# Impacto en la tardanza promedio del modo de controlar sistemas de manufactura mediante agentes utilizando Reforzamiento del Aprendizaje y Lógica Difusa

por

**Claudia Carolina Soto Marín**

Una tesis sometida en completo cumplimiento de los requerimientos para el grado de

MAESTRO EN CIENCIAS
en
Ingeniería Industrial

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ
2008

Aprobado por:

| | |
|---|---|
| _____ | _____ |
| Sonia M. Bartolomei Suárez, Ph.D.<br>Presidente, Comité Graduado | Fecha |
| _____ | _____ |
| Agustín Rullán Toro Ph.D.<br>Miembro, Comité Graduado | Fecha |
| _____ | _____ |
| María de los Ángeles Irizarry, Ph.D<br>Miembro, Comité Graduado | Fecha |
| _____ | _____ |
| Agustín Rullán Toro, Ph.D.<br>Director, Departamento | Fecha |
| _____ | _____ |
| Loida Rivera Betancourt<br>Representante de Estudios Graduados | Fecha |

# RESUMEN

En esta investigación se compararon dos algoritmos para la selección de políticas de despacho en un sistema de tres máquinas. El primer algoritmo tiene un solo agente que controla la toma de decisiones de un sistema de 3 máquinas. El segundo algoritmo utiliza un agente para cada máquina, en este caso tres, los cuales controlan la selección de políticas de despacho sólo de la máquina a la que pertenecen. Los algoritmos seleccionan las piezas a procesar mediante la elección dinámica de políticas de despacho óptima utilizando Reforzamiento del Aprendizaje (RL) y Sistemas de Inferencia Difusa (FIS). Los agentes aprenden a tomar decisiones a lo largo del tiempo mediante prueba y error, recibiendo sólo recompensas o penalidades dependiendo del impacto observado en el objetivo del sistema, que es disminuir la medida de rendimiento tardanza promedio del sistema. En cada máquina se evalúa el estado actual y se recomienda una política de despacho a utilizar en la selección de piezas que esperan en fila. El algoritmo recibe el estado del sistema, convierte los valores de las variables de entrada a valores difusos, para luego evaluarlos mediante la base de reglas difusas. Finalmente decide la política de despacho a utilizar. De acuerdo al impacto obervado en la tardanza promedio del sistema, el agente recompensa o penaliza la acción tomada. Para evaluar si existían diferencias entre los algoritmos utilizados, se utilizó simulación. Se modelaron los sistemas en Arena™ [1] y realizaron corridas experimentales para cada sistema. Se realizó un análisis estadístico de los resultados y se concluyó que para el caso bajo estudio y el tiempo de simulación utilizado, controlar un sistema de manufactura mediante un agente presenta tardanzas promedios del sistema menores que al controlarlo mediante tres agentes.

[1] Rockwell Software Arena es un software de simulación de eventos discretos.

# ABSTRACT

In this investigation it is compared two algorithms in the dispatching rule selection in a system of three machines. The first algorithm has an agent that controls the decision-making of a system of three machines. The second algorithm use an agent for each machine, in this case three, which controls only the dispatching rule selection on the machinery it belong. The algorithms select the parts to be processed through the dynamic selection of the optimal dispatching policy by the use of Reinforcement Learning (RL) and Fuzzy Inference Systems (FIS). The agents learn to make decisions throughout the time by trial-error, receiving rewards or penalties depending on the impact observed in the objective of the system, which is to reduce the performance measure mean tardiness. On each machinery it is evaluated the actual state and it is recommended a dispatching rule to be used in the selection of the parts that waits in the queue. The algorithm receives the state of the system; it transforms the values of the input variables to fuzzy values, to evaluate them through the fuzzy base rules. Finally, the algorithm decides the dispatching rule to use. According to the observed impact in the mean tardiness, the agent rewards or penalizes the taken action. To evaluate if there exist differences between the used algorithms, it was used simulation. The system were modeled in Arena$^{TM}$ and there were performed experimental runs for each system and it was concluded that for the system in study and the simulation time used, to control a manufacturing system by one agent presents mean tardiness lower than to control them by three agents.

# AGRADECIMIENTOS

Quiero agradecer sinceramente a la Dra. Sonia Bartolomei Suárez, presidenta de mi comité graduado por su apoyo, paciencia, ayuda y sobretodo confianza en mí. Le doy gracias también por su gran contribución a la realización de este proyecto, tanto con ideas como con instalaciones físicas. Quiero agradecer también a los miembros del comité, la Dra. María de los Ángeles Irizarry y al Dr. Agustín Rullán por su aporte y apoyo.

Agradezco también al Departamento del Ingeniería Industrial por la oportunidad que me dieron para completar mi Maestría y por la oportunidad brindada de recibir una ayuda económica para financiar mis estudios.

Mi gratitud también a las secretarias del Departamento de Ingeniería Industrial Mayra Colón y Laura González por la paciencia que me tuvieron y por ayudarme con la documentación, la que muchas veces ellas tuvieron que hacer por mí.

Mi especial gratitud a mi esposo Eduardo y a toda mi familia en Chile quienes con su incondicional apoyo y amor me dieron las fuerzas para seguir adelante.

Finalmente, quiero agradecer a mis amigos y compañeros Joan Manuel Castro, Eyad Alí y Eleazar Gil quienes me ayudaron, me apoyaron y dedicaron tiempo valioso en la primera fase de la programación, cuando todo parecía imposible de realizar.

# DEDICATORIA

Esta investigación está dedicada a mi esposo Eduardo y a toda mi familia, mis padres Fernando y Juanita y a mis hermanas Daniela, Catalina y María Fernanda, a mi abuelita Violeta y a mi abuelito Manuel, quien me cuida desde el cielo.

# TABLA DE CONTENIDOS

# LISTA DE TABLAS



# LISTA DE FIGURAS

# CAPÍTULO 1: Introducción

## *1.1 Introducción*

La manufactura se ha convertido en una porción inmensa de la economía del mundo moderno. Según algunos economistas, la manufactura es un sector que produce una parte significativa del producto bruto nacional en una economía, mientras que el sector servicios tiende a ser contabilizado a base de la nómina que se paga.

Los bienes manufacturados, ya sean finales o intermedios, son una fuente significativa de demanda para bienes y servicios de otros sectores de la economía, desde energía y recursos naturales a construcción de nuevas fábricas a servicios provistos por contabilidad, ingeniería, programación y firmas de ayuda temporaria [28].

De los sectores generadores de producción nacional, la manufactura juega un rol único porque no está directamente limitada por recursos naturales y muchos productos son fácilmente transferibles nacional e internacionalmente. Además, es una fuente importante de empleos como también uno de los sectores más dinámicos de la economía norteamericana, siendo responsable del 60% del dinero gastado en el total de la investigación y desarrollo de USA [28].

Debido a la importancia de la manufactura a nivel mundial y a la constante competitividad y dinamismo de los mercados, las empresas de manufactura tienen que encontrar formas eficientes de producción y enfocarse continuamente en mejorar sus procesos. Para mantenerse en el mercado y tener ganancias sostenibles, es necesario sobresalir en la conversión de materias primas en productos con valor agregado que sean de las necesidades del consumidor. Este procedimiento de conversión consiste de un conjunto de actividades tales como diseño, planificación, producción, control de inventario, calidad, entre otros.

Uno de los procesos más importantes en los sistemas de manufactura es la planificación de la producción que define los planes de procesamiento detallados en el piso de producción. Un buen

plan de producción puede proveer beneficios tales como aumento en satisfacción del cliente, niveles bajos de inventario, menores imprevistos e incrementos en la utilización de los recursos. Por lo tanto, existe una gran necesidad de buenas estrategias de planificación de producción.

El control y planificación de la producción es un gran desafío para los encargados del piso de producción. Ésta actividad implica planificar capacidad, evitar que se produzcan cuellos de botella y generar altos niveles de productividad a nivel de piso de producción, entre otros.

En control y planificación de la producción, los Sistemas de Control de piso de producción (SFCS: Shop Floor Control Systems), juegan un rol muy importante. Éstos son una parte central de la manufactura automatizada, la cual ha sido ampliamente estudiada para desarrollar planificación eficiente de producción, secuenciación y control [4]. Estos sistemas se utilizan para facilitar el trabajo de la gerencia de producción, ya que permite realizar la toma de decisiones y ejecutar las funciones necesarias para completar las órdenes de modo eficiente. Un SFCS es un sistema de computadores y controladores (o también llamados agentes) utilizados para programar, despachar y rastrear el progreso de órdenes de trabajo a través del sistema basados en rutas ya definidas. Las funciones más típicas de un SFCS son: 1) seleccionar rutas de procesos para distintos tipos de piezas, 2) ocupar recursos (máquinas) para realizar esos procesos, 3) programar el trabajo que se realizará en cada tipo de pieza para cada máquina, 4) cargar las instrucciones de procesamiento de cada pieza, 5) monitorear el progreso de actividades, 6) detectar y corregir errores, y 7) preparar informes con respecto al estado del sistema [4].

Uno de los aspectos más importantes de un SFCS que afecta directamente el rendimiento de un sistema de manufactura es la toma de decisiones en el piso de producción. Este aspecto ha atraído mucho la atención de los investigadores en las últimas décadas: [30], [31], [38], [20], [4], [5], entre otros.

Existen diversos enfoques que resuelven el problema de la toma de decisiones, unos enfocados a técnicas matemáticas y otros a técnicas de despacho, las que generalmente son utilizadas para tomar decisiones fuera de línea. Las técnicas matemáticas, tales como los algoritmos, pueden encontrar soluciones óptimas pero en muchos casos es muy difícil utilizarlas debido a la

complejidad y al tamaño del problema. Por otro lado, las técnicas de despacho, tales como FIFO, SPT, EDD, entre otras, encuentran soluciones rápidamente y son fáciles de utilizar, pero es conocido que son subóptimas.

Los sistemas modernos de manufactura son dinámicos y tienen que enfrentar situaciones adversas tales como detenciones de maquinarias, tiempos variables de procesamiento de piezas, órdenes de trabajo urgentes, cambios de producto, entre otros. Por lo tanto, se ha optado por un enfoque en tiempo real, en vez de uno fuera de línea.

Debido a que los sistemas modernos de manufactura son dinámicos y tienen que enfrentar situaciones adversas tales como detenciones de maquinarias, tiempos variables de procesamiento de piezas, órdenes de trabajo urgentes, cambios de producto, entre otros, se ha optado por un enfoque en tiempo real en vez de uno fuera de línea.

Muchos enfoques y metodologías se han propuesto en los últimos años para resolver problemas de toma de decisiones en piso de producción. Sin embargo, sólo se ha realizado una investigación en el área de la manufactura relacionado con la programación de agentes que tomen decisiones en cuanto a la selección de las reglas de despacho más eficientes en algún punto de decisión. Esta investigación utilizó conjuntamente las técnicas de Reforzamiento del Aprendizaje (RL)[2] y Lógica Difusa (FL)[3] [18].

En [18], Ritesh estudió tres sistemas conformados por una celda de producción con diferente número de máquinas, número de reglas de despacho y número de variables de entrada para definir el estado del sistema. Para todos los casos se supuso que si se quería minimizar la tardanza promedio del sistema, se podía lograr a través de la minimización de la tardanza promedio en cada máquina de la celda. Para ello, se creó un agente para controlar el comportamiento de todo el sistema de manufactura.

