los vendedores.

arrayOfertasVendedoresOrdenada[]: array de las ofertas de vendedores ordenadas por precio de menor(0) a mayor(n).

arrayOfertasCompradoresOrdenada[]: array de las ofertas de vendedores ordenadas por precio de mayor(0) a menor(n).

arrayOfertasCompradoresAprobadas[]: array de ofertas que tienen un precio igual o mayor al precio establecido, pero no han sido aceptadas todavía.

arrayOfertasVendedoresAprobadas[]: array de ofertas que tienen un precio igual o menor al precio establecido, pero no han sido aceptadas todavía.

clPunto: la clase clPool crea varios objetos clPunto para calcular los precios del mercado y los descarta una vez que haya acabado.

Usa a:

clAgente: Le indica a los objetos agentes que un nuevo turno ha comenzado y recibe sus ofertas para la subasta. Después le informa a los agentes el resultado de cada subasta.

clOferta: Recibe todas las ofertas y calcula el precio de la energía eléctrica cada turno dependiendo de la forma en que el usuario eligió el funcionamiento del mercado.

clPunto: Crea los puntos con los cuales calcula el precio de la energía en el mercado spot.

Clase: clVendedor

Contiene a:

Esta clase contiene lo que contiene clAgente.

Usa a:

clOferta, clLista_de_Reglas: Utilizando a clLista_de_Reglas, crea la oferta que es enviada al objeto clPool para la subasta cada turno y las ofertas de contratos que le hace y envía a otros agentes.

clRegla: Con los resultados que le envía clPool, esta clase crea una nueva regla basada en lo acontecido en la subasta del turno anterior y tambien crea una regla al azar utilizando métodos que le pertenecen a la clase padre clAgente. Califica a las reglas después de utilizarlas de acuerdo al resultado obtenido con ellas.

clOferta: Recibe ofertas de contratos a largo plazo de otros agentes y verifica que ofertas debe aceptar y cuales no.

3.3 Las Especificaciones del Diseño.

Para leer más sobre el diseño de este programa, está disponible en la sección de anexos el diagrama de clases del programa, los casos de uso, el diagrama de secuencias, y una explicación de todos los métodos y variables que tiene cada clase.

CAPITULO 4. Implementación.

En este capítulo se justificará por qué se escogió el lenguaje JAVA para la programación del simulador. También Se justificará y explicará el uso de una base de datos dentro del programa, y se detallarán los productos creados, y los requerimientos de hardware y software.

5.1 El lenguaje de programación.

El lenguaje escogido para programar el simulador de mercado eléctrico fue JAVA. A continuación se enumerarán las razones por las cuales se eligió este lenguaje.

Es un lenguaje orientado a objetos.

Una de las razones más importantes es la capacidad y facilidad que tiene el lenguaje JAVA para trabajar en base a objetos. Aunque teóricamente es posible realizar cualquier programa sin necesidad de un lenguaje orientado a objetos, no cabe duda de que es mucho más sencillo trabajar con objetos para diseñar y programar un simulador que trabaja con agentes independientes. El paradigma de sistemas multiagentes y la programación basada en objetos, como ya se ha tratado en un capítulo anterior, tienen bastantes puntos en común. Es muy sencillo imaginar cada agente como un objeto, o como un grupo de objetos de los cuales se pueden crear múltiples y diferentes instancias, a diferencia de pensar en ellos de cualquier otra forma.

El lenguaje JAVA no es el único lenguaje orientado a objetos que existe, pero ofrece otras ventajas sobre otros lenguajes.

Permite crear hilos de ejecución.

El lenguaje JAVA permite que diferentes partes de código se ejecuten al mismo tiempo gracias a la creación de hilos de ejecución. De esta forma JAVA les permite a los agentes inteligentes que participan en la simulación tomar decisiones sin tener que tomar turnos o esperar que otro agente acabe de decidirse. Esto aumenta la independencia de cada agente de los otros participantes de la simulación.

Funciona en varias plataformas.

Debido a que JAVA es un lenguaje interpretado y no compilado, es muy sencillo llevar cualquier programa en JAVA a múltiples plataformas. Lo único que se necesita para ejecutar un programa de JAVA es instalar la última versión del JAVA runtime enviroment, que interpreta el lenguaje al andar para que la computadora pueda ejecutarlo.

Aunque esto no es necesario para realizar el programa, es una ventaja tentadora que permite la fácil transportación y demostración del programa una vez acabado.

5.2 Problemas con el uso de la base de datos.

El simulador de mercado eléctrico requiere una base de datos Access para su funcionamiento. Cuando se instala el programa, en la carpeta donde se instala se crea una base de datos de Access llamada bdMercadoEléctrico.mdb. Es necesario tener esta base de datos, o alguna otra con una estructura de datos idéntica, para que el programa funcione correctamente.

Access no es una de las bases de datos más poderosas del mercado, y dista mucho de serlo, pero tiene ciertas ventajas que lo convierten en una herramienta útil para los propósitos del simulador de mercado eléctrico.

Las bases de datos Access son fáciles de transportar, ya que no necesitan un servidor de base de datos al cual tiene que conectarse el programa para guardad sus datos. El programa simplemente guarda sus datos directamente en el archivo. Esto permite que el programa pueda funcionar en cualquier computadora sin necesidad de un servidor de base de datos y lo hace mucho más transportable.

Las bases de datos Access son bastante limitadas en lo que respecta a la cantidad de información que pueden guardar. El simulador de mercado eléctrico genera muchos registros, pero hasta ahora ha llegado al límite de lo que permite Access, ya que, aunque es bajo comparado con los límites de otras bases de datos, es más que suficiente para guardar los datos generados por nuestro programa.

Otra ventaja que ofrece Access es que el programa Office es bastante popular y la gran mayoría de los usuarios lo tienen instalado en sus máquinas, lo cual permite que más personas puedan abrir directamente el archivo y trabajar con la información que contiene.

5.3 Requerimientos de hardware y software.

El único requisito especial para ejecutar el programa es la previa instalación del JAVA runtime enviroment. Este programa es necesario para que interprete el programa en JAVA, ya que JAVA no es lenguaje compilado.

No existen requerimientos de memoria ni de espacio en disco duro importantes, ya que el programa no exige muchos recursos para poder funcionar.

5.4 Productos.

El producto creado es un instalador llamado Setup.msi que se encuentra dentro de la carpeta Setup. Dentro de esta carpeta también se encontrarán un archivo llamado Setup.exe y otro llamado Setup.ini.

Al instalar el programa se creará una carpeta llamada ‘Simulador de Mercado Electrico’ dentro de la carpeta ‘Program Files’ a menos de que se haya indicado lo contrario en el momento de la instalación. Dentro de esta carpeta se instalarán los siguientes archivos: Simulador de Mercado Electrico.exe y bdMercadoElectrico.mdb.

CAPITULO 5. Pruebas

En este capítulo se enumerarán las pruebas realizadas, los resultados de cada una y una discusión sobre los mismos.

5.1 El Entorno Para Probar el Simulador.

Todas las pruebas fueron realizadas en una computadora

Las Pruebas de la uno hasta la siete, que no requerían muchos agentes ni turnos, tomaban aproximadamente cuarenta minutos en correr. Las pruebas restantes con diez agentes cada una y con tres mil turnos tomaban siete horas en correr.

Cada prueba tiene los siguientes parámetros y características:

Tipo de Pool: Este parámetro indica si el mercado spot está basado en costos o en precios. Precios significa que los vendedores de energía pueden ofertar cualquier precio que deseen. Costos en cambio significa que los vendedores no pueden ofertar libremente el precio que desea, y solo pueden ofertar el costo de producción.

Se permite la participación de Compradores en el pool: Indica si los compradores pueden enviar ofertas al pool o no. En el caso de que no se les permita, se limitan solamente a

indicarle al pool la cantidad de energía que necesitan para que el pool pueda calcular el precio de la energía.

Precio Máximo: Es el precio máximo que puede tomar la energía eléctrica. Es la mayor cantidad de dinero que un agente está dispuesto a pagar por la energía eléctrica durante todos los turnos que dure la simulación. No tendría sentido que un vendedor trate de vender energía a un precio que ningún comprador esté dispuesto a pagar.

Se permiten Contratos a Largo Plazo: Indica si los compradores y vendedores pueden realizar contratos a largo plazo entre ellos para la compra y venta de energía en lugar de solo vender y comprar por medio del mercado spot.

Compradores y Vendedores: Aquí se indicarán las características de los diferentes compradores y vendedores que participaron en la simulación.

Las características de los agentes que se ingresan como está indicado en la ventana para ingresar compradores y en la ventana para vendedores.

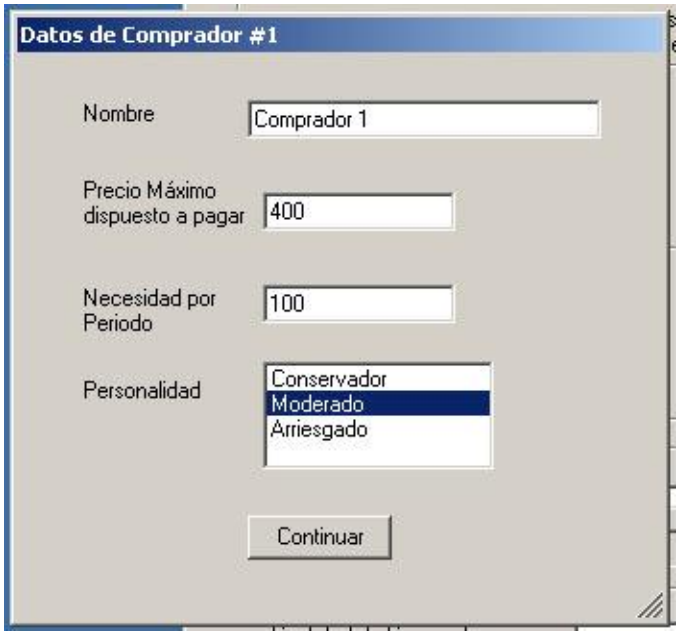


Ilustración 10. Ventana de Datos de Comprador.

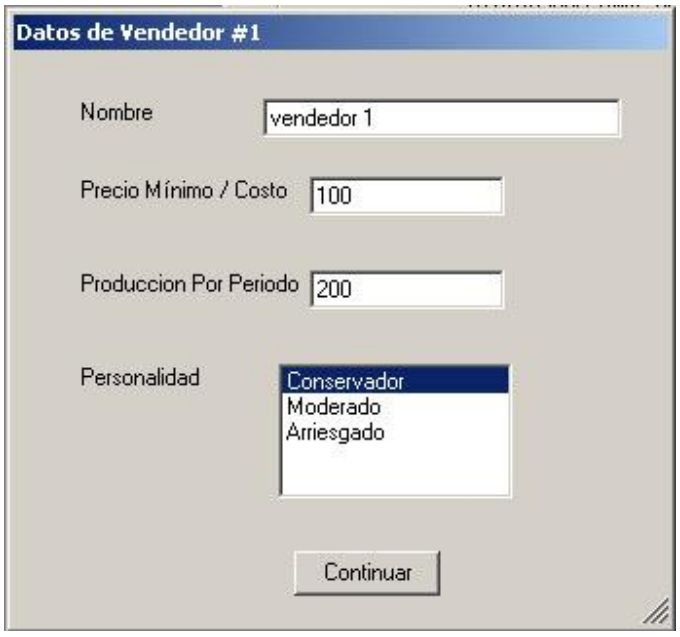


Ilustración 11. Ventana de Datos de Vendedor.

5.2 Resultados

Prueba 1.

En esta primera prueba se ha decidido ejecutar una simulación con las siguientes características:

Tipo de Pool: COSTOS.

Compradores NO participan en el pool.

Precio Máximo: 1000 por unidad.

No se permiten Contratos a largo plazo.

Compradores: Existen cinco compradores en esta simulación. Cada uno de ellos tiene una necesidad de 200 unidades de energía.

Vendedores: Existen cinco vendedores. Cada uno de ellos tiene un costo de 100 por unidad y producen 200 unidades por turno.

Una simulación donde el pool funciona a base de costos, y no se permite la participación de los agentes compradores en el pool, siempre dará el mismo resultado turno tras turno. En esta simulación ni los compradores ni los vendedores de energía eléctrica tiene necesidad de aprender a trabajar en este entorno ya que los vendedores están forzados a ofertar el mismo precio en cada turno y los compradores no pueden participar. Debido a

esto, no es una simulación de mucho interés para los propósitos de la tesis, pero ya que el programa permite la realización de este tipo de simulación se ha incluido.

Resultados.

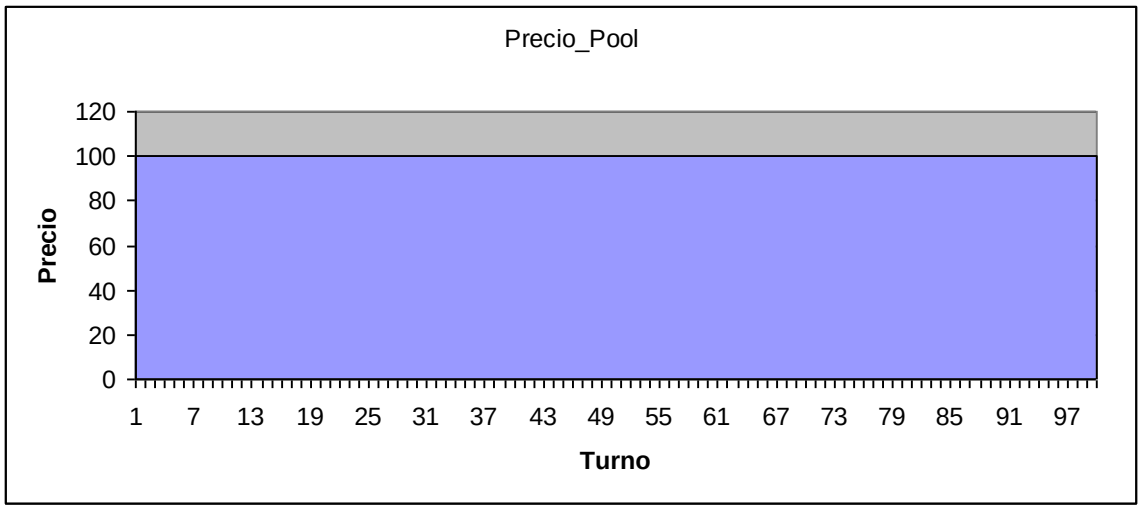


Ilustración 12. Resultados de la Prueba 1.

