UNIVERSIDAD SIMÓN BOLÍVAR

Ingeniería de la Computación

Diseño e Implementación de un Agente de Corretaje en una Cadena de Suministros en un Ambiente Simulado

Por

María Franco y Celso Gorrín

Proyecto de Grado

Presentado ante la Ilustre Universidad Simón Bolívar

como Requerimiento Parcial para Optar el Título de

Ingeniero en Computación

Sartenejas, Octubre de 2007

UNIVERSIDAD SIMÓN BOLÍVAR

DECANATO DE ESTUDIOS PROFESIONALES

COORDINACIÓN DE INGENIERÍA DE COMPUTACIÓN

ACTA FINAL DEL PROYECTO DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN AGENTE DE CORRETAJE EN UNA CADENA DE SUMINISTROS EN UN AMBIENTE SIMULADO

Presentado Por:

MARÍA FRANCO Y CELSO GORRÍN

Este proyecto de Grado ha sido aprobado por el siguiente jurado examinador:

Prof. Blai Bonet

Prof. Carolina Chang

Prof. Ivette Carolina Martínez (Tutor Académico)

SARTENEJAS, 26 de octubre de 2007

Diseño e Implementación de un Agente de Corretaje en una Cadena de Suministros en un Ambiente Simulado

Por

María Franco y Celso Gorrín

RESUMEN

La *Trading Agent Competition* en su categoría *Supply Chain Management* es un foro internacional en el cual equipos de todas partes del mundo elaboran agentes que deben controlar sendas compañías de ensamblaje de computadoras dentro de una simulación. El principal objetivo que persigue esta competencia es fomentar la investigación en las áreas de inteligencia artificial, cadenas de suministros y sistemas multiagentes.

La simulación presenta a los equipos participantes ciertos problemas, algunos de éstos son: escoger las peticiones de presupuesto que se atenderán, determinar el precio de venta de los bienes ofertados, planificar las agendas de fabricación y envíos y procurar los componentes.

El objetivo de esta investigación fue diseñar un agente llamado TicTACtoe cuya principal característica es el empleo del sistema de clasificadores XCSR de Wilson para decidir el precio de venta de los productos finales a sus clientes. Se incluyó una mejora en el sistema de clasificadores, la cual se denominó bloqueo de clasificadores; ésta permite el uso de XCSR en un ambiente donde la recompensa no se recibe inmediatamente, como es el caso de la TAC SCM.

Las distintas pruebas y experimentos realizados sugieren que el aprendizaje mediante XCSR logra generar una serie de reglas que resuelven de manera satisfactoria el problema de determinar el precio de venta de los productos en la TAC SCM. Además muestran que el bloqueo de clasificadores mejora el desempeño del aprendizaje a través de XCSR en ambientes con retraso en la recompensa.

Agradecimientos

A Dios en primer lugar por haber culminado satisfactoriamente esta investigación.

A nuestros padres y familiares: Ninfa La Rosa, Carlos Franco, Celso Gorrín, Elena González, Aneleh Gorrín y Juan Carlos Franco; por su apoyo incondicional en los momentos difíciles.

A nuestra tutora Ivette Carolina Martínez y a David Ojeda por acompañarnos y guiarnos durante todo el desarrollo de nuestro proyecto.

Al Grupo de Inteligencia Artificial de la Universidad Simón Bolívar por proveernos de espacios y recursos para la investigación y la discusión.

A todos nuestros amigos, por su compañía y apoyo.

Índice general

Índice de cuadros

Índice de figuras

Glosario de Términos

API (*Application Program Interface*) Interfaz de programación que facilita el desarrollo de aplicaciones.

Cadena de suministros Red de instalaciones y recursos encargada de obtener un producto o servicio inicial y transformarlo hasta obtener un producto final para finalmente entregarlo al consumidor.

RFQ (*Request-for-Quote*) Petición de presupuesto. Los dos tipos de RFQ existentes son los enviados por los consumidores al agente manufacturador y los enviados por este último a sus proveedores. Para evitar posibles confusiones entre ambos tipos de RFQ, hemos llamado a los primeros RFQc y a los segundos RFQp.

SCM (*Supply Chain Management* o Manejo de Cadena de Suministros) Proceso de planificación, control e implementación de las operaciones relacionadas con el manejo de flujos de bienes desde el proveedor hasta el cliente final, pasando por la fábrica, almacenamiento, etc.

SKU (*Stock Keeping Unit*) Código que identifica unívocamente a cada uno de los tipos de computadoras que pueden armarse en el mercado de la TAC SCM.

Subsumir Incorporar algo como parte de una síntesis o clasificación más amplia.

TAC (*Trading Agent Competition*) Competencia organizada por el Instituto Sueco de Ciencias de la Computación donde los equipos participantes implementan agentes que deben gestionar una cadena de suministros simulada.

XCS Sistema de clasificadores genéticos propuestos por Wilson cuya principal característica es el empleo de una medida de la precisión de la predicción de la recompensa como utilidad (*fitness*) de los clasificadores.

XCSR Variante de XCS que permite el uso de rasgos cuyos valores pertenecen al conjunto de los números reales.

Capítulo 1

Introducción

La *Trading Agent Competition* (TAC) es definida por sus creadores como "un foro internacional diseñado para promover y fomentar investigación de calidad en el problema de los agentes de comercio"[TAC Team, 2007].

El comercio, una de las actividades humanas más importantes, consiste en la compra y venta de bienes y servicios. Las empresas, que cumplen un papel importante en el comercio, deben decidir qué procesos llevar a cabo para transformar los recursos que reciben de sus proveedores en bienes o servicios que finalmente venderán a sus clientes. Al sistema formado por estos procesos, así como por las personas, información y recursos involucrados en la realización de los mismos se le conoce como cadena de suministros.

Las cadenas de suministros de hoy en día son principalmente estáticas, es decir, están basadas en la confianza y en relaciones a largo plazo establecidas entre aliados de negocio; sin embargo, debido a que las condiciones de los mercados suelen ser cambiantes, se cree que estableciendo cadenas de suministros más dinámicas podrían hacerse mejores asociaciones entre clientes y proveedores[Collins et ál., 2006].

La TAC SCM (*Supply Chain Management*) fue diseñada para enfrentar a sus participantes a los retos típicos que representa el manejo de una cadena de suministros dinámica. Cada equipo participante debe crear un agente manufacturador capaz de tomar las decisiones necesarias para manejar una compañía ensambladora de computadoras dentro de una simulación.

El objetivo general de este trabajo es la construcción de una solución para el problema de la TAC SCM. Entre los objetivos específicos de esta investigación se encuentran: analizar el ambiente de la TAC SCM, analizar soluciones previas exitosas a los problemas presentados en esta competencia, diseñar e implementar un agente base capaz de resolver dichos problemas y analizar e integrar el mecanismo de aprendizaje en una de las estrategias del agente base.

En vista del análisis impulsado por estos objetivos y la experimentación con diversos mecanismos de aprendizaje surgieron nuevas metas. La primera es crear una estrategia de decisión del precio final de venta que emplee la computación evolutiva y, más específicamente, el sistema de clasificadores genéticos XCSR (XCS con entradas reales) de [Wilson, 2000]. La segunda meta es atacar los problemas de compra y producción con estrategias estáticas sencillas basadas en soluciones previas exitosas. La tercera es conseguir buenas combinaciones de parámetros de aprendizaje para luego comparar el desempeño de las soluciones obtenidas contra otras soluciones que no emplean técnicas de aprendizaje.

Desde la creación de la competencia en 2003, participan anualmente una treintena de equipos. Algunas de las soluciones consultadas para la elaboración de nuestro trabajo son PhantAgent[Stan et ál., 2006], Mertacor[Mitkas et ál., 2006], CMieux[Benisch et ál., 2006], Botticelli[Benisch et ál., 2004], Tiancalli[Macías et ál., 2006], TacTex[Pardoe y Stone, 2006], KrokodilAgent[Petrić, 2005], RedAgent[Eriksson et ál., 2006], FreeAgent[Eriksson et ál., 2006], DeepMaize[Eriksson et ál., 2006], así como la de SouthamptonSCM[He et ál., 2006]. De todas las anteriores, fueron PhantAgent, TacTex y CMieux las que sirvieron como base y punto de referencia para nuestra solución.

El agente TicTACtoe cuenta con tres estrategias principales para resolver cada uno de los subproblemas de la competencia: la venta, la producción y la compra. La estrategia de compra consiste en evaluar la demanda esperada de los próximos 10 días para ordenar componentes; mientras que la estrategia de producción consiste en fabricar cada orden tan tarde como sea posible. Por otro lado, la estrategia de venta emplea un sistema de clasificadores XCSR para determinar el precio de final de los productos.

Para implementar satisfactoriamente las estrategias del agente TicTACtoe fue necesario diseñar e implementar una estructura de datos que funciona como una agenda. En la agenda el agente guarda registro de los compromisos con sus clientes, así como la cantidad y tipo de componentes que se espera sean entregados en cada uno de los días de la simulación. De este modo, el agente puede tomar decisiones informadas en diversas etapas de la simulación como, por ejemplo, al momento de ordenar componentes o crear la agenda de producción y

envíos que debe entregar a la fábrica en cada día de la simulación.

Por otro lado, los XCS son un mecanismo de aprendizaje por reforzamiento propuesto por [Wilson, 1995] basándose en el trabajo en sistemas de clasificadores genéticos de [Holland, 1976]. El uso de XCS representa ciertas ventajas debido a las cuales fue escogido para ser incorporado al agente. Entre ellas se encuentran una fácil clasificación de datos no separables linealmente, la generación de un conjunto reducido de reglas suficientemente completas, la capacidad de poder detener el algoritmo en cualquier momento y luego seguir aprendiendo sobre la misma población y la posibilidad de analizar las reglas de clasificación una vez que han sido descubiertas.

Al intentar incorporar el XCSR a la solución se consiguió que el algoritmo original no se adaptaba adecuadamente al ambiente de simulación, pues la recompensa a las acciones llevadas a cabo no se recibe en el mismo momento en que se realizan, sino algunos días de simulación después. Para hacer frente a este problema, se ideó un mecanismo al que denominamos bloqueo de clasificadores, que consiste en no permitir que un clasificador sea borrado o submido si está en espera de ser recompensado por alguna acción tomada en el pasado.

Para evaluar el desempeño de los agentes se tomaron en cuenta ciertas variables; principalmente el balance final de la cuenta bancaria y el número de órdenes atendidas. En base a estas medidas, se comparó al agente TicTACtoe con distintas combinaciones de parámetros de aprendizaje contra el agente de muestra proporcionado por los creadores de la competencia y contra dos variantes de TicTACtoe: una que decide el precio de venta de forma aleatoria y otra que emplea una estrategia estática.

Mediante estas comparaciones, se logró concluir que el aprendizaje con el sistema de clasificadores XCSR resuelve de manera satisfactoria el problema de ventas de TAC SCM; además de constituir una mejor solución comparada con otras soluciones alternativas presentadas en este trabajo, como lo son la estrategia estática y la estrategia aleatoria. También se mostró que el bloqueo de clasificadores logra mejorar el desempeño de sistema de clasificadores XCSR en un ambiente en donde se presenta un retraso en la recompensa.

Este informe consta de seis capítulos. El Capítulo 2 presenta la descripción general de la TAC SCM, una introducción al sistema de clasificadores XCSR y algunas soluciones previas propuestas por otros equipos participantes de la TAC SCM. El Capítulo 3 presenta el diseño del agente TicTACtoe y las estrategias adoptadas por el mismo para afrontar los distintos retos de la TAC SCM. El Capítulo 4 describe la implementación del sistema de clasificadores XCSR realizada por el equipo TicTACtoe. El Capítulo 5 presenta el diseño de los experimentos, los resultados obtenidos y un análisis de los mismos. Finalmente, el Capítulo 6 presenta las conclusiones y recomendaciones para futuros trabajos.

Capítulo 2

Marco Teórico

En este capítulo se presenta la descripción general de la *Trading Agent Competition* (TAC) y, más específicamente, de su categoría *Supply Chain Management* (SCM), para la cual se desarrolló nuestro trabajo. También contiene una introducción al sistema de clasificadores XCSR propuesto por [Wilson, 1995] que constituye la herramienta más importante empleada en nuestra solución al problema TAC SCM. Finalmente, se presenta una breve reseña de algunas soluciones exitosas propuestas por equipos participantes en años anteriores que sirvieron de inspiración para nuestro trabajo, así como algunas investigaciones previas donde se aplica aprendizaje a agentes de este tipo. Ésta es la teoría sobre la que se basa la solución que proponemos para la categoría SCM de la TAC.

2.1. *Trading Agent Competition*

La *Trading Agent Competition* (TAC) surge por inspiración de otras competencias en el campo de la inteligencia artificial similares como, por ejemplo, RoboCup[Eriksson et ál., 2006]. La TAC fue concebida con la idea de crear un lugar para que los agentes de *software* pudieran actuar en un mercado simulado comprando y vendiendo bienes[Eriksson et ál., 2006].

Esta competencia cuenta con dos categorías principales:

TAC Classic: fue la primera categoría de la competencia. Consiste en elaborar agentes que construyen planes de viaje al mejor precio posible para luego venderlos a sus clientes[Eriksson et ál., 2006].

TAC Supply Chain Management (SCM): esta categoría se incluyó en la competencia a partir del año 2003 y fue creada en conjunto por la Universidad Carnegie Mellon y el Instituto Sueco de Ciencias de la Computación[TAC Team, 2007].

En esta categoría los agentes controlan una empresa manufacturadora de computadoras y su deber es gestionar la cadena de suministros de la misma; esto es, el proceso de

elaboración del producto final desde la compra de componentes hasta el envío del mismo a sus clientes. Ésta es la categoría hacia la que está orientado el desarrollo de nuestro trabajo.

2.2. TAC SCM

Según [Collins et ál., 2006], "la competencia TAC SCM fue diseñada para capturar muchos de los retos involucrados en el apoyo de prácticas dinámicas en cadenas de suministros, manteniendo las reglas del juego suficientemente simples para atraer un gran número de competidores a presentar sus participaciones.". En este contexto, los equipos participantes desarrollan agentes inteligentes capaces de manejar los puntos claves de una cadena de suministro (decidir cuáles órdenes aceptar, decidir el precio de venta, competir con otros agentes por el mercado, compra de componentes, ensamblaje del producto final y envío al consumidor).

El problema de la competencia TAC SCM es la maximización de la ganancia recibida por concepto de la venta de computadoras por parte de los agentes manufacturadores; entendiéndose como ganancia la diferencia entre ingresos y egresos al final de la simulación[Collins et ál., 2006].

2.3. Ambiente de simulación

La simulación de un juego TAC SCM es manejada por un servidor con el cual los agentes manufacturadores se comunican por medio de conexiones basadas en el protocolo IP (*Internet Protocol*) para enviar al mismo las acciones que desean ejecutar y recibir de éste el estado de la simulación. El resto de los actores y entidades de la simulación (clientes, proveedores, banco, fábrica, etc.) son controlados por el servidor y fueron codificados dentro del mismo por los creadores de la competencia. Este servidor guarda la información de las corridas que puede ser usada posteriormente para reproducir una simulación anterior, por ejemplo.

La codificación de los agentes manufacturadores se hace en el lenguaje de programación JAVA y, para facilitar el desarrollo, los creadores de la competencia proveen a los equipos de

una interfaz de programación (API) con la que se encapsulan los mensajes entre el servidor y el agente. Esta interfaz implementa métodos que permiten al agente notificar al servidor sobre las acciones que desea llevar a cabo y recibir del mismo el estado de la simulación.

2.3.1. Actores

En cada simulación de la TAC SCM intervienen tres actores: los clientes, los agentes manufacturadores y los proveedores. Los detalles sobre los mismos se discuten a continuación.

Clientes

Los clientes son agentes que compran computadoras a los agentes manufacturadores. El comportamiento de éstos fue programado por los creadores de la competencia y están contenidos dentro del servidor.

Los clientes envían al agente manufacturador algunos RFQ[1] (*Request-for-Quote* o petición de presupuesto) solicitando computadoras. Cada RFQc consta de: un tipo de computadora, una cantidad, una fecha de entrega, un precio de reserva (el precio máximo que el cliente está dispuesto a pagar) y una penalización que será cobrada por cada día de retraso en el envío.

Cada tipo de computadora está identificado con un código llamado *Stock Keeping Unit* (SKU) y consiste en una combinación distinta de los cuatro componentes que la conforman; éstos son: procesador, tarjeta madre, memoria RAM y disco duro. De igual manera, cada componente está también identificado con un código, cuenta con un precio base (que no es más que un precio de referencia empleado en el cálculo del precio real) y pertenece a un segmento de mercado específico. En los Cuadros 1 y 2 se muestran los distintos componentes y productos disponibles, respectivamente, así como sus características.

El agente debe enviar a los clientes ofertas en respuesta a los RFQc que éstos le enviaron el día anterior si tiene interés de concretar la venta. Los clientes sólo aceptarán la oferta si cumple las siguientes condiciones:

[1]En adelante, nos referiremos a los RFQ enviados por los clientes a los agentes manufacturadores como RFQc o RFQ de clientes.

Componente	Precio base	Fabricante	Descripción
100	1000	Pintel	CPU Pintel, 2.0 GHz
101	1500	Pintel	CPU Pintel, 5.0 GHz
110	1000	IMD	CPU IMD, 2.0 GHz
111	1500	IMD	CPU IMD, 5.0 GHz
200	250	Basus, Macrostar	Tarjeta madre Pintel
210	250	Basus, Macrostar	Tarjeta madre IMD
300	100	MEC, Queenmax	Memoria, 1 GB
301	200	MEC, Queenmax	Memoria, 2 GB
400	300	Watergate, Mintor	Disco duro, 300 GB
401	400	Watergate, Mintor	Disco duro, 500 GB

Fuente: [Collins et ál., 2006].

Cuadro 1: Catálogo de componentes.

- Se oferta la misma cantidad de productos que el cliente solicitó en el RFQc al que responde.
- La orden promete la entrega en la fecha estipulada en el RFQc.
- El precio de venta ofertado es menor o igual que el precio de reserva especificado en el RFQc correspondiente.

El cliente finalmente informa al agente si acepta la oferta por medio de un recibo que se entrega al día siguiente. La oferta ganadora es aquélla cuyo precio ofrecido es el menor entre todas las que se recibieron. En caso de haber un empate, se resuelve tomando una oferta al azar entre las que lo conforman.

Las órdenes se satisfacen cuando el agente entrega los productos al cliente. El pago al agente se efectúa en la fecha de entrega estipulada en la oferta o al día siguiente a la entrega real (el que ocurra más tarde). Por cada día posterior a la fecha de entrega que pase sin que se haya efectuado la entrega el cliente cobrará al agente manufacturador la respectiva penalización. Las penalizaciones sólo se cobran durante cinco días; al sexto día se cancela la orden y no se cobran más penalizaciones.

Todas las órdenes que no hayan sido entregadas al final del juego son canceladas y se cobran las penalizaciones correspondientes debido a que nunca podrán ser entregadas.

Agente manufacturador

Cada equipo que compite en un juego de la TAC SCM coloca en el mismo un agente que debe procurar un inventario de componentes, tratar de ganar órdenes de los clientes, gestionar las actividades diarias de su fábrica y enviar efectivamente los productos que le son ordenados.

Al inicio de cada uno de los días de la simulación los agentes reciben la siguiente información de sus clientes, el banco, la fábrica y sus proveedores:

De sus clientes: peticiones de presupuestos (RFQc) y las órdenes en respuesta a presupuestos enviados el día anterior.

De la fábrica: la información sobre el inventario tanto de componentes como de productos terminados.

De sus proveedores: ofertas en respuesta a las peticiones de presupuesto que el agente les haya enviado el día anterior y componentes en respuesta a órdenes recibidas con anterioridad.

Del banco: el balance de la cuenta del agente (la cantidad de dinero que el mismo posee).

En base a esta información que recibe del ambiente el agente debe realizar las siguientes tareas cada día:

- Decidir a qué peticiones de presupuesto responder, lo que implica decidir también el precio al que se desea ofrecer al cliente los productos solicitados por éste.
- Enviar a su fábrica una agenda diaria de producción y una de entregas de productos finales a sus clientes.

- Enviar peticiones de presupuesto a sus proveedores para mantener su inventario de componentes y responder a los presupuestos enviados por éstos con órdenes si desea concretar la venta.

Cada agente está dotado de una fábrica que es la entidad encargada de la elaboración, envío y almacenamiento de los productos. Los detalles sobre la misma se explican en la Sección 2.3.2.

Proveedores

Las simulaciones de la TAC SCM cuentan con ocho proveedores. Para cada tipo de componente existen uno o dos proveedores que lo fabrican. El catálogo completo de componentes puede consultarse en el cuadro 1.

Durante cada uno de los días de la simulación, cada uno de los agentes manufacturadores pueden enviar hasta cinco RFQ[2] a cada proveedor por cada uno de los productos que el mismo vende. Un RFQp r equivale a una solicitud de una cantidad q_r de un componente c para su entrega en $i_r + 1$ días (donde i_r es el tiempo que tomará la producción de la orden asociada a r). Los RFQp pueden, opcionalmente, especificar un precio de reserva ρ_r por unidad (el precio máximo por unidad que el agente manufacturador está dispuesto a pagar por el componente). Si la cantidad q_r es igual a cero, se ofertará un precio pero no se responderá al RFQp con una oferta. Esto le permite a un agente conocer el precio de venta de un componente, sin tener que comprarlo. Si la fecha de entrega de un RFQp es posterior al fin de la simulación o menor que dos días en el futuro no son consideradas por el proveedor.

El proveedor deberá responder a cada RFQp r con una oferta al día siguiente. La misma contiene un precio P_r, una cantidad ajustada q'_r y una fecha de entrega. La cantidad reducida q'_r es la cantidad de productos que el agente manufacturador se compromete a entregar, que será menor o igual que la cantidad original solicitada q_r o cero, en caso de que no pueda cumplirse con la restricción del precio de reserva. En el supuesto de que el proveedor no

[2]En adelante, nos referiremos a los RFQ enviados por los manufacturadores a los proveedores como RFQp o RFQ de proveedores.

pueda –o no desee– comprometerse a entregar la orden completa, puede entregar hasta dos de los siguientes tipos de ofertas corregidas:

- Una orden *parcial*, en la que el proveedor se compromete a entregar, para la misma fecha original, una cantidad reducida del producto ordenado.
- Una orden *terminada tan pronto como sea posible* (*earliest complete*, en inglés), en la que el compromiso de entrega es retrasado pero se entrega la cantidad completa.

Todas las ofertas sólo son válidas durante el día en el que llegan. Si un agente manufacturador desea aceptar una oferta debe confirmarlo por medio de una orden enviada al proveedor.

Cada proveedor calcula diariamente la "reputación" de cada agente manufacturador a como sigue:

$$rep_a = \frac{\text{mín}(apr, \zeta_a)}{apr}$$

donde apr es una constante denominada "*acceptable purchase ratio*" y se introduce en la fórmula para evitar que un agente sea penalizado por consultar los precios de dos proveedores y comprar sólo a uno. El apr tiene dos posibles valores: si los productos vendidos por el proveedor que calcula la reputación no los vende ningún otro proveedor (tal es el caso de los CPU), apr tiene por valor 0,75; en caso contrario, su valor es 0,45.

Por otro lado, ζ_a, que se denomina *order ratio* (tasa de órdenes) del agente a se calcula como

$$\zeta_a = \frac{cantidadComprada_a}{cantidadOfertada_a} \quad .$$

Para el cálculo de la tasa de órdenes, $cantidadComprada_a$ es la cantidad total de productos que el agente a ha comprado al proveedor que la calcula y $cantidadOfertada_a$ es, para cada RFQp la mayor de estas tres cantidades:

1. La cantidad en la oferta parcial.
2. La cantidad que realmente se ordenó, en caso de que el agente acepte la orden *earliest-complete*.

3. El 20 % de la cantidad reducida q'_r.

Al inicio del juego $cantidadComprada_a$ y $cantidadOfertada_a$ tienen por valor 2000. Adicionalmente, para que los agentes puedan recuperar su reputación en el tiempo aunque no ordenen o compren nada, ambas cantidades son incrementadas diariamente en 100 unidades.