[2] RL: por las siglas en inglés de “Reinforcement learning”

[3] FL: por las siglas en inglés de “Fuzzy Logic”

Este presente trabajo propone seguir la línea de investigación de [18] de manera tal que se pueda comparar el estudio realizado por estos autores contra uno que utiliza un agente para controlar el comportamiento de cada máquina individualmente mediante RL y FL. Esta investigación propone la siguiente hipótesis: "Para minimizar la tardanza promedio de una celda compuesta por X máquinas se debería minimizar la tardaza promedio en cada máquina en forma individual". Esta hipótesis se probará mediante la comparación de dos sistemas de similares características a través de la combinación de Reforzamiento del Aprendizaje y Lógica Difusa para construir un SFCS en tiempo real. La diferencia entre los sistemas será la manera de controlarlos. En uno de los sistemas se utilizará sólo un agente para controlar todo el sistema y en el otro se construirá un agente para controlar cada máquina, es decir, un total de tres agentes.

El rendimiento de cada decisión tomada por el agente será mejorado vía Reforzamiento del Aprendizaje, en donde el agente aprenderá mediante su propia experiencia a través de prueba y error. Debido a la gran cantidad de datos a manejar se utilizará Lógica Difusa para facilitar los cálculos y disminuir el tiempo de computadora. Además, se utilizará la técnica de simulación para modelar un sistema y tomar decisiones en tiempo real, de manera que se pueda obtener resultados y luego realizar una comparación. El controlador resolverá en tiempo real el problema de la selección de piezas, determinará secuencia de tareas de producción y generará mensajes para ejecutar esas tareas. El objetivo principal de este controlador es evaluar las condiciones actuales del sistema y seleccionar la mejor regla de despacho en cada punto de decisión, es decir, cuando una máquina esté vacía deberá seleccionar la siguiente pieza a procesar.

## *1.2. Definición del Problema*

Para cualquier tipo de empresa, disminuir la tardanza de productos es de vital importancia. Mientras menor sea la tardanza de un producto o servicio, menor es la posibilidad de pérdida de clientes debido a incumplimiento de plazos de entrega. En una empresa de manufactura es posible disminuir la tardanza de productos mediante planificación de producción eficiente. Una manera de disminuir la tardanza promedio, la cual se ha estudiado durante las últimas décadas, es

mediante la selección de la mejor regla de despacho conocida hasta el momento que logre mejorar la tardanza promedio del sistema.

El problema que se desea estudiar en esta investigación es el que se presenta cuando en un sistema de manufactura una máquina termina de procesar una pieza, queda desocupada y debe seleccionar la próxima pieza entre las piezas que esperan en fila. Se denominará a este momento punto de decisión. Para resolver este problema se construyó un caso de estudio muy similar al que utilizó Kanetkar [18], trabajo en el cual se basa esta investigación.

El caso de estudio es el siguiente: cuando la máquina queda desocupada, un agente debe evaluar el estado del sistema y de acuerdo a éste debe seleccionar la regla despacho que logre disminuir la tardanza promedio del sistema. Dentro de las reglas de despacho se pueden mencionar algunas, tales como: primero que llega primero que sale ó "first in first out (FIFO), último que llega primero que sale ó "last in last out" (LIFO), tiempo de procesamiento más corto ó "shortest procesing time" (SPT), costo aparente de tardanza ó "Apparent tardiness cost" (ATC), tiempo de entrega más temprana ó "earliest due date" (EDD), Costo de horas extras ó "Cost overtime" (COVERT), entre otras. Cuando la regla de despacho haya sido seleccionada, la máquina utiliza esta regla para seleccionar la siguiente pieza a procesar entre las piezas que esperan en fila.

La figura 1.1 presenta un ejemplo de cómo cada máquina selecciona una pieza de su fila para procesarla, en forma totalmente independiente de las demás máquinas. Suponer que existe un sistema con 3 máquinas y 3 piezas. Las máquina 1 y 3 han quedado desocupadas y deben seleccionar la próxima pieza para procesarla. El algoritmo propuesto seleccionó la regla de despacho que utilizará cada máquina para seleccionar la siguiente pieza a procesar de las que se encuentran en fila. En la primera máquina se seleccionó FIFO, la segunda máquina ya está procesando una pieza, pero ésta fue seleccionada con SPT y la tercera máquina seleccionó la regla de despacho EDD. Los tiempos de procesamiento y fechas de vencimiento para cada tipo de pieza se presentan en la Tabla 1.1.

**Tabla 1. 1.** Tiempo de procesamiento (min) y fecha de vencimiento (min) para sistema hipotético.

| Pieza | Tiempo de procesamiento | Fecha de entrega (o vencimiento) |
|---|---|---|
|  | 25 min. | 320 min. |
|  | 35 min. | 380 min. |
|  | 30 min. | 200 min. |

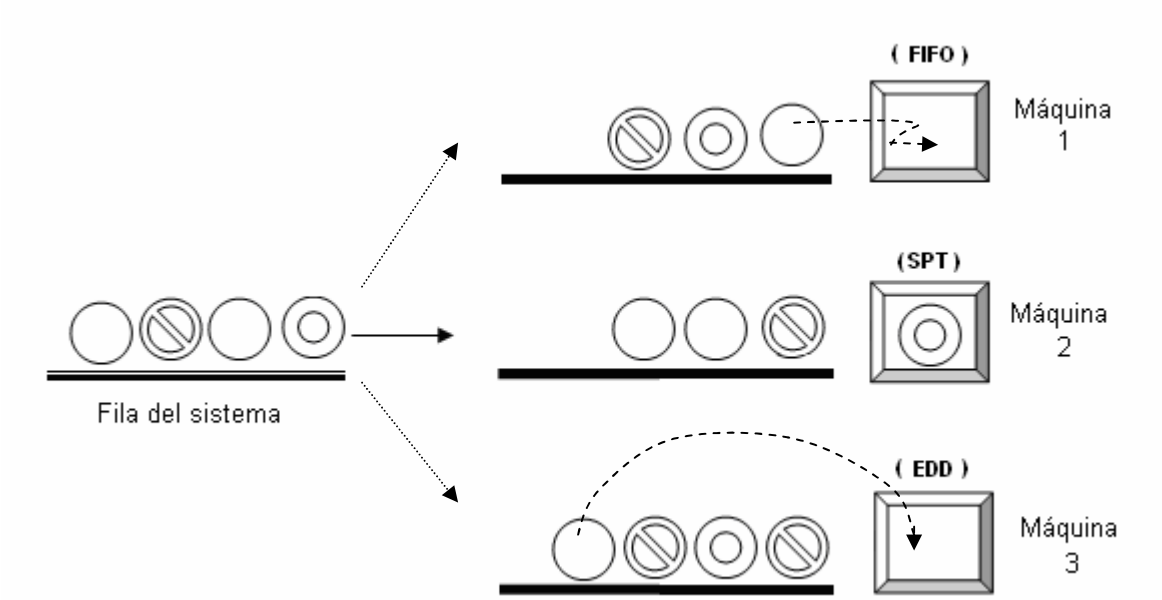



**Figura 1. 1.** Selección de piezas utilizando diferentes reglas de despacho en cada máquina

En este sistema, cada máquina realiza la selección de piezas en forma individual. En la máquina uno, se seleccionó la pieza tipo 3 que se encuentra en la primera posición en la fila, debido a que se está utilizando la regla de despacho FIFO (o el criterio), la cual fue previamente seleccionada con el algoritmo que se propone. La máquina 2 ya se encuentra procesando una pieza, pero fue seleccionada una pieza tipo 1 utilizando la regla de despacho es SPT, que es la que en ese momento poseía el tiempo de procesamiento más bajo entre las que estaban esperando en esa fila. Por último, la máquina 3 selecciona la pieza tipo 3 que se encuentra en la cuarta posición en la fila, debido a que el criterio aplicado es EDD, y de las piezas que hay en fila, la que tiene una fecha de vencimiento más temprana es la de tipo 3.

En esta investigación se utiliza un enfoque denominado algoritmo Q-learning Difuso. Dicho enfoque utiliza una combinación de las técnicas Reforzamiento del Aprendizaje (RL) y Lógica Difusa (FL) para encontrar la regla óptima de despacho que logre disminuir la tardanza promedio del sistema en un momento determinado.

## *1.3. Justificación*

Este estudio pretende demostrar que existe una diferencia entre crear un agente para controlar todo el sistema compuesto por tres máquinas y crear un agente individual para cada máquina del sistema con respecto a ciertas medidas de rendimiento. El objetivo del estudio es identificar la secuencia que minimiza la tardanza promedio para el sistema controlado por un sólo agente, y comparar la misma con la secuencia que minimiza la misma medida para el sistema controlado por un agente para cada máquina. Es decir, si una máquina selecciona una pieza que minimice la tardanza promedio en un instante de tiempo t, puede ocurrir que en la siguiente máquina de la ruta de procesamiento de esa pieza, aumente la tardanza promedio.

Determinar si existe diferencia entre los enfoques propuestos es de gran importancia debido que existen dos estrategias diferentes que hay que evaluar. Por un lado está la precisión de los resultados y suponer posiciones certeras, y por otro lado se encuentra la complejidad del sistema, tiempo de programación y tiempo de computadora (ejecución). Si la diferencia en los resultados entre un enfoque y otro es mínima, se puede optar por construir sólo un agente para controlar el sistema completo, logrando así un ahorro en tiempo de programación, tiempo de computadora y también complejidad. En caso contrario, si la diferencia en la medida de rendimiento de cada enfoque es muy diferente, y se logra un mejor resultado si se programa un agente para cada máquina, hay que determinar si ese rango de error que se quiere evitar es suficientemente grande para justificar un aumento en tiempo de programación de computadora y complejidad del sistema.

Lógicamente la construcción de mayor cantidad de controladores implicará mayores costos, pero, aunque los costos de construir un SFCS sean elevados, las ventajas que puede proveer el sistema

si es que se demuestre que mejora el rendimiento (el rendimiento del sistema aumenta a medida que el controlador aprende a lo largo del tiempo, no se requiere mayor supervisión, por lo cual se disminuyen los costos en esta área, existe menor cantidad de imprevistos, facilita el trabajo de la gerencia de producción, entre otros) justificarían la inversión y a la larga disminuirán los costos debido a malas decisiones que se puedan tomar si se considera como punto de partida los resultados de un sistema que está planteado en forma errónea.

## *1.4. Descripción de los sistemas*

Como se mencionó anteriormente, los sistemas que se modelaron se construyeron como un caso de estudio. El sistema controlado por sólo un agente es el que utilizó Kanetkar [18] y el que es controlado por tres agentes se construyó en esta investigación para poder realizar una comparación entre ellos.

### 1.4.1 Investigación realizada por Kanetkar [18]

Kanetkar estudió tres sistemas, los cuales se diferencian en el número de máquinas, el número de piezas, el número de variables de entrada y de despacho y el tipo de ambiente. El primer sistema era una celda de manufactura compuesta de tres máquinas y tres tipos de pieza. Se consideraron 3 reglas de decisión (FIFO, SPT, EDD) y los estados del sistema estaban definidos por dos variables de entrada (número de piezas y tiempo promedio de espera en fila). El segundo sistema era una celda de manufactura compuesta de diez máquinas que procesaban 3 tipos de pieza. Se consideraron 5 reglas de decisión (FIFO, SPT, EDD, SLACK/OPN y MST) y los estados del sistema estaban determinados por 3 variables de entrada: número de piezas en fila, tiempo promedio de espera en fila y tiempo promedio de procesamiento de piezas en fila. El tercer sistema era similar al segundo, pero se incorporaron disturbios para poder estudiar la ejecución del enfoque propuesto en ambientes cambiantes. Para ello, se consideraron tiempos de procesamiento variable.