Precio Promedio: 100.

Desviación Estándar: 0.

Como era de esperarse, el precio del pool es constante en cada turno, ya que en este tipo de simulación el precio no varía.

Conclusiones.

En un pool basado en costos los vendedores se ven forzados a ofertar su energía siempre al mismo precio, y si no se permite la participación de los compradores, entonces el precio de la energía nunca variará.

Prueba 2.

Este simulación tiene las siguientes características:

Tipo de Pool: COSTOS.

Compradores SI participan en el pool.

No se permiten Contratos a largo plazo.

Compradores: Se crearon tres agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 15, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 20, Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 15, Personalidad = Arriesgado.

Vendedores: Se crearon tres agentes vendedores, cada uno con un costo de 10 por unidad y la capacidad de producir 20 unidades de energía cada turno.

Como se puede apreciar, en esta simulación la demanda es menor que la oferta.

Resultados.

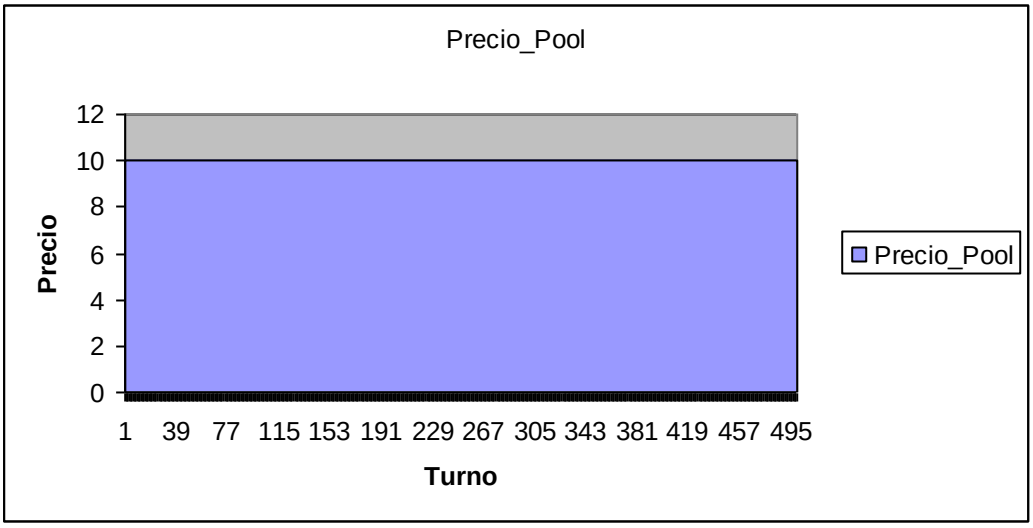


Ilustración 13. Resultados Prueba 2.

Precio Promedio: 10.

Desviación Estándar: 0.

Debido a que los vendedores generan toda la energía que los compradores necesitan y más, los compradores no necesitan pelear por la energía, y los vendedores no pueden ofertan a precios mayores debido a que el pool funciona a base de costos, el precio se asienta en lo mínimo posible.

Conclusiones.

Cuando solo uno de los dos tipos de agentes puede ofertar a su antojo en el pool, a menos que exista una competencia entre ellos por los recursos, el precio tenderá a lo mínimo.

Prueba 3.

Esta simulación tiene las siguientes características:

Tipo de Pool: COSTOS.

Compradores SI participan en el pool.

No se permiten Contratos a largo plazo.

Compradores: Se crearon tres agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 25, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 25, Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 25, Personalidad = Arriesgado.

Vendedores: Se crearon tres agentes vendedores, cada uno con un costo de 10 por unidad y la capacidad de producir 20 unidades de energía cada turno.

Como se puede apreciar, en esta simulación la oferta es menor que la demanda.

Resultado.

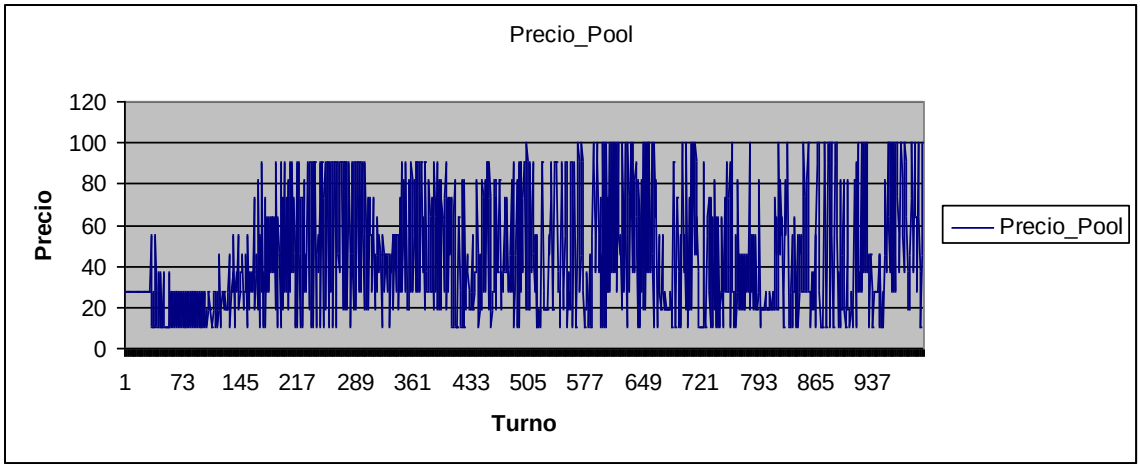


Ilustración 14. Resultados de la Prueba 3.

Precio Promedio: 44.89

Desviación Estándar: 30.13.

Al principio de la simulación, todos los precios eran bajos hasta el turno 39, donde los agentes empezaron a experimentar con precios más altos para conseguir mayor energía.

Conclusión.

Esto demuestra que los agentes compradores son capaces de competir por recursos entre ellos.

Prueba 4.

Esta simulación tiene las siguientes características:

Tipo de Pool: COSTOS.

Compradores SI participan en el pool.

No se permiten Contratos a largo plazo.

Compradores: Se crearon tres agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 20, Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 20, Personalidad = Arriesgado.

Vendedores: Se crearon tres agentes vendedores, cada uno con un costo de 10 por unidad y la capacidad de producir 20 unidades de energía cada turno.

Como se puede apreciar, en esta simulación la oferta es igual a la demanda.

Resultados.

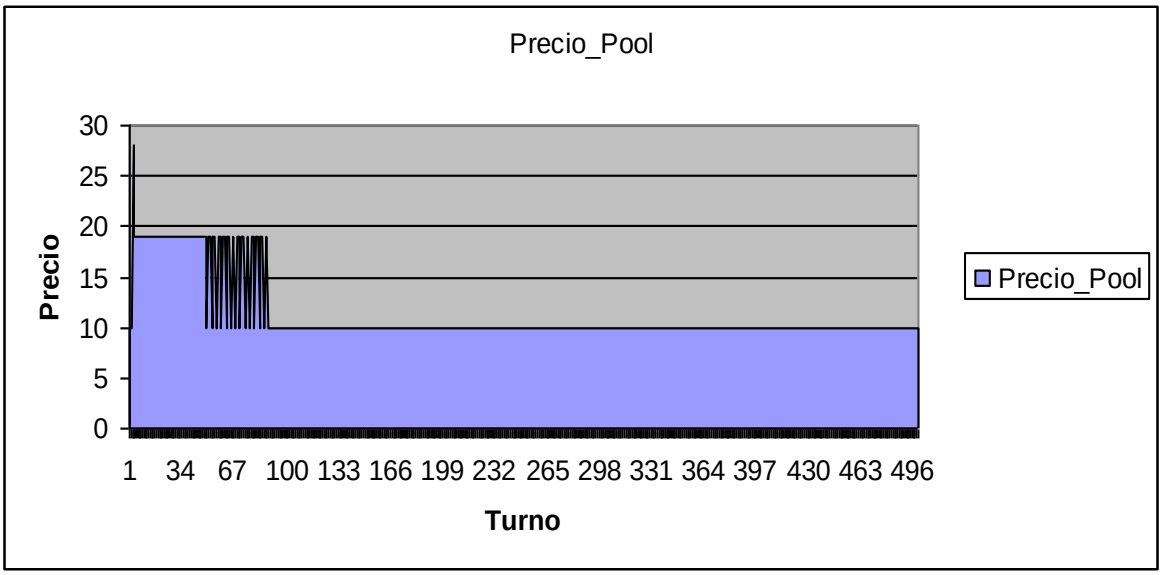


Ilustración 15. Resultados de la Prueba 4.

Precio Promedio: 11.33.

Desviación Estándar: 3.25.

Como la oferta es igual a la demanda, en esta ocasión los agentes compradores estaban confundidos al principio de la simulación y experimentaron con precios altos para satisfacer su necesidad de energía, pero al pasar el tiempo lograron optimizar sus reglas y lograron bajar el precio de la energía para comprarla lo más barata posible.

Conclusión.

Esta simulación demuestra, probablemente de forma más clara que las otras, que los agentes compradores son capaces de optimizar su gasto de dinero para comprar energía

eléctrica, ya que en esta simulación los compradores no tienen necesidad de competir por los recursos y el precio óptimo no es tan obvio para ellos desde el principio.

Prueba 5.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.

Compradores NO participan en el pool.

Precio Máximo: 100 por unidad.

No se permiten Contratos a largo plazo.

Compradores: Se crearon tres agentes compradores con una necesidad de 15 unidades de energía..

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

Como se puede apreciar, en esta simulación la oferta es mayor a la demanda.

Resultados.

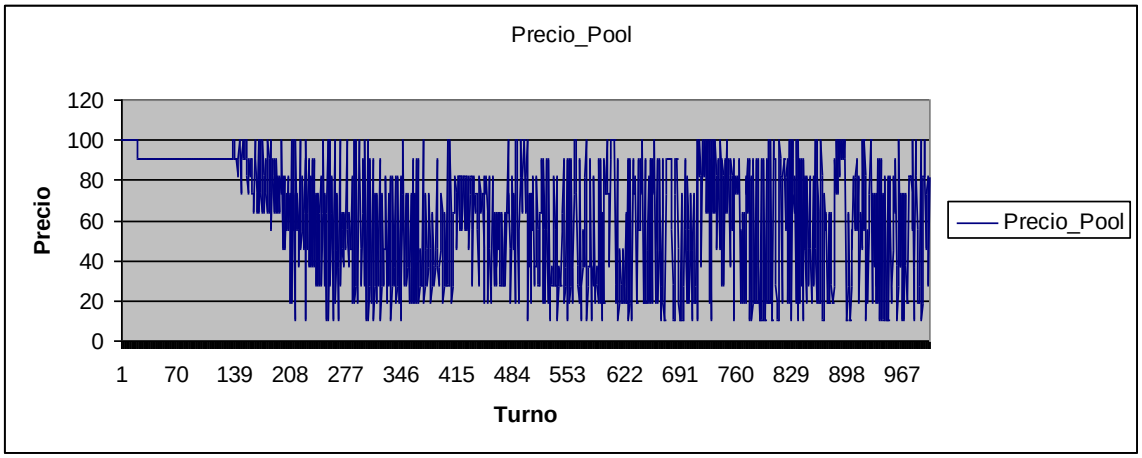


Ilustración 16. Resultados de la Prueba 5.

Precio Promedio: 61.87.

Desviación Estádar: 30.82

Al principio de la simulación los agentes vendedores lograron vender su producción al máximo precio posible, pero al no poder vender toda su producción ya que la oferta es mayor a la demanda, experimentaron con precios más bajos para poder maximizar sus ventas.

Conclusión.

Cuando los compradores no pueden participar en el pool, los vendedores pueden aprovecharse y conseguir precios muy altos para su producción a pesar de que la oferta sea mayor que la demanda. Pero esto solo dura hasta que uno de los vendedores intenta maximizar sus ganancias bajando los precios para vender más que los demás.

Prueba 6.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.

Compradores NO participan en el pool.

Precio Máximo: 100 por unidad.

No se permiten Contratos a largo plazo.

Compradores: Se crearon tres agentes compradores con una necesidad de 20 unidades de energía.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 16, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 17, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 18, Personalidad = Arriesgado.

En esta simulación la demanda es mayor que la oferta.

Resultados.

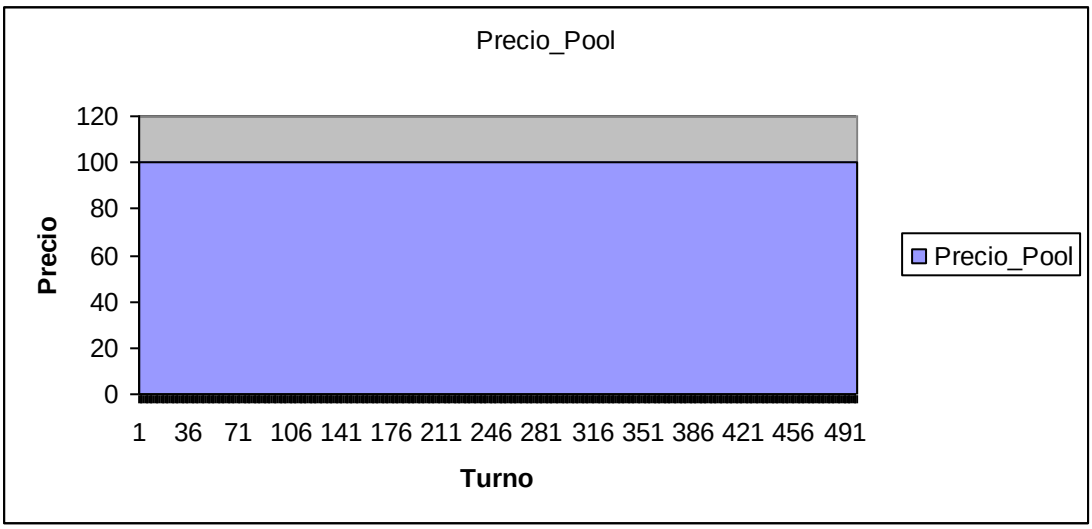


Ilustración 17. Resultados de la Prueba 6.