Este mecanismo de reputaciones evita que un agente malintencionado intente subir los precios de los componentes creando una demanda artificial; esto se lograría solicitando muchos presupuestos que nunca se materializarían en órdenes. De este modo, los proveedores subirían los precios pues estimarían una demanda mayor a la real. Este mecanismo de reputación permite al proveedor tener una idea de la "confiabilidad" de los RFQp enviados por cada agente.

Al día siguiente de recibir los RFQp, los proveedores responden a los mismos por medio de ofertas. Para elegir los RFQp ganadores, los proveedores intentan maximizar sus ganancias mientras tratan a todos los agentes de manera "justa", dándole prioridad a los RFQp de los agentes con mayor reputación y verificando que pueda cumplirse la restricción del precio de reserva.

Finalmente los agentes confirman al proveedor que desean concretar la venta por medio de una orden. En el momento de colocar la orden, los proveedores cobran al agente una inicial de 10 % del valor total del pedido y el resto del dinero será debitado de la cuenta bancaria del agente al momento de la entrega.

2.3.2. Entidades

Los agentes manufacturadores deben interactuar con dos entidades muy importantes en la simulación; éstas son: sus respectivas fábricas y el banco central.

Banco

La simulación cuenta con un banco central en el que todos los agentes manufacturadores tienen una cuenta; inicialmente éstas tienen un balance de cero dólares. El banco informa a diario a los agentes del saldo actual de sus cuentas.

Los movimientos en las cuentas bancarias son realizados automáticamente durante la si-

mulación. Se ingresa dinero en las mismas cuando algún cliente paga por el envío de productos y se retira cuando el agente paga por la compra de componentes o recibe penalizaciones por entregas tardías. Las cuentas bancarias no tienen saldos máximos ni mínimos; los agentes pueden tener saldos negativos.

Las cuentas están sujetas a los intereses que son cargados a diario. Si en el día d el saldo b_d de una cuenta es negativo, el saldo b_{d+1} de la misma para el día siguiente se calcula como

$$b_{d+1} = \left(1 + \frac{\alpha'}{E}\right) b_d + creditos_d - debitos_d$$

donde E es la duración del juego en días y α' es la tasa anual de interés para préstamos y su valor es escogido al azar entre 6 % y 12 %.

Si, en caso contrario, b_d es positivo, b_{d+1} se calcula como sigue:

$$b_{d+1} = \left(1 + \frac{\alpha}{E}\right) b_d + creditos_d - debitos_d$$

donde E es nuevamente la duración de juego en días y α es la tasa anual de interés para ahorros y su valor es $\alpha = \alpha'/2$.

Fábrica

Cada agente manufacturador cuenta con una fábrica con capacidad diaria de producción limitada. Cada tipo de producto requiere de una cantidad distinta de ciclos de fábrica para ser elaborado (ver Cuadro 2).

Luego de recibir del agente la agenda de producción para el día en curso, la fábrica la procesa en orden, elaborando sólo los productos para los que se cuenta con inventario necesario hasta que se agoten los ciclos de producción disponibles para la jornada.

Todos los productos que se fabrican son colocados en inventario al final de la jornada y podrán ser enviados a partir del día siguiente.

El inventario no tiene capacidad máxima y cada producto o componente que permanece almacenado genera un cargo que será descontado de la cuenta bancaria del agente diaria-

SKU	Componentes	Ciclos	Segmento de mercado
1	100, 200, 300, 400	4	Rango bajo
2	100, 200, 300, 401	5	Rango bajo
3	100, 200, 301, 400	5	Rango medio
4	100, 200, 301, 401	6	Rango medio
5	101, 200, 300, 400	5	Rango medio
6	101, 200, 300, 401	6	Rango alto
7	101, 200, 301, 400	6	Rango alto
8	101, 200, 301, 401	7	Rango alto
9	110, 210, 300, 400	4	Rango bajo
10	110, 210, 300, 401	5	Rango bajo
11	110, 210, 301, 400	5	Rango bajo
12	110, 210, 301, 401	6	Rango medio
13	111, 210, 300, 400	5	Rango medio
14	111, 210, 300, 401	6	Rango medio
15	111, 210, 301, 400	6	Rango alto
16	111, 210, 301, 401	7	Rango alto

Fuente: [Collins et ál., 2006].

Cuadro 2: Lista de materiales (*Bill of Materials* o BOM).

mente. El costo del almacenamiento consiste en un porcentaje aleatorio entre el 25 % y el 50 % del valor nominal del producto. Este valor aleatorio es fijo durante toda la simulación.

2.3.3. Día de simulación TAC SCM

Una simulación de TAC SCM consiste de 220 "días" que duran 15 segundos cada uno. En cada día d de la simulación ocurren los siguientes eventos, que son presentados aquí sin orden particular:

- Los proveedores:
  - Reciben los RFQp enviados en el día d.
  - Responden por medio de ofertas a los RFQp enviados el día $d - 1$.

- Reciben de los agentes manufacturadores las órdenes en respuesta a las ofertas del día $d-1$.
- Entregan de componentes a los agentes manufacturadores.

- Los agentes:
  - Reciben los RFQc enviados en el día d.
  - Responden por medio de ofertas a los RFQc enviados el día $d-1$.
  - Reciben de los clientes las órdenes en respuesta a las ofertas del día $d-1$.
  - Envían a la fábrica las agendas de producción y entregas para el día $d+1$.
  - Envían a los proveedores los RFQp del día d.
  - Reciben las ofertas de los proveedores en respuesta a los RFQp del día $d-1$ y responden a las mismas por medio de órdenes.
  - Reciben del banco el saldo de su cuenta en el mismo.
  - Reciben de sus proveedores los componentes pautados para ser entregados durante este día d.

- Los clientes:
  - Envían los RFQc del día d.
  - Reciben las ofertas de los agentes manufacturadores en respuesta a los RFQc del día $d-1$ y responden a las mismas por medio de órdenes.
  - Reciben productos de los agentes manufacturadores.

2.4. Soluciones previas exitosas

En esta sección presentamos algunas soluciones exitosas al problema de la competencia TAC SCM elaboradas por equipos que han participado en la misma en años anteriores. Adicionalmente a las presentadas en esta sección se consultaron otras soluciones; sin embargo, son las listadas a continuación las que sirvieron como base para nuestra solución propuesta.

Para facilitar la comprensión de la estrategia empleada por cada agente hemos seguido el ejemplo de [Eriksson et ál., 2006], quien separa la estrategia global del agente en tres subestrategias: de compra, de producción y de venta.

2.4.1. PhantAgent

PhantAgent[Stan et ál., 2006] fue desarrollado por un equipo del Departamento de Ciencias de la Computación de la Universidad "Politehnica" de Bucarest, en Rumanía. El mismo fue participante en la competencia del año 2006 obteniendo el segundo lugar.

Estrategia de compra

Para determinar la cantidad de componentes que deben ser ordenados se fija un tamaño inicial para el inventario. El tamaño deseado de inventario para el resto de los días se calcula haciendo descender linealmente hasta cero (al final de la simulación) el tamaño inicial.

PhantAgent estima el tamaño del inventario para todos los días hasta el final del juego como sigue

$$quant_i = quant_{i-1} + arv_i - est_i$$

donde $quant_i$ es la cantidad estimada de inventario para el día i, arv_i es el inventario que debería llegar el día i y est_i es la cantidad estimada de componentes usados por día. Esta cantidad comienza como un valor fijo y luego, durante la simulación, es ajustada de acuerdo a la demanda.

Para determinar cuántos componentes deben ordenarse cada día se calcula la diferencia entre el tamaño esperado de inventario para ese día y el tamaño deseado de inventario para el mismo.

PhantAgent coloca tres tipos distintos de órdenes: a largo plazo, a mediano plazo y a corto plazo. Las órdenes a largo plazo se colocan al inicio del juego (entre los días 2 y 5) de modo que estos componentes sean entregados uniformemente a lo largo de la segunda mitad del juego. Su estrategia se basa en la heurística de que si hay muy pocos componentes de algún tipo, entonces estos son más costosos que si los tuviera en grandes cantidades. Esto

le permite comprar un determinado componente a precios un poco más altos si tiene poca existencia del mismo.

Estrategia de producción

PhantAgent fabrica primero los productos que tienen la fecha de entrega prometida más cercana y los entrega apenas estén disponibles. La decisión de cuáles RFQc pueden entregarse a tiempo y la estimación de ocupación de la fábrica se hace en base a los ciclos de fabricación requeridos por las órdenes pendientes de ser manufacturadas. Adicionalmente, PhantAgent tiene como regla que toda orden debe poder fabricarse en un rango de 3 a 4 días.

Si en un determinado día no hay más producción prometida pendiente, PhantAgent fabrica uniformemente productos de todos los tipos hasta que las cantidades almacenadas de cada tipo de producto alcancen un umbral o *threshold* que para la competencia de 2006 fue fijado en 50.

Estrategia de venta

La estrategia de fijación de precio de ventas también emplea una heurística similar a la que emplea en la estrategia de compra. Si un producto final requiere de un componente del cual se tiene poca existencia, éste puede venderse a un precio mayor al que podría ser vendido en caso de tener más existencia de dicho componente.

El precio de venta de un artículo para un determinado día se calcula en base al precio máximo en que fue comprado el mismo a PhantAgent en los días inmediatamente anteriores. Dicho factor aumenta si la fábrica está muy llena y se reducirá cuando la fábrica tenga más capacidad de producción disponible.

2.4.2. TacTex-06

TacTex-06[Pardoe y Stone, 2006] fue desarrollado en la Universidad de Texas en los Estados Unidos de América. Este agente participó en la competencia del año 2006 resultando ganador.

Estrategia de compra

Un módulo al que los autores de TacTex-06 llamaron Manejador de Demanda calcula qué componentes necesita cada día por medio de una predicción de la demanda; la misma la realiza manteniendo una distribución de probabilidad calculada en base a los RFQc recibidos.

Por otro lado, un módulo denominado Manejador de Suministros es el encargado de minimizar el costo de la obtención de los mismos. El Manejador de Suministros, empleando un modelo de predicción entrenado con el algoritmo *Additive Regression with Decision Stumps* de la librería Weka[Witten y Frank, 1999] empleando datos de simulaciones pasadas, estima la disponibilidad y precio de los distintos componentes en el futuro. Adicionalmente, los componentes tienen valores mínimos de inventario (umbrales o *thresholds*) que decrecen linealmente hasta cero en la etapa final del juego.

Una gran mejora con respecto a su versión del año 2005 es que TacTex-06 sólo envía inmediatamente las solicitudes de componentes a los proveedores si estima que, de esperar más tiempo, los precios subirían drásticamente. De este modo se evita crear un inventario innecesario de componentes.

Otro aspecto importante de TacTex-06 es la forma en que decide qué componentes comprar al inicio del juego (cuando los precios son impredecibles). En este sentido, el agente planifica producir cantidades iguales de cada tipo de producto para 80 días y planifica sus compras basado en esto usando datos de juegos pasados para determinar qué componentes se deben comprar al inicio para garantizar suficiente existencia y aprovechar los bajos precios.

Estrategia de producción

Como ya se dijo antes, el Manejador de Demanda cuenta con un modelo de predicción que puede estimar el número de RFQc que serán recibidos en cualquier día futuro. Basado en estas predicciones, dicho manejador puede planificar la producción para los próximos días. Por otro lado, éste también se encarga de ofertar a los RFQc recibidos, producir y entregar las órdenes.

Una de las característica más resaltantes del Manejador de Demanda es la de planificar la producción de los próximos 10 días usando un algoritmo codicioso (*greedy*) para evitar el

uso de la programación lineal que implicaría un costo adicional. Este produce las órdenes tan tarde como sea posible para así no comprometer *a priori* los recursos de fábrica; y de esta manera, darle prioridad a las órdenes que generen ganancias importantes y tengan que ser entregadas a corto plazo.

Estrategia de venta

TacTex-06 cuenta con un sistema de predicción de aceptación de ofertas que supone que los precios ganadores de las ofertas a lo largo del juego siguen una distribución normal y crea una función que asigna a cada precio de oferta posible la probabilidad estimada de que el mismo sea aceptado. Esta función se crea empleando un método llamado *Sampling Importance Resampling Particle Filter*[Arulampalam et ál., 2002]. De este modo, el agente determina a cuáles RFQc ofertar de modo que se maximicen las ganancias suponiendo que toda oferta será aceptada y sujeto a la planificación de la producción elaborada en días anteriores (y, por tanto, a la capacidad de la fábrica).

Una consideración especial de TacTex-06 en cuanto a las ventas es la elección de la estrategia a tomar al final de la competencia. Para tomar esta decisión, el agente determina, usando datos de juegos pasados y técnicas de aprendizaje de máquinas, si se encontrará en uno de los dos escenarios típicos del final del juego: o bien los precios son muy bajos (dado que la mayoría de los agentes está intentando reducir la cantidad de productos finales en inventario a toda costa) o muy altos (dado que la mayoría de los otros agentes prefirieron no producir más). Luego de haber estimado el caso típico en el que se encuentra, TacTex-06 sólo vende al final del juego si cree que las ventas tendrán un alto margen de ganancia.

2.4.3. CMieux

El agente CMieux[Benisch et ál., 2006] fue diseñado e implementado en la Universidad Carnegie Mellon, en el Laboratorio e-Supply Chain Management. Participó en el año 2005 quedando en semifinales. Este agente está constituido por diferentes módulos que manejan cada uno el itinerario, las predicciones, los precios de oferta y la estrategia, entre otros.

Estrategia de compra

El módulo de previsión (*forecasting*) predice un precio estimado para cada componente para los días inmediatamente siguientes. El módulo de procura se encarga de aceptar las ofertas de los proveedores, seleccionando aquéllas que tengan un precio, una cantidad y una fecha de entrega satisfactorios basado en información de juegos anteriores.

El agente, en un esfuerzo por mantener su reputación tan alta como sea posible, primero acepta las órdenes que satisfacen tanto la cantidad como la fecha de entrega estipulados (o "*full orders*"). Al momento de pedir presupuesto a los proveedores el agente pide los componentes que le faltan y que aún no ha ordenado, para mantener sus niveles de producción.

Estrategia de producción

El módulo de planificación ordena, de acuerdo al tiempo que falta para entregarlas y las penalizaciones que involucran, las órdenes de los clientes. Luego se utiliza una técnica de despacho codiciosa (*greedy*) para seguir el itinerario de producción que el agente se planteó al principio.

Estrategia de venta

El módulo de previsión predice el precio de venta de los productos para los días inmediatamente posteriores. Además el módulo de estrategia determina qué parte de toda la demanda va a constituir el objetivo del agente. El objetivo del módulo de estrategia en este caso es orientarse hacia la fracción de la demanda que va a producir el mayor beneficio en total.

Este agente posee la filosofía de que si trata de abarcar mucho mercado va a tener un margen de ganancia muy bajo, pero si abastece una parte muy pequeña va a dejar ciclos de fábrica libres, que constituyen potencial de ganancia no aprovechado. Por otra parte, el módulo de ofertas maneja una distribución probabilística que le indica la probabilidad de obtener una orden en particular a partir de una oferta.

2.5. Técnicas de aprendizaje aplicadas a TAC SCM

David Pardoe y Peter Stone realizaron experimentos aplicando diferentes técnicas de aprendizaje en las decisiones de venta[Pardoe y Stone, 2004]. El aprendizaje se empleó para determinar la probabilidad de que un cliente acepte una orden considerando un precio de venta específico. Se compararon diferentes técnicas de aprendizaje entre las cuales se encuentran: redes neurales, árboles de regresión y árboles de decisión.

Estas técnicas de aprendizaje se incluyeron al agente TacTex-03 para determinar cuál era el impacto sobre el desempeño de éste. Se dividió el rango de precios en varias clases y se utilizó una máquina de aprendizaje para cada una de las clases. La técnica de aprendizaje que obtuvo los mejores resultados fue los árboles de regresión.

En general, las técnicas de aprendizaje mejoraron el funcionamiento del agente. Entre las conclusiones más importantes que se derivaron de este trabajo está que ganar ofertas en la competencia TAC SCM es un problema muy complejo debido a que los precios ganadores pueden fluctuar muy rápidamente gracias a las condiciones cambiantes del juego. Por esto, en este trabajo se afirma que tomar decisiones basándose en estados pasados del juego actual, no lleva a predicciones acertadas, mientras que utilizar información recogida en numerosos juegos pasados presenta mejores resultados, y puede ser usada exitosamente en la competencia cuando el agente se enfrenta a diferentes grupos de agentes que cambian su comportamiento en el transcurso de la simulación.

2.6. Sistema de clasificadores genéticos XCSR

Los sistemas de clasificadores genéticos son un mecanismo de aprendizaje por reforzamiento propuesto por [Holland, 1976] y consisten en un conjunto de reglas (también llamadas clasificadores) de la forma "SI condición ENTONCES acción", un algoritmo genético para explorar de manera inteligente el espacio de todas las reglas posibles y un sistema de reforzamiento para asignar una medida de utilidad (*fitness*[3]) a las reglas existentes y, de esta forma, guiar la búsqueda de nuevas reglas hacia otras de igual o mejor calidad[Bull, 2004].

[3]La función de *fitness* es una medida de la utilidad de los individuos de la población en algoritmos genéticos o, en nuestro caso, la utilidad de una regla en un sistema de clasificadores genéticos.

Lo que pretenden los sistemas de clasificadores genéticos es crear un conjunto finito de reglas que permita tomar buenas decisiones ante un problema determinado.

Posteriormente, [Wilson, 1995] propone un sistema propio de clasificadores genéticos llamado XCS. La principal particularidad del mismo es que se separa la noción de la fuerza o recompensa esperada de cada regla de la de su utilidad o *fitness*. En XCS, la predicción de la recompensa (o recompensa esperada) es empleada al momento de escoger la acción a realizar. Por otro lado, la función de *fitness* de las reglas, tomada en cuenta por el algoritmo genético para crear nuevos clasificadores, está basada en la exactitud de la predicción de éstas. Es decir, para el algoritmo genético de XCS una regla es "mejor", en la medida en que su predicción de la recompensa sea más exacta; a diferencia del sistema propuesto por Holland, donde una regla era mejor en la medida en que su predicción de la recompensa fuera alta (aunque no necesariamente cercana a la recompensa real).

Esta forma de definir la función de *fitness* estimula que las reglas tengan una buena predicción de la recompensa, ya que con un sistema como el de Holland, la predicción de una regla puede ser muy buena pero a la vez estar muy alejada de la recompensa real. Adicionalmente, XCS permite construir un sistema de reglas que abarquen un espacio más completo del dominio, ya que tendremos reglas tanto para los buenos escenarios como para los malos.

Conjuntos importantes

En XCS existen cuatro conjuntos que merecen especial consideración. Éstos son:

- La población $[P]$, que contiene todos los clasificadores existentes en el sistema.
- El *match set* $[M]$, que consiste en los clasificadores existentes en el sistema que se ajustan a la situación o estado actual.
- El *action set* $[A]$, que contiene todos los clasificadores de $[M]$ que proponen la acción que es finalmente llevada a cabo.
- El *action set* previo $[A]_{-1}$, que contiene los clasificadores que pertenecieron a $[A]$ en la

iteración anterior.

Atributos de los clasificadores

Los clasificadores en XCS son reglas formadas por una condición, una acción y una predicción[Butz y Wilson, 2001]. Cada una de estas reglas tiene los siguientes atributos:

- La condición C, que indica cuáles son los estados en los cuales puede ser aplicado dicho clasificador.
- La acción A, que es la acción (o clasificación) propuesta por dicha regla.
- La predicción p, que es un estimado de la recompensa que se obtendrá si se aplica la acción propuesta por el clasificador en un estado en el que éste pudiera ser empleado.

Además de estos atributos básicos, cada clasificador posee también los siguientes atributos adicionales:

- Una estimación ϵ del error cometido por la predicción.
- La utilidad F.
- La experiencia exp; esto es, el número de veces que el clasificador ha pertenecido a un *action set*.
- La marca de tiempo (*time stamp*) ts, que indica la última vez que se ejecutó el algoritmo genético en un *action set* que contenía a este clasificador.
- El estimado as del tamaño promedio de los *action set* a los que este clasificador ha pertenecido.
- La numerosidad n del clasificador; esto es, cuántos clasificadores ordinarios (o micro-clasificadores) representa este macroclasificador.

Es importante tener en cuenta que todos los clasificadores en XCS son en realidad macro-clasificadores; en otras palabras, cada clasificador cl en realidad representa $cl.n$ clasificadores idénticos.

2.6.1. XCSR

En el sistema XCS original, las condiciones de los clasificadores (reglas) están codificadas como cadenas de ceros, unos y *don't care*. Cada uno de estos dígitos indica si el cumplimiento o no de una condición en el estado del ambiente es necesaria para poder emplear la regla o, en el caso de los *don't care*, si dicha condición es irrelevante. Esto produce la siguiente limitación: las variables que modelan el estado del ambiente deben ser únicamente booleanas. Para otro tipo de variables es necesario hacer una conversión, lo cual es aún más difícil para variables continuas porque introduce el problema de discretizar el dominio de la variable.

Posteriormente, [Wilson, 2000] extiende su sistema original de clasificadores XCS para crear lo que llamó XCSR o XCS con entradas continuas (número reales). Para ello introduce la noción de condiciones basadas en rangos, permitiendo que las condiciones booleanas de las reglas empleen variables continuas. En este sentido, cada característica del ambiente se toma como una variable real x_i y cada entrada de la condición de cada clasificador es de la forma (c_i, r_i), creando así una representación de tipo centro-radio (con centro en c_i y radio r_i). De este modo, para evaluar la condición del clasificador se verifica que se cumpla lo siguiente:

$$(\,\forall\, i \;:\; 0 \leqslant i < n \;:\; c_i - r_i \;\leqslant\; x_i \;\leqslant\; c_i + r_i\,) \quad .$$

Más tarde [Wilson, 2001] vuelve a extender XCS, pero esta vez para crear un sistema de clasificadores para entradas enteras al que llamó XCSI. En esta oportunidad crea una nueva representación de intervalos para las condiciones de las reglas consistente de pares del tipo (l_i, u_i), donde l_i y u_i son las cotas inferior y superior del intervalo, respectivamente. Con esta nueva representación la evaluación de la condición booleana se realiza verificando la siguiente condición:

$$(\,\forall\, i \;:\; 0 \leqslant i < n \;:\; l_i \;\leqslant\; x_i \;\leqslant\; u_i\,) \quad .$$

Aunque inicialmente esta última representación fue introducida para clasificadores con entradas enteras, [Stone y Bull, 2003] muestran los beneficios de la misma sobre la representación centro-radio inicial en el caso de clasificadores con entradas de valores reales. El

principal problema de la primera es la no-biyectividad de la relación genotipo-fenotipo como consecuencia del acotamiento de las variables asociadas a las características del ambiente; esto es, dos rangos iguales pueden ser representados con muchos pares centro–radio distintos, lo que no sucede con la notación de cotas inferior y superior, pues cada intervalo tiene una única representación.

2.6.2. Estrategia de selección de acción

XCS no prescribe un método específico de selección de acción[Butz y Wilson, 2001]. Puede entonces emplearse cualquier estrategia: seleccionar la acción con la que se presume que se obtendrá la mayor recompensa (explotación pura), escoger la acción al azar entre las propuestas por los clasificadores del *match set* (exploración pura), escoger la acción usando la técnica de la rueda de la ruleta, etc.

Un método muy común es el conocido como ϵ-*greedy selection*[Sutton y Barto, 1998], que consiste en hacer explotación pura con una probabilidad previamente establecida y exploración pura el resto de las veces. Usando esta estrategia de selección de acción se toma en cuenta la predicción de la recompensa que se obtendrá por ejecutar cada acción durante la explotación, pero a la vez se evita que se estanque la búsqueda durante la exploración.

2.6.3. Mecanismos importantes dentro del XCS

Durante la ejecución de XCS se emplean dos mecanismos importantes y que merecen una explicación detallada. Éstos son: el mecanismo de *covering* y el de subsunción, y son abordados a continuación.

Covering

Cuando, para un determinado estado del ambiente, no hay ningún clasificador cuya condición se satisfaga, se crea un clasificador que sí se adapte a dicho estado.