Para cada sistema se realizó diseño de experimentos para estudiar el comportamiento de la medida de rendimiento tardanza promedio del sistema a diferentes niveles. Finalmente, se comparó los resultados obtenidos de cada uno de los sistemas utilizando el enfoque propuesto

(Reforzamiento del Aprendizaje en conjunto con Lógica Difusa) con los resultados entregados por el optimizador OptQuest de Arena™.

#### 1.4.1.1 Características de los sistemas

Cada máquina tenía capacidad de procesar una pieza a la vez. El sistema constaba de 4 filas distribuidas de la siguiente manera: cada máquina tenía una fila de entrada con capacidad de 10 piezas y había una fila a la entrada del sistema con capacidad infinita a la cual llegaban todas las piezas. A la salida de cada máquina, existía un amortiguador de salida para evitar que el sistema eliminara las piezas debido a la capacidad limitada de las filas de la máquina siguiente en la ruta de procesamiento.

Todas las piezas llegaban a la fila del sistema, la cual poseía capacidad infinita, en donde se asignaba a cada pieza su ruta de procesamiento dependiendo del tipo de pieza de la que se tratase. Las piezas se dirigían luego a la fila de entrada correspondiente a la primera máquina de su ruta de procesamiento en donde esperaban hasta que ellas eran seleccionadas para procesamiento y eran luego enviadas a la siguiente máquina de su ruta. La fila de cada máquina recibía piezas que venían tanto de la fila del sistema como de las filas locales de salida de las otras máquinas. Cuando una máquina terminaba de procesar una pieza quedaba desocupada y tenía que seleccionar la próxima pieza entre las piezas que estaban esperando en su fila para procesarla de acuerdo a una política de despacho, que en este caso era la regla de decisión que había sido previamente establecida. Este momento fue denominado punto de decisión. Las reglas de decisión cambiaban en cada punto de decisión y la selección de cada regla dependía de los estados percibidos del sistema.

La Figura 1.2 muestra el primer sistema estudiado por Kanetkar [18], compuesto por tres máquinas, tres piezas, tres reglas de despacho disponibles, una fila para cada máquina y la fila del sistema que recibe las piezas que llegan al sistema. En esta figura, la máquina 2 está procesando una pieza, la cual utilizó la regla de despacho SPT para seleccionarla. Esta regla no cambiaba hasta que se seleccionara una regla de despacho distinta a SPT al momento que se debía seleccionar otra pieza en esa máquina. Las máquina 1 y 3 están desocupadas y deben seleccionar la próxima pieza a procesar entre las piezas que esperan en su fila. Para hacerlo, el agente debe

seleccionar primero la regla de despacho a utilizar más adecuada, de acuerdo a la información que se le entrega con respecto al ambiente de cada máquina.

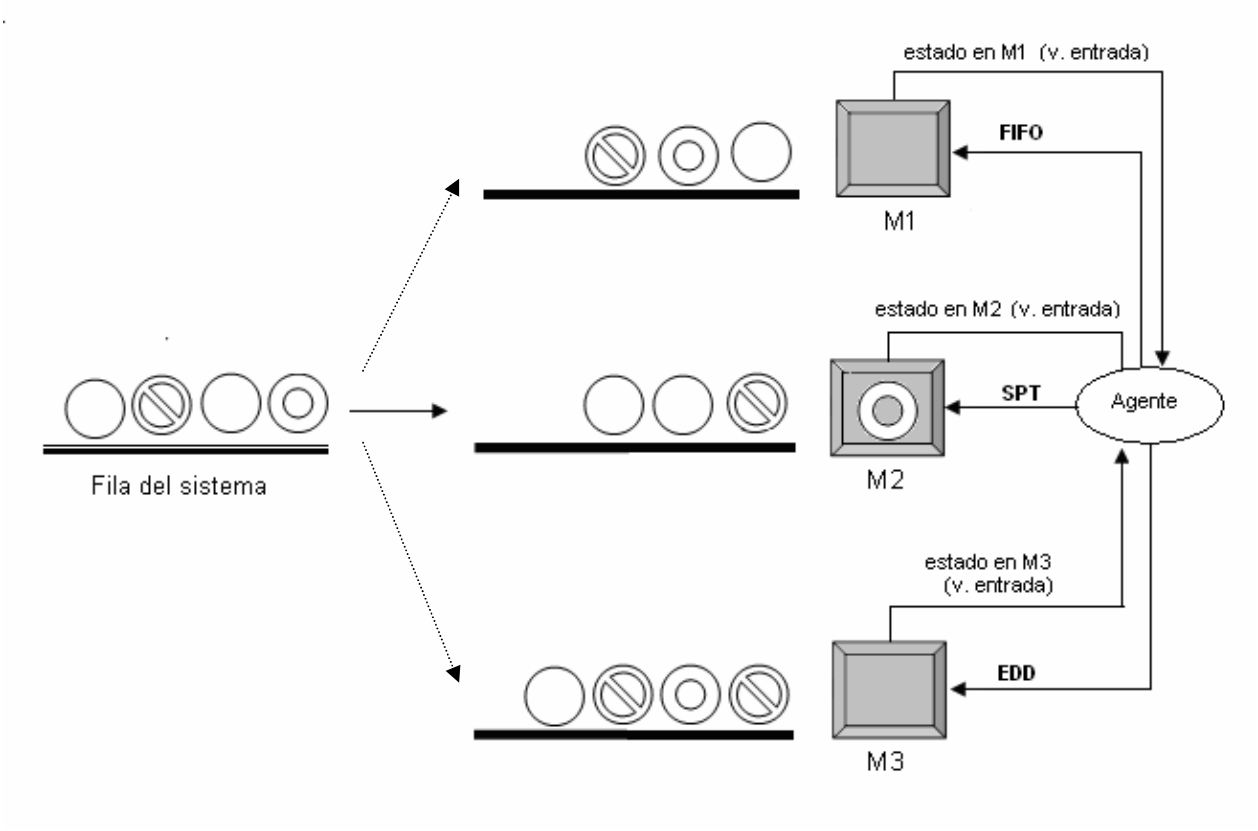



**Figura 1. 2.** Sistema estudiado por Kanetkar [18], utilizando un agente para controlar todo el sistema.

Cada una de las máquinas le entrega al agente los valores de las variables de entrada que determinan el estado de cada máquina. La regla de despacho es seleccionada mediante un algoritmo denominado Q-learning difuso propuesto Jouffe (1998), el cual combina las técnicas RL y Sistemas de Inferencia Difuso (FIS). Con la regla ya actualizada en la máquina, es posible seleccionar la próxima pieza para procesar.

El objetivo de cada sistema era minimizar la tardanza promedio total. Para ello, se asumió que la tardanza promedio del sistema completo podía ser reducida mediante la reducción de la tardanza promedio en cada máquina individualmente, utilizando sólo un agente que tomaría las decisiones para todas las máquinas en el sistema.

### 1.4.2 Descripción de los sistemas en la presente investigación

Como se mencionó anteriormente, esta investigación pretende seguir la línea de estudio de Kanetkar [18]. Se propone comparar dos sistemas de similares características. El primero de ellos tendrá sólo un agente que seleccionará las reglas de despacho para todas las máquinas. El segundo sistema tendrá tres agentes, es decir, un agente para máquina donde cada uno de ellos controlará la selección de la regla de decisión de la máquina a la que pertenece. El objetivo de ambos sistemas es minimizar la tardanza promedio del sistema, pero diferenciándose en la forma de controlar los sistemas y en la forma de calcular su recompensa. Uno de los sietmas se controlará mediante un agente y el otro se controlará mediante tres agentes. En el caso de asignación de recompensas, en el sistema controlado por un agente se asignará recompensa de acuerdo a la comparación entre la tardanza promedio que se ha calculado hasta el momento y una tardanza estimada. Ésta se obtendrá por comparar el tiempo en el sistema estimado con la fecha de entrega de la pieza, en donde el tiempo en el sistema se calculará como la suma del tiempo real que le ha tomado completar el procesamiento hasta la estación donde se encuentra más un estimado del tiempo que le tomará completar el procesamiento en las próximas máquinas de su ruta. En el caso de tres agentes se asignará de acuerdo a la tardanza real obtenida por estación.

Ambos sistemas de manufactura poseen las mismas características que el primer sistema que estudió Kanetkar [18], con 3 máquinas, 3 filas, 3 reglas de despacho disponibles (FIFO, SPT y EDD), pero utilizando 3 variables de entrada en vez de 2.

El primer sistema a estudiar opera de la misma manera que opera el sistema de Kanetkar [18], y está representado en la Figura 1.2. El segundo sistema está representado en la Figura 1.3, donde cada máquina es controlada por su propio agente. Cada agente recibe información de su ambiente, el cual comprende los valores de las variables de entrada seleccionadas de esa máquina en particular. Con esta información, el agente selecciona mediante las técnicas RL y FL la regla de despacho a utilizar, la cual se utiliza para seleccionar la próxima pieza a ser procesada en la máquina.