Precio Promedio: 100.

Desviación Estándar: 0.

Rápidamente los vendedores se aprovecharon de la falta de oferta para maximizar sus ganancias.

Conclusión.

Los vendedores en este programa también son capaces de maximizar sus ganancias al igual que los compradores, como ya se demostró en las anteriores simulaciones.

Prueba 7.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.

Compradores NO participan en el pool.

Precio Máximo: 100 por unidad.

No se permiten Contratos a largo plazo.

Compradores: Se crearon tres agentes compradores con una necesidad de 20 unidades de energía.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

En esta simulación la demanda es igual a la oferta.

Resultados.

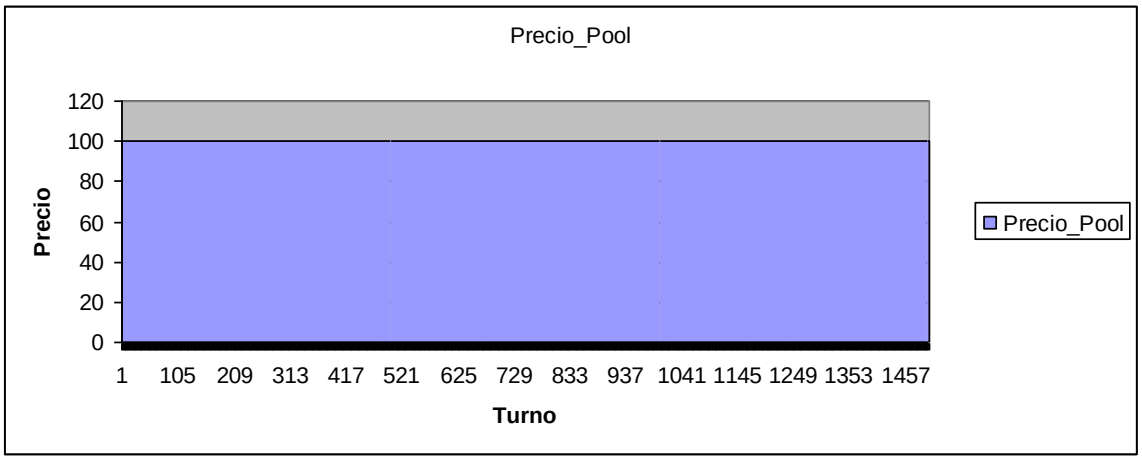


Ilustración 18. Resultados de la Prueba 7.

Precio Promedio: 99.99.

Desviación Estándar: 0.46.

Aunque es imposible verlo en el gráfico, en el primer turno el precio fue de 82 por unidad, pero rápidamente los agentes maximizaron sus ganancias al ver que no había necesidad de competir para vender sus productos.

Conclusión.

A pesar de que la inteligencia artificial de compradores y vendedores es prácticamente idéntica, los vendedores lograron maximizar sus ganancias en menos turnos que los compradores en una situación similar (Prueba 4). Esto seguramente se debe a que la ofertas al principio dependen del azar cuando los agentes no tienen mucha experiencia.

Los agentes vendedores tuvieron mejor suerte y llegaron a un estado óptimo de forma más rápida.

Prueba 8.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.

Compradores SI participan en el pool.

No se permiten Contratos a largo plazo.

Compradores: Se crearon cinco agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 15, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 15, Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 20, Personalidad = Arriesgado.

Comprador 4 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 5 – Precio Máximo = 100, Necesidad = 15, Personalidad = Moderado.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

Vendedor 4 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 5 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

En esta simulación la demanda es menor a la oferta.

Resultados.

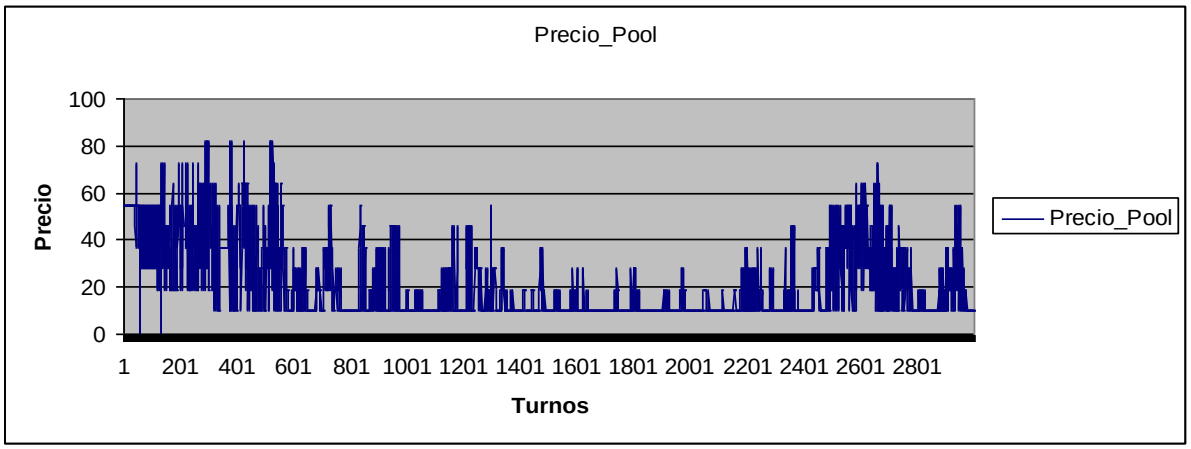


Ilustración 19. Resultados de la Prueba 8.

Precio promedio: 21.02 por unidad. (Sin contar los turnos en que el precio fue cero, lo cual significa que en esos turnos no se vendió ni compró nada.)

Desviación Estándar: 16.00.

Como se puede apreciar en el gráfico, si se lo contrasta con los gráficos de las siguientes pruebas, los compradores supieron aprovechar su ventaja, que la oferta era mayor que la

demanda, y consiguieron en promedio precios bajos para la adquisición de la energía que necesitaban.

Conclusión.

Aunque en promedio el precio fue bajo, existen muchos picos altos en el gráfico y la simulación nunca encuentra estabilidad. Esto se debe a que los agentes se encuentran en un estado de constante experimentación causado por la constante competencia para comprar y vender a los mejores precios posibles lo cual causa que una regla que funciona bien en un turno, no funcione de igual manera en el siguiente. Solamente en aquellos escenarios donde los agentes no compiten entre ellos se puede tener estabilidad.

Prueba 9.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.
Compradores SI participan en el pool.
No se permiten Contratos a largo plazo.

Compradores: Se crearon cinco agente compradores.
Comprador 1 – Precio Máximo = 100, Necesidad = 25, Personalidad = Conservador.
Comprador 2 – Precio Máximo = 100, Necesidad = 25, Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 20, Personalidad = Arriesgado.

Comprador 4 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 5 – Precio Máximo = 100, Necesidad = 25, Personalidad = Moderado.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

Vendedor 4 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 5 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

En esta simulación la demanda es mayor a la oferta.

Resultados.

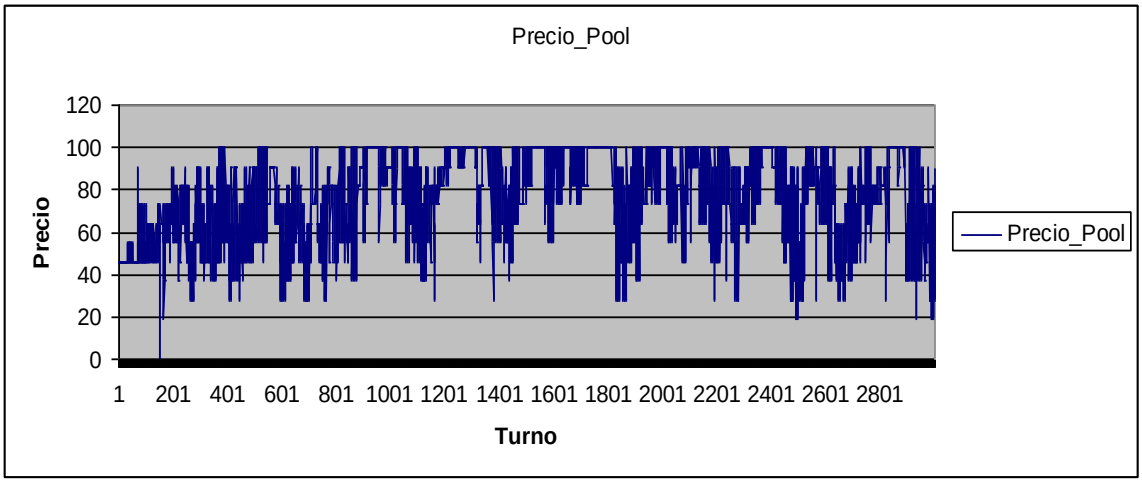


Ilustración 20. Resultados de la Prueba 9.

Precio promedio: 77.07 por unidad. (Sin contar los turnos en que el precio fue cero, lo cual significa que en esos turnos no se vendió ni compró nada.)

Desviación Estándar: 20.82.

En esta ocasión los vendedores se aprovecharon y obtuvieron un precio más alto por la producción.

Conclusión.

Tanto los compradores como los vendedores son capaces de aprovechar las ventajas que ocurran en una simulación. En este tipo de simulación donde tanto los compradores como los vendedores pueden ofertar en el pool, parece ser imposible maximizar la ventaja y esto favorece a los agentes con desventaja para que los otros no se aprovechen.

Prueba 10.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.
Compradores SI participan en el pool.

No se permiten Contratos a largo plazo.

Compradores: Se crearon cinco agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 20, Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 20, Personalidad = Arriesgado.

Comprador 4 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 5 – Precio Máximo = 100, Necesidad = 20, Personalidad = Moderado.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

Vendedor 4 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 5 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

En esta simulación la demanda es igual a la oferta.

Resultados.

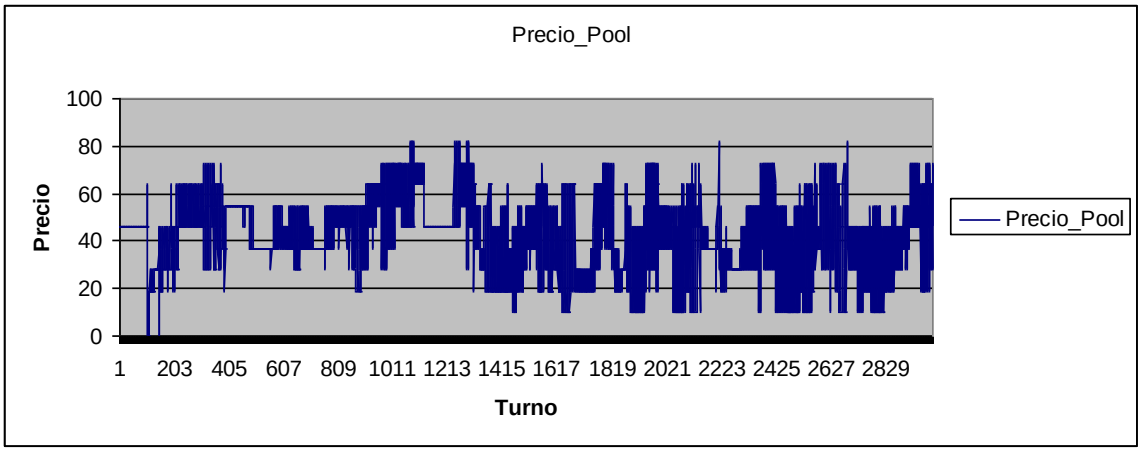


Ilustración 21. Resultados de la Prueba 10.

Precio promedio: 41.84 por unidad. (Sin contar los turnos en que el precio fue cero, lo cual significa que en esos turnos no se vendió ni compró nada.)

Desviación Estándar: 16.76.

En esta ocasión el promedio ha sido prácticamente la mitad del máximo.

Conclusión.

A diferencia de las simulaciones donde solo un tipo de agente podía ofertar libremente en el pool, y donde cuando se daba el caso de que la oferta era igual a la demanda el agente que podía oferta optimizaba sus ganancias o compras, en este caso el precio es un acuerdo más razonable entre compradores y vendedores.

Prueba 11.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.

Compradores SI participan en el pool.

Se permiten Contratos a largo plazo.

Número de turnos sin Contratos a Largo Plazo: 1000.

Compradores: Se crearon cinco agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 15, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 15, Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 20, Personalidad = Arriesgado.

Comprador 4 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 5 – Precio Máximo = 100, Necesidad = 15, Personalidad = Moderado.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

Vendedor 4 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 5 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

En esta simulación la demanda es menor a la oferta.

Resultados.

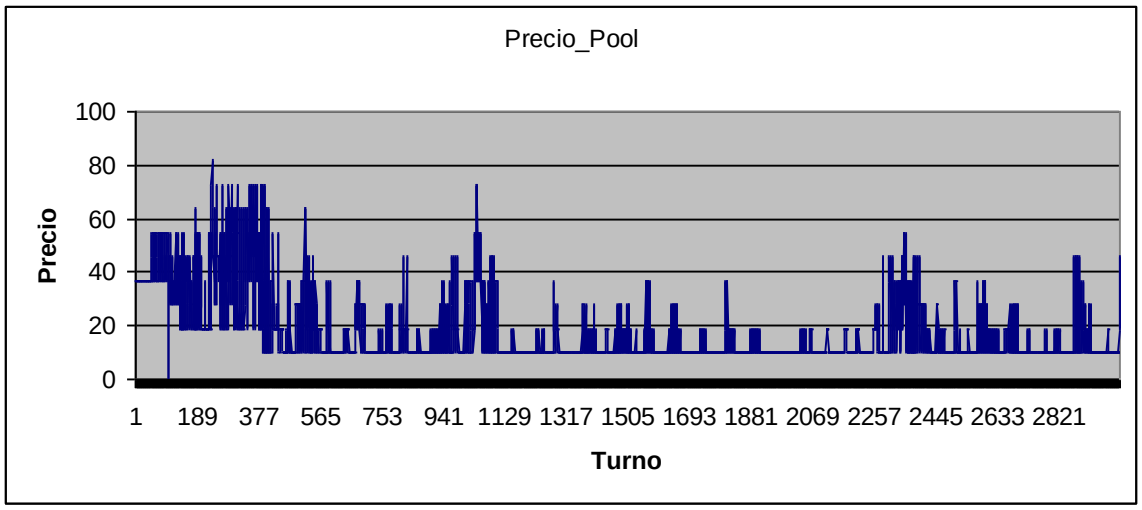


Ilustración 22. Resultados de la Prueba 11.