En el caso de XCSR con notación centro-radio, esto se logra creando, para cada valor x_i del estado del ambiente, un par (x_i, r), donde r es un número real aleatorio con distribución uniforme entre 0 y una constante s_0[Wilson, 2000].

Sin embargo, si se emplea XCSR con notación de cota inferior y cota superior, se crea, para cada x_i, un par $(x_i - r_1, x_i + r_2)$, donde r_1 y r_2 son valores aleatorios con distribución uniforme entre 0 y una constante s_0[Stone y Bull, 2003].

En ambos casos, la acción propuesta por el nuevo clasificador se escoge aleatoriamente entre todas las acciones posibles, con una distribución uniforme.

Subsunción

Un clasificador es más general que otro cuando el conjunto de estados que cumplen con la condición del primero contiene al conjunto de estados que cumplen con la condición del segundo. Cuando un clasificador c_1 clasifica de manera más general que otro clasificador c_2, se puede decir que c_1 subsume a c_2. En este caso se aumenta la numerosidad de c_1 y se elimina c_2 de la población. Esto constituye una manera de crear reglas más generales que tengan mayor peso en la población.

Existen dos momentos en el algoritmo de XCS en los cuales se realiza la subsunción: durante la ejecución del algoritmo genético y durante la creación del *action set*.

En el caso de la subsunción durante el algoritmo genético simplemente se verifica que los clasificadores hijos no sean subsumidos por sus padres. En caso de ser subsumido por el padre, el nuevo clasificador nunca se agrega a la población sino que la numerosidad del padre aumenta.

En caso de la subsunción dentro del *action set* se busca en el *action set* el clasificador más general que tenga suficiente precisión y experiencia[4]. Luego todos los clasificadores del *action set* se comparan con el más general. Si se consigue uno que sea subsumido por el clasificador más general, se aumenta la numerosidad del más general y se elimina el menos general de la población.

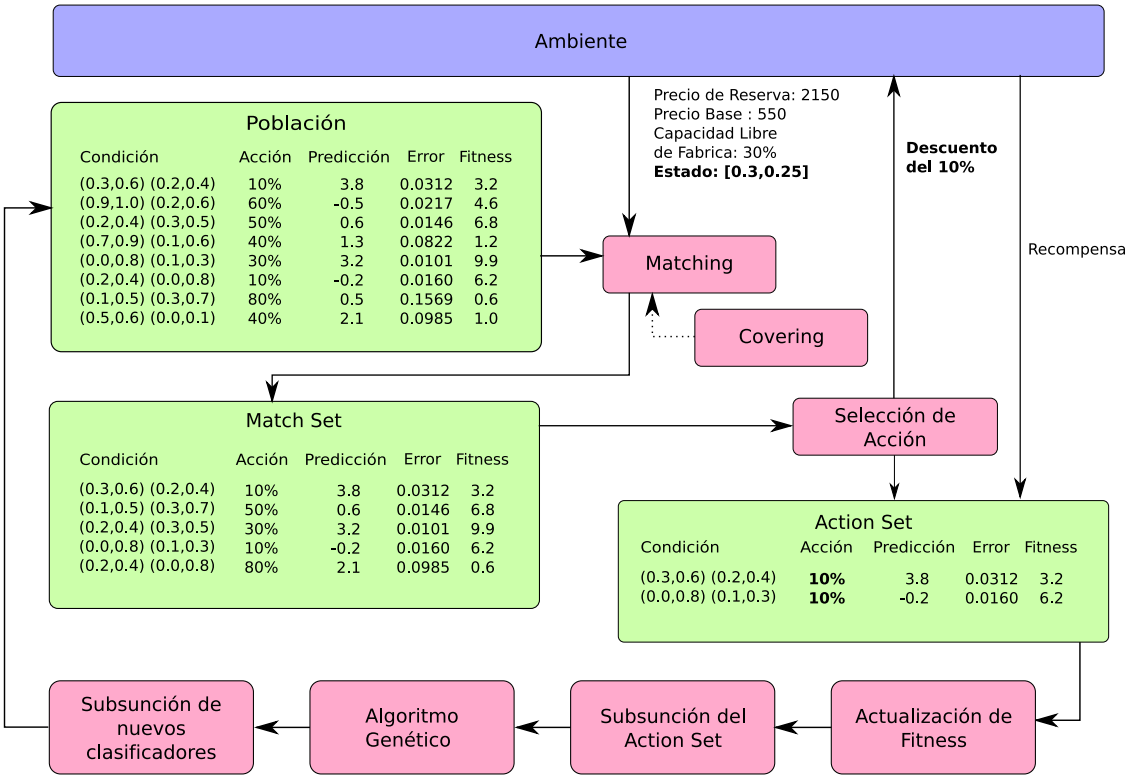


Figura 1: Esquema general de la ejecución del algoritmo de XCSR

2.6.4. Ciclo de ejecución

Según [Butz y Wilson, 2001], durante el proceso de entrenamiento de un sistema de clasificadores XCS (o XCSR) se repiten ciertos pasos mientras la condición de parada[5] no se satisfaga. En la Figura 1 se muestra un diagrama de la ejecución del algoritmo, el cual consta de los siguientes pasos:

1. Se percibe el estado del ambiente.

2. Se crea el *match set* con los clasificadores cuya condición es satisfecha por el estado actual del ambiente. Si el número de distintas acciones propuestas por los clasificadores de $[M]$ no superan un umbral dado, se ejecuta el procedimiento de *covering* hasta que se supere dicho umbral.

[4]Para determinar que un clasificador tenga suficiente precisión y experiencia se verifica que estos valores hayan superado un umbral dado que es fijado por quien realiza el modelo.

[5]La condición de parada puede ser haber realizado un cierto número de iteraciones, la reducción del error hasta un cierto valor o cualquier otra que se desee emplear.

3. Se crea un arreglo de predicciones al que llamaremos PA donde se calcula la predicción de la recompensa que se recibirá por la aplicación de cada una de las posibles acciones. Este arreglo se elabora a partir del conjunto $[M]$. Si existe al menos un clasificador en $[P]$ que proponga el uso de la acción i, $PA[i]$ se calcula como sigue:

$$PA[i] = \frac{\sum_{cl \in \{x \,:\, x \in [P] \,\wedge\, x.A = i \,:\, x\}} cl.p \cdot cl.F}{\sum_{cl \in \{x \,:\, x \in [P] \,\wedge\, x.A = i \,:\, x\}} cl.F}$$

   Si ningún clasificador en $[P]$ propone la acción i, se le asigna a $PA[i]$ un valor especial (como "nil" o "null", por ejemplo).

4. Se escoge la acción que se llevará a cabo utilizando el método se haya escogido para ello (ver Sección 2.6.2).

5. Se crea el *action set* con los clasificadores del *match set* que proponen la acción escogida.

6. Se ejecuta la acción escogida.

7. Se recibe la recompensa ρ por la acción ejecutada.

8. Se recompensa a los clasificadores de $[A]_{-1}$. Para ello se crea P como $\rho_{-1} + \gamma \,\text{máx}(PA)$ donde ρ_{-1} es la recompensa obtenida en la iteración anterior y γ (conocido como factor de descuento) es fijado por quien hace el modelo y sólo se emplea en problemas *multistep*. En los problemas *singlestep* tiene por valor 1. Luego, en base a P se llevan a cabo las actualizaciones necesarias en los atributos de los clasificadores y se hace subsunción en $[A]_{-1}$ (en caso de que se desee hacer subsunción en los *action sets*). Esto se hace utilizando una tasa de aprendizaje β, un umbral ϵ_0 para la consideración del error, y las constantes α y ν establecidas por quien hace el modelo. Para realizar estas actualizaciones se ejecuta el procedimiento ACTUALIZAR CONJUNTO definido en el Algoritmo 2.6.1 con $[A]_{-1}$, P y $[P]$ como sus parámetros reales. Dentro de este proceso se realiza la actualización del *fitness* la cual corresponde al Algoritmo 2.6.2.

9. Se ejecuta el algoritmo genético sobre el *action set* de la iteración anterior si la diferen-

cia entre el tiempo actual y el momento promedio de la última aplicación de algoritmos genéticos sobre cada uno de los clasificadores de $[A]_{-1}$ supera un umbral dado. En caso de que se desee hacer subsunción en las ejecuciones del algoritmo genético, se verifica si alguno de los clasificadores generados por entrecruzamiento es subsumible por alguno de sus padres y, de ser así, se lleva a cabo la subsunción.

Algoritmo 2.6.1: ACTUALIZAR CONJUNTO($[A], P, [P]$)

for each cl **in** $[A]$
  $cl.exp \leftarrow cl.exp + 1$
  if $cl.exp < 1/\beta$
    $cl.p \leftarrow cl.p + (P - cl.p)/cl.exp$
  else
    $cl.p \leftarrow cl.p + \beta(P - cl.p)$
  if $cl.exp < 1/\beta$
    $cl.\epsilon \leftarrow cl.\epsilon + (|P - cl.p| - cl.\epsilon)/cl.exp$
  else
    $cl.\epsilon \leftarrow cl.\epsilon + \beta(|P - cl.p| - cl.\epsilon)$
  if $cl.exp < 1/\beta$
    $cl.as \leftarrow cl.as + (\sum_{c \in [A]} c.n - cl.as)/cl.exp$
  else
    $cl.as \leftarrow cl.as + \beta(\sum_{c \in [A]} c.n - cl.as)$
  ACTUALIZAR FITNESS($[A]$)
  if *doActionSetSubsumption*
    hacer subsunción en $[A]$ actualizando $[P]$

Fuente: [Butz y Wilson, 2001]

Algoritmo 2.6.2: ACTUALIZAR FITNESS($[A]$)

$accuracySum \leftarrow 0$
inicializar vector de precisiones κ
for each cl **in** $[A]$
  if $cl.exp < \epsilon_0$
    $\kappa(cl) \leftarrow 1$
  else
    $\kappa(cl) \leftarrow \alpha(cl.\epsilon/\epsilon_0)^{-\nu}$
  $accuracySum \leftarrow accuracySum + \kappa(cl) * cl.n$
for each cl **in** $[A]$
  $cl.F \leftarrow cl.F + \beta(\kappa(cl) \cdot cl.n/accuracySum - cl.F)$

Fuente: [Butz y Wilson, 2001]

Capítulo 3

TicTACtoe

En este capítulo se presenta el diseño de nuestro agente, TicTACtoe, y las estrategias adoptadas.

TicTACtoe consta de tres módulos: Compra, Venta y Producción (ver Figura 2). Estos módulos interactúan de manera independiente con el agente; sin embargo sus acciones están muy relacionadas entre sí. Básicamente cada uno de los módulos toma decisiones en su área e informa al agente de su decisión final. El agente es el encargado de pasar esta información a los demás módulos para tomar decisiones posteriormente. También se implementó una estructura que denominaremos Agenda, que hace el papel de organizador para el agente. Ésta permite llevar un registro de las decisiones tomadas en el pasado y tener en cuenta los eventos que se presentarán a futuro para, de este modo, realizar predicciones más acertadas. Mediante la Agenda se provee al agente de una estructura dinámica para ingresar y buscar información que pueda ser relevante para futuras decisiones.

3.1. Módulo de Compras

El módulo de compras es el encargado de realizar las licitaciones a los proveedores y aceptar las ofertas para comprar los componentes necesarios durante la simulación.

3.1.1. Creación de Licitaciones (RFQp)

Al momento de enviar las licitaciones a los proveedores, TicTACtoe calcula la cantidad de componentes que va a necesitar en los próximos diez días de simulación. Esto se realiza tomando en cuenta el inventario actual, los compromisos pautados para los próximos diez días y las llegadas de componentes confirmadas, lo que permite estimar la cantidad faltante de componentes para cumplir con los compromisos de los próximos días.

Si la cantidad requerida de un componente es mayor que cero, se envía la licitación al proveedor favorito de dicho componente, que es aquél que últimamente ha dado los mejores

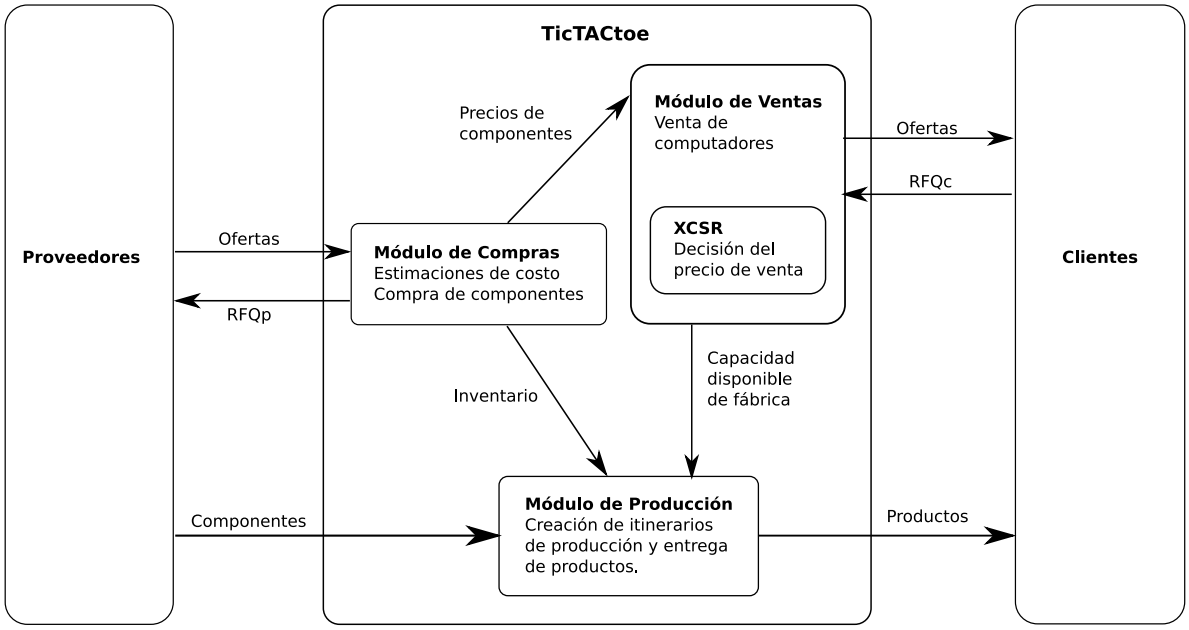


Figura 2: Arquitectura del Agente TicTACtoe

precios. Sólo se envía una licitación con cantidad mayor que cero por componente para no tener que rechazar ofertas y bajar la reputación del agente.

El mecanismo de determinación del proveedor favorito se implementó por medio de un registro con los últimos precios dados por los proveedores para cada uno de los componentes que ofrecen. De esta manera, el proveedor favorito de un producto es aquél que, para dicho producto, tiene el mejor precio en el registro.

La selección de un proveedor favorito se realiza para procurar precios más bajos. Esto se hace basado en la suposición de que el estado de los agentes proveedores no cambia drásticamente; y de que si un proveedor da el mejor precio, lo más probable es que en los días subsiguientes continúe dándolo. A todos los demás proveedores se les envía una licitación con cantidad igual a cero para conocer a que precio están ofertando cada componente y cambiar de proveedor favorito en caso de ser necesario. Cabe destacar que cada componente tiene un proveedor favorito independiente a los demás. Si la cantidad requerida de un componente es igual a cero, se envían licitaciones con cantidad igual a cero a todos los proveedores que vendan este componente.

3.1.2. Revisión y aceptación de ofertas

Al momento que llegan las diferentes ofertas por cada uno de los proveedores (ofertas normales, ofertas con cantidad menor pero a tiempo y ofertas con cantidad completa pero fecha de entrega retrasada) TicTACtoe acepta preferiblemente las ofertas completas que llegarán a tiempo y, en caso de no haberlas, aquéllas que varían la cantidad.

Una vez enviada la orden, el agente agrega a la agenda la llegada de estos componentes para tener un estimado más exacto del inventario a futuro. También se actualizan los precios base de cada uno de los componentes y el proveedor favorito tomando en cuenta el mejor precio ofertado por los proveedores. La actualización del precio base se realiza utilizando un promedio ponderado como se muestra en la ecuación 1.

$$P_c = B_c \cdot \alpha + P_c \cdot (1 - \alpha) \tag{1}$$

El proveedor favorito para un componente se sustituye por aquél que haya dado el mejor precio durante ese día.

3.2. Módulo de Producción

El módulo de producción se encarga de la manufacturación de las órdenes que se encuentran abiertas, es decir, todas las órdenes que se encuentran esperando por ser fabricadas o ser entregadas al cliente.

Este módulo todos los días organiza las órdenes activas primero de acuerdo a la fecha de entrega y luego de acuerdo al *penalty* que producirían en caso de no ser entregadas. Luego para cada una de estas órdenes, en el orden en que quedaron, verifica si hay suficiente inventario de computadores producidos para que dicha orden pueda ser entregada y, en este caso, es entregada al cliente. Ésta es una estrategia utilizada por PhantAgent para evitar costos de almacenamiento[Stan et ál., 2006]. En caso de no poder ser entregada una orden se verifica si ésta ya sobrepasó la fecha límite de entrega[1]. En este caso el cliente no está dispuesto a esperar más y cancela su pedido. Si esto ocurre, entonces se deja de tomar

[1]La fecha límite de entrega es fijada al momento en que el cliente realiza la primera licitación.

en cuenta esta orden en la planificación futura y se liberan los componentes asociados a ésta para que puedan ser utilizados en la fabricación de cualquier otro pedido.

Si la orden aún no va a ser cancelada por el cliente y tampoco hay suficientes computadoras armadas para entregarlas, se intenta producirla. Primero se verifica si en la agenda la orden está marcada para ser producida ese día. El agente TicTACtoe sigue con rigurosidad los itinerarios de fabricación que se planteó previamente. En caso de encontrarse la orden en el itinerario de producción de ese día, se intenta producir la diferencia entre la cantidad que se estipulaba vender y la cantidad que ya se encuentra en inventario. Si se emplean productos en inventario, se marcan para que no sean tomados en cuenta en futuras órdenes y, de este modo, evitar los cálculos erróneos. En caso de que la cantidad faltante de computadoras no pueda ser producida por falta de ciclos o de componentes, se produce la máxima cantidad posible.

Si la orden no está marcada para ser producida ese día, se descarta y se espera a que llegue su día de fabricación. Si una orden que se encuentra marcada para un día no puede ser fabricada ese mismo día, ya sea completa o parcialmente, se asigna en la agenda para el día siguiente para continuar con su fabricación.

Al final del día el módulo de producción determina el número de órdenes atrasadas y el número de órdenes activas y envía esta información al módulo de ventas para que ajuste la cantidad de ciclos libres que se ofertan diariamente. Esto le permite al agente ofertar menos ciclos de los que en realidad tiene y ponerse al día con las órdenes atrasadas.

3.3. Módulo de Ventas

El módulo de ventas todos los días revisa las licitaciones de los clientes (RFQc) y escoge para ofertar aquéllas cuyo precio de reserva es mayor al precio base[2] calculado por el agente y cuyo momento de entrega no es posterior al fin de la simulación. Luego, para cada una de estas licitaciones el agente utiliza una serie de reglas generadas utilizando XCSR que le permiten determinar cuánto descuento d dar sobre el precio de reserva sugerido por el cliente.

[2]El precio base es la suma de los precios estimados que maneja el agente, de cada uno de los componentes que conforman un producto.

La incorporación de XCSR al agente se explica en la Sección 3.5.

El precio de oferta final viene determinado por la ecuación 2, donde *BasePrice* es el costo calculado por el agente por sus experiencias recientes y *ReservePrice* es el precio de referencia determinado por el cliente. Los clientes nunca están dispuestos a pagar un precio superior a su precio de referencia.

$$OfferedPrice = BasePrice + (ReservePrice - BasePrice) \cdot (1 - d) \tag{2}$$

Una vez calculado el precio de oferta para cada una de las licitaciones, se ordenan descendentemente de acuerdo a la ganancia obtenida, y se revisa en la agenda si hay ciclos disponibles para fabricar el pedido. Siempre se trata de fabricar lo más tarde posible ya que esto permite dejar espacio para órdenes que necesiten ser entregadas con más urgencia[Pardoe y Stone, 2006].

Por otro lado, los ciclos libres son multiplicados por un factor entre 0 y 1 que es inversamente propocional a la cantidad de órdenes retrasadas que posee el agente (esto le permite al agente ponerse al día en caso de que su retraso se deba a una saturación de la capacidad productiva). Si existen ciclos libres para fabricar en el día escogido[3], se oferta a la licitación y se deja en la agenda constancia del posible compromiso; reservando así los ciclos de fábrica necesarios para su producción. De esta manera se puede calcular correctamente la cantidad de ciclos de fabricación disponibles para un día específico. En caso de que no se disponga de suficientes ciclos libres se verifica si el pedido se puede cubrir con el inventario esperado no reservado para el día siguiente. En caso de que esto sea cierto se acepta la orden y se reserva el inventario correspondiente del día siguiente.

No todas las ofertas que se envían en un día son aceptadas por el cliente. Si el cliente acepta el presupuesto, entonces el compromiso tentativo pasa a ser una orden inamovible. Por el contrario, si el cliente rechaza el presupuesto, el compromiso tentativo se elimina y se liberan de esta forma los ciclos libres apartados para fabricarlo.

[3]Los ciclos libres ofertados no son iguales a los ciclos efectivos y esta tasa de oferta varía dependiendo del porcentaje de órdenes atrasadas que posea el agente en un momento dado.

3.4. Agenda

El agente posee una estructura de datos que simula la función de una agenda. Esta estructura permite guardar las decisiones de producción, compra y venta para un día específico para así poder recordar estos datos en futuras decisiones. También permite guardar acontecimientos de cada uno de los días de la simulación que le permitan al agente tomar decisiones informadas. La agenda está constituida por 220 registros, cada uno correspondiente a un día de la simulación. Cada uno de los registros posee la siguiente información del día:

Compromisos de producción inamovibles. Corresponden a todas aquellas órdenes ya aceptadas por el cliente que se espera producir durante ese día.

Posibles compromisos de producción. Corresponden a aquellos RFQ a los cuales se ha ofertado y que se fabricarían ese día en caso de que la orden sea aceptada. Junto con los compromisos de producción inamovibles permite saber qué cantidad de ciclos de fábrica aún se tiene disponible en el día en cuestión para ofertarlos.

Órdenes producidas del día. Órdenes que efectivamente se produjeron un día determinado, debido a que es posible que algunas de las órdenes marcadas para ser producidas en una fecha no puedan ser manufacturadas y tengan que se pospuestas para fechas posteriores.

Inventario tentativo. Productos que ya fueron ordenados a los proveedores y que tienen como fecha de llegada el día en cuestión. Permite calcular el inventario esperado en días futuros.

3.5. Incorporación de XCSR al agente TicTACtoe

La incorporación del algoritmo de XCSR se hizo en el módulo de ventas del agente utilizando la librería que se explicará en el Capítulo 4. El módulo de ventas posee una población de clasificadores que puede ser leída de un archivo al inicio de la simulación y ser guardada nuevamente en el mismo al finalizar la simulación. La decisión que es tomada

utilizando XCSR es el precio de venta para una oferta dada. Debido a esto, al módulo de ventas (el encargado de tomar esta decisión) se le agregaron dos métodos importantes:

getPriceDiscountFactor(): Recoge el estado del ambiente y busca el *match set* correspondiente a este estado. Luego determina la acción a efectuarse (que sería el descuento que devuelve este método) y guarda el *action set* junto con la licitación para poder recompensarlo posteriormente.

rewardAndRunGA(actionset, reward): Recompensa el *action set* que es pasado como parámetro y guarda la información del error para estadísticas futuras. Este método también se encarga de desbloquear los clasificadores involucrados; esto se explicará con más detalle en la Sección 4.1.6.

3.5.1. Estructura del clasificador

A continuación se explica el modelo que se realizó del problema de decisión del precio de venta para incorporar XCSR en esta área.