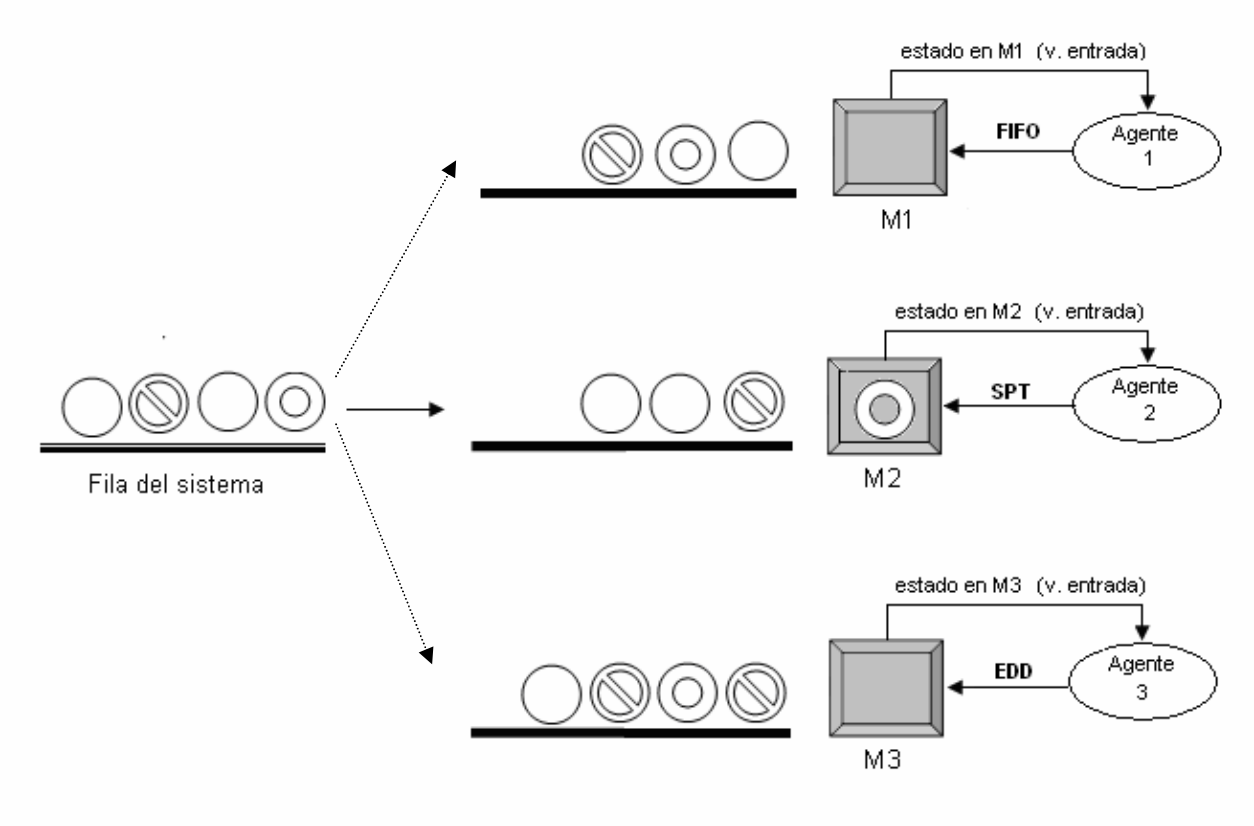



**Figura 1. 3**. Sistema de manufactura que posee un agente para cada máquina para controlar sus decisiones.

Cada máquina posee una fila con capacidad de 10 piezas para recibir a las piezas que esperarán hasta ser atendidas en esa máquina. Además, cada máquina posee una fila de salida para evitar que las piezas sean eliminadas debido a falta de espacio en la fila siguiente de la ruta de procesamiento. Todas las piezas llegan a la fila del sistema, la cual posee capacidad infinita. Al salir de ésta, las piezas se dirigen a la fila local correspondiente a la primera máquina de su ruta de procesamiento en donde esperan hasta que son seleccionadas. Después de ser procesadas, las piezas son enviadas a la siguiente máquina de su ruta si es que no se han completado y si las piezas ya han completado su ruta, abandonan el sistema. Cuando una máquina termina de procesar una pieza queda desocupada y entonces la máquina tiene que seleccionar la próxima pieza a procesar entre las piezas que están esperando en su fila. Este momento fue denominado punto de decisión.

En cada punto de decisión, el agente tiene que seleccionar la regla de despacho a utilizar para poder seleccionar la pieza que será procesada a continuación, utilizando como información el estado del sistema. El agente percibe el estado del sistema mediante las variables de entrada consideradas, que en este caso son tres: número de piezas en fila, tiempo promedio de espera en fila y tiempo promedio de procesamiento de piezas en fila. La selección de reglas de despacho se hará utilizando las técnicas de Reforzamiento del Aprendizaje en conjunto con Sistemas de Inferencia Difuso que utilizan Lógica Difusa. Se dispone de tres reglas de despacho para seleccionar: la primera pieza que llega es la que primero se atiende (FIFO), tiempo de procesamiento más corto (SPT), y tiempo de entrega más corto (EDD).

## *1.5 Objetivos*

### 1.5.1 Objetivo general

Evaluar si existe una diferencia significativa en la medida de desempeño "tardanza promedio" entre controlar el aprendizaje con un agente para el sistema completo mediante reforzamiento del aprendizaje y lógica difusa, y controlar el aprendizaje con un agente para cada máquina, cada uno manejado por reforzamiento del aprendizaje y lógica difusa.

### 1.5.2 Objetivos específicos

- Construir dos modelos de un sistema de manufactura utilizando un programa de simulación de eventos discretos, como es el caso de Arena$^{TM}$. El primero con sólo un agente que controle el aprendizaje para todo el sistema mediante Reforzamiento del Aprendizaje y Lógica Difusa. El segundo sistema con un agente para cada máquina, cada uno con RL y Lógica Difusa que controlen el aprendizaje de cada máquina en forma individual.
- Comparar el comportamiento de la medida de rendimiento "tardanza promedio" entre ambos sistemas.
- Analizar las diferencias que se detecten entre ambos sistemas en cuanto a la medida de rendimiento.

## *1.6 Resumen*

En este capítulo se presentó una introducción al tema de estudio en esta investigación. Se explicó que este trabajo está basado en una investigación previa realizada por [18]. Además se presentó una justificación de porqué se realiza esta investigación, los supuestos realizados y se detallaron los objetivos del estudio. Para un mayor conocimiento de lo que se discutirá a lo largo de la investigación, se explicó brevemente la investigación realizada por Kanetkar [18] y las características de los sistemas que se estudiaron. También se presenta una descripción de los sistemas que se estudiarán en esta investigación.

En el capítulo dos se discutirán algunas investigaciones realizadas previamente en el ámbito de la ingeniería e inteligencia artificial. Ëstas están relacionadas con las técnicas que se utilizarán en esta investigación: Reforzamiento del Aprendizaje y Lógica Difusa.

En el capítulo tres se discute la teoría de las técnicas de Reforzamiento del Aprendizaje y de los Sistemas de Inferencia Difus. También se discute un enfoque propuesto por Jouffe en 1998 que combina ambas técnicas y que denominó Reforzamiento de Aprendizaje Difuso.

En el capítulo cuatro, se discuten los componentes algoritmo de Reforzamiento de Aprendizaje Difuso. Se incluyen las variables de entrada del sistema, las variables de salida, las etiquetas lingüísticas, las funciones de membresía, las reglas de despacho seleccionadas, la base de reglas.

En el capítulo cinco se discute la implementación del algoritmo Q-learning Difuso propuesto. Se especifican los programas utilizados para el funcionamiento de los sistemas bajo estudio. Se describen también, los elementos utilizados para realizar las interfaces de comunicación entre los programas y el funcionamiento secuencial del sistema de manufactura utilizando el algoritmo propuesto.

El capítulo 6 presenta todo lo relacionado con la experimentación. Se definen los modelos de simulación, los supuestos realizados, se describe con detalle cada sistema, se definen los tiempos de corrida, los tiempos de procesamiento de piezas y los tiempos de entrega. En este capítulo se

presentan a demás los resultados obtenidos por cada uno de los modelos de simulación y se realiza un análisis de estos.

El capítulo 7 presenta las conclusiones de la investigación. Se incluye la comparación de los resultados obtenidos de cada sistema en estudio, se presentan los hallazgos encontrados, las conclusiones obtenidas de la investigación y finalmente se presentan recomendaciones para trabajos futuros.

# CAPITULO 2: Revisión de Literatura

## *2.1 Introducción*

La complejidad y la naturaleza de la toma de decisiones a corto plazo de los problemas de planificación de la producción llevan a un creciente interés de parte de los investigadores para encontrar métodos heurísticos que sean capaces de producir soluciones de calidad en un tiempo computacional razonable.

En el área de control y planificación de la producción, la selección de políticas de despacho es un problema crítico para la toma de decisiones, la cual afecta el rendimiento del piso de producción. Con la programación en tiempo real es posible responder rápidamente a situaciones imprevistas y adversas que puedan presentarse en el sistema. Muchas técnicas se han utilizado para responder a este problema, utilizándose principalmente algoritmo genético (GA), redes neuronales (NN), lógica difusa (FL), programación dinámica (PD) y reforzamiento del aprendizaje (RL) y muchas veces una combinación de las anteriores. Lo mismo ocurre para otros campos de estudio.

Este capítulo presenta y describe brevemente investigaciones realizadas en distintos campos que utilizan ya sea Reforzamiento del Aprendizaje, Lógica Difusa o una combinación de ellas con otros algoritmos heurísticos. Posteriormente se presentan investigaciones realizadas en el ámbito de manufactura que utilizan RL, FL o una combinación de ellas y otras que tienen como objetivo disminuir la tardanza promedio del sistema.

## *2.2 Investigaciones previas*

Las ideas de Reforzamiento del Aprendizaje han estado presentes en ingeniería por muchas décadas e.g. [24] y en inteligencia artificial desde mucho antes [25] y [33], entre otros. Sin embargo, es recientemente que el desarrollo y aplicación de los métodos de reforzamiento del

aprendizaje han ocupado un número significativo de investigadores en este campo. En el área de inteligencia artificial e ingeniería es posible citar las siguientes investigaciones:

El el 2005 Duan [8] aplicó RL y FL en un problema de navegación de un robot móvil inteligente en un ambiente desconocido y cambiante, el cual debe aprender a llegar a una meta haciendo cambios entre estrategias de evitar obstáculos o seguir una pared cuando éste queda atrapado. Se hicieron experimentos para evaluar el comportamiento de un robot en un laberinto. Un controlador con FL es aplicado al sistema de control del sistema reactivo del robot. El consecuente de las reglas difusas fueron refinadas mediante Q (λ)-learning. Los experimentos de simulación de la navegación del robot fueron implantados en un ambiente desconocido, el cual comprendió obstáculos cóncavos. Para acelerar el proceso de aprendizaje, se utilizó conocimiento experto para la toma de decisiones del algoritmo. Los resultados del experimento indicaron eficiencia y efectividad del enfoque propuesto. El algoritmo aprendido presentaba robustez y adaptabilidad, pudiendo aplicarse a diferentes ambientes.

En el año 2002 Gu [12] presentó un enfoque de aprendizaje al desarrollo de controladores difusos basados en recompensas tardías desde el mundo real. Las recompensas tardías son aportadas a las reglas difusas individuales utilizando el reforzamiento de Q-learning. Se desarrolló un algoritmo genético para encontrar un "trade-off" entre explotación y exploración. El propósito del estudio fue evaluar el comportamiento reactivo de robots jugando fútbol. Este trabajo propuso una solución al problema de hacer una correspondencia desde información sensorial con ruido a acciones imperfectas para alcanzar una meta. Los experimentos realizados evaluaron el enfoque propuesto, pero quedó como un trabajo futuro demostrar los efectos de este enfoque en la tasa de aprendizaje, mediante la comparación con algoritmo genético puro o algoritmo Q-learning puro. Además, el tiempo de aprendizaje estaba limitado en robots reales.

Kretchmar en [21] estudió la situación en la cual múltiples agentes intentan acelerar el proceso de aprendizaje compartiendo información utilizando la técnica de reforzamiento del aprendizaje. El sistema intenta simular un juego de niños consistente en que una persona piensa en un número y otras personas tienen que adivinarlo, recibiendo como refuerzo sólo 3 respuestas: está correcto, en cuyo caso se gana el juego y termina; muy alto; muy bajo. La investigación simula el juego

mediante agentes, los cuales no interactúan entre ellos sino que cada uno está aprendiendo la misma tarea aislado de los otros agentes. Se asume que todos los agentes son homogéneos, es decir, idénticos e interactúan con el mismo entorno para aprender la misma tarea. Los agentes comparten eventualmente los valores Q de Q-learning para llegar a una política óptima utilizando pocos episodios y menos tiempo de corrida.

Esta investigación mostró cómo sistemas multiagentes pueden lograr aprendizaje intercambiando información, pero los resultados indican que "acelerar" el aprendizaje eficientemente es difícil de obtener. Si un agente presenta dos episodios con el mismo estado consecutivos, el segundo episodio utiliza todos los valores Q que fueron actualizados durante el primer episodio. En el caso que dos agentes paralelos presenten el mismo estado, los valores Q de ese estado no han sido actualizados aún, lo cual genera dos efectos: el valor Q no se actualiza en la misma forma y segundo, el espacio de estado es buscado menos eficientemente porque hay duplicidad. La conclusión de la investigación fue que implementaciones masivas en paralelo de procesadores son probablemente poco realistas. Además el aceleramiento de aprendizaje no incrementa en la misma medida que el aumento en la cantidad de agentes en paralelo debido al sobrelapamiento en la búsqueda de valores Q.