Precio promedio: 17.51 por unidad. (Sin contar los turnos en que el precio fue cero, lo cual significa que en esos turnos no se vendió ni compró nada.)

Desviación Estándar: 12.98.

El precio promedio fue más bajo en esta ocasión que en la prueba 8, que es bastante parecida a esta pero sin contratos a largo plazo, pero por muy poco.

El precio promedio de la energía vendida por contratos fue 34.16

El precio promedio de la energía vendida en el pool sin contar los turnos donde no se permitían contratos a largo plazo o los turnos en que el precio fue cero: 13.70.

Desviación Estándar: 8.41.

Conclusión.

A primera vista parecería que permitir contratos a largo plazo aumenta el precio de la energía en el pool. Por lo menos eso ocurre con la forma en que se han implementado los contratos a largo plazo en el programa.

Prueba 12.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.

Compradores SI participan en el pool.

Se permiten Contratos a largo plazo.

Número de turnos sin Contratos a Largo Plazo: 1000.

Compradores: Se crearon cinco agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 25, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 25 Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 20, Personalidad = Arriesgado.

Comprador 4 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 5 – Precio Máximo = 100, Necesidad = 25, Personalidad = Moderado.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

Vendedor 4 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 5 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

En esta simulación la demanda es mayor a la oferta.

Resultados.

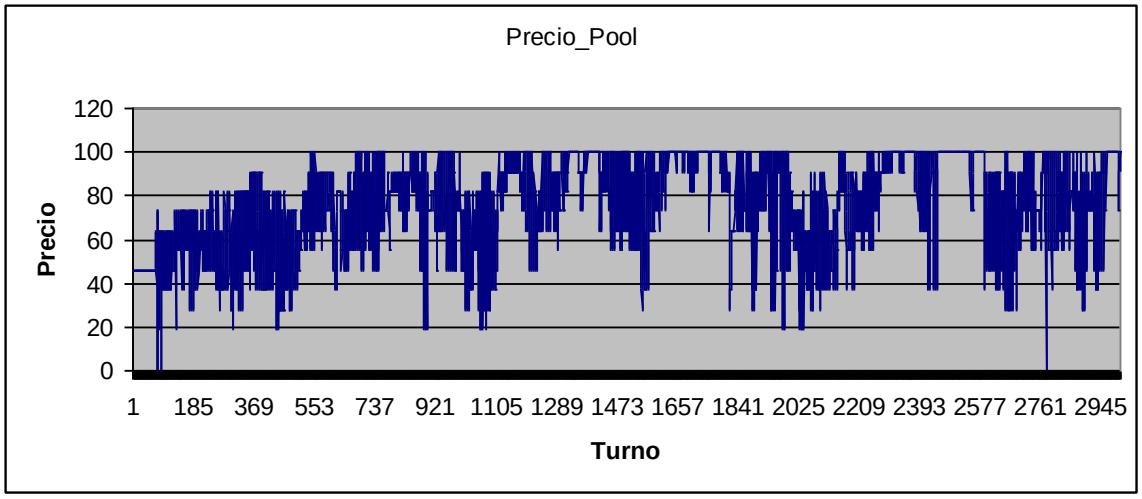


Ilustración 23. Resultados de la Prueba 12.

Precio promedio: 76.90 por unidad. (Sin contar los turnos en que el precio fue cero, lo cual significa que en esos turnos no se vendió ni compró nada.)

Desviación Estándar: 21.09.

El precio promedio fue más alto en esta ocasión que en la prueba 9, que es bastante parecida a esta pero sin contratos a largo plazo, pero al igual que en el caso de la prueba 11 con la prueba 8, la diferencia es muy baja.

El precio promedio de la energía vendida por contratos fue 81.39.

Desviación Estándar: 16.52.

El precio promedio de la energía vendida en el pool sin contar los turnos donde no se permitían contratos a largo plazo o los turnos en que el precio fue cero: 82.05.

Desviación Estándar: 19.89.

Conclusión.

Esta simulación también muestra un pequeño aumento en el precio promedio de la energía eléctrica en comparación con la simulación similar sin contratos a largo plazo. Comparándolo con la prueba anterior, también podemos ver que el precio de la energía vendida por contratos es más alto que la del pool.

Prueba 13.

Esta simulación tiene las siguientes características:

Tipo de Pool: PRECIOS.

Compradores SI participan en el pool.

Se permiten Contratos a largo plazo.

Número de turnos sin Contratos a Largo Plazo: 1000.

Compradores: Se crearon cinco agente compradores.

Comprador 1 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 100, Necesidad = 20 Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 20, Personalidad = Arriesgado.

Comprador 4 – Precio Máximo = 100, Necesidad = 20, Personalidad = Conservador.

Comprador 5 – Precio Máximo = 100, Necesidad = 20, Personalidad = Moderado.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

Vendedor 4 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 5 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

En esta simulación la demanda es mayor a la oferta.

Resultados.

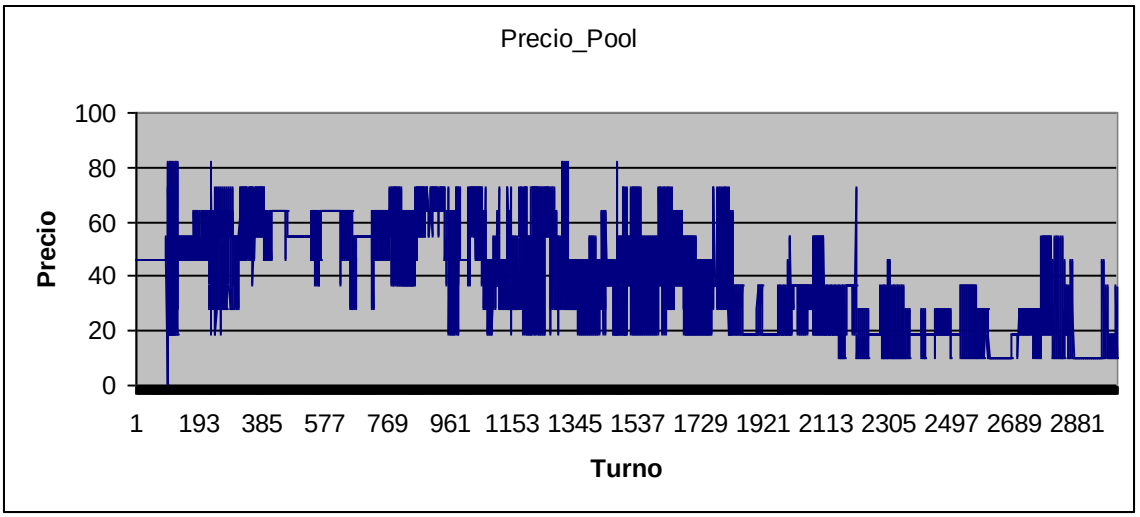


Ilustración 24. Resultados de la Prueba 13.

Precio promedio: 38.62 por unidad. (Sin contar los turnos en que el precio fue cero, lo cual significa que en esos turnos no se vendió ni compró nada.)
Desviación Estándar 19.68.

El precio promedio fue más alto en esta ocasión que en la prueba 10, que es bastante parecida a esta pero sin contratos a largo plazo.

El precio promedio de la energía vendida por contratos fue 38.26.
Desviación Estándar: 22.35.

El precio promedio de la energía vendida en el pool sin contar los turnos donde no se permitían contratos a largo plazo o los turnos en que el precio fue cero: 30.81.
Desviación Estándar: 17.72

Conclusiones.

En esta ocasión el precio promedio fue ligeramente menor al precio promedio que se obtuvo en la prueba 10, la cual es prácticamente idéntica a esta prueba pero sin contratos a largo plazo.

Prueba 14.

Esta simulación tiene las siguientes características:

Tipo de Pool: COSTOS.

Compradores SI participan en el pool.

No se permiten Contratos a largo plazo.

Número de turnos sin Contratos a Largo Plazo: 1000.

Compradores: Se crearon cinco agente compradores.

Comprador 1 – Precio Máximo = 150, Necesidad = 25, Personalidad = Conservador.

Comprador 2 – Precio Máximo = 125, Necesidad = 23 Personalidad = Moderado.

Comprador 3 – Precio Máximo = 100, Necesidad = 25, Personalidad = Arriesgado.

Vendedores: Se crearon tres agentes vendedores.

Vendedor 1 – Precio Mínimo = 10, Producción = 20, Personalidad = Conservador.

Vendedor 2 – Precio Mínimo = 10, Producción = 20, Personalidad = Moderado.

Vendedor 3 – Precio Mínimo = 10, Producción = 20, Personalidad = Arriesgado.

En esta simulación la demanda es mayor a la oferta.

Resultados.

El primer gráfico representa la fluctuación del precio de la energía en el mercado.

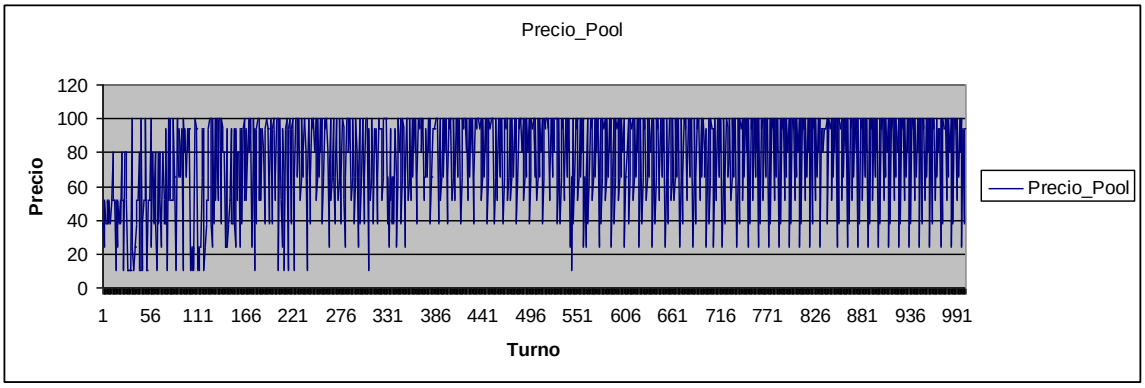


Ilustración 25. Fluctuación del precio en el mercado spot de la energía en la prueba 14.

Precio Promedio: 74.41.

Desviación Estándar: 27.36.

El siguiente gráfico representa la fluctuación de los precios ofertados por el Comprador 1.

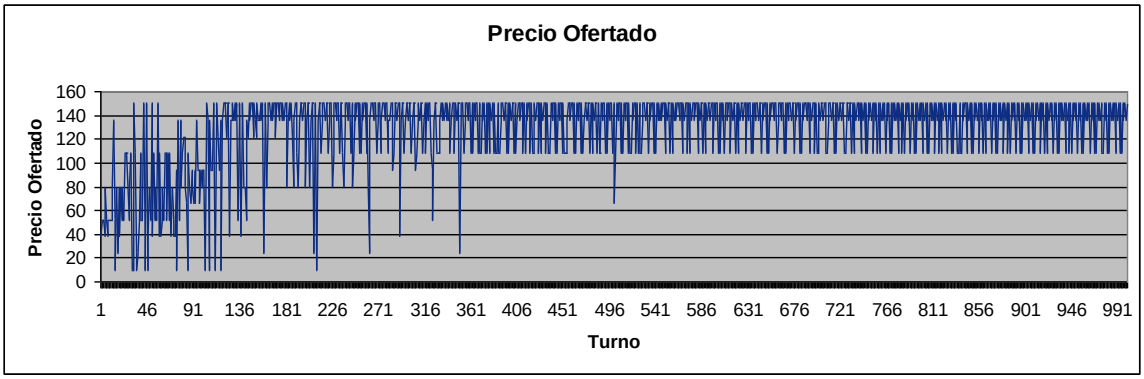


Ilustración 26. Fluctuación de los precios ofertados por el comprador 1 en la prueba 14.

Precio Ofertado Promedio: 127.88.

Desviación Estándar: 30.95.

El siguiente gráfico representa la cantidad de energía adquirida por el comprador 1 en cada turno.

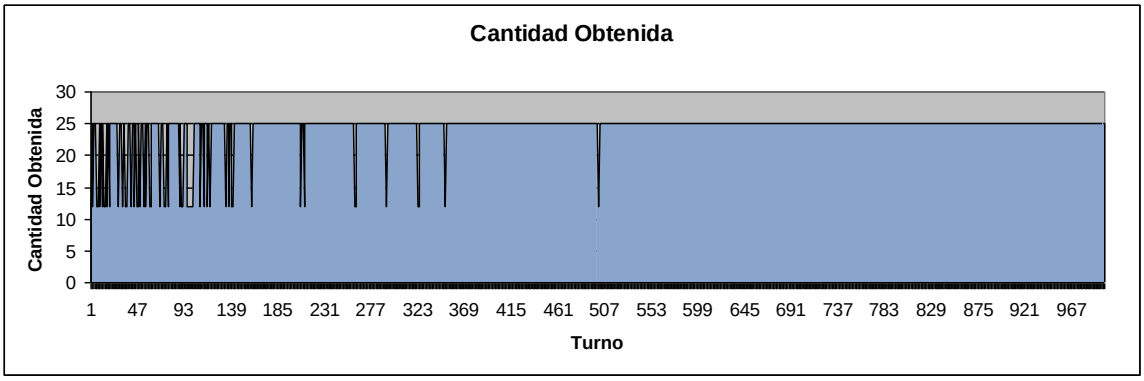


Ilustración 27. Energía adquirida por el comprador 1 en la prueba 14.

Cantidad Aceptada Promedio: 24.22 de 25.

Desviación Estándar: 3.09.

Como se puede apreciar, el Comprador 1, que tenía la capacidad de ofertar la mayor cantidad de dinero, aprovechó e incrementó su precio de oferta para garantizar la compra de energía. También se debe resaltar que el precio escogido por el comprador oscila por 127, y no por su máximo precio posible. Es interesante notar que el agente logró encontrar el precio más bajo que le garantizaba la compra de energía eléctrica.