Condición

Existe una serie de valores durante la simulación que son conocidos por el agente y que proveen información necesaria para la toma de decisiones. Debido a que no se pueden incluir todos estos valores en la toma de decisiones (pues aumentaría considerablemente el espacio de búsqueda del problema), se decidió incluir en la estructura de la condición sólo los cinco valores que consideramos los más importantes para la decisión del descuento. Los rasgos o *features* que fueron tomados en cuenta para generar las reglas son los siguientes:

x_1 Porcentaje de órdenes pendientes atrasadas con respecto al número total del órdenes pendientes. Permite determinar cuánto trabajo se encuentra atrasado y decidir qué tan conveniente es hacer una buena oferta y obtener una nueva orden. El valor de x_1 se determina como sigue:

$$x_1 = \frac{lateOrders}{totalOrders}$$

x_2 Porcentaje de fábrica que quedó disponible el día anterior con respecto al cupo diario de ciclos libres del agente. Le permite al agente determinar si debe aumentar o disminuir el descuento. Por ejemplo, si la fábrica se encuentra copada, el agente debería dar poco descuento para tratar de fabricar primero los pedidos que tiene pendientes antes de comprometerse con nuevos pedidos. Pero, en cambio, si la fábrica se encuentra desocupada, el agente debería dar buenos descuentos para tratar de aprovechar esos ciclos de fábrica que no se están utilizando. El valor de x_2 se determina como sigue:

$$x_2 = \frac{freeFactoryCapacity}{2000}$$

x_3 Porcentaje del precio base con respecto al precio de reserva indicado por el comprador. Le permite saber al agente la máxima ganancia posible para una la orden. Los casos en los que el precio base es superior al precio de reserva son descartados por el agente automáticamente ya que no proveen ganancia alguna. El valor de x_3 se determina como sigue:

$$x_3 = \text{mín}\left(\frac{basePrice}{reservePrice}, 1\right)$$

x_4 Holgura entre la fecha actual y la fecha en la cual debe ser entregado el pedido en cuestión. Le permite al agente determinar de cuánto tiempo dispone para fabricar la orden. Se escala este valor entre 0 y 1 tomando en cuenta que la máxima holgura permitida es de 12 días. El valor de x_4 se determina como sigue:

$$x_4 = \frac{(dueDate - day)}{12}$$

x_5 Día del juego normalizado entre 0 y 1 tomando en cuenta la cantidad de días que dura un juego. Este valor es muy importante debido a que en la simulación con el pasar del tiempo se presentan situaciones diferentes. Por ejemplo, conforme pasan los días, los componentes comienzan a escasear y a aumentar su valor, o la demanda comienza

a bajar. Sin embargo, estas variaciones dependen directamente de las decisiones que tomen los agentes, y es por esto que este rasgo es fundamental en la condición, porque le permite al agente aprender a tomar decisiones tomando en cuenta estos cambios en la simulación. El valor de x_5 se determina como sigue:

$$x_5 = \frac{day}{220}$$

Todos los rasgos $x_1,\ x_2,\ \ldots\ , x_5$ estan normalizados entre 0 y 1, para poder utilizar estos valores como cotas inferiores y superiores. Esto es debido a que la librería implementada posee esta limitación, lo cual se explicará a profundidad en la Sección 4.1.1.

Acción

La implementación de XCSR posee 10 acciones, las cuales significan descuentos sobre el precio final de venta y vienen dados como un porcentaje de la diferencia entre el precio base calculado por el agente y el precio de reserva determinado por el cliente. Los descuentos considerados varían de 10 % en 10 % y van desde un 0 % hasta un 90 %.

Recompensa

La recompensa viene determinada principalmente por la ganancia o pérdida que se obtuvo a partir de una licitación, escalada por la cantidad de dinero que implicaba la fabricación de la misma para el agente al momento de ofertar a esa licitación. Existen tres momentos diferentes en los cuales un *action set* puede recibir su recompensa:

Cuando la oferta a la licitación no es aceptada. En este caso se considera que la licitación no produjo ninguna ganancia o pérdida, porque no se invirtió en ella ni se obtuvo nada a cambio. En este caso la recompensa es 0.

Cuando se entrega la orden. En este caso se considera la ganancia que se obtuvo por la venta y las pérdidas que surgieron si se entrega algunos días tarde. La ganancia se obtiene multiplicando el precio de oferta por producto por la cantidad de productos mientras que las pérdidas vienen determinadas por la cantidad de *penalty* multiplicado

por la cantidad de días que la orden tardó en entregarse después de lo estipulado por el cliente. Si la orden es entregada a tiempo no se producen pérdidas. Tanto las pérdidas como las ganancias se escalan por la inversión que realiza el agente en fabricar esa orden; esto es, el precio base de los productos multiplicado por la cantidad que se fabricó. La recompensa recibida en este caso se calcula por medio de la ecuación 3.

$$reward = \left(\frac{offeredPrice}{basePrice}\right)^2 - \left(\frac{\text{máx}(day - duedate, 0) \cdot penalty}{basePrice \cdot quantity}\right)^2 \tag{3}$$

Un detalle importante de la ecuación 3 es que al escalar por el precio base se obtiene un valor porcentual de la ganancia obtenida. Podría pensarse que una buena aproximación de función de recompensa sería restar los gastos a la ganancia. Sin embargo, esto nos devuelve un valor neto de ganancia, que no es el mismo para los diferentes tipos de productos. No todos los productos producen la misma ganancia, debido a las diferencias en los costos, pero sí pueden tener el mismo margen de ganancia, por lo tanto es más indicado premiar el clasificador en base al porcentaje de ganancia que se obtuvo.

Cuando se cancela la orden sin ser entregada. En este caso al agente no logró fabricar y entregar la orden. Después de una serie de días el cliente no está dispuesto a esperar más y la orden es cancelada. En este caso la orden sólo produce pérdidas para el agente, correspondientes a todas las penalizaciones que fueron cobradas antes de cancelar dicho pedido. El dinero invertido en materia prima destinada para la fabricación de esta orden no es considerado una pérdida, puesto que estos componentes pueden ser utilizados para completar otras órdenes. La recompensa en este caso viene dada por la ecuación 4, que consiste una variación de la ecuación 3, haciendo el término correspondiente a la ganancia igual a cero.

$$reward = -\left(\frac{\text{máx}(day - duedate, 0) \cdot penalty}{basePrice \cdot quantity}\right)^2 \tag{4}$$

En las ecuaciones 3 y 4 los términos correspondientes tanto a la pérdida como a la ganancia se elevan al cuadrado para que las situaciones borde en las cuales el porcentaje de ganancia o pérdida es significativamente mayor, la recompensa de los clasificadores sea más pesada.

Capítulo 4

Implementación de XCSR

En este capítulo se presenta la implementación de XCSR que se realizó para resolver el problema de determinar cuál es el descuento adecuado a aplicar sobre el precio de venta dada una situación determinada durante el juego.

4.1. Detalles de implementación y validación de la librería

La implementación de la librería de XCSR se realizó teniendo como base la versión 1.0 de la librería de XCS de [Butz, 2006]. Ésta tenía la limitación de ser sólo para XCS tradicional y, debido a esto, fue necesario adaptarla a XCSR con notación de cotas superior e inferior.

Dicha librería proveía soporte para problemas *singlestep* y *multistep*. Un problema *singlestep* es un problema no secuencial, es decir, la recompensa de una acción depende únicamente de esa acción y no de las acciones tomadas anteriormente; mientras que en un problema *multistep* cada una de las acciones afecta la recompensa de las acciones subsiguientes. Nuestra implementación de XCSR sólo abarca los problemas *singlestep*, puesto que en el problema que queremos resolver, la recompensa depende directamente de la acción que se tomó y no de una secuencia de acciones.

En el futuro se podría investigar cómo tratar este problema como un problema *multistep*, buscando la correlación entre la toma de decisiones. Por ejemplo, se podría tomar en cuenta que la ganancia de un determinado momento viene dada también por las decisiones tomadas con respecto a órdenes anteriores y no únicamente a la orden en cuestión. Sin embargo, no se realizó durante esta investigación una implementación de XCSR de tipo *multistep* dado que se consideró que el tiempo que se requeriría para elaborar el mismo era mayor de aquél del que se disponía para la elaboración de este trabajo.

4.1.1. Implementación del *don't care*

En la librería de XCS de Butz, el *don't care* o la generalización de los clasificadores se encuentra implementado sustituyendo la información de un alelo por #, un caracter que se interpreta como más general y puede ser igual tanto a 1 como a 0. La idea es que el alelo modificado no influya en la clasificación que realiza a través de esta regla.

En nuestra librería de XCSR el *don't care* se implementó como una ausencia de cota superior o inferior, según sea el caso del alelo a modificar. Para poder implementar el *don't care* de esta forma, se restringió la librería de manera de que todos los rasgos del cromosoma deben estar acotados entre 0 y 1. Cuando se coloca un *don't care* en el cromosoma, se desplaza la cota al máximo límite permitido; en caso de que el alelo sea cota superior se desplaza hasta 1 y en caso de que sea cota inferior se desplaza hasta 0. De esta manera el alelo no influirá en la clasificación.

4.1.2. Subsunción de los clasificadores

Las reglas de subsunción de la librería de M. Butz estaban orientadas a rasgos booleanos, por lo que fue necesario rediseñar las reglas de subsunción para que se adaptaran a la nueva estructura del clasificador, donde todos los rasgos se encontraban acotados entre 0 y 1.

Para implementar la subsunción se utilizaron las reglas propuestas por [Loiacono, 2004] donde un clasificador es más general que otro, si cada uno de los rangos del primero contiene a los rangos del segundo. Por ejemplo (l_i, u_i) subsume a (l_j, u_j) si $u_i > u_j \wedge l_i < l_j$. Además, para que ocurra la subsunción, la acción de los clasificadores debe ser la misma.

4.1.3. Creación dinámica de la población

La librería utilizada como base para implementar XCSR poseía un método de creación dinámica de la población, mediante el cual se comenzaba con una población vacía y cada vez que se generaba un *action set* se buscaba que todas las acciones estuvieran cubiertas, es decir, que al menos hubiera un clasificador para cada acción existente. En caso de que no hubiera un clasificador para una acción en el *action set* se hacía *covering* y se agregaba el nuevo clasificador a la población. La ventaja de este método es que la población se crea

dinámicamente conforme aparecen los estados del ambiente en el experimento.

Sin embargo, debido a que la población tiene un tamaño limitado, al querer tener cubiertas todas las acciones es necesario borrar viejos clasificadores mediante el algoritmo de rueda de la ruleta para agregar los nuevos. Esto puede resultar muy desfavorable, ya que por lo estocástico del algoritmo, en etapas avanzadas de ejecución cuando ya los individuos hayan evolucionado, se puede estar borrando clasificadores con un mejor *fitness* que el nuevo clasificador que está entrando, por lo tanto, se está disminuyendo el *fitness* general de la población. Debido a esto se decidió cambiar el método de creación dinámica de la población para que fuese únicamente utilizado en las primeras etapas del algoritmo genético, es decir, cuando la población aún no ha llegado a su límite. De esta forma se estará creando la población de manera uniforme, pero al momento de tener la población completa se dejan de borrar clasificadores que puedan contribuir a encontrar mejores soluciones. Sin embargo, se continúan agregando y borrando individuos de la población mediante la aplicación el algoritmo genético sobre el *action set*.

4.1.4. Operadores de entrecruzamiento

Se implementó un operador de entrecruzamiento de dos puntos entre rangos de la condición que permitiese generar hijos con rangos válidos. Esto significa que los puntos de corte sólo pueden ser escogidos entre la cota superior de uno de los rangos y la cota inferior del siguiente. Este operador es equivalente al operador de entrecruzamiento para entradas binarias, ya que se cruzan las condiciones completas. Esto garantiza que los rangos de los hijos sean válidos, ya que son una combinación de los rangos de los padres.

4.1.5. Validación de la librería de XCSR

Luego de realizadas las adaptaciones necesarias a la librería de Butz para que la misma trabajara con XCSR y no sólo con el XCS original, era necesario llevar a cabo ciertas pruebas para determinar que la librería estaba funcionando correctamente.

Se hicieron puebas sencillas que consistían en la clasificación de puntos en la región del plano $x_1 \times x_2$ en la que los valores de ambas variables varían entre 0 y 1. De este modo,

x_1 y x_2 fueron empleadas como rasgos (*features*) de los clasificadores y se puso a prueba el sistema con varios experimentos en los que debía clasificar los puntos según si pertenecían o no a círculos dados. En las Figura 3 se puede observar los resultados de estas pruebas.

En la Figura 3(a), los puntos marcados en azul fueron clasificados como pertenecientes a la clase formada por la unión de los dos círculos dibujados en negro; los puntos marcados en rojo fueron clasificados como no pertenecientes a dicha clase.

Por otro lado, en la Figura 3(b), los puntos marcados en morado fueron clasificados como no pertenecientes a ninguna de las dos clases formadas por los dos círculos. Los marcados en azul y rojo fueron clasificados como pertenecientes a las clases formadas por los círculos izquierdo y derecho, respectivamente.

En la Figura 3(c) los puntos dibujados en amarillo fueron clasificados por el sistema como no pertenecientes a ningún círculo, los rojos fueron clasificados como pertenecientes al círculo superior izquierdo, los morados como pertenecientes al círculo superior derecho y los azules como pertenecientes al círculo inferior.

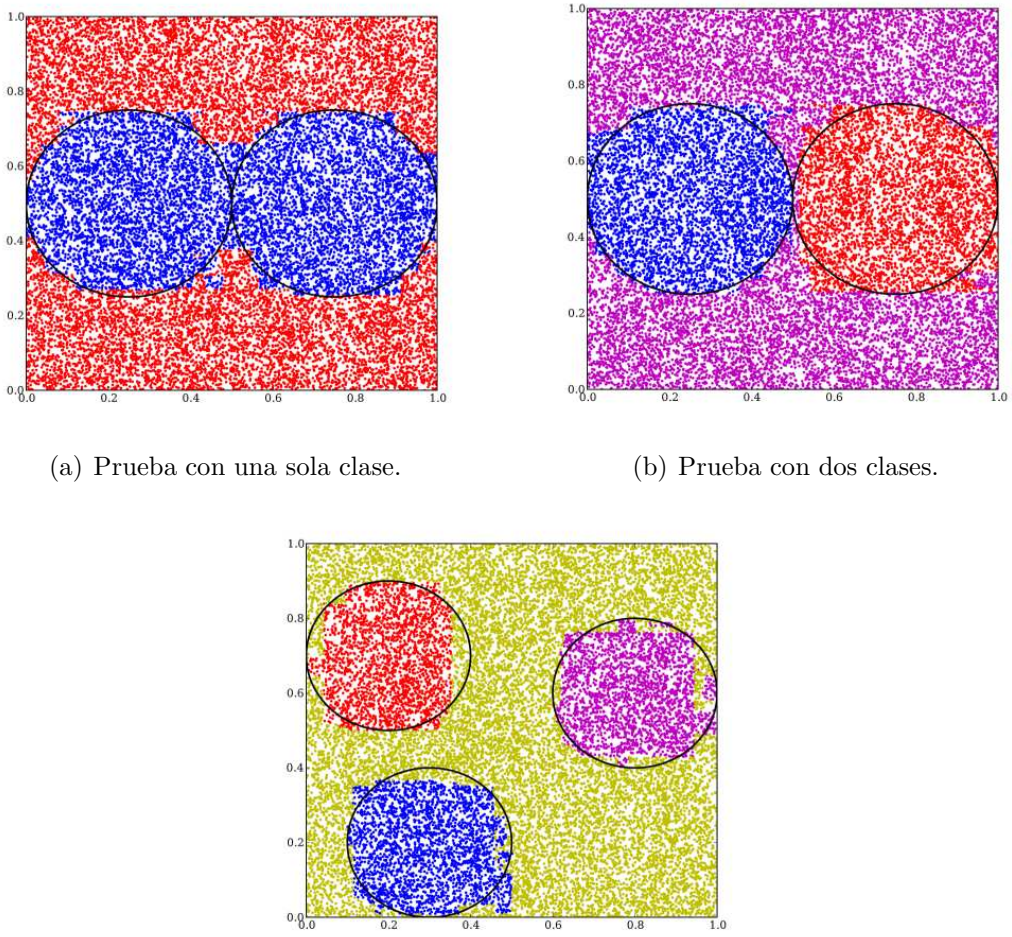
Estos resultados demuestran que esta implementación de XCSR logra clasificar correctamente conjuntos de datos que no son linealmente separables. A partir de esto se consideró que esta implementación era apta para ser usada en el agente TicTACtoe.

4.1.6. Otras adaptaciones

Además de la implementación estándar de la librería de XCSR fueron necesarias adaptaciones especiales de los mecanismos para incorporar esta librería en el agente TicTACtoe. Estas adaptaciones vienen dadas por características especiales del problema que se quiere resolver.

Recompensa con retraso y bloqueo de clasificadores

Debido a que la recompensa de un *action set* depende de la cantidad de dinero que se gane o se pierda con la oferta con la cual están relacionados, es indispensable poder guardar el *action set* asociado a ésta para en el futuro, cuando se conozca la utilidad que tuvo aceptar esta orden para el agente, poder recompensar los clasificadores correspondientes.



(a) Prueba con una sola clase.

(b) Prueba con dos clases.

(c) Prueba con tres clases.

Figura 3: Pruebas de validación de la librería de XCSR.

Para guardar el *action set* asociado a una licitación se modificó la estructura *CustomerRFQ* para que permitiera almacenar este conjunto de clasificadores. Luego, todas las licitaciones que tengan un *action set* esperando por recompensa se indexan por número de licitación en un *HashMap* que es manejado por la estructura matriz del agente. Se escogió indexar por número de licitación debido a que este número siempre se conoce y puede ser accedido fácilmente en todas las etapas de la negociación (cuando la licitación se convierte en oferta, y posteriormente cuando se convierte en orden de compra).

El siguiente problema era el de evitar que los clasificadores que estaban esperando por una recompensa fueran borrados durante la dinámica del algoritmo, ya que se borrarían

tomando en cuenta información que no se encuentra totalmente actualizada. Para esto se implementó un sistema sencillo de semáforos, donde cada clasificador tiene un contador que indica cuántas veces el clasificador se encuentra en un *action set* esperando para obtener recompensa. De esta manera, sólo se consideran dentro del algoritmo de eliminación por rueda de la tuleta aquéllos cuyo contador se encuentre en cero.

Otra restricción importante al momento de realizar la subsunción es que un clasificador bloqueado no puede ser subsumido, debido a que este clasificador se encuentra esperando por una recompensa y su información no está totalmente actualizada para formar parte de un nuevo clasificador. Los clasificadores bloqueados pueden participar en todos los demás mecanismos de XCSR, es decir, entrecruzamiento, mutación, exploración y explotación.

Tasas variables de exploración y explotación

Otro cambio importante realizado a la librería base fue con respecto a la cantidad de veces que se exploraba y se explotaba durante el aprendizaje. La librería base siempre explotaba y exploraba intercaladamente, recompensando los clasificadores únicamente en los casos en los que exploraba y tomando para las estadísticas los casos en los cuales se explotaba. Como el algoritmo de XCSR está inmerso en el agente y cada acción que éste tome, ya sea por exploración o explotación, repercute en el desempeño del agente en la simulación, esta metodología de exploración y explotación tuvo que ser modificada.

Se modificó el algoritmo para que se recompensaran los clasificadores tanto en los casos en los que explora como los casos en los que se explota. Además, se modificó para que la exploración y la explotación no se produjeran de manera intercalada, sino que se produjeran de manera estocástica con una tasa de explotación que empieza en 0 y va aumentando linealmente conforme pasa el tiempo en el aprendizaje. Esto permite que al inicio del algoritmo de aprendizaje el agente explore más frecuentemente, y conforme pasa el tiempo vaya disminuyendo la probabilidad de explorar y aumentando la probabilidad de explotar hasta llegar a un umbral (o tasa máxima de explotación) donde esta probabilidad permanece constante. Este incremento inicial se escogió de forma lineal principalmente por simplicidad. En la Sección 5.2 se presenta un experimento donde se varía el valor de este umbral.

Capítulo 5

Experimentos y Resultados

En este capítulo mostraremos los experimentos realizados, sus resultados, y las conclusiones que se derivan de éstos. Se realizó un experimento principal que muestra la mejoría al utilizar aprendizaje por medio de XCSR en la toma de decisiones de un agente TAC SCM. También se realizó un experimento que demuestra la importancia de utilizar el algoritmo de bloqueo de clasificadores en problemas que implican un retraso en la recompensa. Adicionalmente se realizaron una serie de experimentos para conseguir los valores más favorables para ciertos parámetros como la tasa de explotación[1] y el tamaño de la población.

Cada uno de los agentes comparados compitió por separado con 5 agentes *dummy* proporcionados en la librería del servidor, sin ninguna modificación adicional. Debido a que los comportamientos de los *dummy* no varían entre sí significativamente, se puede escoger uno de ellos al azar para compararlo con el agente en algunos experimentos. Esta comparación es importante debido a que, al tratarse de un juego competitivo, el desempeño de un agente afecta el desempeño de sus competidores, por lo tanto, es importante ver en qué medida el agente logró afectar el rendimiento de su respectivo *dummy*.

Para cada experimento se hicieron 40 corridas o juegos. En la primera de ellas, el agente comenzó con una población de clasificadores vacía. Al finalizar cada juego el agente guardó la población y al iniciarse el siguiente, estableció como su población inicial de clasificadores la población guardada en el juego anterior. Los datos reportados en los gráficos representan el desempeño promedio de los agentes en las últimas 25 corridas, ya que en las primeras 15 consideramos que el agente se encuentra en el proceso de aprendizaje y no ha alcanzado su desempeño óptimo.

Cada uno de los juegos tuvo una duración de 220 días simulados[2] (al igual que en la

[1] Es importante recordar que esta tasa de explotación es en realidad la tasa máxima de explotación de la que se habla en la Sección 4.1.6.

[2] Cabe destacar que cada uno de los clasificadores puede pertenecer a un *action set* más de una vez en un mismo día.

competencia) y, luego de verificar que ninguno de los agentes necesitaba más de 5 segundos para realizar todas sus tareas diarias dentro de la simulación, se estableció la duración de los días de la misma en 5 segundos (recordemos que en la competencia cada día tiene una duración de 15 segundos).

Las corridas con poblaciones de 1000 clasificadores o menos se realizaron en 20 computadoras Pentium IV 3.0Ghz con 2MB de cache y 512 MB de memoria RAM. Las corridas con poblaciones de 2000 y 3000 clasificadores, que requerían más memoria para correr satisfactoriamente, se realizaron en 2 computadoras Pentium IV 2.8Ghz con 1MB de Cache y 2GB de memoria RAM.

El desempeño de los agentes se evaluó en términos de la simulación utilizando dos medidas principales:

(a) Resultado final: balance final promedio de la cuenta bancaria. En base al resultado final se determina cuál es el agente ganador en una simulación. El resultado final indica en general qué tan buenas fueron las inversiones del agente y cuánta ganancia logró acumular al final de la simulación.

(b) Órdenes atendidas: número promedio de órdenes atendidas por cada agente durante las simulaciones de prueba. Esta variable es un indicador del porcentaje del mercado que abarcó el agente. Esta medida de desempeño está directamente relacionada con la estrategia de descuento determinada por el XCSR, ya que si el agente logra dar el mejor precio en la mayoría de los casos entonces atenderá más órdenes.

La combinación de las órdenes atendidas y el resultado final determinará la efectividad del XCSR, ya que lo que se espera aprender es una buena estrategia de determinación del descuento que garantice ganar la orden pero a su vez maximice la ganancia del agente.

Existen además otras medidas de desempeño que permiten evaluar al agente en términos de la simulación:

(c) Utilización de la fábrica: indica el porcentaje de ciclos de fábrica utilizados en promedio durante las corridas de prueba. Los ciclos de fábrica para el agente representan

capacidad productiva por lo tanto debe aprovecharlos al máximo, pero sin exceder la capacidad de la misma.

(d) Intereses: muestra qué porcentaje de la ganancia final del agente es producto de intereses abonados por el banco en su cuenta; un porcentaje negativo indica pérdidas por intereses.

(e) Penalizaciones: indica el porcentaje de la ganancia del agente que representan las pérdidas por penalizaciones debidas a entregas tardías.

(f) Costo de componentes: muestra el porcentaje de la ganancia del agente que representan los gastos por adquisición de componentes.