En el 2002 Gu junto a otros investigadores utilizaron RL en conjunto con FL para construir un controlador para programa movimientos de robots cuyo objetivo era alcanzar una pelota y anotar un gol. Ellos utilizaron el algoritmo AHC (Adaptative Heuristic Critic) de RL para el algoritmo de reforzamiento y también GA para predecir el reforzamiento interno. Los resultados experimentales mostraron que el robot podía aprender mediante el propuesto esquema de RL. El aprendizaje continuo en robots probó que la simulación de aprendizaje es útil y ahorra tiempo debido a la reducción es búsqueda de estados. El tema de la convergencia del algoritmo de reforzamiento del aprendizaje quedó abierto para futuras investigaciones [13].

Crites [7] aplicó la versión de un paso de Q-learning a un problema de despachos de 4 elevadores, en un sistema de 10 pisos, donde cada elevador hizo su elección independientemente del resto. Se utilizaron dos métodos de RL. En el primer caso, a cada elevador se le dio su propia función acción-valor y su propia red neuronal. En el segundo caso, había sólo una red y una sola

función valor, con la experiencia de todos los elevadores contribuyendo al aprendizaje en la red. En este estudio se aplicó redes neuronales para representar la función acción-valor y las redes fueron entrenadas por simulación.

Mientras estos algoritmos consumen un considerable tiempo de computadora, es despreciable comparado con lo requeriría cualquier método convencional de programación dinámica, debido a la gran cantidad de estados que presentaba el problema, que se estimaban cerca de $10^{22}$. Para todas las medidas de rendimiento, RL sobrepasó a los otros métodos heurísticos comúnmente utilizados en la industria. Las técnicas de RL y FL también han sido utilizadas en las últimas décadas en ambientes de manufactura, las cuales intentan resolver diversos problemas de planificación de la producción, tales como selección de piezas utilizando diferentes combinaciones de reglas de despacho, heurísticos o combinación de técnicas para disminuir tardanza promedio de productos u otras medidas de rendimiento, entre otros. Planificación de la producción basada en reglas, tal como selección de reglas de toma de decisiones (despacho, selección de piezas o selección de máquinas) es un problema crítico que afecta el rendimiento del piso de producción. En el caso específico de la tardanza de productos con respecto a sus fechas de entrega, es una medida de rendimiento muy importante, ya que puede significar la pérdida de clientes. A pesar de la importancia de disminuir la tardanza de productos, los problemas en la literatura son más bien limitados. A continuación se presentan algunas investigaciones realizadas en el área de manufactura que utilizan RL, FL y otras que intentan disminuir la tardanza promedio mediante diferentes técnicas.

En [34] Wang junto a otros investigadores aplicaron RL para resolver un problema de selección de reglas de despacho para una simple máquina. Ellos utilizaron el algoritmo Q-Learning para lograr que una máquina aprendiera a seleccionar reglas de despacho para tres diferentes casos en los cuales las mejores reglas habían sido previamente definidas. El objetivo del sistema era disminuir la tardanza promedio del sistema. Se utilizó una tabla de políticas de selección de reglas (estado-acción), y también una tabla de recompensas (tardanza-recompensa) para la construcción de la función de recompensas. En este estudio se utilizó el método tabular porque es mucho más simple de implantar, pero los resultados experimentales revelaron que es necesario considerar una mayor cantidad de estados, con lo cual el método tabular ya no sería efectivo. Los

resultados mostraron que más estados en la tabla de política y más rangos en la tabla de recompensas para determinar el monto de reforzamiento mejorarían el aprendizaje.

En [3] Cambolat junto a otros investigadores intentaron resolver la limitación que existe en secuenciación al seleccionar una regla de despacho, ya que al hacerlo sólo se está satisfaciendo un criterio y no varios a la vez. La investigación propone resolver un problema multi-criterio mediante la utilización de lógica difusa para combinar las reglas de despacho SPT, Razón crítica (CR) y la tasa de carga de la próxima máquina (NML) del trabajo a ser procesado a continuación para satisfacer no sólo un único objetivo, sino todos los objetivos. Lógica difusa calcula el valor de la prioridad considerando SPT, CR y NML. Se denominó a esta regla de prioridad regla de prioridad difusa (FPR). El trabajo con el valor más alto de prioridad es asignado a la máquina para ser procesado. Los objetivos del sistema son minimizar el tiempo de flujo, el tiempo para completar un trabajo, la tardanza, trabajo en proceso y el tiempo de espera promedio y, por otro lado, maximizar el flujo. Se comparó este enfoque mediante simulación con el resultado obtenido de las reglas de despacho SPT, EDD, entre otras, en forma individual. Los resultados indicaron significantes mejoras en el tiempo de flujo promedio, tardanza promedio, trabajo en proceso y flujo simultáneamente. Obviamente el objetivo de cada sistema de producción es diferente, pero este enfoque se puede adaptar a cualquier ambiente cambiando los elementos del algoritmo como son los pesos de las reglas, funciones de membresía, entre otros.

En el 2004 Hong junto a otros investigadores [15] presentaron un enfoque que es adecuado para producción “just in time” (JIT) en problemas de planificación con multiobjetivos y cambios dinámicos en el piso de producción. Presentan un algoritmo de piso de producción distribuido, donde las piezas y las máquinas controlan su propio comportamiento para minimizar los costos de la suma del cuadrado de la desviación (MSD) de los plazos de entrega y los costos de ajuste. Minimizando los MSD aseguran producción JIT simultáneamente minimizando inventario (mediante minimización de prontitud) y minimizando tardanza. Con este algoritmo cada parte del controlador utiliza sólo su información local para minimizar la desviación de la fecha de entrega de la pieza. El problema de control de maquinarias es modelado como procesos de decisión semi Markov y resuelto utilizando Q-learning mediante el método tabular. Los algoritmos de

aprendizaje fueron evaluados utilizando simulación para dos tipos de sistemas de manufactura: programación de familias y tamaños dinámicos de lotes.

Los resultados mostraron que el algoritmo propuesto alcanza una significante mejora en rendimiento por sobre la usual reglas de despacho en problemas complejos de control de piso de producción en tiempo real para producción JIT. La investigación muestra la factibilidad de incorporar agentes de RL en tiempo real sistemas de manufactura altamente complejos. Queda abierta para investigación futura utilizar función aproximación en vez del método tabular para la utilización de multiagentes RL en máquinas en paralelo y grandes familias, también como otros objetivos tales como utilización de las máquinas y balanceo de línea en tiempo real.

En el 2000 Armentano junto a otros investigadores [1] presentaron un enfoque de programación de trabajos utilizando búsqueda Tabu en máquinas idénticas en paralelo con el objetivo de minimizar la tardanza promedio con respecto a su fecha de entrega. La búsqueda local fue realizada en la vecindad definida por dos tipos de movimientos: transferencia de trabajos de una máquina a otra y movimientos de intercambio de trabajos entre dos máquinas. A continuación se analizó la incorporación de dos estrategias de diversificación con el objetivo de explorar regiones no visitadas de la solución de espacios. La primera estrategia utilizaba memoria a largo plazo para almacenar la frecuencia de movimientos ejecutados a través de la búsqueda y la segunda estrategia hace uso de los movimientos influenciales.

En esta investigación se examinaron 900 problemas mostrando un rango de mejora de 2.98% a 10.69% mayor que los mejores heurísticos reportados en la literatura. La diversificación involucró 150 trabajos y 10 máquinas, pero debido al alto costo computacional se calculó un límite inferior para 540 problemas. Los resultados mostraron que en cerca del 62% de tales problemas, las soluciones que proveía la búsqueda Tabu estaban dentro del 1% del límite inferior. El uso de búsqueda Tabu para minimizar la tardanza promedio en máquinas en paralelo resultó ser efectiva. La búsqueda Tabu sin diversificación rindió buenos resultados para muchos problemas. Además, la inclusión de estrategias de diversificación tiende a producir mejores resultados para el mismo criterio de detenimiento. Los resultados mostraron que reglas involucrando la prohibición de movimientos reversibles no trabajaron bien. Como extensión de

esta investigación se propone establecer estrategias para evaluar una lista más reducida de candidatos en la vecindad con el fin de obtener soluciones de alta calidad en menos tiempo computacional.

## *2.3 Resumen*

Este capítulo describió brevemente investigaciones relacionadas con Reforzamiento del Aprendizaje, Lógica Difusa, combinación de ambas técnicas y combinación de estas técnicas con otros algoritmos para la programación de agentes en el área de inteligencia artificial, ingeniería y manufactura.

En el próximo capítulo se discutirán en detalle la técnica de Reforzamiento del Aprendizaje (RL) y sus distintos algoritmos, se discutirá los Sistemas de Inferencia Difusos (FIS), los cuales utilizan la teoría de la lógica difusa, y por último, se discutirá los sistemas de Reforzamiento del Aprendizaje Difusos, que son una propuesta de Jouffe en 1998 [16] que combina ambas técnicas para resolver problemas de aprendizaje de maquinarias con múltiples estados.

# CAPITULO 3: Marco Teórico

## *3.1 Introducción*

Este capítulo trata con detalle las técnicas utilizadas en esta investigación para la programación y control de agentes computarizados. Entre estas se incluye la técnica de Reforzamiento del Aprendizaje (RL), los Sistemas de Inferencia Difuso basados en la teoría de Lógica Difusa (FL), y los Sistemas de Reforzamiento del Aprendizaje Difusos.

En los recientes años, la tecnología de agentes ha ofrecido una solución para controlar sistemas de manufactura requiriendo flexibilidad, confiabilidad, adaptabilidad y reconfigurabilidad [29]. Un agente puede utilizarse para representar componentes físicos, tales como piezas, máquinas, herramientas e incluso seres humanos. Cada agente está a cargo de recolectar información, almacenar datos y tomar decisiones para la correspondiente área del piso de producción. Un agente puede ser visto como un módulo computacional capaz de actuar autónomamente para alcanzar una meta. En la aplicación de sistemas multiagentes, un tema importante para mejorar la capacidad de un agente autónomo es cómo aumentar la inteligencia del agente. El Aprendizaje de Maquinarias es uno de los mecanismos que provee la habilidad a un agente para incrementar su inteligencia durante la operación.

El término *Aprendizaje de máquinarias* se refiere a la capacidad de un sistema de adquirir e integrar conocimiento autónomamente. Esta capacidad de aprender de la experiencia, de la observación analítica y otros, resulta en un sistema que puede mejorarse continuamente por sí mismo y por lo tanto incrementar su eficiencia y efectividad [2].

En la segunda sección de este capítulo se describe la técnica Reforzamiento del Aprendizaje (RL), un tipo de aprendizaje de maquinarias. Ésta técnica es muy prometedora debido a que es una forma de programar agentes mediante recompensas y penalizaciones sin necesidad de especificarle al sistema cómo se debe realizar la tarea [17].

RL ha sido estudiado en cibernética, estadística, psicología y ciencia computacional desde hace algunas décadas a causa de su generalidad y utilidad para resolver problemas que anteriormente era imposible resolver. Ha sido sólo entre los últimos 5 a 10 años que ha atraído un creciente interés en el área de aprendizaje de máquinas y de inteligencia artificial.