Ahora examinaremos los gráficos del Comprador 2.

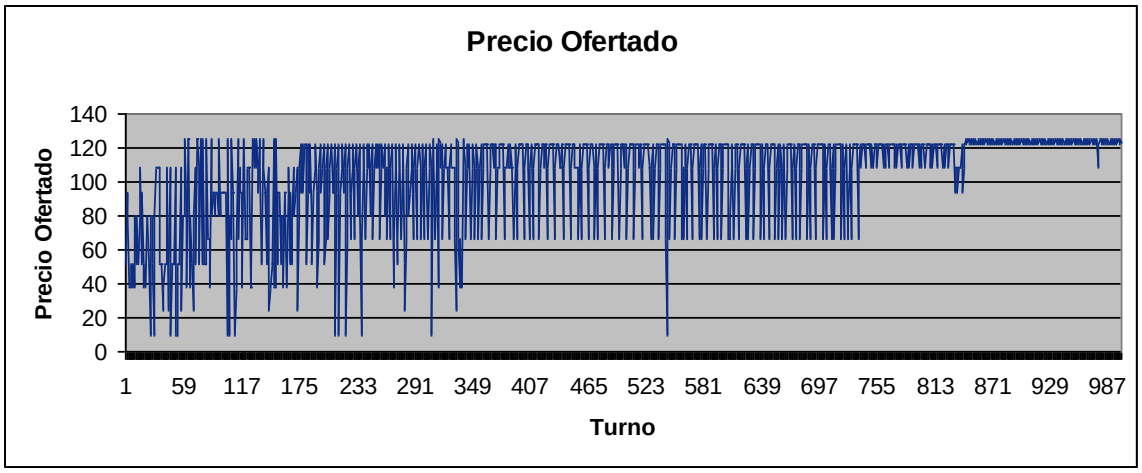


Ilustración 28. Fluctuación de los precios ofertados por el comprador 2 en la prueba 14.

Precio Ofertado Promedio: 103.45.

Desviación Estándar: 27.59.

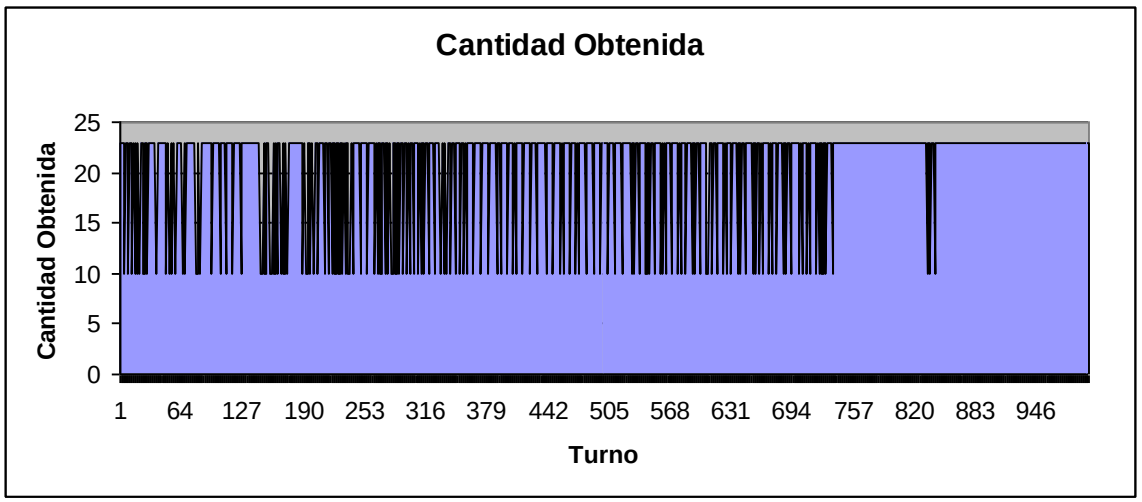


Ilustración 29. Energía adquirida por el comprador 2 en la prueba 14.

Cantidad Promedio: 20.67 de 23.

Desviación Estándar: 4.99

El comprador 2 pasó por una situación similar a la del comprador 1, pero a pesar de no poder ofertar tanto dinero como el primero comprador, al ofertar todo lo que podía logró conseguir la cantidad de energía que necesitaba. Se puede apreciar como alrededor del turno 750 mejoró bastante su forma de ofertar y empezó a conseguir toda la energía más constantemente.

Para finalizar, se examinará los gráficos del comprador 3.

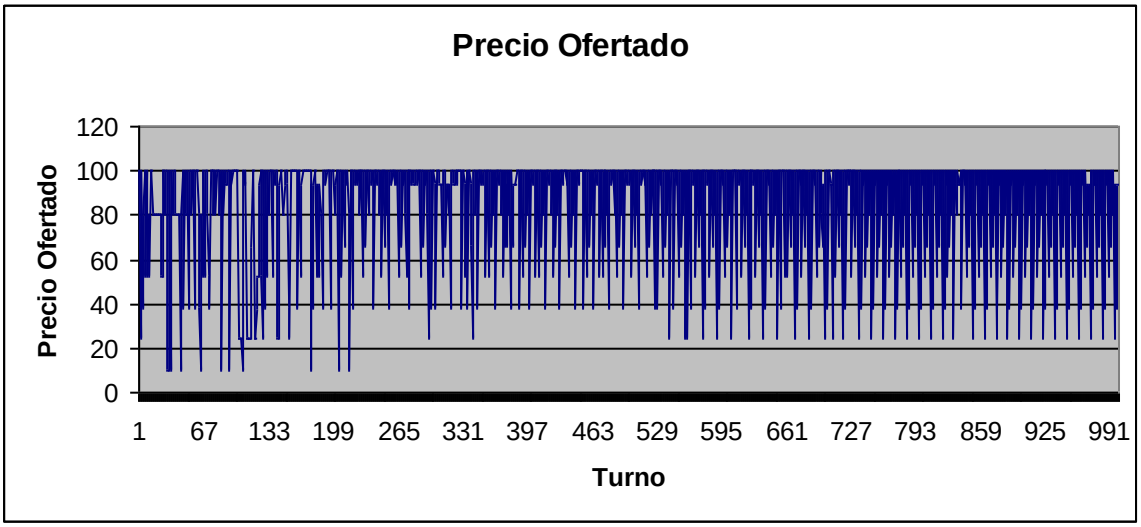


Ilustración 30. Fluctuación de los precios ofertados por el comprador 3 en la prueba 14.

Precio Ofertado Promedio: 82.61.

Desviación Estándar: 25.42.

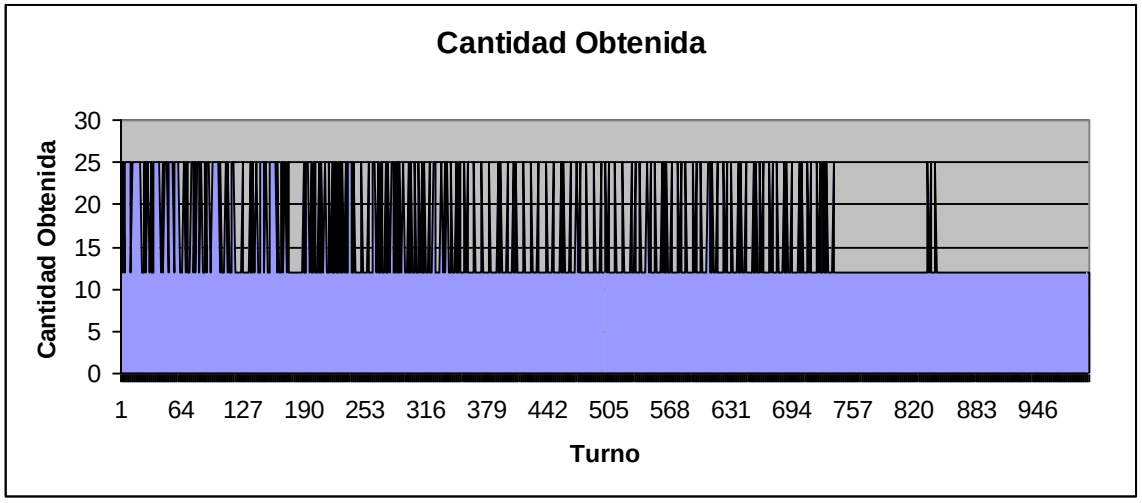


Ilustración 31. Energía adquirida por el comprador 3 en la prueba 14.

Cantidad Obtenida Promedio: 15.11 de 25.

Desviación Estándar: 5.55.

A diferencia de los otros dos compradores, el tercer comprador no podía ofertar la cantidad de dinero necesaria para conseguir toda la energía que terminaba siendo comprada por los otros dos compradores. Se puede apreciar que alrededor del turno 750, al mismo tiempo que el comprador 2 mejoró su estrategia y empezó a conseguir energía de forma más constante, el comprador 3 empezó a conseguir menos energía de la que necesitaba. Cuando los otros compradores aprendieron a ofertar mejor, dejaron sin energía al comprador que podía ofertar menos dinero.

Conclusión:

Con esta simulación se puede comprobar que cuando un agente, en este caso agentes compradores, tiene una ventaja sobre los otros para competir en un mercado, el algoritmo de inteligencia artificial le permite aprender como aprovecharlo y maximizar la cantidad de energía eléctrica que puede comprar. También se puede observar que los gráficos son menos caóticos al avanzar los turnos cuando algunos agentes tienen ventajas.

5.3 Discusión de los resultados

Antes de discutir los resultados obtenidos con el simulador, hay que recalcar que el simulador no es el mundo real. En el mundo real pueden aparecer nuevos compradores y nuevos vendedores de energía, especialmente si se percibe una gran demanda que podría causar una mayor inversión en la producción de energía. Desde ese punto de vista, el entorno de estas simulaciones es estático ya que no aparecerán más agentes que los especificados por el usuario al comenzar la simulación. Este simulador no puede predecir ni simular la entrada de nuevos agentes al mercado eléctrico; solo se limita a simular el mercado con un número fijo de agentes.

También hay que tomar en cuenta que los agentes tienen límites. Los agentes de este programa nunca terminan de aprender, lo cual puede dar situaciones como las que se pueden apreciar en la ilustración 30. El comprador 3 de la prueba 14 no puede comprar energía porque no posee la cantidad de recursos económicos que los otros dos

compradores poseen. Sin importar que precio intente, nunca podrá ganarles a los otros dos compradores, por lo cual seguirá intentando diferentes precios y esto afecta el precio promedio de la simulación.

La primera prueba (Prueba 1), como era de esperarse, no ofrece resultados muy interesantes más allá de comprobar que el mecanismo de subasta funciona. Los vendedores no pueden ofertar otro precio que no sea el costo de producir la energía, y los compradores no participan en el pool. Por ende, el precio en este tipo de simulación nunca variará.

Las pruebas dos, tres y cuatro comprueban que los agentes compradores son capaces de aprender a funcionar en el pool del mercado. En estas simulaciones los vendedores solo ofertan su energía al costo de producirla. En estos casos los compradores solo tienen que competir entre sí.

Las pruebas cinco, seis y siete comprueban que los agentes vendedores son capaces de aprender a ofertar en el pool del mercado. En estos casos los compradores prácticamente no participan en el pool y solo informan su necesidad al pool. Los vendedores solo deben competir entre sí.

Una vez que hemos comprobado que ambos tipos de agentes funcionan de la forma deseada, es decir, son capaces de aprender a maximizar sus ganancias o a comprar

energía de la forma más barata posible, podemos analizar las simulaciones más complejas donde participan ambos tipos de agentes.

Las pruebas ocho, nueve y diez muestran que los agentes pueden aprovecharse de las ventajas que les ofrece el entorno, pero que los agentes desaventajados son capaces de defenderse hasta cierto punto, lo cual causa que los precios en estas simulaciones sean más justos para compradores y vendedores.

Cabe notar que en este simulador los agentes tienen muy poca información sobre los otros agentes. Los compradores no saben cuanto producen los vendedores, ni cual es el mínimo precio que cada vendedor está dispuesto a aceptar excepto por el mínimo global. Los vendedores no saben cuanta energía necesitan los compradores, ni el máximo precio que cada uno está dispuesto a pagar para obtenerla. Es posible en el mundo real que los compradores se unan y solo oferten el mínimo hasta que los vendedores acepten, o que los vendedores se unan y solo oferten el precio al máximo y los compradores se vean forzados a comprar a ese precio. Estos eventos, aunque posibles, están más allá de lo que se quiere lograr con este programa. Uno de los objetivo que queremos lograr es comprobar que los agentes puedan tomar sus decisiones basados únicamente en su algoritmo de aprendizaje y no con una inteligencia artificial reactiva con conocimiento y estrategias pre-programadas. También, como se puede comprobar con los resultados obtenidos, los agentes con capaces de aprovechar las circunstancias; pero la capacidad de unirse por una causa común va más allá de sus habilidades como agentes inteligentes. En otras palabras, estas simulaciones son “todos contra todos.”

Las pruebas once, doce y trece son idénticas a las tres anteriores con la diferencia de que permiten la creación de contratos a largo plazo. En estos casos parece ser que cuando existe una diferencia entre oferta y demanda, el precio de la energía eléctrica es mayor con los contratos a largo plazo. Cuando la oferta y la demanda son iguales, la diferencia entre precios de una simulación con contratos y una sin contratos es mínima.

Después de haber examinado todas las simulaciones podemos hacer las siguientes observaciones:

- Un pool donde ambos tipos de agentes, compradores y vendedores, pueden ofertar el precio que consideran apropiado ofrece una ventaja muy grande sobre los mercados asimétricos y los simétricos basados en costos. Como hemos podido comprobar, en estos dos casos es muy fácil para los agentes que pueden ofertar precios libremente aprovecharse de cualquier ventaja. Por ejemplo en las pruebas dos y tres, donde los compradores no participan en el pool, se puede apreciar que los vendedores se aprovechan de que la oferta es mayor que la demanda, o igual a esta y maximizan el precio sin ningún problema. En el caso de la prueba cuatro, donde los compradores tampoco participan en el pool pero la demanda es mayor que la oferta, nos encontramos con el caso de que los vendedores ahora deben competir entre ellos para obtener mejores ganancias. En este caso podemos observar que los resultados se parecen a los resultados obtenidos con un mercado simétrico a base de precios. Conclusiones similares podemos sacar al observar las

pruebas cinco, seis y siete donde solo los compradores pueden libremente elegir el precio a ofertar en el pool.