(g) Costo de almacenamiento: indica el porcentaje de la ganancia del agente que representan los gastos por almacenamiento de componentes y productos manufacturados.

Con respecto al desempeño del sistema de clasificadores, éste fue evaluado en base a:

(h) Precisión: Es un factor que indica qué tan acertada es la recompensa que va a recibir el clasificador.

(i) Experiencia: Es el número de veces que un clasificador ha pertenecido a un *action set* y por consecuencia a recibido una recompensa.

En cada experimento se muestran únicamente las gráficas correspondientes a las medidas en las cuales se consiguieron diferencias considerables de rendimiento entre los agentes.

Tomando en cuenta la varianza observada en los resultados de algunos agentes, se realizaron tablas de frecuencia de la posición obtenida por cada uno de los agentes. Esta posición es relativa de los agentes entre si tomando en cuenta los resultados finales de cada una de las 25 corridas tomadas en cuenta para los resultados. De esta forma el agente que obtiene el mejor resultado final es el que queda en la primera posición y el que tiene el peor resultado final queda en la última. De esta tabla de frecuencia también se determinó la posición promedio de cada uno de los agentes con respecto a los demás.

5.1. Experimento 1: Desempeño de TicTACtoe

El objetivo de este experimento es comparar la versión del agente TicTACtoe que realiza aprendizaje con otras versiones del mismo. Estas otras versiones utilizan estrategias diferentes para resolver el problema de decisión del precio de venta. También es objeto de estudio comparar al agente TicTACtoe con el agente *dummy* provisto por la librería del servidor, al cual llamaremos `Dummy`.

El agente TicTACtoe que realiza aprendizaje, al cual llamaremos `L-TicTACtoe` en este experimento, utiliza una tasa de explotación del 30 %, un tamaño de población de 1000 clasificadores y el algoritmo de bloqueo. Se escogieron estos valores para los parámetros de aprendizaje debido a que fueron los que tuvieron mejor desempeño en sus experimentos correspondientes (ver Secciones 5.2, 5.3 y 5.4). Las otras dos versiones de TicTACtoe involucradas son `Random` y `Static`. La primera decide qué descuento dar sobre el precio de reserva de forma aleatoria mientras que la segunda decide qué descuento otorgar en el día d de la simulación como se explica a continuación:

$$descuento(d) = \begin{cases} 80\,\% & \text{si } freeFactoryCapacity_{d-1} > 80\,\%, \\ 10\,\% & \text{si } freeFactoryCapacity_{d-1} < 5\,\%, \\ 30\,\% & \text{en el resto de los casos.} \end{cases}$$

donde $freeFactoryCapacity_{d-1}$ es el porcentaje de ciclos de fábrica libres en el día $d-1$ de la simulación. Estas reglas "ingenuas" intentan evitar la saturación de la fábrica elevando los precios cada vez que el número de ciclos libres del día anterior estuvo por debajo del 5 % y atraer compradores cuando este valor supera el 80 %. Es importante recordar que los agentes `L-TicTACtoe`, `Static` y `Random`, calculan el precio final de venta de la misma forma (ver Sección 3.3), lo que varía es la forma como calculan el descuento que deben dar en cada caso.

Resultados

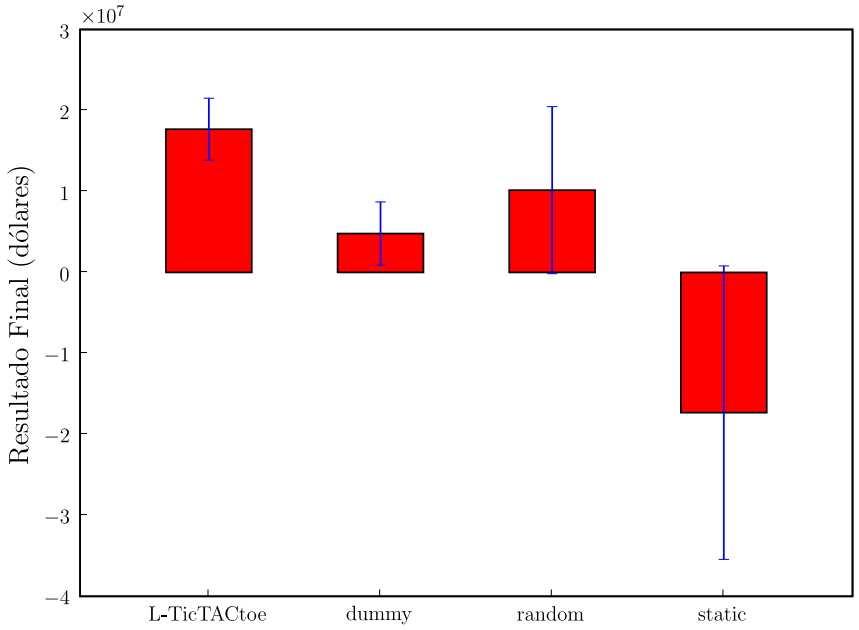
En la Figura 4(a) se observa que el agente con mejor desempeño global (en términos de resultado final) es `L-TicTACtoe`, mientras que los agentes `Random` y `Static` obtienen, en promedio, resultados finales peores que los obtenidos por `L-TicTACtoe`. Se observa también que el desempeño de este agente es casi el cuádruple que el desempeño de `Dummy` y el doble que el de `Random`. Como los agentes de este experimento sólo difieren en su estrategia de selección de descuento, entonces se puede afirmar que esta estrategia sí afecta el desempeño global del agente.

Por otro lado, en la Figura 4(b) se observa que los agentes `Random` y `Static` ganan más ofertas que `L-TicTACtoe`. Sin embargo, las Figuras 5(a) y 5(b) indican que tanto `Random` como `Static` están entregando menos órdenes a tiempo y obteniendo más penalizaciones. Esto significa que las estrategias de oferta de estos agentes no son las mejores, porque se comprometen con órdenes que no pueden entregar a tiempo y, en consecuencia, son penalizados.

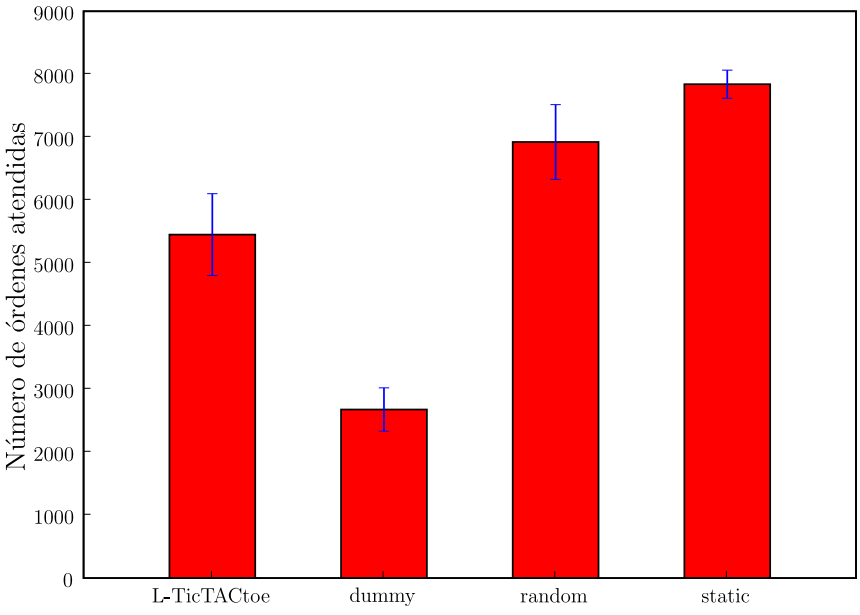
En la Figura 5(c), podemos observar que `Static` obtiene intereses negativos, es decir, tuvo que pagar intereses al banco por tener un balance negativo en su cuenta. Esto se traduce en que, en promedio, la estrategia tomada por `Static` es deficiente, ya que presenta balances bancarios negativos en la mayoría de los días de la simulación. Por otro lado, `L-TicTACtoe` es el agente que gana más intereses y presenta una menor varianza. Esto evidencia un comportamiento más estable con respecto a los balances bancarios.

Con repecto a la utilización de la fábrica, Figura 5(d), podemos apreciar que los agentes `Random` y `Static` presentan una mayor utilización de la fábrica. Esto podría significar un mejor uso de su capacidad productiva pero, al ver las penalizaciones que han sido cobradas a estos dos agentes, podemos notar que en realidad se encuentran superando su capacidad de producción. Por otro lado, `L-TicTACtoe`, no utiliza tanto la fábrica como los agentes anteriores, pero en términos generales, presenta una mejor solución al problema, porque se mantiene atendiendo eficientemente una porción considerable del mercado.

En el Cuadro 3 podemos ver la tabla de frecuencia de posiciones en la que se aprecia

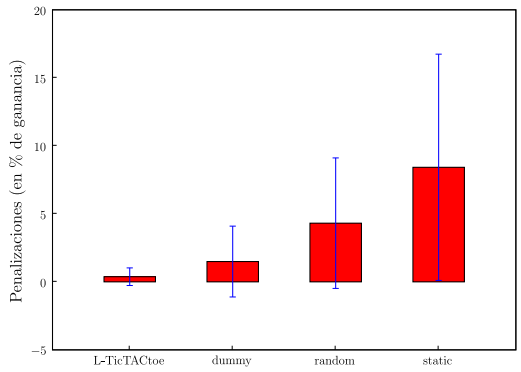


(a) Resultado Final

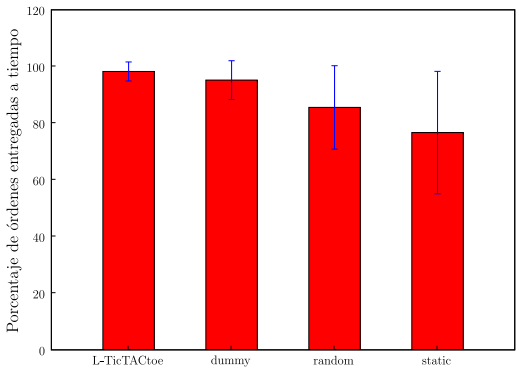


(b) Órdenes Atendidas

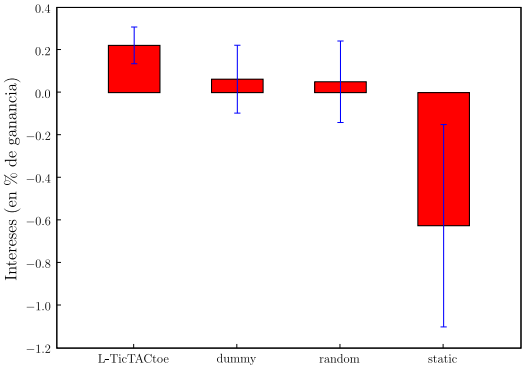
Figura 4: Comparación del desempeño de un agente *dummy* y del agente TicTACtoe empleando diferentes estrategias de selección de descuento. Medidas principales de desempeño.



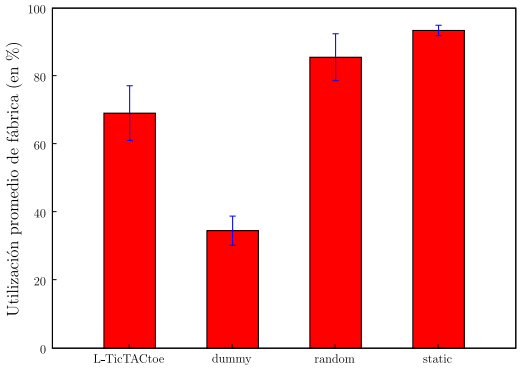
(a) Penalizaciones



(b) Porcentaje de órdenes entregadas a tiempo



(c) Intereses



(d) Utilización de la fábrica

Figura 5: Comparación del desempeño de un agente *dummy* y del agente TicTACtoe empleando diferentes estrategias de selección de descuento. Medidas secundarias de desempeño.

que el agente `L-TicTACtoe` queda en una posición promedio más alta que los demás agentes. Esto concuerda con la gráfica de resultado final. También podemos observar que el agente `Static` presenta los peores resultados al obtener 22 veces la última posición con respecto a los otros agentes y no obtener en ninguna corrida el primero o el segundo lugar.

Finalmente, podemos afirmar que la estrategia de oferta que utiliza `L-TicTACtoe` (que hace uso de aprendizaje de máquinas y de computación evolutiva) sí mejora el desempeño de nuestra aproximación a una solución para el problema TAC SCM. Por otro lado, la estrategia estática es la que presenta los peores resultados, por su incapacidad de adaptarse a nuevas situaciones durante la simulación. Incluso muestra peores resultados que la estrategia aleatoria de venta. Debido a la superioridad observada en el agente `L-TicTACtoe` comparado

Agente	1ro	2do	3ro	4to	**Promedio**
L-TicTACtoe	17	8	0	0	**1.32**
Dummy	0	9	15	1	**2.68**
Random	8	8	7	2	**2.12**
Static	0	0	3	22	**3.88**

Cuadro 3: Frecuencia de posiciones de los agentes en el experimento de desempeño de TicTACtoe

con las demás soluciones, podemos afirmar que el aprendizaje evolutivo cumplió con su objetivo, es decir, crea un sistema de reglas para la estrategia de venta de un agente TAC SCM.

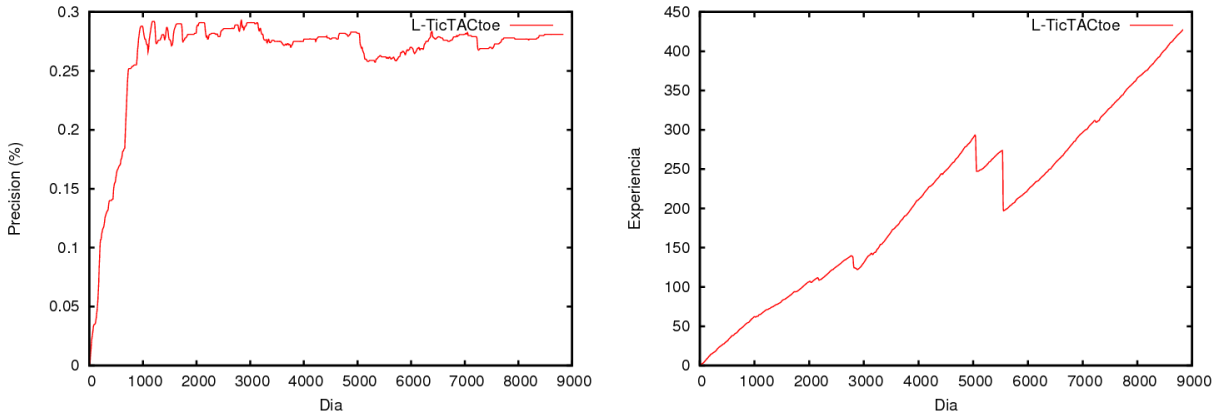
Aprendizaje de L-TicTACtoe

Con respecto al aprendizaje que realizó `L-TicTACtoe`, en la Figura 6 podemos observar cómo varía la precisión y la experiencia de la población conforme pasan los días de la simulación. Estas gráficas determinan el desempeño del aprendizaje que realizó `L-TicTACtoe`.

En la Figura 6(a) podemos observar que la precisión de la población aumenta rápidamente hasta llegar a un punto en el que se estanca. El hecho de que la precisión del agente no sea tan alta se debe a que el ambiente de la simulación es muy dinámico y, por ende, el agente debe adaptarse constantemente a nuevas situaciones. Sin embargo, podemos observar una evolución en la precisión promedio de la población. Por otro lado, en la Figura 6(b) podemos observar la experiencia promedio de los clasificadores de `L-TicTACtoe`, que aumenta conforme pasa el tiempo. Esto quiere decir que la población se encuentra evolucionando y que existen clasificadores con un tiempo prolongado en la población que toman buenas decisiones. Las repentinas disminuciones de experiencia, observadas entre los días 4000 y 6000, representan la eliminación de clasificadores experimentados de la población. Estas eliminaciones se deben a la alta tasa de exploración que tiene el agente, lo cual puede causar que buenos clasificadores dejen de ser usados. Después de pasar un tiempo prolongado sin pertenecer a ningún *action set*, éstos pueden ser borrados de la población con mayor facilidad.

En general, se puede observar a través de estas gráficas que el agente `L-TicTACtoe`, logra

evolucionar satisfactoriamente una serie de reglas adecuadas para tomar la decisión del precio de venta.



(a) Precisión (b) Experiencia

Figura 6: Evolución de la precisión y la experiencia de la población del XCSR del agente `L-TicTACtoe` durante el entrenamiento de 8800 días (40 juegos) .

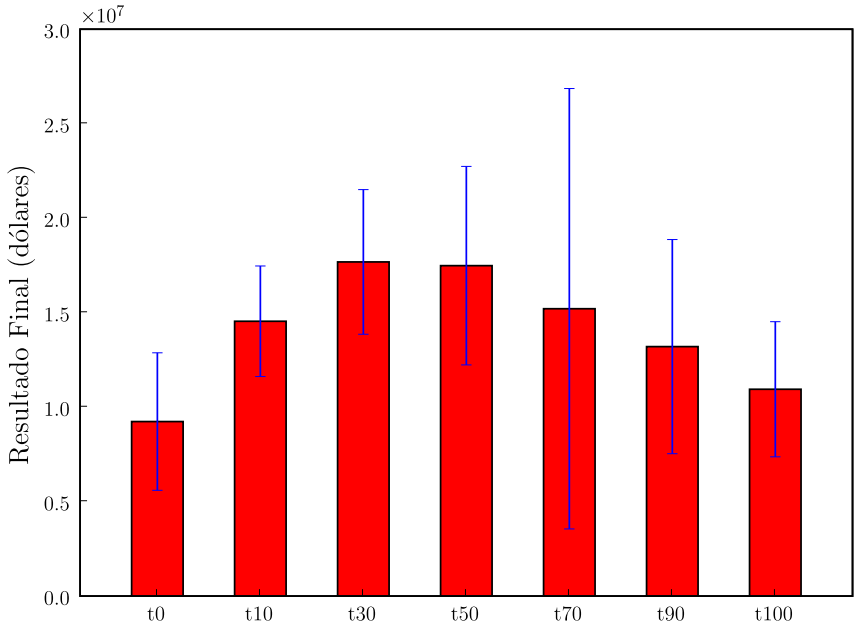
5.2. Experimento 2: Tasa de explotación

Se experimentó con el agente TicTACtoe utilizando diferentes tasas de explotación (0,9; 0,7; 0,5; 0,3), para determinar cuál es la tasa que produce mejores resultados (ver Sección 4.1.6). Los demás parámetros del agente TicTACtoe permanecen iguales, utilizando una población de 1000 individuos y utilizando el algoritmo de bloqueo de clasificadores. El tamaño de la población se fijó en 1000 individuos debido a que fue la que obtuvo mejores resultados (ver Seccion 5.4). Por otro lado, se experimentó con el algoritmo de bloqueo de clasificadores activado debido a que de esta forma se obtuvieron resultados más favorables (ver Sección 5.3).Los agentes involucrados los denominaremos como se indica en el Cuadro 4.

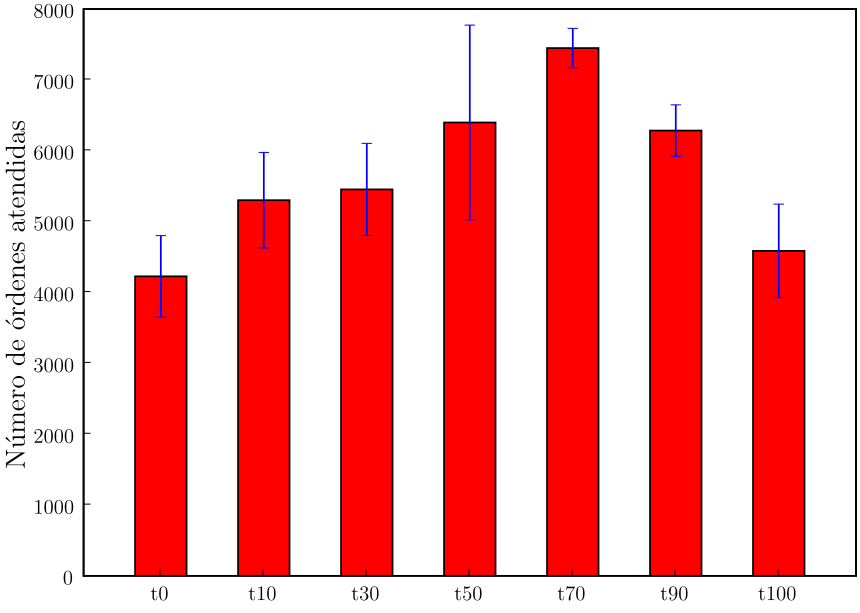
Resultados

En la Figura 7(a) podemos observar que el agente con el menor balance bancario al final del juego es `t0` seguido por `t100`. Lo mismo ocurre en la Figura 7(b) donde `t0` y `t100` tienen la menor cantidad de órdenes atendidas. Esto se traduce en que realizar constantemente exploración, es decir, dar siempre el descuento de venta de forma aleatoria[3] produce los

[3]El descuento se escoge de forma aleatoria entre las opciones que presenta el *match set*.



(a) Resultado Final



(b) Órdenes Atendidas

Figura 7: Comparación del desempeño del agente TicTACtoe empleando diferentes tasas de explotación. Medidas principales.

Nombre	**Descripción**
`t0`	Agente TicTACtoe con tasa de explotación del 0 %
`t10`	Agente TicTACtoe con tasa de explotación del 10 %
`t30`	Agente TicTACtoe con tasa de explotación del 30 %
`t50`	Agente TicTACtoe con tasa de explotación del 50 %
`t70`	Agente TicTACtoe con tasa de explotación del 70 %
`t90`	Agente TicTACtoe con tasa de explotación del 90 %
`t100`	Agente TicTACtoe con tasa de explotación del 100 %
`d30`	Agente *dummy* que compitió con tic30
`d70`	Agente *dummy* que compitió con tic70

Cuadro 4: Agentes involucrados en el experimento de influencia de la tasa de explotación sobre el desempeño del agente TicTACtoe que emplea aprendizaje.

peores resultados. Pero por otro lado, realizar explotación pura, como el agente `t100`, también produce resultados desfavorables. Esto se debe a que el agente `t100` no explora nuevas soluciones y no se adapta a los cambios que puedan ocurrir durante la simulación. En general, se puede observar que lo agentes que combinan explotación y exploración obtienen mejores resultados debido a las características dinámicas del ambiente en el que se desenvuelven.

En la Figura 7(b) podemos observar que el agente con más órdenes atendidas es el agente `t70`. Sin embargo, en la Figura 7(a) podemos observar que el agente `t70`, no es el que termina con el mayor balance monetario. El agente con el mayor balance monetario es `t30`.

Esto significa que el agente `t70` logra abarcar la mayor cantidad de mercado. Sin embargo, no logra finalizar el juego con el mayor resultado porque presenta muchas penalizaciones por entregas tardías, como lo muestra la Figura 8(c). Por otro lado, se puede observar que el agente `t30` no obtiene tantas órdenes. Esto lo lleva a obtener menos penalizaciones que `t70` porque no sobrepasa su capacidad de manufactura y compra. De esto podemos derivar que el agente `t30` aprende a manejar un número de órdenes que le permite minimizar las penalizaciones obtenidas, y de esta manera maximizar su ganancia final.

Si observamos la Figura 8(a), podemos ver que la implementación de TicTACtoe, ya sea con una tasa de explotación de 30 % ó de 70 %, obtiene mucha más ganancia que el agente

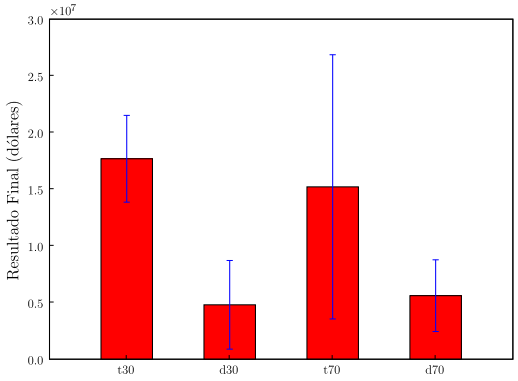
dummy que compitió con él. Lo mismo ocurre en la Figura 8(b), en la cual se observa que la implementación de TicTACtoe abarca mucho más mercado que los agentes *dummy*. De hecho, si observamos el número de órdenes atendidas por `d70` podemos ver que es un poco menor al de las atendidas por `d30`. Esto quiere decir que la estrategia de precios de `t70` hizo que fuese más difícil para su correspondiente *dummy* obtener órdenes.