En la tercera sección se explica con detalle los Sistemas de Inferencia Difuso, los cuales se basan en la teoría de la Lógica Difusa (FL). La Lógica Difusa es de gran utilidad debido a puede reducir la complejidad de un sistema con muchos estados a un simple sistema con valores difusos para hacerlo más manejable. Ésta técnica se ha desarrollado rápidamente debido a sus potencialidades de aplicación, principalmente en el área de diseño de controladores electrónicos. Sus aplicaciones van desde el control de complejos procesos industriales, hasta el diseño de dispositivos artificiales de deducción automática, pasando por la construcción de artefactos electrónicos de uso doméstico y de entretenimiento, así como también de sistemas de diagnóstico. Desde el punto de vista tecnológico, la Lógica Difusa se encuadra en el área de la llamada Inteligencia Artificial y ha dado origen a sistemas expertos de tipo difuso y a sistemas de control automático.

Por último, la tercera sección habla de los Sistemas de Reforzamiento del Aprendizaje Difusos, propuesto por Jouffe [1998], los cuales son una combinación de las técnicas de Reforzamiento del Aprendizaje y Lógica Difusa. Éste sistema es el que se utilizará para la construcción y control de los sistemas bajo investigación.

## *3.2 Reforzamiento de Aprendizaje*

El Reforzamiento del Aprendizaje (RL) es un tipo de aprendizaje de máquinas en el cual un agente interactúa con su entorno y aprende a comportarse tomando acciones para mejorar la recompensa recibida por el entorno. En RL un agente interactúa con su entorno en una secuencia de pasos discretos de tiempo, la cual puede observarse en la Figura 3.1. A cada paso de tiempo, el agente recibe alguna representación del estado actual $s_t$, de su entorno y en base a ese estado realiza una acción $a_t$ que cambia el estado actual del entorno a $s_{t+1}$. Un paso de tiempo más tarde,

el agente recibe una recompensa o una penalización en forma de un número escalar por la acción recién realizada indicando la deseabilidad de ese nuevo estado [2]. Finalmente, el agente se encuentra nuevamente enfrentando un nuevo estado.

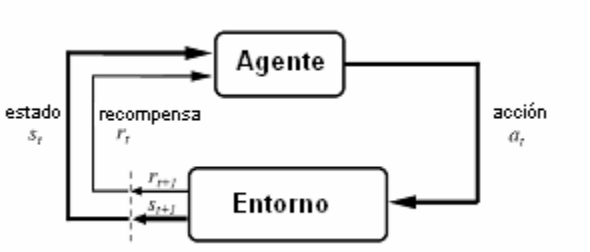



**Figura 3. 1.** Interacción entre el agente y su entorno en RL [32]

Algunos investigadores clasifican RL como un tipo de aprendizaje supervisado, ya que la recompensa es un tipo de supervisión, y hay algunos que la clasifican como un tipo de aprendizaje no supervisado debido a que la "recompensa" no se considera un ejemplo provisto por un supervisor externo. Hay algo de verdad en cada uno de esos puntos de vista pero reforzamiento del aprendizaje es realmente diferente de ambos enfoques [2].

Un aspecto clave que distingue RL del aprendizaje supervisado y no supervisado es la presencia del conflicto entre exploración y explotación, el cual está ausente en los otros dos tipos de aprendizaje a menos que el sistema de aprendizaje esté también influenciando los ejemplos de entrenamiento que éste ve [32]. El agente tiene que balancear entre dos objetivos: explotar lo que ya se ha aprendido en el pasado para continuar obteniendo altas recompensas ó comportarse de una nueva manera (explorar) para aprender más.

Las características más importantes del Reforzamiento del aprendizaje son:

- *Prueba y error:* no se le indica al agente qué acción tomar, sino que debe descubrir qué acciones producen la mayor recompensa.
- *Recompensa tardía:* Las acciones pueden afectar no sólo la recompensa inmediata sino que también las siguientes situaciones. Un sistema de reforzamiento del aprendizaje

frecuentemente debe sacrificar recompensas inmediatas para obtener mejores recompensas más tarde o en un largo plazo. [2]

Esta técnica está basada en el principio de los Procesos de Decisión de Markov (MDPs. Estos tienen tienen la propiedad que el entorno satisface la propiedad markoviana que implica lo siguiente: el estado del entorno en cualquier lapso de tiempo $t>0$ provee la misma información acerca de lo que pasará a continuación como lo hace la completa historia del proceso hasta el tiempo t.

El problema de RL está constituido por cuatro componentes fundamentales, incluyendo: entorno, política, función de reforzamiento o función de recompensa y función de valor. Cada una de ellas es discutida en detalle a continuación.

### 3.2.1 Componentes de Reforzamiento del Aprendizaje

- **Entorno:** Está **c**ompuesto por un conjunto discreto de estados $S_t$. El agente interactúa con un entorno dinámico y aprende a hacer una correspondencia desde situaciones a acciones mediante prueba y error. Si el sistema RL puede observar perfectamente toda la información que puede influir en la acción a realizar, entonces el sistema RL elige acciones basadas en verdaderos estados del entorno.
- **Política:** Define la forma en que un agente aprende a comportarse en un momento determinado [32]. Es decir, una política es la regla que el agente utiliza para seleccionar acciones y está formada por una correspondencia desde estados del entorno a acciones a ser tomadas cuando se está en esos estados. En términos matemáticos, es una función que para cada estado asigna una probabilidad a cada posible acción, es decir, es la probabilidad que el agente ejecute una acción *a* cuando está en el estado *s*.

- **Función de reforzamiento (o función de recompensa):** Es la función exacta de futuras recompensas que el agente busca maximizar [14]. Este traza un mapa desde cada estado percibido (par estado-acción) del entorno a un simple número, una recompensa o penalización, indicando la deseabilidad intrínseca de ese estado, es decir, cuales son los eventos favorables para el agente.

Entonces, el agente RL ajusta su política basada en su experiencia acumulada para tratar de mejorar el monto de recompensa que recibe a lo largo del tiempo.

- **Función de valor:** Son funciones escalares de estados, o pares de estado-acción, que dicen cuán bueno es para el agente estar en ese estado o tomar una acción en ese estado [2]. Mientras una función de recompensa indica qué es bueno en un sentido inmediato, una función de valor especifica qué tan bueno es a largo plazo. En otras palabras, indica la deseabilidad de estados a largo plazo después de tomar en cuenta los estados que son probables a seguir y las recompensas disponibles en esos estados [14] y [2].

El *valor del estado* es definido como la suma de las recompensas recibidas que un agente puede esperar acumular en un futuro cuando se comienza en un estado y se sigue alguna política determinada hasta un estado terminal. [14] y [2].

### 3.2.2. Técnicas para estimar funciones de valor

Muchos algoritmos han sido desarrollados para estimar la función de valor. Algunos de ellos son discutidos en esta sección, incluyendo el método Montecarlo, Programación Dinámica y Diferencias Temporales.

- **Método Montecarlo (MC).**

Son métodos para resolver problemas de reforzamiento del aprendizaje basados en retornos de muestra promedio. Para asegurar que están disponibles los retornos bien definidos, los métodos MC están definidos únicamente por tareas en episodios. Estos métodos requieren únicamente experiencia, muestra de secuencia de estados, acciones y recompensas desde interacción en línea o simulada con el entorno. Aprendizaje desde la experiencia en línea no requiere de conocimiento previo de la dinámica del entorno. Aunque se requiere de un modelo, el modelo necesita únicamente generar transiciones de la muestra, no la distribución completa de probabilidad de todas las posibles transiciones que son requeridas por los métodos de programación dinámica [32].

- **Programación Dinámica (PD)**

En PD cada copia de respaldo actualiza el valor de un estado basado en el valor de todos los posibles estados sucesores y sus probabilidades de ocurrir más bien que en una muestra del siguiente estado. Esto es, ellos actualizan los estimados en base a otros estimados. Esto es lo que se llama *bootstrapping o autoarranque*.

Aunque las ideas PD pueden ser aplicadas a problemas con estados continuos y espacios de acción, soluciones exactas son posibles únicamente en casos especiales. PD puede no ser práctico para muchos problemas grandes debido a que el tiempo que le toma a un PD encontrar una política óptima es polinomial en el número de estados y acciones (peor caso). Un método PD está garantizado a encontrar una política óptima en tiempo polinomial.

- **Diferencias Temporales (TD)**

Es una combinación de las ideas de MC y DP. Como MC, los métodos TD pueden aprender directamente a través de la experiencia sin un modelo dinámico del entorno. Como DP, los métodos TD actualizan los estimados basados en parte en otros estimados aprendidos, sin esperar por la respuesta final [32].

### 3.2.3 Comparación entre métodos TD, MC y DP

Cada tipo de método tiene sus fortalezas y debilidades. Las más importantes son:

- Los métodos PD están bien desarrollados matemáticamente, pero requieren un modelo completo y certero del entorno. En cambio, los métodos TD no requieren modelo del entorno, de su recompensa y las distribuciones de probabilidad del siguiente estado. Los métodos MC tampoco requieren un modelo del entorno.
- Los métodos MC son conceptualmente simples, pero no son adecuados para cómputos incrementales paso a paso. En cambio los métodos TD son implantados naturalmente en línea y son completamente incrementales, pero son más complejos de analizar [32].
- Mientras los métodos MC deben esperar hasta el final de un episodio para determinar el incremento a $V(s_t)$, donde únicamente $R_t$ es conocido, los métodos TD necesitan esperar únicamente hasta el próximo lapso de tiempo [32].

- Se ha encontrado que los métodos TD convergen más rápido que la constante $\alpha$ de los métodos MC en tareas estocásticas [32].
- Algunos métodos MC deben ignorar o descontar episodios en los cuales las acciones experimentales son tomadas, ya que pueden reducir enormemente el aprendizaje. Los métodos TD son mucho menos susceptibles a esos problemas, porque ellos aprenden de cada transición independientemente de las acciones subsecuentes tomadas [32].

Dadas las ventajas presentadas por el método de Diferencias Temporales (TD), en esta investigación se utilizará el método TD para estimar la función de valor. A continuación los métodos TD se explican con más detalle.

### 3.2.4 Método de diferencias Temporales (TD)

El aprendizaje mediante diferencias temporales es un intento de resolver el problema de aprendizaje de máquinas de reforzamiento retardado. En muchos dominios de AI, es muy difícil para un programa aprender de una serie de eventos a causa de la separación temporal entre eventos y sus resultados.

Los métodos TD fueron introducidos por Sutton como una clase de métodos para aprender a realizar predicciones en problemas de múltiples pasos. En cada paso la diferencia entre dos sucesivas predicciones es usada como error de entrenamiento. Cada predicción es modificada tal que la haga más cercana a la próxima. De hecho, TD es una clase de métodos referido como TD ($\lambda$), donde $0<\lambda<1$ es llamado un factor de ponderación que es un parámetro heurístico que controla la asignación de crédito temporal. El factor $\lambda > 0$ permite incorporar diferencias de predicciones de más paso de tiempo, con la esperanza de acelerar el aprendizaje [6].

El estimado actual (más nuevo) es asumido a ser más correcto que el estimado producido la última vez que el sistema observó un patrón y la diferencia entre los dos es usado para ajustar el peso en la red [10]. Estos métodos estiman el retorno esperado mediante la próxima recompensa más el valor del próximo estado. La actualización a la función de valor toma la diferencia de estimados sucesivos de la función de valor [22].