- En las simulaciones donde existe competencia ya sea entre compradores y vendedores, compradores y compradores, o vendedores y vendedores, podemos observar que el precio nunca encuentra un punto fijo, es decir que los agentes no pueden encontrar un conjunto de reglas óptimas. En los mercados asimétricos, la competencia se crea porque el número de recursos por los que compiten es limitado y menor a la cantidad necesaria para satisfacer todas sus necesidades. En el caso de los mercados simétricos, siempre existe competencia porque el objetivo principal de los vendedores es maximizar el precio, y esto entra en conflicto con los compradores que intentan minimizar el precio. La constante competencia lleva a que una regla que haya funcionado bien en un turno, no tenga el mismo rendimiento en el siguiente, por lo cual se experimentará con otra regla. Si un comprador oferta un precio mayor, obtendrá más energía eléctrica en ese turno, pero eso causará que otro agente obtenga menos energía de la que obtuvo en el turno anterior, por lo cual el segundo agente se verá forzado a utilizar otra regla y experimentar con un precio diferente. Esto causa el constante ciclo de experimentación que nunca acaba a menos que unos de los agentes tenga una gran ventaja sobre los otros; por ejemplo la capacidad de pagar más dinero por la energía, en cuyo caso ofrecerá a pagar más dinero para garantizar que siempre obtendrá energía. En el caso de que tal ventaja no exista, uno podría pensar que el resultado sería un completo caos, pero podemos observar que no es así. Las

pruebas ocho y nueve ofrecen clara evidencia de que aún con constante competencia, el precio promedio responde a las leyes de la oferta y la demanda. Si contrastamos los gráficos obtenidos con ambas pruebas se puede apreciar la diferencia de precios. Es prueba que la inteligencia artificial es capaz de aprender a adaptarse al mercado aún cuando sea imposible encontrar un grupo de reglas óptimas.

CONCLUSIONES Y RECOMENDACIONES

1. Hemos construido un simulador que puede simplificar el proceso de experimentación y permite estudiar varios tipos de mercados eléctricos.

2. Pudimos comprobar que los agentes inteligentes pueden ser utilizados para la simulación de mercados eléctricos y programas similares, ya que son capaces de aprender y adatarse en este entorno. Por lo tanto hemos cumplido con el primer objetivo planteado para esta tesis.

3. Hemos simulado diferentes formas de funcionamiento del mercado eléctrico y estudiado los resultados, cumpliendo así con el segundo objetivo planteado para esta tesis.

4. Pudimos verificar que los resultados obtenidos son razonables y consistentes con los conocimientos de economía aprendidos según lo visto en el capítulo de los resultados.

5. Se recomienda probar más a fondo este simulador con especialistas en el campo para poder verificar su efectividad en la simulación de los diferentes mercados.

6. Para acelerar la ejecución del programa se recomienda hacerlo en otro lenguaje, ya que JAVA es lento debido a que es un lenguaje interpretado y no compilado.

7. Se puede mejorar el programa separando un poco más la toma de decisiones sobre contratos a largo plazo de la toma de decisiones sobre el mercado spot. En vez de que la primera sea una reacción automática sobre la segunda, como se ha implementado en este programa, es posible hacer que los agentes consideren a los contratos como algo aparte y hacer la toma de decisiones algo más complejo.

APÉNDICE 1 – CASOS DE USO

Nombre:	Simulación con Pool por Costos sin Compradores.
Autor:	Juan Arteaga
Descripción: El usuario inicia una simulación donde el pool trabaja por costos y no se le permite participar a los compradores en el mercado spot.	
Actores: Usuario del programa.	
Precondiciones: El usuario ha seleccionado las opciones para una simulación con Pool por costos y sin Compradores, y ha llenado los datos de los compradores y vendedores que participarán en la simulación. La base de datos está vacía.	
Flujo Normal: 1. El actor termina de llenar los datos de los compradores y vendedores e inicializa la simulación. 2. El pool le indica a los vendedores que oferten por costos. 3. Los vendedores enviarán sus ofertas al pool. 4. El pool escogerá las ofertas que se deben aceptar de acuerdo a las reglas del mercado asimétrico. 5. El pool le indicará el resultado de la subasta a los vendedores. 6. El pool escribirá los resultados en la base de datos. 7. El pool repetirá estas acciones hasta acabar con la simulación.	
Flujo Alternativo: 7. Si ya se realizaron todos los turnos de la simulación, el pool dará por terminada la simulación y se regresará a la ventana principal.	
Poscondiciones: La base de datos contendrá datos y su estada constará como “llena”.	

Nombre:	Simulación con Pool por Precios sin Compradores.
Autor:	Juan Arteaga

Descripción: El usuario inicia una simulación donde el pool trabaja por precios y no se le permite participar a los compradores en el mercado spot.

Actores: Usuario del programa.

Precondiciones: El usuario ha seleccionado las opciones para una simulación con Pool por precios y sin Compradores, y ha llenado los datos de los compradores y vendedores que participarán en la simulación. La base de datos está vacía.

Flujo Normal:

1. El actor termina de llenar los datos de los compradores y vendedores e inicializa la simulación.
2. El pool le indica a los vendedores que oferten por precios.
3. Los vendedores enviarán sus ofertas al pool.
4. El pool escogerá las ofertas que se deben aceptar de acuerdo a las reglas del mercado asimétrico.
5. El pool le indicará el resultado de la subasta a los vendedores.
6. El pool escribirá los resultados en la base de datos.
7. El pool repetirá estas acciones hasta acabar con la simulación.

Flujo Alternativo:

7. Si ya se realizaron todos los turnos de la simulación, el pool dará por terminada la simulación y se regresará a la ventana principal.

Poscondiciones: La base de datos contendrá datos y su estada constará como “llena”.

Nombre:	Simulación con Pool por Costos con Compradores.
Autor:	Juan Arteaga

Descripción: El usuario inicia una simulación donde el pool trabaja por costos y se le permite participar a los compradores en el mercado spot.

Actores: Usuario del programa.

Precondiciones: El usuario ha seleccionado las opciones para una simulación con Pool por costos y con Compradores, y ha llenado los datos de los compradores y vendedores que participarán en la simulación. La base de datos está vacía.

Flujo Normal:

1. El actor termina de llenar los datos de los compradores y vendedores e inicializa la simulación.
2. El pool le indica a los vendedores que oferten por costos.
3. El pool le indica a los compradores que oferten.
4. Los vendedores enviarán sus ofertas al pool.
5. Los compradores enviarán sus ofertas al pool.
6. El pool escogerá las ofertas que se deben aceptar de acuerdo a las reglas del mercado simétrico.
7. El pool le indicará el resultado de la subasta a los vendedores y compradores.
8. El pool escribirá los resultados en la base de datos.
9. El pool repetirá estas acciones hasta acabar con la simulación.

Flujo Alternativo:

9. Si ya se realizaron todos los turnos de la simulación, el pool dará por terminada la simulación y se regresará a la ventana principal.

Poscondiciones: La base de datos contendrá datos y su estada constará como “llena”.

Nombre:	Simulación con Pool por Precios con Compradores.
Autor:	Juan Arteaga

Descripción: El usuario inicia una simulación donde el pool trabaja por precios y se le permite participar a los compradores en el mercado spot.

Actores: Usuario del programa.

Precondiciones: El usuario ha seleccionado las opciones para una simulación con Pool por precios y con Compradores, y ha llenado los datos de los compradores y vendedores que participarán en la simulación. La base de datos está vacía.

Flujo Normal:

1. El actor termina de llenar los datos de los compradores y vendedores e inicializa la simulación.
2. El pool le indica a los vendedores que oferten por precios.
3. El pool le indica a los compradores que oferten.
4. Los vendedores enviarán sus ofertas al pool.
5. Los compradores enviarán sus ofertas al pool.
6. El pool escogerá las ofertas que se deben aceptar de acuerdo a las reglas del mercado simétrico.
7. El pool le indicará el resultado de la subasta a los vendedores y compradores.
8. El pool escribirá los resultados en la base de datos.
9. El pool repetirá estas acciones hasta acabar con la simulación.

Flujo Alternativo:

9. Si ya se realizaron todos los turnos de la simulación, el pool dará por terminada la simulación y se regresará a la ventana principal.

Poscondiciones: La base de datos contendrá datos y su estada constará como “llena”.

Nombre:	Simulación con Pool por Precios con Compradores y con contratos a largo plazo.
Autor:	Juan Arteaga

Descripción: El usuario inicia una simulación donde el pool trabaja por precios, se le permite participar a los compradores en el mercado spot, y se permiten los contratos a largo plazo.

Actores: Usuario del programa.

Precondiciones: El usuario ha seleccionado las opciones para una simulación con Pool por precios y con Compradores, y ha llenado los datos de los compradores y vendedores que participarán en la simulación. La base de datos está vacía.

Flujo Normal:

1. El actor termina de llenar los datos de los compradores y vendedores e inicializa la simulación.
2. El pool le indica a los vendedores que oferten por precios.
3. El pool le indica a los compradores que oferten.
4. Vendedores y Compradores se envían ofertas de contratos a largo plazo.
5. Vendedores y Compradores aceptan o rechazan estas ofertas.
6. Los vendedores enviarán sus ofertas al pool.
7. Los compradores enviarán sus ofertas al pool.
8. El pool escogerá las ofertas que se deben aceptar de acuerdo a las reglas del mercado simétrico.
9. El pool le indicará el resultado de la subasta a los vendedores y compradores.
10. El pool escribirá los resultados en la base de datos.
11. El pool repetirá estas acciones hasta acabar con la simulación.

Flujo Alternativo:

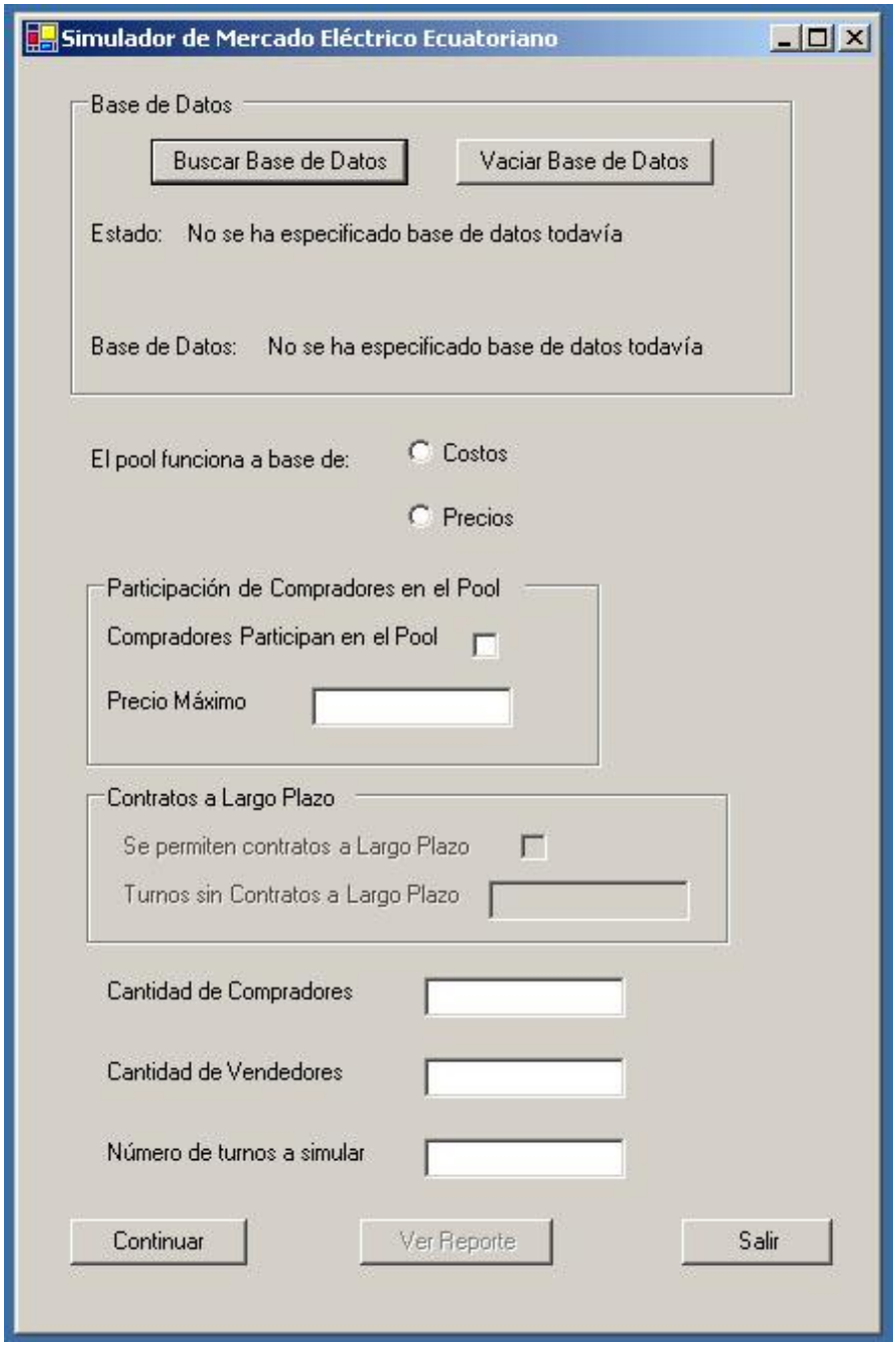
11. Si ya se realizaron todos los turnos de la simulación, el pool dará por terminada la simulación y se regresará a la ventana principal.

Poscondiciones: La base de datos contendrá datos y su estada constará como “llena”.

APÉNDICE 2 – DIAGRAMA DE CLASES.