Con respecto a la utilización de la fábrica, consideremos que idealmente un agente utiliza la fábrica tanto como pueda, para completar más pedidos y, de esta forma, obtener más ganancia[Stan et ál., 2006]. Aunque `t30` y `t70` tienen la misma estrategia para utilizar la fábrica, `t70` presenta un porcentaje mayor de utilización por 20 puntos porcentuales, como se puede observar en la Figura 8(d). Esto se debe a que la cantidad de órdenes atendidas de `t70` es mayor que la de `t30`. Podemos entonces afirmar que, considerando la utilización de la fábrica, `t70` resulta mejor que `t30` porque este último gana menos órdenes.

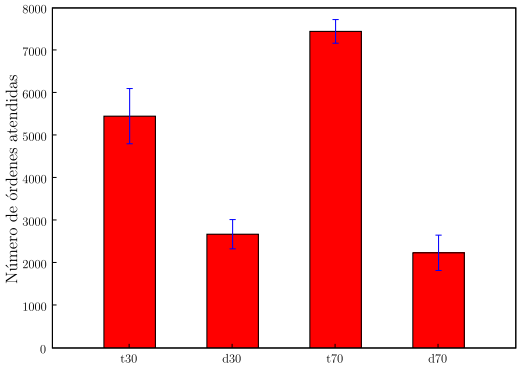
La gráfica anterior concuerda con la gráfica de los ingresos totales, Figura 8(e), en la cual se observa que el agente que obtiene la mayor cantidad de ingresos es `t70`. En esta figura también se puede observar que los ingresos de `d70` son menores a los ingresos de `d30`. Esto es debido a que `t70` ofrece sus productos a precios más competitivos que `t30`. Los ingresos obtenidos por `t70` son proporcionales a sus órdenes atendidas, lo cual evidencia también que este agente propone precios competitivos pero acordes con el costo del producto, es decir, no ofrece productos a precios que le produzcan pérdidas sólo para abarcar mercado.

En el Cuadro 5 se puede observar que el agente `t30` queda en una posición promedio de 2.84, que es la más alta en comparación con los otros agentes. Esto concuerda con los resultados finales de los diferentes agentes en la Figura 7(a). También se puede ver que el agente que alcanza el primer lugar con más frecuencia es el agente `t70`; sin embargo, este agente también queda múltiples oportunidades de último lugar, lo cual explica la gran varianza que podemos notar en las barras correspondientes al mismo. Esto explica por qué la posición promedio de este agente no es tan alta como la del agente `t30`.

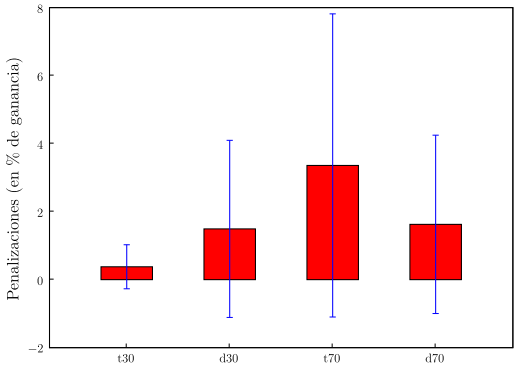
De este experimento se puede concluir que el agente `t30` es el que presenta los mejores resultados en términos del balance final. Sin embargo, la estrategia de ventas que desarrolla



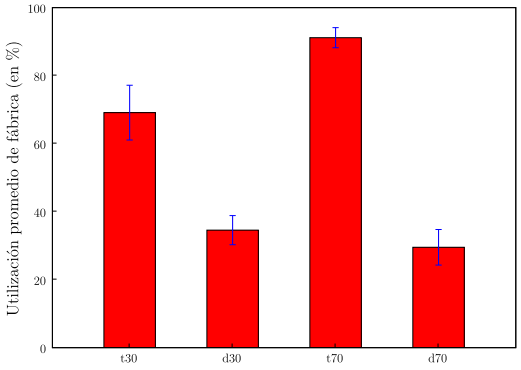
(a) Resultado Final



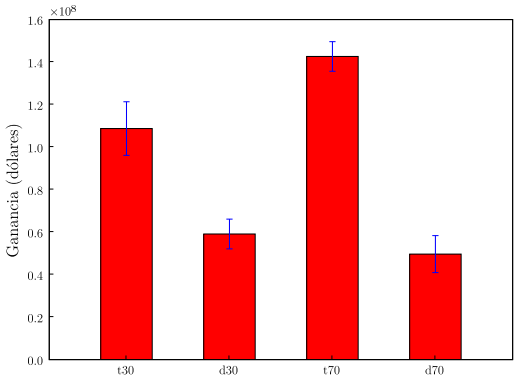
(b) Órdenes Atendidas



(c) Penalizaciones



(d) Utilización de la fábrica



(e) Ingresos

Figura 8: Comparación del desempeño del agente TicTACtoe empleando diferentes tasas de explotación. Medidas secundarias.

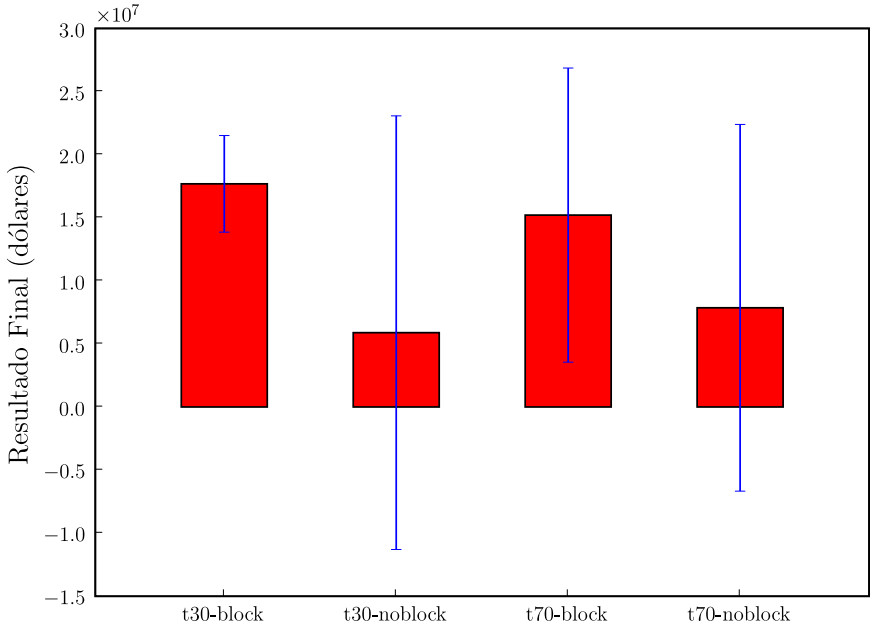
Agente	1ro	2do	3ro	4to	5to	6to	7mo	**Promedio**
t0	0	0	1	2	7	6	9	**5.8**
t10	0	3	7	6	6	2	1	**4.0**
t30	4	6	9	4	1	0	1	**2.84**
t50	7	7	3	2	3	1	2	**2.92**
t70	10	7	0	1	0	0	7	**3.08**
t90	4	2	4	5	2	4	4	**4.08**
t100	0	0	1	5	6	12	1	**5.28**

Cuadro 5: Frecuencia de posiciones de los agentes en el experimento de tasa de explotación

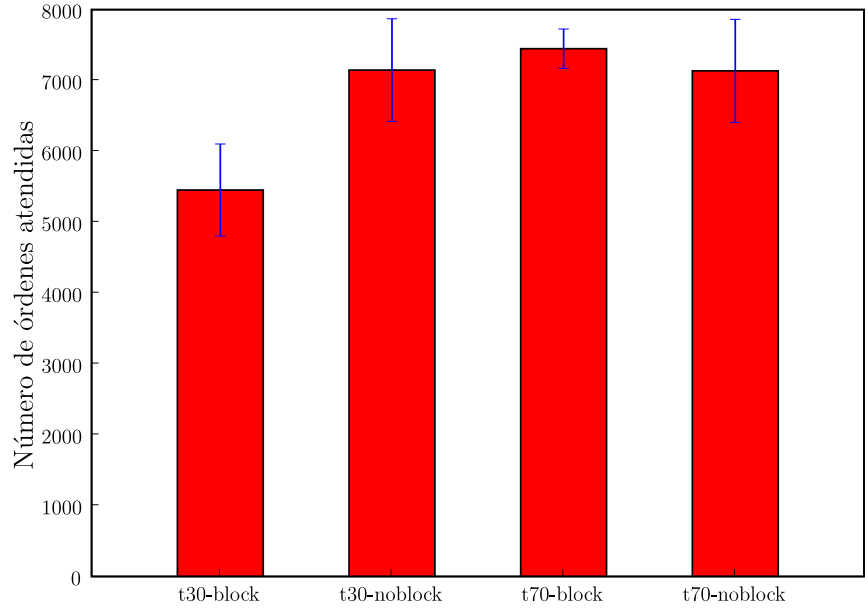
el agente `t70` es bastante prometedora y se esperaría que diese mejores resultados si se mejoran otros mecanismos del agente, por ejemplo, la planificación de manufactura, la compra de componentes o inclusive el mismo XCSR. La implementación de XCSR que se realizó únicamente toma en cuenta la capacidad disponible de la fábrica como un factor de demora en la fabricación de órdenes, pero no considera otros problemas adicionales como la dificultad que existe en la obtención los componentes necesarios por parte de los proveedores.

5.3. Experimento 3: Bloqueo de clasificadores

En este experimento, se analizó el agente TicTACtoe utilizando el algoritmo de bloqueo de clasificadores y sin hacer uso de éste, para corroborar la importancia del mismo (ver Sección 4.1.6). Al no hacer uso del algoritmo se permite borrar clasificadores libremente sin importar si éstos se encuentran esperando recompensa. Para este experimento, se utilizaron las tasas de explotación de 30 % y 70 %, por ser éstas las que dieron los resultados más interesantes del experimento anterior. Todos los agentes que realizan aprendizaje en este experimento utilizaron una población de 1000 clasificadores debido a que fue la que obtuvo mejores resultados (ver Sección 5.4). Este parámetro se estableció debido a los requerimientos de memoria al utilizar poblaciones mayores. En este experimento utilizaremos la leyenda del Cuadro 6 para denominar los agentes involucrados.



(a) Resultado Final



(b) Órdenes Atendidas

Figura 9: Comparación del desempeño del agente TicTACtoe empleando tasas de explotación de 30 % y 70 % con y sin emplear el algoritmo de bloqueo. Medidas principales.

Nombre	Descripción
`t30-block`	Agente con algoritmo de bloqueo de clasificadores (ver Sección 4.1.6) y tasa de explotación de 30 %.
`t30-noblock`	Agente sin algoritmo de bloqueo de clasificadores y tasa de explotación de 30 %.
`t70-block`	Agente con algoritmo de bloqueo de clasificadores (ver Sección 4.1.6) y tasa de explotación de 70 %.
`t70-noblock`	Agente sin algoritmo de bloqueo de clasificadores y tasa de explotación de 70 %.

Cuadro 6: Agentes involucrados en el experimento de influencia del algoritmo de bloqueo sobre el desempeño del agente TicTACtoe que emplea aprendizaje.

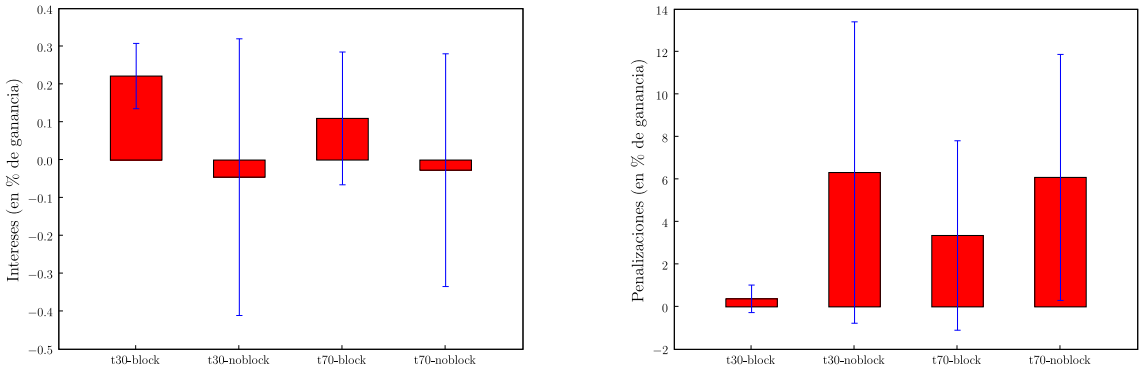
Resultados

En este experimento se puede observar la diferencias en términos de la simulación que presentan los agentes que aprenden utilizando y sin utilizar el algoritmo de bloqueo de clasificadores. En la Figura 9(b) podemos observar que el agente `t30-block` tiene muchas menos órdenes atendidas que el agente `t30-noblock`. También podemos observar que el agente `t70-block` tiene más órdenes atendidas que `t70-noblock`, aún cuando esa diferencia es de apenas un 4 %.

Si contrastamos las órdenes atendidas de `t70-block` con las de `t70-noblock`, Figura 9(b), vemos que no existe mucha diferencia entre estos dos agentes; pero al ver el balance monetario al final del juego, Figura 9(a), podemos ver que `t70-block` dobla la ganancia del agente que no bloquea los clasificadores. Esta diferencia se debe a las penalizaciones que tiene que enfrentar `t70-noblock` lo cual podemos evidenciar en la Figura 10(b). Las penalizaciones obtenidas por los agentes que no realizan bloqueo son mucho mayores, lo cual indica que estos agentes no logran desarrollar un mecanismo apropiado para determinar un precio de oferta según los requerimientos del cliente. Por lo tanto, se puede afirmar que estos agentes hacen ofertas a precios muy bajos, a clientes que quieren cobrar altas penalizaciones y a órdenes que son muy difíciles de fabricar (debido a que los componentes que las conforman

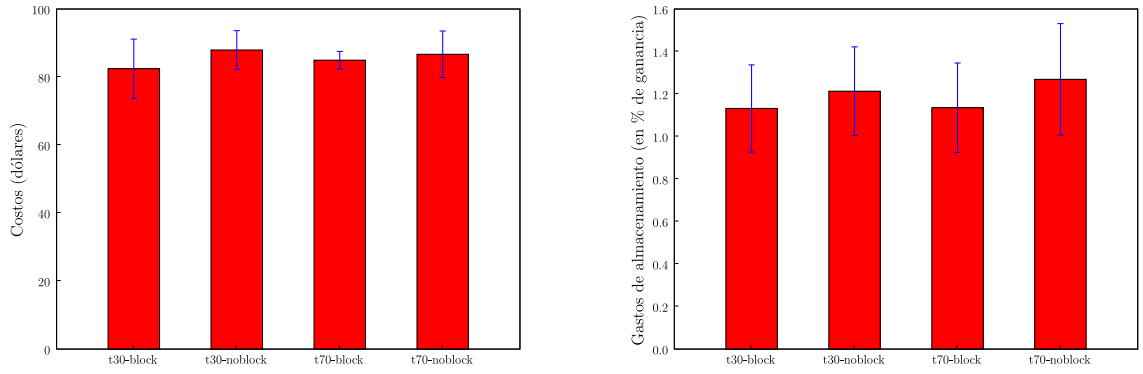
se encuentran escasos). Esto les permite tener muchas órdenes atendidas, pero éstas en su mayoría no representan un buen mercado pues su retraso implica mayores penalizaciones. En el caso de `t30-noblock` se observa el mismo problema de aprendizaje encontrado en `t70-noblock`.

Con respecto a `t30-block` se evidencia lo mismo que en el experimento pasado, es decir, que obtiene su ventaja debido a que obtiene menos órdenes y esto le permite tener menos penalizaciones ya que no tiene que lidiar con el colapso por falta de componentes.



(a) Intereses

(b) Penalizaciones



(c) Costo de componentes

(d) Costo de almacenamiento

Figura 10: Comparación del desempeño del agente TicTACtoe empleando tasas de explotación de 30 % y 70 % con y sin emplear el algoritmo de bloqueo. Medidas secundarias.

En la Figura 10(a) se puede evidenciar que los agentes que no realizan bloqueo obtienen intereses negativos; mientras que los agentes que bloquean tienen intereses positivos. Esto quiere decir que, en promedio, la cuenta bancaria de los agentes que no bloquean se mantiene

con un balance negativo, mientras que las de lo agentes que bloquean se mantiene positiva. Este factor junto con las penalizaciones explica por qué en la Figura 9(a) los agentes que no bloquean son los que terminan con la menor cantidad de dinero al final de la simulación.

Con respecto a los costos, Figuras 10(c) y 10(d), podemos apreciar también que los agentes que realizan bloqueo de clasificadores gastan menos en la compra de componentes y en costos de almacenamiento. Como ambos agentes tienen la misma estrategia para obtener los mejores precios de componentes, se puede afirmar que los agentes que realizan bloqueo se involucran con órdenes cuyo costo de fabricación es menor y pueden obtener mucha más ganancia.

En el Cuadro 7 podemos observar que en cuanto a las posiciones que tomaron los agentes respecto a los otros, los agentes que bloquean en promedio quedan en una mejor posición que los agentes que no bloquean. Además de esto podemos ver que los agentes con la arquitectura de TicTACtoe en promedio se posicionan mejor que los agentes *dummy*. Esto concuerda con las gráficas observadas anteriormente donde se puede observar que los agentes que bloquean desarrollan una mejor estrategia.

Agente	1ro	2do	3ro	4to	5to	6to	7mo	8vo	**Promedio**
t30-block	6	9	8	1	0	1	0	0	**2.32**
t30-noblock	4	4	5	2	1	1	2	6	**4.32**
t70-block	9	6	2	2	2	1	0	3	**3.0**
t70-noblock	6	2	3	4	2	0	2	6	**4.28**
d30-block	0	2	1	5	3	6	5	3	**5.48**
d30-noblock	0	1	1	3	8	4	6	2	**5.56**
d70-block	0	1	3	3	6	5	5	2	**5.36**
d70-noblock	0	0	2	5	3	7	5	3	**5.68**

Cuadro 7: Frecuencia de posiciones de los agentes en el experimento de algoritmo de bloqueo

Es importante analizar la experiencia del sistema XCSR de cada agente, ya que indica cuántas veces los clasificadores han evolucionado. En la Figura 11 podemos observar que la experiencia de los clasificadores de los agentes que realizan bloqueo es mayor que las de los

agentes que no lo realizan. Esto de debe a que los agentes que no bloquean los clasificadores permiten borrar un clasificador sin esperar que éste tenga toda su información actualizada. Los clasificadores entonces son borrados de la población basándose en información atrasada. Esto permite en muchos casos que clasificadores que llevaron a buenas decisiones, sean borrados antes de que puedan ser recompensados y, de esta forma, se pierde el conocimiento que se hubiese obtenido a través de esta regla si se hubiese bloqueado.

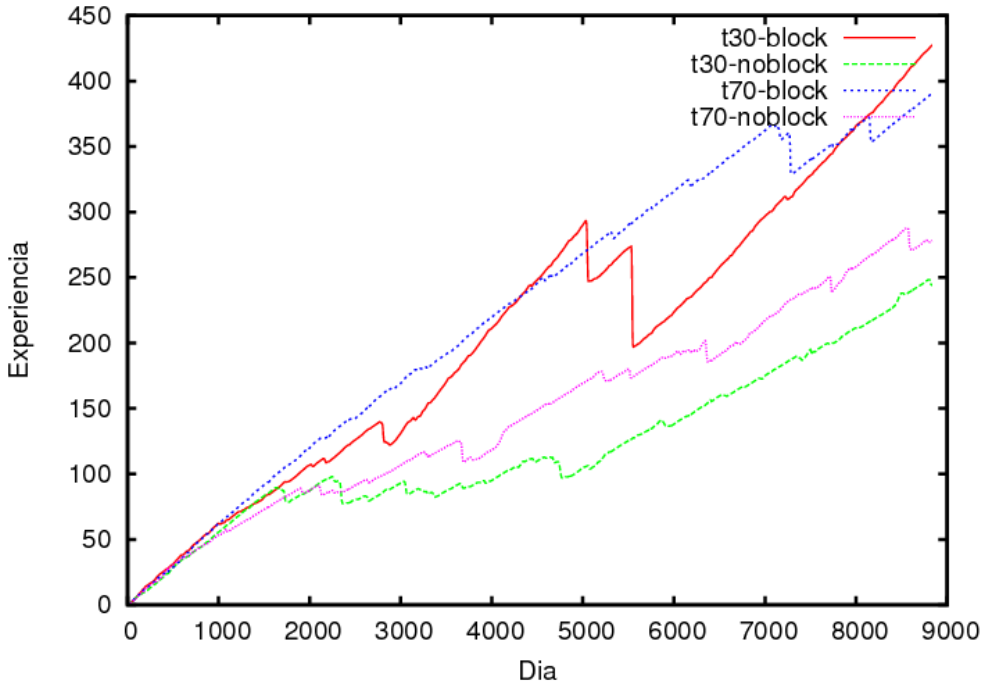


Figura 11: Evolución de la experiencia de la población del XCSR del agente TicTACtoe empleando tasas de explotación de 30 % y 70 % con y sin emplear el algoritmo de bloqueo durante el entrenamiento de 8800 días (40 juegos).

Podemos afirmar que los resultados de este experimento comprobaron que los agentes que manejan un mecanismo de bloqueo de clasificadores tienen un mejor desempeño que los agentes que no bloquean. Además se observó que el bloqueo incrementa la experiencia del XCSR, por lo tanto aumenta el desempeño del aprendizaje.

Nombre	Descripción
t500	Agente TicTACtoe con población de 500 individuos
t1000	Agente TicTACtoe con población de 1000 individuos
t2000	Agente TicTACtoe con población de 2000 individuos
t3000	Agente TicTACtoe con población de 3000 individuos

Cuadro 8: Agentes involucrados en el experimento de influencia del tamaño de la población sobre el agente TicTACtoe que emplea aprendizaje.

5.4. Experimento 4: Tamaño de la población

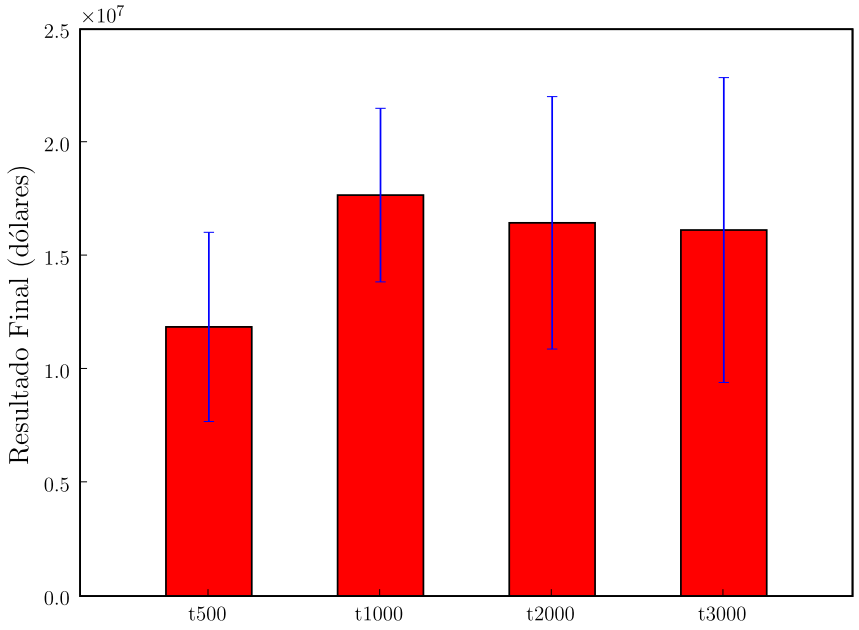
Este experimento tiene el objetivo de determinar cuál es el tamaño de población que presenta los mejores resultados para el problema de ventas con el que se enfrenta TicTACtoe. En esta oportunidad se experimentó con el agente TicTACtoe con poblaciones de 500, 1000, 2000 y 3000 clasificadores. El resto de los parámetros permanecieron fijos: la tasa de explotación en 30 % y utilizando el algoritmo de bloqueo, ya que demostraron ser los mejores valores para dichos parámetros (ver Secciones 5.2 y 5.3). Los nombres con los que se identificaron los distintos agentes pueden consultarse en el Cuadro 8.