#### *3.2.4.1 Tipos de algoritmos TD*

Los algoritmos TD más importantes son: Adaptive Heuristic Critic (AHC), Tabular TD(0), TD (λ), Q-learning y Sarsa. Éstos se discuten a continuación.

Los algoritmos AHC son una versión adaptada de la política de iteración en la cual los cálculos de la función de valor no son implementados para resolver un conjunto de ecuaciones lineales, sino que más bien calculados por un algoritmo llamado TD(0) en vez de actuar para maximizar la recompensa inmediata. Esto actuará para maximizar el valor heurístico, *V*, que es calculado por el crítico. El crítico utiliza la señal de reforzamiento externo real para aprender a trazar un mapa de estados a sus valores esperados descontados.

La regla TD(0) es realmente una instancia de clase de algoritmos más general llamados TD(λ), con λ=0, donde λ es el parámetro que determina la característica temporal de las copias de respaldo con rango desde 0 (trazos de no elegibilidad ) a 1 (resultando en un simple método MonteCarlo). TD(0) mira únicamente un paso más adelante cuando ajusta estimados de valores. Aunque éste eventualmente llegará al resultado correcto, puede tomarse mucho tiempo [17]. La regla TD(λ) es similar a la regla TD(0), pero ésta es aplicada a cada estado de acuerdo a su elegibilidad, en vez de sólo al estado inmediatamente previo, *s*. La elegibilidad de un estado *s* es el grado al cual éste ha sido visitado en el pasado reciente. Cuando el reforzamiento es recibido, éste es usado para actualizar todos los estados que han sido recientemente visitados, de acuerdo a su elegibilidad. Es computacionalmente más costoso ejecutar el TD(λ) general, aunque frecuentemente converge considerablemente más rápido para grandes valores de λ [17].

Sarsa y Q-learning son un poco similares, pero mientras Q-learning converge a Q* independientemente del comportamiento del agente (tan pronto como las condiciones de convergencia son satisfechas), Sarsa converge a una función acción-valor que es la óptima dado el modo de exploración del agente. Como los algoritmos TD para valores de estados, ambos, Q-learning y Sarsa pueden ser extendidos para incluir elegibilidad.

#### *3.2.4.2 Ventajas de Q-Learning*

Q-learning es insensible a la exploración, lo que significa que los valores Q convergerán al óptimo independiente de cómo se comporten los agentes mientras los datos están siendo recolectados. Esto significa que aunque el tema de la exploración-explotación debe ser indicado en Q-learning, los detalles de la estrategia de exploración no afectará la convergencia del algoritmo de aprendizaje. Por esas razones, Q-learning es el algoritmo más popular y ha demostrado ser el más efectivo para aprendizaje retardado [17]. El método de Q-learning es más fácil de implantar que los otros algoritmos TD. La función acción-valor aprendida, $Q_t$, directamente aproxima a $Q^*$, la función acción-valor, independiente de la política seguida. Esto simplifica dramáticamente el análisis del algoritmo y permite pruebas tempranas de convergencia. La política sigue teniendo efecto ya que determina qué par estado-acción será visitado y actualizado, sin embargo, el requisito para la correcta convergencia es que todos los pares continúen siendo actualizados. Bajo estos supuestos y una variante de las condiciones usuales de aproximación estocástica en la secuencia de parámetros paso-tamaño, se ha demostrado que $Q_t$ converge a $Q^*$ con probabilidad igual a 1. [32].

### 3.2.5 Q-learning

Uno de los acontecimientos más importantes en RL fue el desarrollo de un algoritmo TD de control con política fuera de línea conocido como Q-learning. [36]. Su forma más simple, el algoritmo *Q-learning* de un paso está definido por:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma \max_{a'} Q(s',a') - Q(s,a)], \qquad (3.\ 1)$$

en donde,

***s*** : estado *S,* donde s es el estado actual y s' es el siguiente estado

***a*** : acción *A(s),* donde a es la acción actual y a' es la siguiente acción

**Q (s,a)** : función acción-valor. Valor de tomar una acción a en el estado *s* bajo una política determinada. (Retorno esperado comenzando desde el estado *s*, tomando la acción *a* y siguiendo la política determinada)

**α** : parámetro positivo del tamaño del paso

**γ Є [0, 1)** : es el factor de descuento. Determina el presente valor de recompensas futuras.

NOTA: Si $\gamma = 0$, el agente está únicamente interesado con maximizar las recompensas inmediatas.

La meta de este algoritmo es aprender a hacer una correspondencia entre valores estado-acción, Q(s,a), los cuales representan las recompensas esperadas a largo plazo para cada par estado-acción. Se ha probado que los valores Q aprendidos con este algoritmo convergen a valores óptimos de estado-acción, Q* [36]. Los valores estado-acción representan la política óptima que el agente intenta aprender.

### 3.2.6 Estrategia Explotación-exploración

El conflicto explotación-exploración está presente en todo problema de Reforzamiento del Aprendizaje. Explotación significa utilizar soluciones conocidas, es decir, explotar lo que ya se ha aprendido en el pasado (seleccionando acción egoísta ó "greedy", que es la óptima) para continuar obteniendo altas recompensas, y exploración significa buscar nuevas soluciones (seleccionar otra acción que no sea la acción egoísta) para aprender más. Si siempre se selecciona la acción egoísta (explotación absoluta), puede ser imposible conocer el valor de las otras acciones, que podrían resultar, a la larga, mejores que la acción egoísta. Si nunca se toma la acción egoísta, la estimación de la función de valor de acción no serviría para nada, ya que no se estaría utilizando su conocimiento para mejorar la política. Por lo anterior, se hace necesario un balance adecuado entre exploración y explotación, de manera que el agente explote su conocimiento para obtener buenas recompensas a la vez que explora el ambiente para realizar una mejor selección de acciones en el futuro.

Para establecer en qué momento realizar explotación o exploración, existen muchos métodos, entre los cuales se encuentran softmax y ε-greedy. En esta investigación se utiliza el método ε-

greedy, el cual involucra seleccionar con probabilidad 1-ε la acción con el más alto valor (explotación) y con probabilidad ε seleccionar la acción en forma aleatoria.

En las etapas iniciales de aprendizaje la estrategia está orientada más hacia la exploración que la explotación, ya que no se conoce los resultados de aplicar un cierto tipo de acción. A medida que el aprendizaje progresa la estrategia cambia más a explotación que explotación, disminuyendo el valor de ε a medida que el tiempo transcurre.

## *3.3. Sistemas de Inferencia Difusos (FIS)*

Un Sistema de Inferencia Difuso ó "Fuzzy Inference System (FIS) es una de muchas aplicaciones de la Lógica Difusa. Para entender cómo es la lógica detrás de estos sistemas, es importante entender primero, qué es la Lógica Difusa y en qué se diferencia de la lógica convencional.

La lógica difusa es una teoría matemática frecuentemente aplicada a problemas de inteligencia artificial y que permite representar problemas con un alto grado de imprecisión e información subjetiva. La Lógica Difusa es una forma más general de la lógica convencional. Mientras la lógica convencional estudia la membresía de un elemento a un conjunto, donde las posibilidades son únicamente verdaderas o falsas, la Lógica Difusa estudia la membresía parcial de un elemento en un conjunto, donde las posibilidades varían entre 0 y 1. Los sistemas de inferencia difusos constan de 3 etapas: fusificación, reglas de inferencia y defusificación. Estas etapas están representadas en la Figura 3.2.

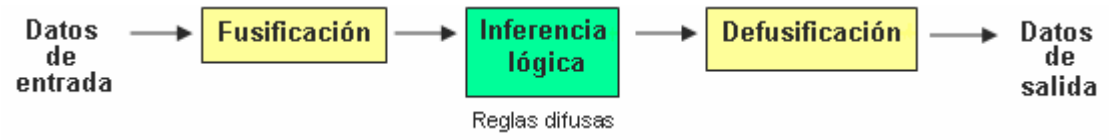



**Figura 3. 2.** Etapas de un sistema de inferencia difuso.

### 3.3.1 Fusificación.

Fusificación es la transformación de los datos desde valores del mundo real (normalmente valores numéricos) a valores difusos a través de grados de membresía. Los *grados de membresía* indican el grado de pertenencia de un elemento a un conjunto difuso. Cada elemento presenta un grado de membresía a un conjunto difuso que puede tomar valores entre 0 y 1, donde 0 significa que el elemento no pertenece al conjunto y 1 significa pertenencia total al conjunto. Este grado de membresía es definido por una *función de membresía* asociada al conjunto difuso. Los grados de membresía se determinan mediante dos componentes: las etiquetas lingüísticas que definen el nombre de los conjuntos difusos, tales como adverbios o adjetivos que son característicos del conjunto difuso, y mediante la función de membresía representadas por funciones matemáticas.

La *función característica o de membresía* es una curva que define cómo a cada punto en el espacio de la variable de entrada (numérica) le corresponde un grado de membresía entre 0 y 1. La forma de la función de membresía utilizada será dependiente del criterio aplicado en la resolución de cada problema. Normalmente son usados dos criterios: el conocimiento humano y la colección de datos para diseñar una función. Las funciones de membresía más frecuentemente utilizadas a causa de su simplicidad matemática y manejabilidad son: triangular, trapezoidal, gaussiana, gamma, etc. En la etapa de fusificación, un grado de membresía a cada conjunto difuso que ha sido considerado es asignado a cada variable de entrada, a través de las funciones características asociadas a esos conjuntos difusos. La Figura 3.3 muestra un ejemplo de fusificación para la variable velocidad. Se definieron 3 etiquetas lingüísticas para esta variable: lento, medio, rápido. Si un valor de entrada de la variable velocidad es de 55 (m/s), los grados de membresía de este valor son los siguientes: 0.25 para la etiqueta lento y 0.75 para la etiqueta medio y 0 para la etiqueta rápido. Este ejemplo posee una mezcla de funciones de membresía con forma trapezoidal para las etiquetas lento y rápido y triangular para la etiqueta medio.

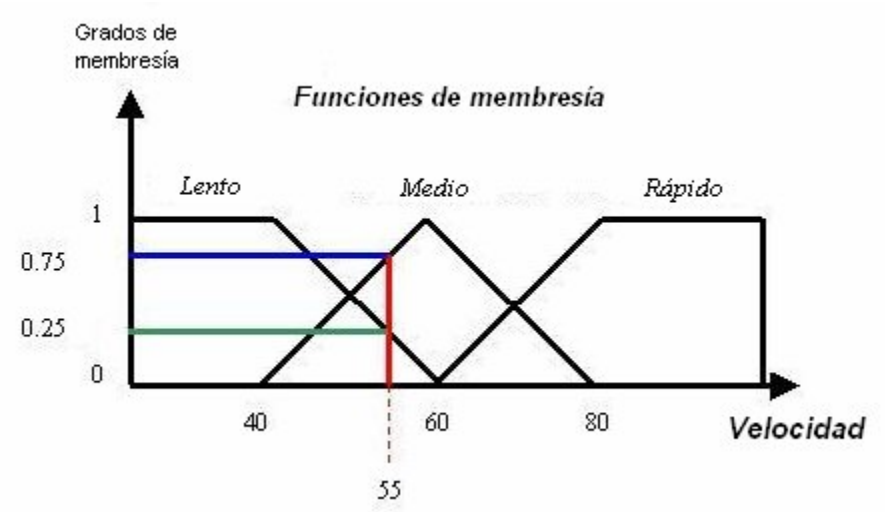



**Figura 3. 3.** Grados de membresía de la variable velocidad para la entrada 55 (m/sg). [ 11]

### 3.3.2 Inferencia Lógica

Esta etapa representa la base de reglas que definen el sistema. Aquí es donde se relacionan los conjuntos de entrada y las entradas difusas a través de algunas reglas que son del tipo IF/ELSE – THEN. Las entradas en esta etapa son conjuntos difusos (grados de membresía) y las salidas son también conjuntos difusos asociados a las variables de entrada. En el proceso de inferencia lógica se llevan a cabo 3 pasos: aplicación de los operadores difusos (AND, OR ó NOT) en el antecedente de cada regla, la implicación desde el antecedente al consecuente y la agregación de los consecuentes a través de todas las reglas activas.