APÉNDICE 3 – FLUJO DE VENTANAS

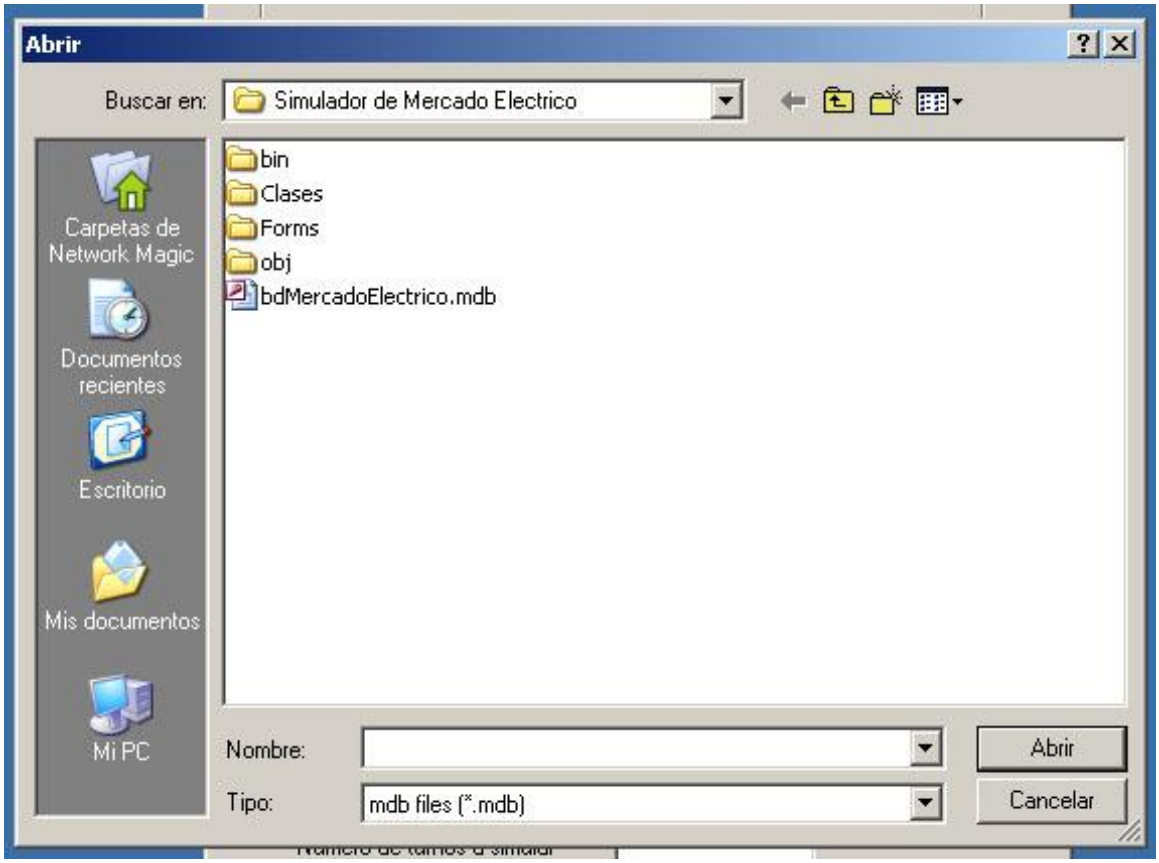
Ventana Principal.



Esta es la ventana principal del programa y es la primera ventana que se le muestra al usuario al ejecutar el programa.

De esta ventana se puede acceder a todas las otras ventanas.

Ventana para seleccionar Base de Datos.



A esta ventana se accede haciendo clic sobre el botón 'Buscar Base de Datos' en la Ventana Principal.

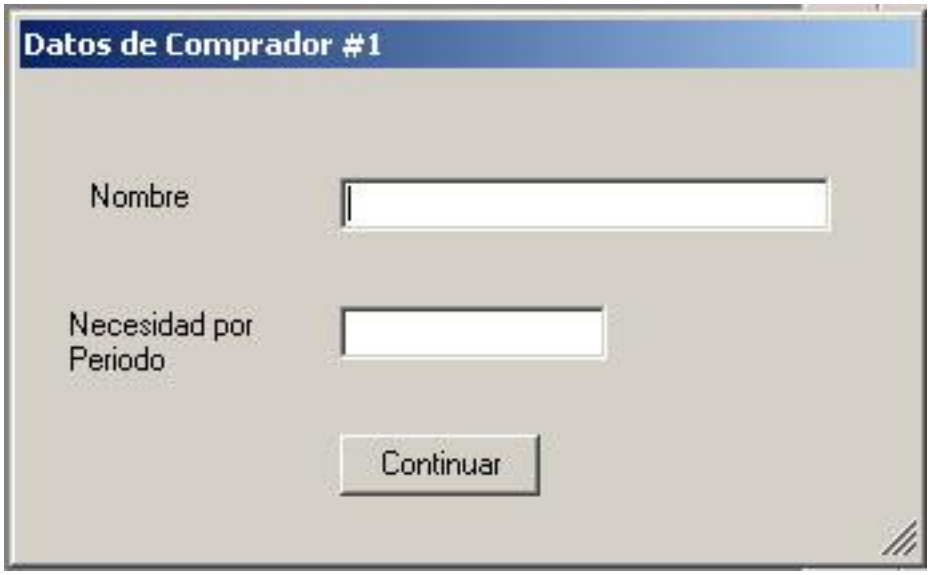
En esta ventana se elije la base de datos que se utilizará en el programa para guardar los resultados.

Ventana para especificar comprador.

A esta ventana se accede haciendo clic sobre el botón ‘Continuar’ en la Ventana Principal.

En esta ventana se escriben los datos de los compradores que se desea tener en la simulación.

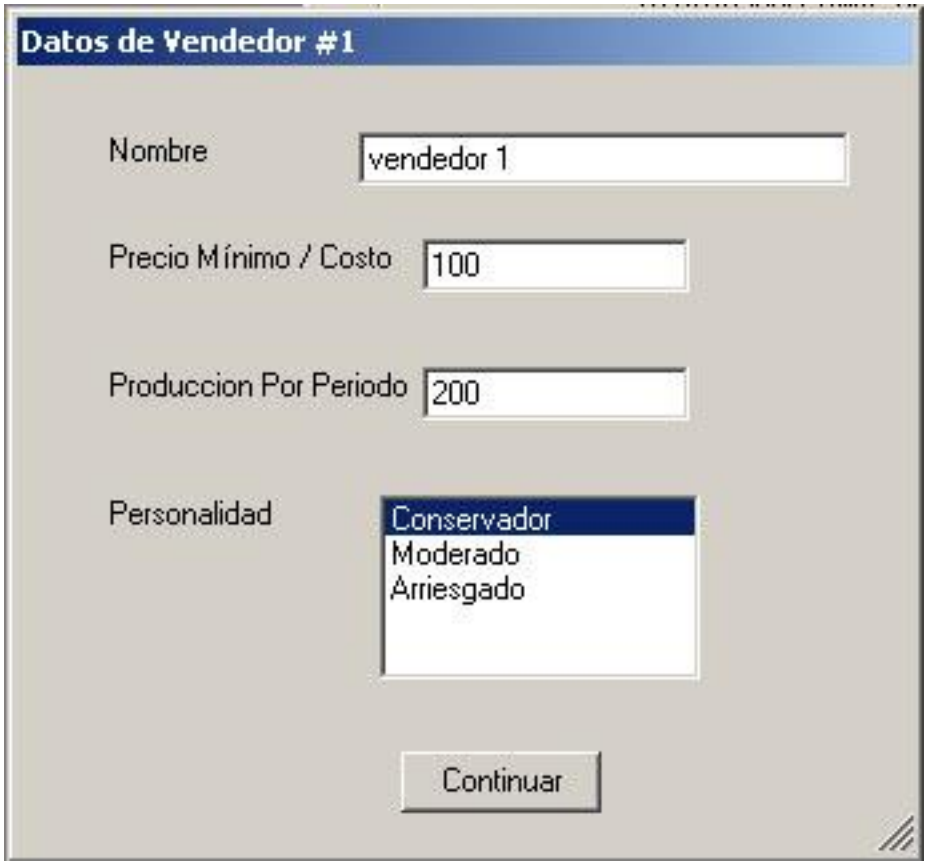
Ventana para especificar comprador simplificada.



A esta ventana se accede haciendo clic sobre el botón ‘Continuar’ en la Ventana Principal, pero solo si los compradores no participan en el pool.

En esta ventana se escriben los datos de los compradores que se desea tener en la simulación. Esta ventana tiene menos opciones que la anterior ya que si los compradores no participan en el pool, no es necesario especificar ciertas opciones.

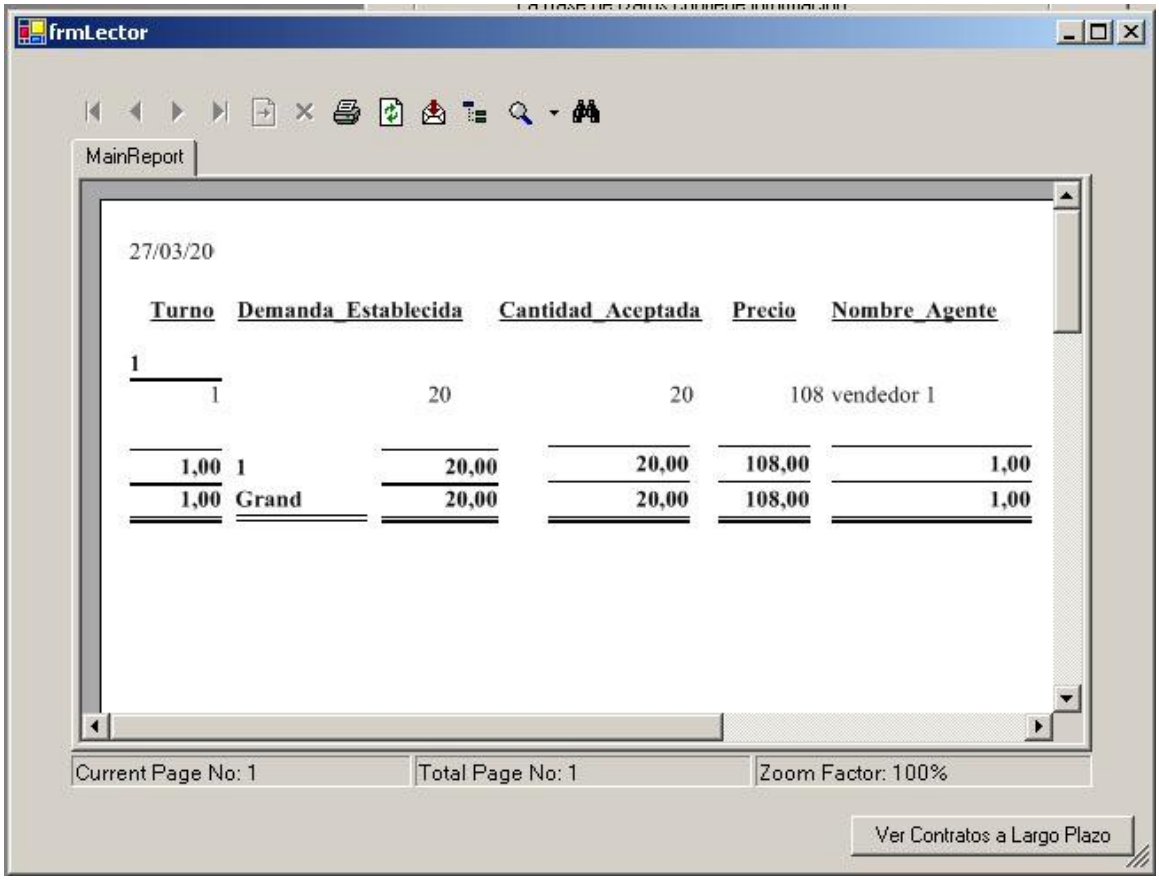
Ventana para especificar vendedor.



A esta ventana se accede después de haber terminado de especificar todos los compradores que participan en la simulación.

En esta ventana se escriben los datos de los vendedores que se desea tener en la simulación.

Ventana de Reporte.



A esta ventana se accede haciendo clic sobre el botón ‘Ver Reporte’ en la Ventana Principal.

En esta ventana se pueden ver los reportes de las ofertas hechas al pool en cada turno y sus resultados, y también el reporte de los contratos a largo plazo.

APÉNDICE 4 – MANUAL DE USUARIO

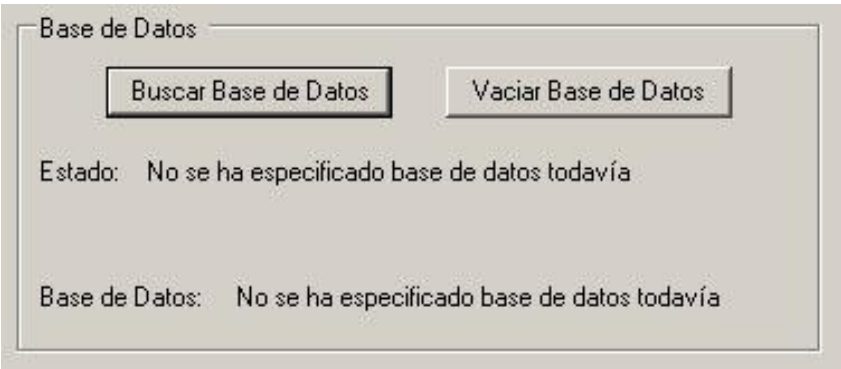
Instalación.

Para instalar el programa solo hay que correr el instalador que se llama Setup.msi

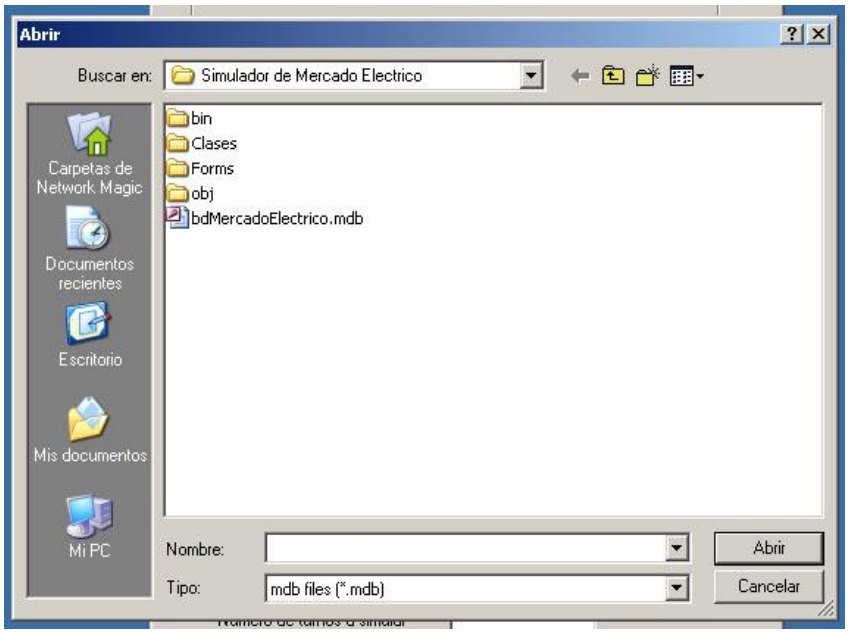
Para realizar una simulación.