Resultados

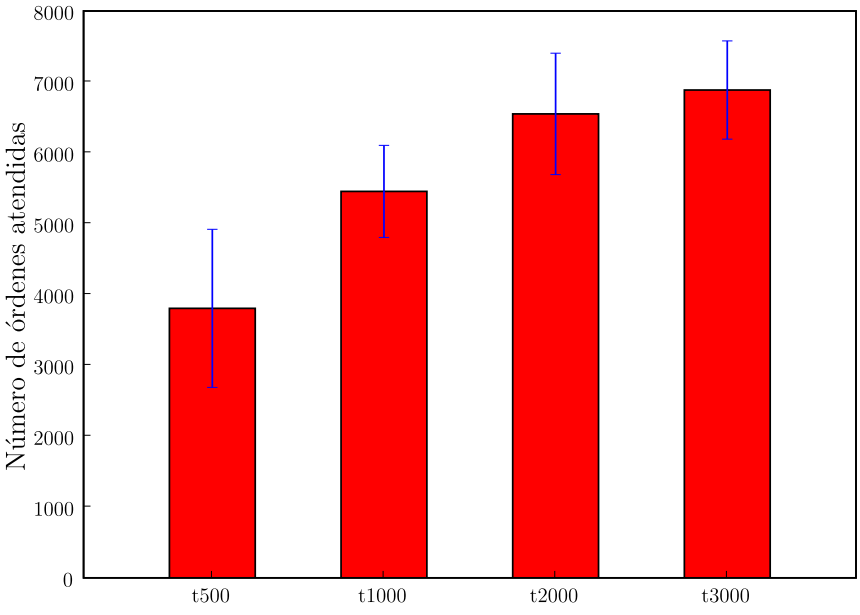
En la Figura 12(b) observamos que, conforme se aumenta el tamaño de la población, aumenta también el número de órdenes atendidas por el agente (lo que representa que el agente dio mejores ofertas).

Por otro lado, en la Figura 13(a), observamos que la cantidad de dinero pagada por penalizaciones aumenta conforme aumenta el tamaño de la población. Si relacionamos esto con la gráfica de órdenes atendidas, podemos ver claramente que cuando los agentes pasan cierto umbral de órdenes atendidas, ya no pueden manejar su mercado eficientemente y obtienen muchas más penalizaciones.

Al observar la Figura 12(a), notamos que el agente que finaliza las simulaciones con mayor balance en su cuenta es t1000. Esto es gracias a que obtuvo un numero de órdenes considerable y se mantuvo con un bajo nivel de penalizaciones como se hizo notar anteriormente. Los



(a) Resultado Final



(b) Órdenes Atendidas

Figura 12: Comparación del desempeño del agente TicTACtoe empleando diferentes tamaños de población. Medidas principales.

agentes `t2000` y `t3000` presentan un resultado final inferior debido a las penalizaciones obtenidas por la cantidad de órdenes aceptadas que no pudieron despachar o fueron entregadas tarde.

También podemos observar que `t500` es el que obtiene el resultado final más bajo. Esto se debe al limitado número de órdenes que atendió; lo cual se traduce en menos ganancia, como se observa en la Figura 13(b). Se puede observar mediante los resultados que obtuvo este agente que una población de 500 clasificadores resulta insuficiente para el tamaño del problema de venta, ya que no presenta buenos resultados.

Por último, en la Figura 13(c) podemos observar que el agente que gana más intereses a lo largo del juego es `t1000`, es decir, su balance bancario fue el más elevado a lo largo de las simulaciones. Esto quiere decir que la estrategia de este agente lo mantiene en primer lugar en la mayoría de las etapas del juego por ser el que tuvo los balances bancarios en promedio más elevados.

Tomando en cuenta las posiciones en las cuales quedaron los agentes con respecto a los otros, en el Cuadro 9, podemos ver que en promedio el agente que queda en una mejor posición es `t1000`. También podemos ver que la estrategia desarrollada por `t500` es deficiente, ya que no obtiene en ninguna oportunidad el primer lugar y su posición promedio es más baja que la de los demás agentes TicTACtoe.

Agente	1ro	2do	3ro	4to	5to	6to	7mo	8vo	**Promedio**
t500	0	6	6	9	1	2	1	0	**3.6**
t1000	6	12	5	1	1	0	0	0	**2.16**
t2000	8	4	9	1	2	1	0	0	**2.52**
t3000	11	3	4	5	0	0	2	0	**2.52**
d500	0	0	0	2	6	5	6	6	**6.32**
d1000	0	0	1	2	6	4	8	4	**6.12**
d2000	0	0	0	1	4	8	7	5	**6.44**
d3000	0	0	0	4	5	5	1	10	**6.32**

Cuadro 9: Frecuencia de posiciones de los agentes en el experimento de tamaño de población

En conclusión, este experimento demostró que aumentar el tamaño de la población incrementa el número de órdenes atendidas. Con respecto al desempeño global, la población de 1000 clasificadores muestra los mejores resultados. Si se mejoran las estrategias de producción y compra del agente para atender satifactoriamente un mayor número de órdenes, es posible que el uso de poblaciones de 2000 y 3000 clasificadores mejore el desempeño del agente. Sin embargo, sería necesario estudiar el compromiso entre el uso de recursos que implica el empleo de poblaciones tan grandes y la mejora obtenida mediante su uso.

5.5. Experimento 5: Comparación con otras soluciones al problema TAC SCM

El objetivo de este experimento es medir el desempeño del agente TicTACtoe frente a otras soluciones del problema TAC SCM que han tenido buenos resultados en esta competencia. En el Cuadro 10 se muestran los agentes participantes en este experimento.

La metodología de este experimento difiere de la de los experimentos anteriores. En esta ocasión los cuatro agentes compitieron en la misma simulación para observar el comportamiento de éstos cuando competían entre ellos. El experimento se realizó de esta forma debido a que son implementaciones completamente diferentes y es interesante ver como se afectan entre sí. Para completar los seis agentes participantes se incluyeron 2 agentes *dummy* proporcionados por la librería del servidor (los cuales no figuran en las gráficas). Con respecto a los parámetros de aprendizaje de `TicTACtoe` se estableció una tasa de explotación del 30 %, una población máxima de 1000 clasificadores y se activó el algoritmo de bloqueo.

Resultados

Con respecto al resultado final podemos ver en la Figura 14(a) que el agente `TicTACtoe` tiene un balance bancario final mucho menor que los agentes `Phant` y `Maxon`. Por otro lado los balances del agente `TicTACtoe` están en el mismo rango que los balances del agente `Tiancalli`.

En la Figura 14(b) podemos ver también que el agente `TicTACtoe` obtiene aproxima-

Agente	Descripción
`TicTACtoe`	El agente desarrollado en esta investigación.
`Phant`	Mejor conocido como PhantAgent. Fue desarrollado por la Universidad Politécnica de Bucarest en Rumanía y obtuvo el segundo lugar en la competencia TAC SCM del año 2006 y en primer lugar en año 2007.
`Maxon`	Fue desarrollado en la Universidad de Ciencias Aplicadas de Alemania por Wolfram Conen y obtuvo el 4to lugar en las competencias TAC SCM de los años 2006 y 2007.
`Tiancalli`	Fue desarrollado en la Benemérita Universidad Autónoma de Puebla en México y llegó a cuartos de final en la competencia TAC SCM del año 2005.

Cuadro 10: Agentes participantes en el experimento de comparación con otras soluciones de TAC SCM

damente la mitad de órdenes que los agentes `Phant` y `Maxon`. También podemos ver que nuestro agente obtiene la misma cantidad que el agente `Tiancalli`. Esto puede evidenciar que la estrategia de ventas de los agentes `Phant` y `Maxon` es mucho mejor que las del agente `TicTACtoe`.

Otro factor interesante a analizar entre estos agentes es el costo que fue pagado para adquirir componentes. En la Figura 15(a) podemos observar que los agentes `Maxon` y `Phant` incurren en menos costos que el agente `TicTACtoe`. Esto indica que se puede mejorar considerablemente la estrategia de compra de nuestro agente para que, reduciendo estos costos, pueda obtener mucha más ganancia al final del juego.

Con respecto a la utilización de la fábrica en la Figura 15(b) podemos ver que los agentes `Maxon` y `Phant` utilizan la fábrica de manera eficiente, mientras que `TicTACtoe`, debido a la poca cantidad de órdenes que puede acaparar frente a estos agentes, no utiliza tantos ciclos de fábrica como ellos. De hecho, podemos observar que el agente `Phant` utiliza su fábrica al máximo, lo cual indica que utilizar la capacidad productiva en una mejor manera y sin caer en demasiadas penalizaciones lleva a mejores balances bancarios al final del juego.

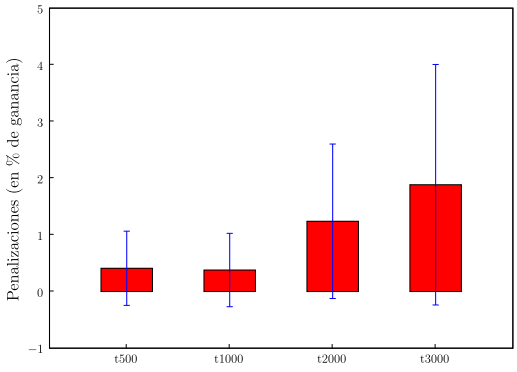
Con respecto a las posiciones en las cuales quedaron los agentes con respecto a los otros,

Agente	1ro	2do	3ro	4to	**Promedio**
tictactoe	0	0	15	10	**3.4**
phant	25	0	0	0	**1.0**
maxon	0	25	0	0	**2.0**
tiancalli	0	0	10	15	**3.6**

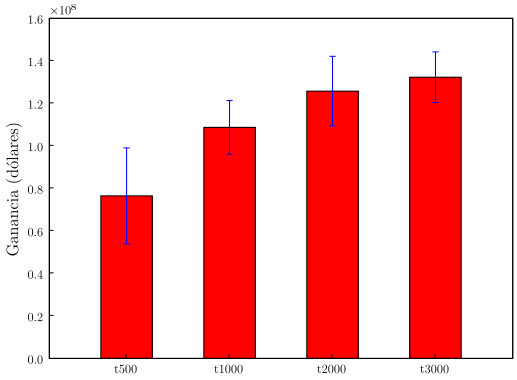
Cuadro 11: Frecuencia de posiciones de los agentes en el experimento de comparación con otras soluciones al problema TAC SCM.

podemos ver que los agentes `Phant` y `Maxon` obtienen el primero y el segundo lugar respectivamente en todas las corridas. Por otro lado nuestro agente en promedio obtiene una posición mejor que el agente `Tiancalli`.

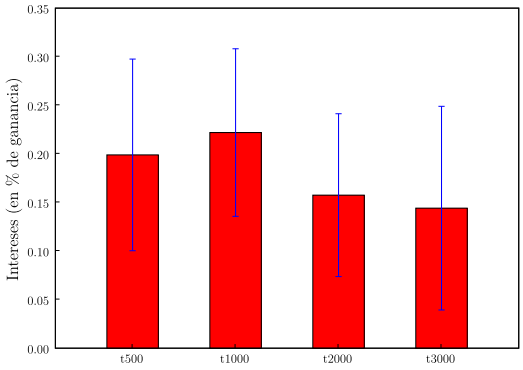
En conclusión, a través de este experimento podemos ver que la solución presentada en esta investigación puede mejorarse en muchos aspectos, ya que pudimos ver que las soluciones ganadoras de la competencia TAC SCM se comportan mucho mejor que la solución que presentamos actualmente. En nuestro caso tanto la estrategia de venta como la estrategia de compra y producción pueden mejorarse aún más, para de esta forma mejorar el desempeño global del agente.



(a) Penalizaciones

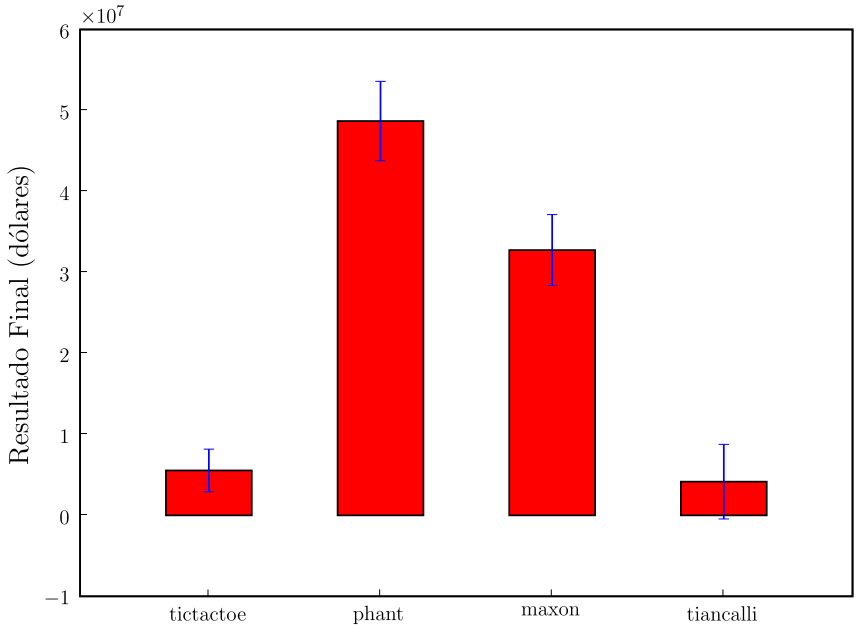


(b) Ingresos

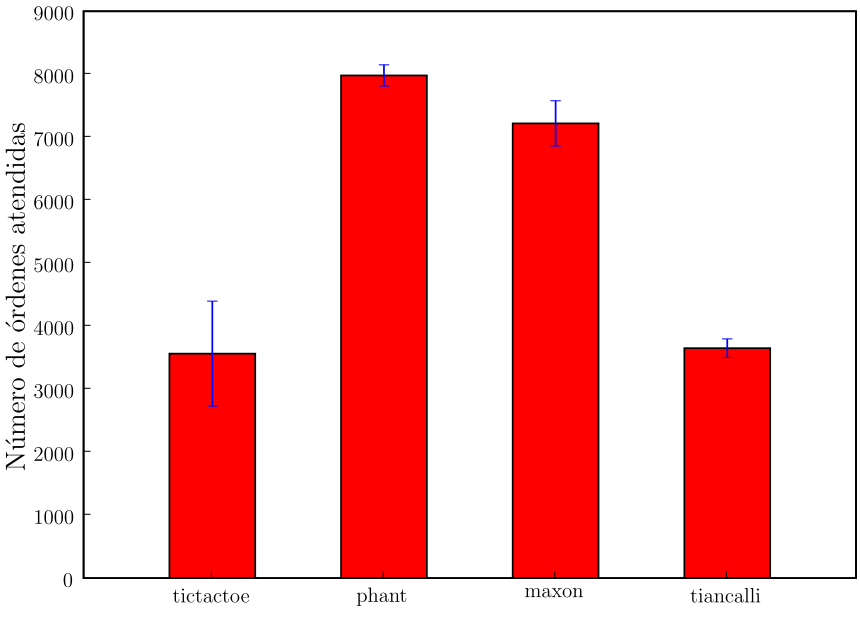


(c) Intereses

Figura 13: Comparación del desempeño del agente TicTACtoe empleando diferentes tamaños de población. Medidas secundarias.

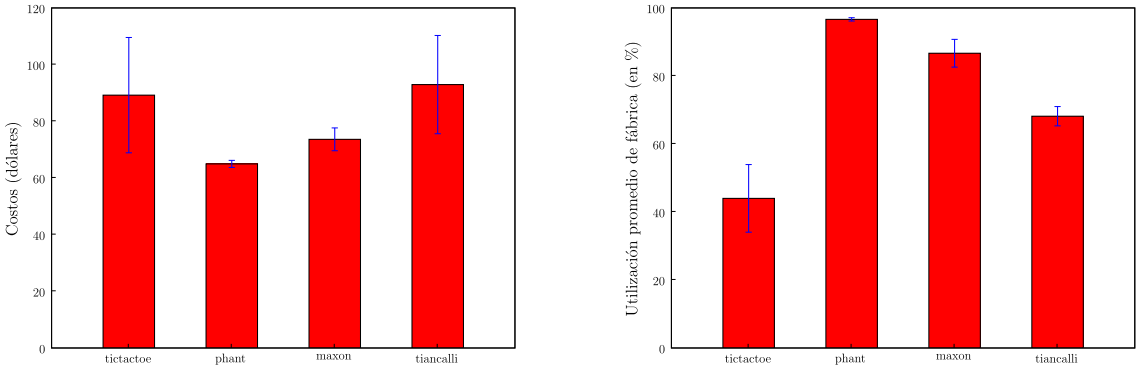


(a) Resultado Final



(b) Órdenes Atendidas

Figura 14: Comparación del desempeño del agente TicTACtoe comparado con otros agentes participantes en la competencia TAC SCM. Medidas principales.



(a) Costo de componentes

(b) Utilización de la fábrica

Figura 15: Comparación del desempeño del agente TicTACtoe comparado con otros agentes participantes en la competencia TAC SCM. Medidas secundarias.

Capítulo 6

Conclusiones y Recomendaciones

Se cumplieron con los objetivos planteados al inicio de la investigación entre los cuales están: analizar el ambiente de la TAC SCM, analizar soluciones previas exitosas a los problemas presentados en esta competencia, diseñar e implementar un agente base capaz de resolver dichos problemas y analizar e integrar el mecanismo de XCS en uno de los subproblemas asociados dentro del agente base.

Se implementó un agente de corretaje que constituye una solución al problema TAC SCM y está compuesto por los siguientes elementos:

- Una agenda que permite guardar y centralizar la información de las decisiones tomadas por el agente.
- Una estrategia de compras estática basada en la obtención de componentes tomando en cuenta los compromisos adquiridos en los días próximos de la simulación.
- Una estrategia de producción estática que se encarga de fabricar componentes priorizando las órdenes de acuerdo a la ganancia que representan y su fecha límite de entrega.
- Una estrategia de ventas que incluye técnicas de aprendizaje evolutivo como medio de soporte para la toma de decisiones, específicamente un sistema de clasificadores genéticos XCSR.

Se evolucionó, mediante el algoritmo de XCSR, un conjunto de reglas adecuadas al problema de ventas de TAC SCM, que representan una mejor solución que las soluciones alternativas presentadas en este trabajo: la estrategia aleatoria y la estrategia estática. Si se mejorasen las estrategias actuales del agente como la compra de componentes o la planificación de la producción podrían obtenerse resultados aún mejores.

Se mostró la influencia de la tasa de explotación y el tamaño de la población en el aprendizaje. Dada la arquitectura actual del agente, las tasas de explotación pequeñas mejoran el desempeño del aprendizaje debido al carácter dinámico de la simulación. Por otro lado se mostró que para el tamaño del problema las poblaciones de 500 clasificadores resultaban insuficientes. Las poblaciones muy grandes tampoco dieron resultados sobresalientes y se caracterizaron por acaparar demasiados recursos.

Se presentó el algoritmo de bloqueo de clasificadores como una solución al problema de retraso de la recompensa en XCS. Se mostró que el bloqueo de clasificadores en el algoritmo de XCSR aumenta el desempeño del mismo en problemas que implican este tipo de retraso. Así mismo, se mostró que utilizando el algoritmo de bloqueo se aumenta la experiencia promedio de los clasificadores, lo cual implica una evolución más rápida del sistema dadas las dificultades del retraso en la recompensa.

Recomendaciones y direcciones futuras

En vista de los problemas de producción observados en el agente al superar un cierto umbral de órdenes atendidas, es recomendable mejorar las estrategias de compra y producción del mismo. Entre las mismas está agregar un planificador a la estrategia de producción. Al realizar estas mejoras será necesario volver a determinar cuáles serían los parámetros de XCSR que mejor se ajustan a la nueva arquitectura del agente.

Con el fin de mejorar la estrategia de ventas, sería interesante incorporar como uno de los rasgos del XCSR una medida de dificultad para la obtención de los componentes involucrados con la orden específica, para dar precios de venta tomando en cuenta qué tan escasos son los componentes necesarios para fabricarla.

También se sugiere realizar pruebas en profundidad utilizando diferentes operadores de cruce y mutación para determinar cuáles serían los más apropiados al problema.

Debido a que el problema de las ventas también podría ser visto como un problema de "multiples pasos" (*multistep*), sería interesante modelar el problema de ventas como un problema de este tipo para comparar de esta forma ambas soluciones.

Bibliografía

[Arulampalam et ál., 2002] Arulampalam, S., Maskell, S., Gordon, N., y Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*.

[Benisch et ál., 2004] Benisch, M., Greenwald, A., Grypari, I., Lederman, R., Naroditskiy, V., y Tschantz, M. (2004). Botticelli: a supply chain management agent designed to optimize under uncertainty. *SIGecom Exch.*, 4(3):29–37.

[Benisch et ál., 2006] Benisch, M., Sardinha, A., Andrews, J., y Sadeh, N. (2006). CMieux: adaptive strategies for competitive supply chain trading. En *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pages 47–58, New York, NY, USA. ACM Press.

[Bull, 2004] Bull, L. (2004). *Applications of Learning Classifier Systems*. SpringerVerlag.

[Butz, 2006] Butz, M. (2006). Illigal java-xcs - lcs web. Disponible en: `http://lcsweb.cs.bath.ac.uk/software/software.2006-02-15.6004875636/`.

[Butz y Wilson, 2001] Butz, M. V. y Wilson, S. W. (2001). An algorithmic description of XCS. *Lecture Notes in Computer Science*, 1996:253–?? Disponible en: `http://citeseer.ist.psu.edu/butz01algorithmic.html`.

[Collins et ál., 2006] Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., y Janson, S. (2006). The Supply Chain Management Game for the 2007 Trading Agent Competition. Disponible en: `http://www.sics.se/tac/`.

[Eriksson et ál., 2006] Eriksson, J., Finne, N., y Janson, S. (2006). Evolution of a supply chain management game for the Trading Agent Competition. *AI Commun.*, 19(1):1–12.

[He et ál., 2006] He, M., Rogers, A., Luo, X., y Jennings, N. R. (2006). Designing a successful trading agent for supply chain management. En *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1159–1166, New York, NY, USA. ACM Press.

[Holland, 1976] Holland, J. H. (1976). Adaptation. En Rosen, R. y Snell, F. M., editores, *Progress in theoretical biology IV*, pages 263–293. Academic Press, Nueva York.

[Loiacono, 2004] Loiacono, D. (2004). Evolving rules with xcsf: Analysis of generalization and performance. Proyecto de Grado, Politecnico di Milano, Facoltà di Ingegneria dell' Informazione.

[Macías et ál., 2006] Macías, D., Vilariño, D., y López, F. (2006). An agent for the Supply Chain Management game 2006.

[Mitkas et ál., 2006] Mitkas, P. A., Kontogounis, I., Chatzidimitriou, K. C., y Symeonidis, A. L. (2006). A Robust Agent Design for Dynamic SCM Environments. En *SETN*, pages 127–136.

[Pardoe y Stone, 2004] Pardoe, D. y Stone, P. (2004). Bidding for customer orders in TAC SCM: A learning approach. Disponible en: `http://citeseer.ist.psu.edu/655704.html;http://www.eecs.harvard.edu/tada04/tacauctions.pdf`.

[Pardoe y Stone, 2006] Pardoe, D. y Stone, P. (2006). An autonomous agent for supply chain management.

[Petrić, 2005] Petrić, A. (2005). A. Petrić y K. Jurasović: KrokodilAgent: a supply chain management agent. En *8th International Conference on Telecommunications – ConTEL 2005*, pages 297–302, Zagreb, Croacia. Faculty of Electrical Engineering and Computing of the University of Zagreb.