Una vez que haya aparecido la pantalla principal, se podrá proceder con la elección del tipo de simulacro que se desea realizar.

1 – Escoger una base de datos.



El primer paso que se debe tomar es la elección de la base de datos donde se van a guardar los resultados obtenidos con la simulación. Para esto hay que hacer clic sobre el botón 'Buscar Base de Datos'. Se abrirá otra ventana donde podrá recorrer todos los archivos en la computadora hasta encontrar la base de datos que se desea usar.



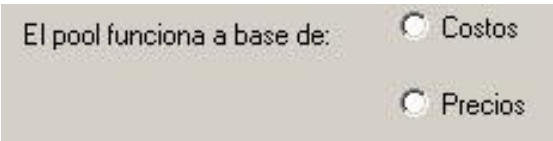
El programa solo acepta bases de datos de Access, y solo las que tienen la misma estructura adecuada para guardar la información de los resultados. Si no cumple con este requisito, la simulación no se podrá realizar y saldrá un mensaje de error advirtiéndole que no se puede utilizar la base de datos escogida. Una vez escogida la base de datos en la sección de 'Estado' deberá decir que la base de datos ha sido aceptada.

Se recomienda solo utilizar la base de datos que viene con el programa, bdMercadoEléctrico.mdb o una copia de la misma.

Si la base de datos contiene información, lo cual se indicará también en la sección de 'Estado' se debe vaciar la base de datos, ya que de lo contrario tampoco se podrá realizar la simulación. Para vaciar la base de datos hay que hacer clic en el botón 'Vaciar Base de Datos'.

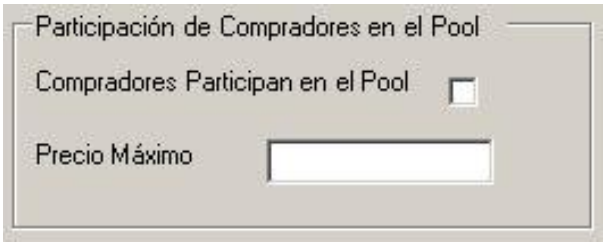
2 – Seleccionar el tipo de simulación a realizar.

Hay que seleccionar si el pool funciona a base de costos o a base de precios.



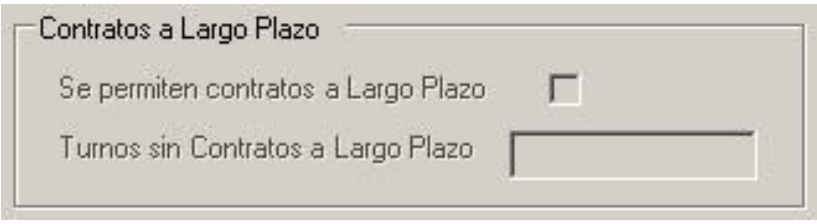
Si el pool funciona a base de costos, eso significa que los precios de los vendedores serán fijos y escogidos por el usuario antes de iniciar la simulación. Si el pool funciona a base de precios, eso significa que los vendedores pueden escoger sus propios precios.

En la sección 'Participación de Compradores en el Pool' el usuario puede escoger si los compradores pueden enviar ofertas al pool, o solo los vendedores.



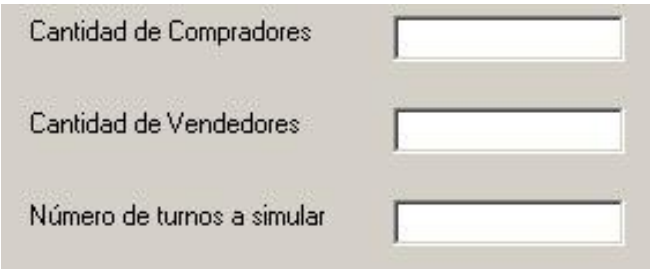
En el bloque de texto 'Precio Máximo' se debe poner el precio máximo que puede alcanzar la energía eléctrica si es que no se permite a los compradores participar. Esto se debe a que sin los compradores ofertando precios, no existe un precio tope para la energía eléctrica.

En la sección de 'Contratos a Largo Plazo' el usuario escoge si se permiten los contratos a largo plazo o no.



Los contratos a largo plazo solo se pueden permitir en una simulación con un pool que funcione a base de precios y donde se permita a los compradores participar. Esto se debe a que en un pool a costo los agentes vendedores no aprenden a predecir el mercado ya que solamente ofertan el mismo precio, y no es posible para ellos saber si un contrato es bueno para ellos o no. Lo mismo ocurre con los compradores cuando no se les permite ofertar en el pool. El usuario también tiene la opción de poner cuantos turnos sin contratos a largo plazo tendrá la simulación. Esto sirve para darle a los agentes compradores y vendedores la oportunidad de aprende como se comporta el mercado antes de que intenten hacer contratos entre ellos.

3 – Elegir cuantos compradores, vendedores y turnos tendrá la simulación.

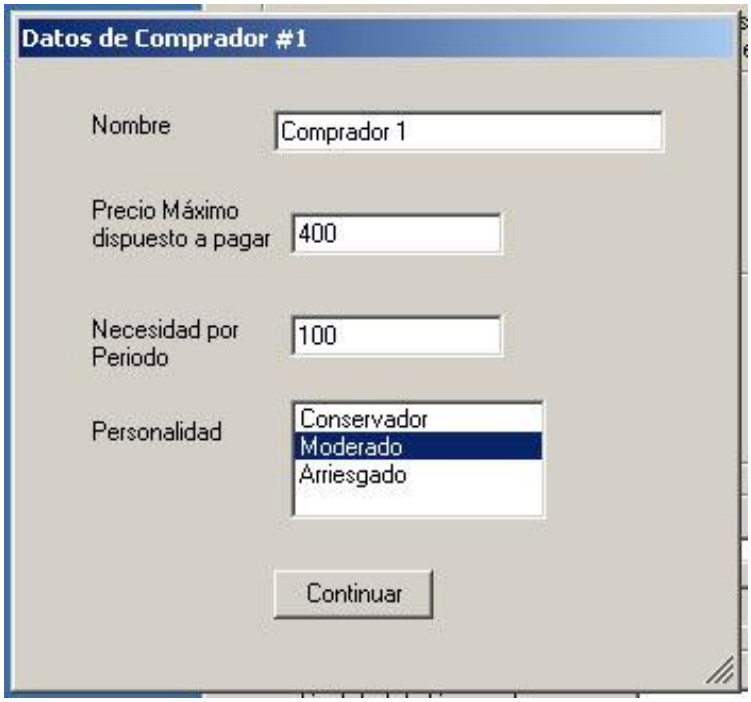


En los cuadros de texto de Compradores, Vendedores y turnos se debe escribir el número deseado de estos. Teóricamente se pueden poner 9999 compradores, y 9999 vendedores y un número gigantesco de turnos, pero no es recomendado ya que una simulación con cinco vendedores, cinco compradores y tres mil turnos puede tardar alrededor de ocho horas.

4 – Especificar los compradores y vendedores.

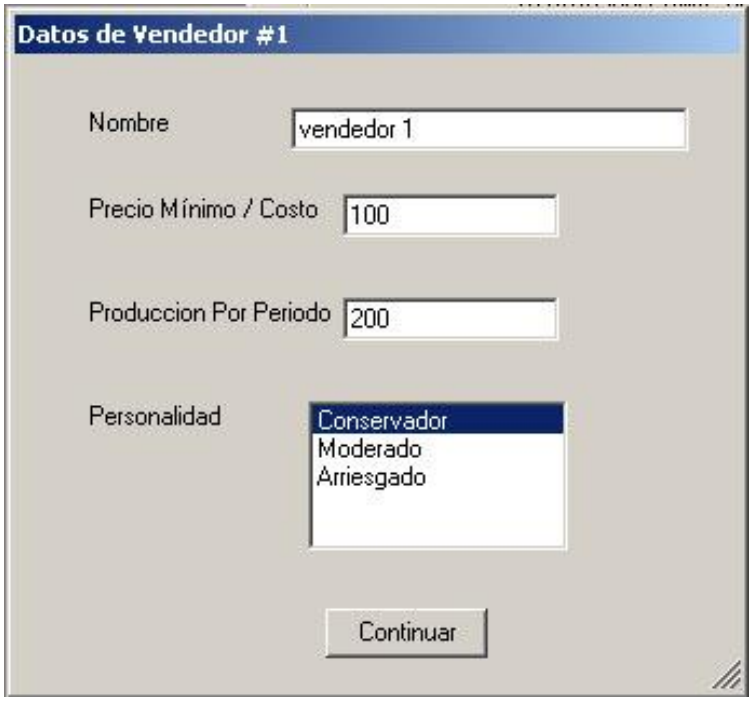
Una vez que se haya escogido el tipo de simulación, hay que hacer clic en el botón ‘Continuar’. Si se ha cometido un error, como por ejemplo escribir letras donde se debía escribir un número, o no se ha escogido una base de datos apropiada, el programa dará una advertencia y no procederá con la simulación hasta que se haya corregido el error.

Sí todos los datos han sido ingresados correctamente, el programa mostrará la ventana de ‘Datos del Comprador’ donde se deben ingresar los datos de cada comprador que participará en la simulación.



Personalidad solo afecta que tan arriesgado es el comprador al momento de ofertar si no tiene una regla específica que le diga lo que tiene que hacer. Un comprador conservador preferirá enviar precios bajos, uno moderado precios ni muy bajos ni muy altos, mientras que el arriesgado enviará precios altos.

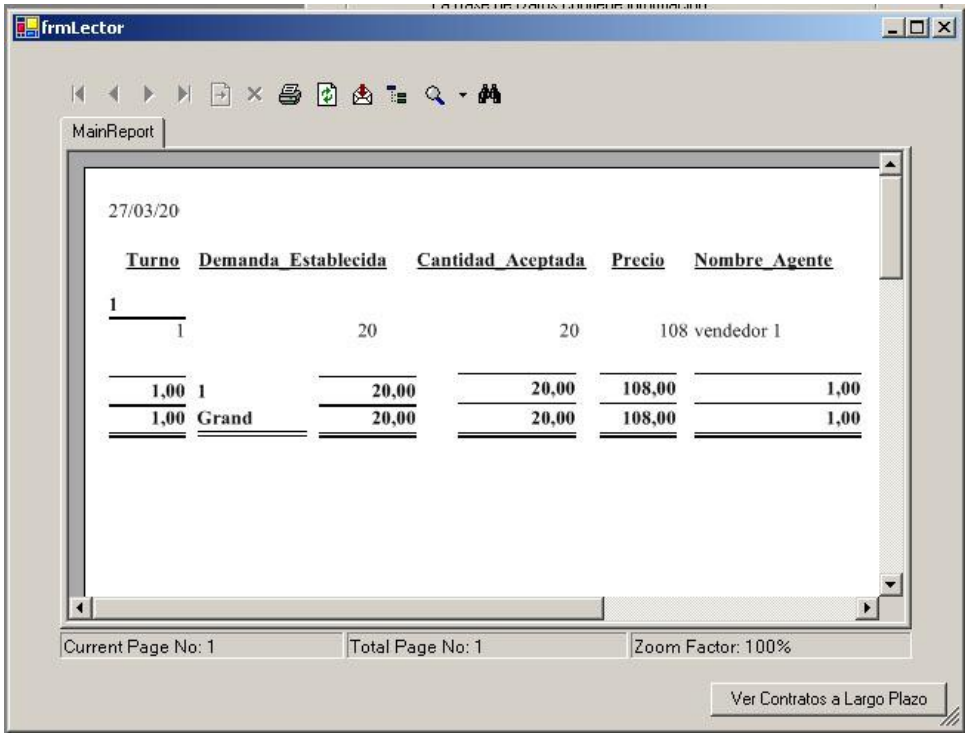
Después de haber especificado los compradores deseados aparecerán las ventanas para especificar los vendedores.



La única diferencia entre vendedores y compradores es que las personalidades funcionan de diferente manera: Vendedores conservadores piden precios altos, los moderados ni muy bajos ni muy altos y los arriesgados precios bajos.

Una vez que se hayan especificado todos los vendedores correctamente, empezará la simulación. El programa indicará cuando esta haya terminado.

Revisión de los Resultados.



Para revisar los resultados de una simulación hay que hacer clic en el botón 'Ver Reporte' que abrirá otra ventana conteniendo el reporte de las acciones de cada comprador y vendedor y el resultado de la subasta.

Para ver los resultados de los contratos a largo plazo hay que hacer clic en el botón 'Ver Contratos a Largo Plazo'.

El reporte está desplayado en páginas, y para moverse entre las páginas debe hacer clic en los botones de flechas.

El resto de botones en el reporte son los usuales botones de Windows con los cuales el usuario ya está familiarizado.

Bibliografía.

1 - José Layana Ch. Características del Mercado Eléctrico del Ecuador. ESPOL.

2 - Anthony J. Bagnall, George D. Smith. A Multi-Agent Model of the UK Market in Electricity Generation. 2005.

3 - V. Vishwanathan, J. McCalley, V. Honavar. A Multiagent System Infrastructure and Negotiation Framework for Electric Power Systems. Iowa State University. 2001.

4 - Isabel Praca, Carlos Ramos, Zita Vale, Manuel Cordeiro. MASCEM: A Multiagent System That Simulates Competitive Electricity Markets. 2003

5 - Stuart Russell, Peter Norvig. Inteligencia Artificial: Un enfoque moderno. 1996.

6 - Michael Wooldridge. Intelligent Agents. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.

7 - Larry Bull. Learning Classifying Systems: A Brief Introduction.