[Stan et ál., 2006] Stan, M., Stan, B., y Florea, A. M. (2006). A dinamic strategy agent for supply chain management.

[Stone y Bull, 2003] Stone, C. y Bull, L. (2003). For real! XCS with continuous-valued inputs. *Evol. Comput.*, 11(3):299–336.

[Sutton y Barto, 1998] Sutton, R. y Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

[TAC Team, 2007] TAC Team (2007). Lugar *web* de la competencia TAC. Disponible en: `http://www.sics.se/tac/`.

[Wilson, 1995] Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175. Disponible en: `citeseer.ist.psu.edu/wilson95classifier.html`.

[Wilson, 2000] Wilson, S. W. (2000). Get real! XCS with continuous-valued inputs. *Lecture Notes in Computer Science*, 1813:209–?? Disponible en: `citeseer.ist.psu.edu/233869.html`.

[Wilson, 2001] Wilson, S. W. (2001). Mining oblique data with XCS. *Lecture Notes in Computer Science*, 1996:158–?? Disponible en: `citeseer.ist.psu.edu/article/wilson00mining.html`.

[Witten y Frank, 1999] Witten, I. H. y Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Apéndice A

Tabla de parámetros fijos de XCSR

Este apéndice contiene los valores empleados para los parámetros personalizables de XCSR en la implementación de TicTACtoe.

Parámetro	**Valor**	**Descripción**
α	0.1	Tasa de aprendizaje para modificar la precisión de un clasificador.
β	0.2	Tasa de aprendizaje para la actualización del *fitness*, la predicción y el error de predicción.
δ	0.1	Fracción del *fitness* promedio de la población debajo la cual el *fitness* de un clasificador es tomado en cuenta en el cálculo de la probabilidad de eliminarlo.
ν	5	Exponente de la función fuerza para la evaluación del *fitness*.
θ_{GA}	25	Es el umbral para la aplicación de el algoritmo genético sobre un *action set*.
ϵ_0	10	El umbral del error a partir del cual la precición del clasificador se coloca en 1.
θ_{del}	20	El umbral sobre el cual el *fitness* de un clasificador debe ser considerado en su probabilidad de borrarse.
pX	0.8	Probabilidad de realizar el cruce entre clasificadores.
pM	0.04	Probabilidad de mutar un alelo o la acción de un clasificador.
$p_{dontcare}$	0.1	Probabilidad de aplicar la función de *don't care* durante el proceso de *covering*.
$predictionError$ $Reduction$	0.25	Reducción de la predicción del error cuando se crea un nuevo clasificador.
$fitnessReduction$	0.1	Reducción del *fitness* cuando se crea un nuevo clasificador.
θ_{sub}	20	Experiencia requerida por un clasificador para subsumir a otro.
$doGA$ $Subsumption$	*true*	Determina si la subsunción debe ser realizada durante el algoritmo genético.

doActionSet Subsumption	*true*	Determina si la subsunción debe ser realizada sobre un *action set*.
predictionIni	10.0	Valor inicial de la predicción al generar un nuevo clasificador.
prediction ErrorIni	0.0	Valor inicial del error de predicción de un nuevo clasificador.
$fitnessIni$	0.01	Valor inicial del *fitness* de un nuevo clasificador.
$dontCareMin$	0.0	Cota inferior para la función de *don't care*.
$dontCareMax$	1.0	Cota superior para la función de *don't care*.

Apéndice B

Resultados detallados de los experimentos

B.1. Experimento 1: Desempeño de TicTACtoe

	Agente	Promedio	Varianza
Resultado Final (US$)	L-TicTACtoe	17,684,004.68	3,827,810.34427
	Dummy	4,800,379.28	3,909,834.59393
	Random	10,157,938.28	10,313,748.0073
	Static	-17,317,410.84	18,120,669.0059
Órdenes Atendidas	L-TicTACtoe	5,453.96	649.655644682
	Dummy	2,676.88	344.13167538
	Random	6,924.28	593.058985262
	Static	7,841.8	223.251353113
Ingresos (US$)	L-TicTACtoe	108,718,063.44	12,611,823.1562
	Dummy	59,104,351.36	6,999,544.47046
	Random	129,693,391.12	10,135,822.7452
	Static	125,074,861.76	6,238,071.14129
Costo (%)†	L-TicTACtoe	82.4446968276	8.70175892256
	Dummy	89.7694276972	15.1416649353
	Random	86.7953352657	5.81285079476
	Static	103.286314949	3.661228607
Costo de Almacenamiento (%)†	L-TicTACtoe	1.13215633268	0.204951204161
	Dummy	0.678682619418	0.160817587025
	Random	1.1080883826	0.246903886956
	Static	1.51102836606	0.253585443642

Penalizaciones (%)†	L-TicTACtoe	0.379154178208	0.647384659544
	Dummy	1.49316830266	2.60405440949
	Random	4.31533905596	4.79727942651
	Static	8.42308068284	8.31749255035
Intereses (%)†	L-TicTACtoe	0.221937875239	0.0862373559729
	Dummy	0.0631498005496	0.159391200298
	Random	0.0510340268139	0.191407435416
	Static	-0.625212603873	0.475252214144
Utilización de fábrica (%)	L-TicTACtoe	69.12	8.05357063668
	Dummy	34.56	4.28252262107
	Random	85.56	6.90458784674
	Static	93.44	1.52970585408
Órdenes entregadas a tiempo (%)	L-TicTACtoe	98.24	3.35757849251
	Dummy	95.2	6.83739716559
	Random	85.56	14.7113334995
	Static	76.68	21.6019289262
Órdenes entregadas tarde (%)	L-TicTACtoe	1.52	2.97377425729
	Dummy	4.04	5.4273996229
	Random	12.4	12.5133262298
	Static	19.24	17.9194493963
Órdenes nunca entregadas (%)	L-TicTACtoe	0.32	0.556776436283
	Dummy	0.84	1.67531091642
	Random	2.16	2.44404037064
	Static	4.16	4.74938592522

† Este porcentaje es calculado en base a los ingresos totales obtenidos por el agente.

B.2. Experimento 2: Tasas de Explotación

	Agente	Promedio	Varianza
Resultado Final (US$)	t0	9,235,641.44	3,638,978.42787
	d0	1,049,531.48	5,417,504.49016
	t10	14,543,995.56	2,925,682.01164
	d10	4,605,412.12	2,723,731.47353
	t30	17,684,004.68	3,827,810.34427
	d30	4,800,379.28	3,909,834.59393
	t50	17,486,431.16	5,252,774.33645
	d50	5,273,648.76	2,980,199.164
	t70	15,206,827.28	11,654,157.5278
	d70	5,606,321.16	3,163,776.25106
	t90	13,201,892.52	5,666,869.09688
	d90	5,070,082.76	2,121,620.23737
	t100	10,945,369.92	3,575,704.99771
	d100	5,038,982.68	4,761,610.87159
Órdenes Atendidas	t0	4,226.76	574.715457132
	d0	3,105.64	325.406909371
	t10	5,301.56	674.026710648
	d10	2,778.88	336.301243927
	t30	5,453.96	649.655644682
	d30	2,676.88	344.13167538
	t50	6,398.0	1,374.5722668
	d50	2,457.32	496.135878566
	t70	7,448.44	278.09277229
	d70	2,239.24	415.77580898
	t90	6,284.28	362.909598477
	d90	2,558.32	338.307887385
	t100	4,586.12	660.448794382
	d100	2,790.0	433.507400936

Ingresos (US$)	t0	82,787,498.92	9,772,299.87673
	d0	66,978,199.92	6,898,350.88653
	t10	104,414,075.44	12,378,910.4629
	d10	61,330,024.2	7,350,961.30892
	t30	108,718,063.44	12,611,823.1562
	d30	59,104,351.36	6,999,544.47046
	t50	122,604,489.4	25,776,490.2459
	d50	54,541,499.6	10,872,511.0173
	t70	142,639,166.96	6,966,361.06167
	d70	49,645,852.84	8,701,694.9434
	t90	122,519,881.44	9,527,047.13692
	d90	56,230,095.76	7,231,801.71835
	t100	88,533,515.84	11,350,416.3206
	d100	60,583,297.76	9,653,242.69952
Costo (%)†	t0	87.0807227908	8.73141813688
	d0	94.6008440294	14.5431185876
	t10	84.6795499433	9.65623307281
	d10	91.027963038	14.2744633562
	t30	82.4446968276	8.70175892256
	d30	89.7694276972	15.1416649353
	t50	84.4562936535	17.227298091
	d50	89.0134774365	22.8435428633
	t70	84.9543046154	2.57308372958
	d70	86.6659281666	20.1363776691
	t90	86.0517487291	5.54125180499
	d90	89.6936023287	14.3174926187
	t100	86.0651591627	10.1227742829
	d100	89.8772584743	19.7999752878

Costo de Almacenamiento (%)†	t0	1.27745533299	0.255938877105
	d0	0.6764619541	0.149884095873
	t10	0.997535624973	0.194200060353
	d10	0.688398619611	0.134211854719
	t30	1.13215633268	0.204951204161
	d30	0.678682619418	0.160817587025
	t50	1.17390412622	0.330719272179
	d50	0.632044521196	0.173667835355
	t70	1.13565893192	0.210352786314
	d70	0.574336311472	0.16023079373
	t90	1.03271911883	0.219118213287
	d90	0.679726816812	0.143563508202
	t100	0.939563183623	0.164615894065
	d100	0.709836433308	0.161562751949
Penalizaciones (%)†	t0	0.572696658536	1.01087929272
	d0	3.01828219094	3.61762080224
	t10	0.589097204958	0.681552952033
	d10	0.840887455577	1.17187337726
	t30	0.379154178208	0.647384659544
	d30	1.49316830266	2.60405440949
	t50	0.296433875936	0.608762775757
	d50	0.803413626713	1.57190034575
	t70	3.35904533244	4.45734977658
	d70	1.62658065841	2.62241687241
	t90	2.26382318314	2.27050729069
	d90	0.69463449194	0.947485180836
	t100	0.786112505978	1.56384992978
	d100	1.13070120863	2.50315253153

Intereses (%)†	t0	0.0867154110663	0.0959040533178
	d0	-0.137437076705	0.301451269599
	t10	0.195335043806	0.0768916011734
	d10	0.0664783693335	0.149422598182
	t30	0.221937875239	0.0862373559729
	d30	0.0631498005496	0.159391200298
	t50	0.18910380944	0.0855074626866
	d50	0.117992722004	0.143094184209
	t70	0.11005493326	0.175412901652
	d70	0.159472494621	0.169787161558
	t90	0.123597344545	0.107009172175
	d90	0.0846346771365	0.0964224160326
	t100	0.15380249921	0.0762289882977
	d100	0.0352414622337	0.187496378986
Utilización de fábrica (%)	t0	54.24	7.34438561079
	d0	39.44	4.03195568759
	t10	67.12	8.41783820229
	d10	36.16	4.29806157859
	t30	69.12	8.05357063668
	d30	34.56	4.28252262107
	t50	78.96	16.7642476718
	d50	32.44	6.40364479548
	t70	91.16	2.95352896267
	d70	29.52	5.22908532473
	t90	79.08	4.75149099406
	d90	33.36	4.31933636878
	t100	59.12	8.55531024179
	d100	35.72	5.71197572357

Órdenes entregadas a tiempo (%)	t0	98.04	3.14218607554
	d0	90.36	9.99533224394
	t10	97.0	3.57071421427
	d10	96.8	3.73050488093
	t30	98.24	3.35757849251
	d30	95.2	6.83739716559
	t50	98.16	4.2
	d50	97.28	4.88637561662
	t70	88.6	13.6930639376
	d70	93.96	7.64024432419
	t90	90.84	8.7924209787
	d90	96.92	3.65057073163
	t100	95.52	6.81371166595
	d100	96.8	6.09644705272
Órdenes entregadas tarde (%)	t0	1.52	2.48529005685
	d0	7.8	7.68114574787
	t10	2.84	3.31260219968
	d10	2.88	3.25729949498
	t30	1.52	2.97377425729
	d30	4.04	5.4273996229
	t50	1.6	3.5
	d50	2.6	4.71699056603
	t70	9.76	11.627123462
	d70	5.2	5.93717104352
	t90	7.76	7.38399169375
	d90	2.88	3.29545141066
	t100	4.16	5.8
	d100	2.44	4.36921045499

Órdenes nunca entregadas (%)	t0	0.4	0.763762615826
	d0	1.76	2.80297461042
	t10	0.24	0.435889894354
	d10	0.32	0.748331477355
	t30	0.32	0.556776436283
	d30	0.84	1.67531091642
	t50	0.28	0.737111479583
	d50	0.2	0.57735026919
	t70	1.6	2.17944947177
	d70	0.88	2.04776300712
	t90	1.4	1.52752523165
	d90	0.24	0.435889894354
	t100	0.36	1.22065556157
	d100	0.76	2.63438797446

† Este porcentaje es calculado en base a los ingresos totales obtenidos por el agente.

B.3. Experimento 3: Bloqueo de Clasificadores

	Agente	Promedio	Varianza
Resultado Final (US$)	t30-block	17,684,004.68	3,827,810.34427
	d30-block	4,800,379.28	3,909,834.59393
	t70-block	15,206,827.28	11,654,157.5278
	d70-block	5,606,321.16	3,163,776.25106
	t30-noblock	5,889,070.92	17,182,259.6609
	d30-noblock	4,052,750.32	5,924,192.30144
	t70-noblock	7,860,693.72	14,530,705.274
	d70-noblock	4,146,764.6	4,539,732.83111
Órdenes Atendidas	t30-block	5,453.96	649.655644682
	d30-block	2,676.88	344.13167538
	t70-block	7,448.44	278.09277229
	d70-block	2,239.24	415.77580898
	t30-noblock	7,147.0	723.809079339
	d30-noblock	2,332.4	382.053552093
	t70-noblock	7,136.2	726.637919921
	d70-noblock	2,267.68	434.691530475
Ingresos (US$)	t30-block	108,718,063.44	12,611,823.1562
	d30-block	59,104,351.36	6,999,544.47046
	t70-block	142,639,166.96	6,966,361.06167
	d70-block	49,645,852.84	8,701,694.9434
	t30-noblock	131,834,692.32	14,089,813.5645
	d30-noblock	51,100,180.6	6,809,018.33264
	t70-noblock	132,703,666.52	14,161,478.1238
	d70-noblock	49,856,088.92	9,240,726.83684

Costo (%)†	t30-block	82.4446968276	8.70175892256
	d30-block	89.7694276972	15.1416649353
	t70-block	84.9543046154	2.57308372958
	d70-block	86.6659281666	20.1363776691
	t30-noblock	87.9535306523	5.68592820102
	d30-noblock	88.4435901974	16.1478042481
	t70-noblock	86.6891241341	6.84286668371
	d70-noblock	88.5489218997	22.022575004
Costo de Almacenamiento (%)†	t30-block	1.13215633268	0.204951204161
	d30-block	0.678682619418	0.160817587025
	t70-block	1.13565893192	0.210352786314
	d70-block	0.574336311472	0.16023079373
	t30-noblock	1.21314516828	0.208923423072
	d30-noblock	0.651136250583	0.178615480173
	t70-noblock	1.26906883899	0.262300654277
	d70-noblock	0.669503298856	0.222970821108
Penalizaciones (%)†	t30-block	0.379154178208	0.647384659544
	d30-block	1.49316830266	2.60405440949
	t70-block	3.35904533244	4.45734977658
	d70-block	1.62658065841	2.62241687241
	t30-noblock	6.32113973443	7.09066351041
	d30-noblock	3.03387546149	5.96254580621
	t70-noblock	6.09159226115	5.78720637435
	d70-noblock	2.51446599033	4.2769298367
Intereses (%)†	t30-block	0.221937875239	0.0862373559729
	d30-block	0.0631498005496	0.159391200298
	t70-block	0.11005493326	0.175412901652
	d70-block	0.159472494621	0.169787161558
	t30-noblock	-0.0451729047582	0.365525601193
	d30-noblock	0.0595921181539	0.288258418319
	t70-noblock	-0.0267207688603	0.307474840437
	d70-noblock	0.050359906972	0.252661509793

Utilización de fábrica (%)	t30-block	69.12	8.05357063668
	d30-block	34.56	4.28252262107
	t70-block	91.16	2.95352896267
	d70-block	29.52	5.22908532473
	t30-noblock	86.8	8.00520663902
	d30-noblock	30.4	4.01040313851
	t70-noblock	86.4	8.21583836258
	d70-noblock	29.56	5.52328404726
Órdenes entregadas a tiempo (%)	t30-block	98.24	3.35757849251
	d30-block	95.2	6.83739716559
	t70-block	88.6	13.6930639376
	d70-block	93.96	7.64024432419
	t30-noblock	80.04	20.6104342506
	d30-noblock	91.04	12.0536301586
	t70-noblock	80.36	18.0321010053
	d70-noblock	92.4	9.09212113132
Órdenes entregadas tarde (%)	t30-block	1.52	2.97377425729
	d30-block	4.04	5.4273996229
	t70-block	9.76	11.627123462
	d70-block	5.2	5.93717104352
	t30-noblock	16.56	17.149052452
	d30-noblock	7.52	9.04673053281
	t70-noblock	16.52	14.9753130184
	d70-noblock	5.56	5.23672925912
Órdenes nunca entregadas (%)	t30-block	0.32	0.556776436283
	d30-block	0.84	1.67531091642
	t70-block	1.6	2.17944947177
	d70-block	0.88	2.04776300712
	t30-noblock	3.28	3.74744357307
	d30-noblock	1.4	3.31662479036
	t70-noblock	3.16	3.21039976742
	d70-noblock	1.96	4.99566478726

† Este porcentaje es calculado en base a los ingresos totales obtenidos por el agente.

B.4. Experimento 4: Tamaño de la población

	Agente	Promedio	Varianza
Resultado Final (US$)	t500	11,870,582.76	4,172,077.79331
	d500	4,430,928.68	3,974,578.64915
	t1000	17,684,004.68	3,827,810.34427
	d1000	4,800,379.28	3,909,834.59393
	t2000	16,460,850.96	5,566,664.47827
	d2000	3,936,415.0	4,416,866.45419
	t3000	16,143,555.88	6,722,174.58116
	d3000	3,846,441.44	5,473,290.64765
Órdenes Atendidas	t500	3,803.4	1,115.58299407
	d500	3,143.36	457.983795201
	t1000	5,453.96	649.655644682
	d1000	2,676.88	344.13167538
	t2000	6,547.64	856.744024393
	d2000	2,424.56	442.543319913
	t3000	6,884.56	693.277486533
	d3000	2,396.6	540.18152813
Ingresos (US$)	t500	76,489,307.48	22,563,951.1123
	d500	69,653,644.4	10,379,883.9405
	t1000	108,718,063.44	12,611,823.1562
	d1000	59,104,351.36	6,999,544.47046
	t2000	125,750,943.56	16,396,627.8571
	d2000	53,393,286.8	8,977,305.0614
	t3000	132,304,739.76	11,916,468.1455
	d3000	52,695,805.4	11,358,246.8295

Costo (%)†	t500	83.2502799906	24.0462368481
	d500	92.2047856551	18.3338064696
	t1000	82.4446968276	8.70175892256
	d1000	89.7694276972	15.1416649353
	t2000	84.5998370336	10.2118217541
	d2000	90.2985244205	21.2454873727
	t3000	84.9150941106	6.01139112655
	d3000	90.2753928114	27.2426228614
Costo de Almacenamiento (%)†	t500	1.0193847816	0.381302260934
	d500	0.699499881445	0.172213500531
	t1000	1.13215633268	0.204951204161
	d1000	0.678682619418	0.160817587025
	t2000	1.22820381007	0.243466849832
	d2000	0.699125343975	0.165391885512
	t3000	1.14294058001	0.24876395066
	d3000	0.667623157725	0.217928894061
Penalizaciones (%)†	t500	0.410010248925	0.654696889968
	d500	0.725057883691	1.47719968004
	t1000	0.379154178208	0.647384659544
	d1000	1.49316830266	2.60405440949
	t2000	1.23942952305	1.36271405354
	d2000	1.67434742002	3.11483227666
	t3000	1.88430011239	2.12185879009
	d3000	1.74077362142	2.67129082341
Intereses (%)†	t500	0.198945820028	0.0986295590127
	d500	-0.00928278779337	0.201462849051
	t1000	0.221937875239	0.0862373559729
	d1000	0.0631498005496	0.159391200298
	t2000	0.157512074576	0.0838065519082
	d2000	0.0444876901641	0.202440078689
	t3000	0.144131888507	0.104816915316
	d3000	-0.0168790664313	0.430024956597

Utilización de fábrica (%)	t500	49.0	14.5487685619
	d500	40.92	6.15033874406
	t1000	69.12	8.05357063668
	d1000	34.56	4.28252262107
	t2000	80.52	10.4885652022
	d2000	31.76	5.24626851518
	t3000	84.88	8.08970539225
	d3000	31.44	6.85006082698
Órdenes entregadas a tiempo (%)	t500	97.72	3.68012680941
	d500	97.32	4.74095630297
	t1000	98.24	3.35757849251
	d1000	95.2	6.83739716559
	t2000	94.6	5.0579969685
	d2000	94.6	7.49444238531
	t3000	92.96	7.64024432419
	d3000	94.24	6.97782200977
Órdenes entregadas tarde (%)	t500	2.12	3.3950945004
	d500	2.52	4.39810565282
	t1000	1.52	2.97377425729
	d1000	4.04	5.4273996229
	t2000	4.56	4.29224106189
	d2000	4.52	5.54616984954
	t3000	5.88	6.41170804076
	d3000	4.76	5.43353782846
Órdenes nunca entregadas (%)	t500	0.12	0.331662479036
	d500	0.24	0.522812904712
	t1000	0.32	0.556776436283
	d1000	0.84	1.67531091642
	t2000	0.76	0.925562891794
	d2000	0.96	2.47453699373
	t3000	0.96	1.20692446602
	d3000	0.96	1.96807858922

† Este porcentaje es calculado en base a los ingresos totales obtenidos por el agente.

B.5. Experimento 5: Comparación con otras soluciones al problema TAC SCM

	Agente	Promedio	Varianza
Resultado Final (US$)	tictactoe	5,546,332.120	2,629,511.523
	phant	48,693,391.400	4,902,925.984
	maxon	32,756,991.840	4,383,549.826
	tiancalli	4,161,840.200	4,606,918.299
Ordenes Atendidas	tictactoe	3,564.080	833.481
	phant	7,980.840	168.518
	maxon	7,219.720	359.239
	tiancalli	3,650.680	147.486
Ingresos (US$)	tictactoe	64,588,150.240	14,764,094.829
	phant	144,894,462.080	5,777,518.826
	maxon	128,836,919.240	9,078,930.533
	tiancalli	83,445,041.440	17,902,383.328
Costo (%)†	tictactoe	89.226	20.352
	phant	64.965	1.247
	maxon	73.634	4.026
	tiancalli	92.947	17.340
Costo de Almacenamiento (%)†	tictactoe	1.301	0.330
	phant	1.677	0.246
	maxon	1.122	0.164
	tiancalli	1.637	0.874
Penalizaciones (%)†	tictactoe	0.917	1.511
	phant	0.107	0.074
	maxon	0.065	0.071
	tiancalli	0.336	0.154
Intereses (%)†	tictactoe	0.031	0.097
	phant	0.356	0.106
	maxon	0.246	0.082
	tiancalli	-0.092	0.284
Utilizacion de fábrica (%)	tictactoe	44.000	9.941
	phant	96.640	0.490
	maxon	86.680	4.080
	tiancalli	68.160	2.838

Órdenes entregadas a tiempo (%)	tictactoe	96.640	4.949
	phant	100.000	0.000
	maxon	100.000	0.000
	tiancalli	99.040	0.351
Órdenes entregadas tarde (%)	tictactoe	2.840	4.110
	phant	0.000	0.000
	maxon	0.000	0.000
	tiancalli	0.840	0.374
Órdenes nunca entregadas (%)	tictactoe	0.440	1.044
	phant	0.000	0.000
	maxon	0.000	0.000
	tiancalli	0.080	0.277

† Este porcentaje es calculado en base a los ingresos totales obtenidos por el agente.