Una simple regla difusa IF-THEN es de la forma: Si X es A entonces Y es B, donde A y B son valores lingüísticos definidos por conjuntos difusos y "Si X es A" es el antecedente y "entonces Y es B" es el consecuente. Un ejemplo de esto es el caso de la propina que se deja en un restaurante:

Si el servicio es **bueno** y la comida es **excelente** entonces la propina es **generosa.**

Los conceptos “bueno” y “excelente” son etiquetas lingüísticas para las variables de entrada “servicio” y “comida” y el concepto “generoso” es etiqueta lingüística de la variable de salida “propina”. El antecedente correspondería a la frase completa antes del entonces: “Si el servicio es bueno y la comida es excelente” y el consecuente al a la oración después del entonces: “la propina es generosa”. Si el antecedente de una determinada regla tiene más de una parte, entonces se utilizan los operadores difusos para obtener un número que represente el resultado del antecedente para esa regla. Este número es entonces aplicado a la función de salidas.

Zadeh [39] propuso operaciones elementales entre conjuntos difusos. Esto implica definir cómo calcular las funciones de membresía. Sean $\mu_{(x)}$ y $\mu_{(y)}$ las funciones de pertenencia correspondientes a los conjuntos difusos X e Y, los operadores difusos son [23]:

*Intersección (AND):* $\mu_{(X \text{ and } Y)} = \text{mínimo de } (\mu_{(x)}, \mu_{(y)})$ (3. 2)

*Unión (OR) :* $\mu_{(X \text{ or } Y)} = \text{máximo de } (\mu_{(x)}, \mu_{(y)})$ (3. 3)

*Complemento (NOT)* : $\mu_{(\text{Complemento } X)} = 1 - \mu_{(x)}$ (3. 4)

En el ejemplo de la propina se utiliza el operador lógico AND.

Luego de aplicar los operadores difusos se realiza la implicación, la cual debe ser implantada para cada regla. En este proceso se utilizan varios métodos, conocidos como T-Normas o T-Conormas. Los más comunes son los tipos T-Normas y los más comúnmente utilizados son: *Mínimo*, el cual “trunca” el conjunto difuso de salida, y el *Producto*, el cual “escalona” el conjunto difuso de salida. En muchos casos, las variables de entrada tienen una membresía en más de una etiqueta lingüística, por lo tanto, más de una regla puede tener la misma combinación de variables de entrada en su estamento “IF”. Esto significa que más de una regla puede activarse. Este problema se soluciona mediante el cálculo del verdadero valor de cada regla activada utilizando la T-Norma operador producto. La ecuación 3.5 representa el cálculo del verdadero valor α.

$$\alpha_{R_I}(S) = T\ [\ \mu^i_{L_1}(S_1),\ \ \mu^i_{L_2}(S_2),.....,\mu^i_{L_{NI}}(S_{N_I})], \qquad (3.5)$$

donde T representa la T-Norma, que en nuestro caso será el operador mínimo, $\mu_{L_1}, \mu_{L_2}, ....., \mu_{L_{NI}}$ son las funciones de membresía de las etiquetas lingüísticas $L_1, L_2, \ldots L_{NI}$ y, $S_1, S_2, ....., S_{NI}$ son los valores de las variables de entrada.

Como las decisiones están basadas en evaluar todas las reglas del sistema de inferencia, las reglas deben ser combinadas de alguna manera para tomar una decisión final. Para ello se realiza la agregación de reglas, que es el proceso por el cual los conjuntos difusos que representan las salidas de cada regla son combinados en un simple conjunto difuso. La agregación ocurre únicamente una vez para cada variable de salida. Al proceso de agregación entra la lista de funciones de salida truncadas o escalonadas retornadas del proceso de implicación y la salida de este proceso es un conjunto difuso para cada variable de salida. Se utilizan 3 métodos para realizar la agregación: el método del máximo (OR), el método del Probor (Or probabislítico) y el método de la suma (simplemente la suma del conjunto de salida de cada regla), entre los cuales el método del máximo es el más comúnmente utilizado.

### 3.3.3 Defusificación.

Es la etapa que transforma los conjuntos difusos obtenidos desde la etapa de inferencia en valores concretos de entrada (los resultados) a través de métodos matemáticos de defusificación. En otras palabras, es la acción inversa de la fusificación. Las entradas de la defusificación son valores concretos y las entradas son grados de membresía a los conjuntos difusos. Para obtener un resultado escalar a partir del conjunto difuso de salida que resulta de la agregación de todas las reglas se utilizan métodos matemáticos, donde los más populares son el método del máximo, el método de la altura y el método del centroide o método del centro de gravedad. Este último es el más utilizado en aplicaciones de lógica difusa en la ingeniería porque es posible encontrar una única solución, aunque es difícil de calcular.

## *3.4 Enfoque Reforzamiento del Aprendizaje Difuso*

El sistema con reforzamiento del aprendizaje difuso propuesto por Jouffe [1998], utiliza las técnicas de Reforzamiento del Aprendizaje y Sistemas de Inferencia Difuso (FIS) en conjunto, para sacar provecho de las ventajas de ambas técnicas. Para ello, se incorpora Reforzamiento del Aprendizaje en las etapas de inferencia lógica y defusificación de un sistema de inferencia difuso. En la etapa de Inferencia Lógica, cada regla en la base de reglas no sólo entregará a la siguiente etapa un conjunto de posibles variables de salida, sino que también entregará los pesos asociados a cada una de esas variables en forma de vector $w_d$. Este vector representa la probabilidad de seleccionar las diferentes acciones disponibles en la base de reglas para aproximar la política óptima [16].

Por otro lado, la etapa de defusificación de un FIS cambia completamente, ya que en un FIS común las variables de entrada se defusifican mediante métodos matemáticos para obtener un resultado de tipo escalar. En el caso del enfoque propuesto por Jouffe, las variables de salida del sistema serán dos, una es la mejor regla de decisión entregada por el Actor y otra es la función de evaluación de estado $Q_t$ (Función Q-learning) entregada por el Crítico. Para un mejor entendimiento, la Figura 3.4 presenta el sistema de inferencia difuso modificado para incorporar Reforzamiento del Aprendizaje.

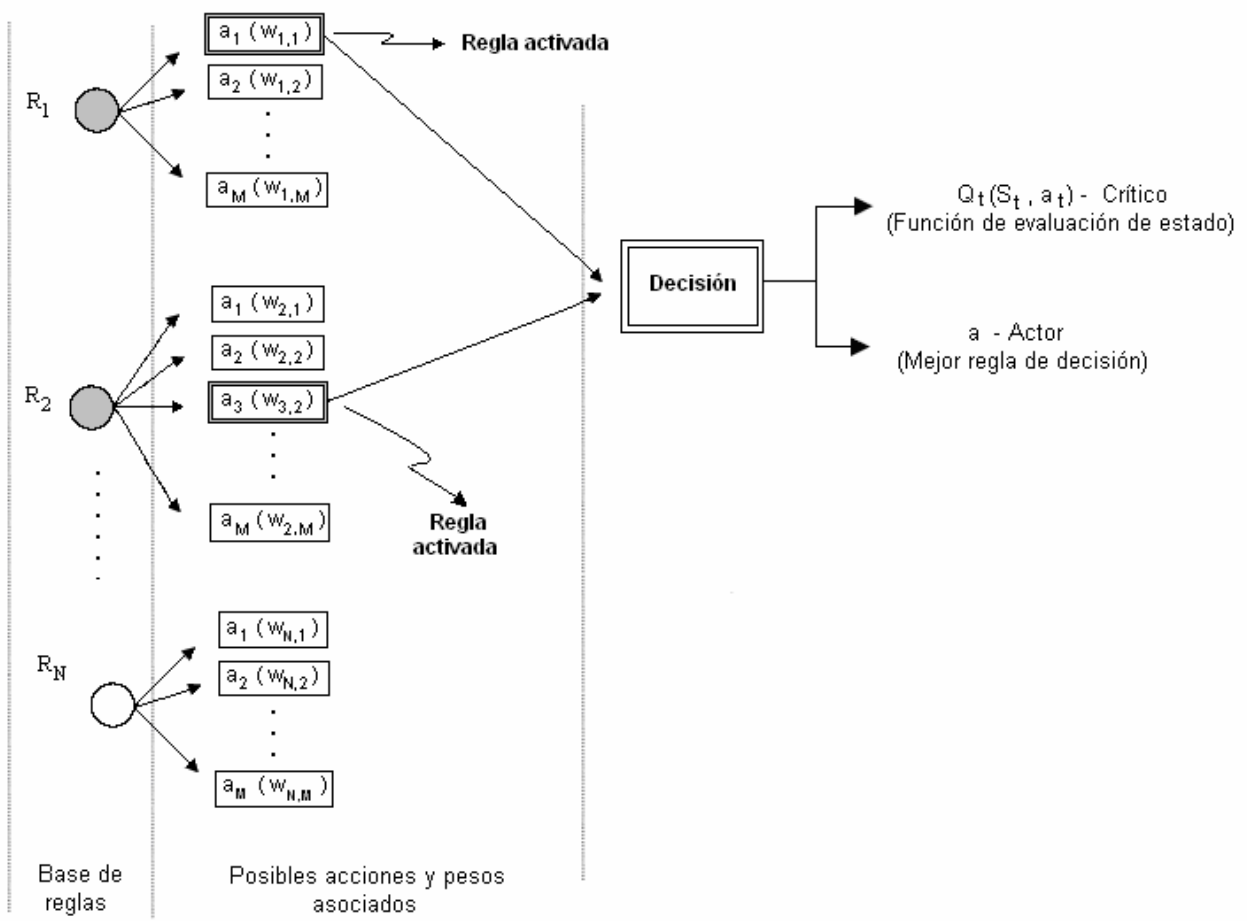



**Figura 3. 4.** FIS modificado para incorporar Reforzamiento del Aprendizaje.

El aprendizaje se llevará a cabo mediante la actualización del vector de pesos $w_d$ asociado a cada acción a ser realizada en una regla en particular. Esta actualización es un proceso que se realiza para lograr que el actor seleccione la mejor regla de decisión (o de despacho) de un conjunto de alternativas disponibles, las cuales dirigirán a un mejor valor crítico $Q_t$.

Como se mencionó anteriormente, en un principio los pesos de cada una de las acciones en la base de reglas es el mismo, de modo que la probabilidad de seleccionar cualquiera de ellas es la misma. Por lo tanto, la selección de la mejor regla de decisión en un comienzo es a prueba y error para cada regla. La etapa de defusificación en este enfoque consiste en combinar las reglas de decisión seleccionadas para obtener la mejor regla de decisión como un dato de salida del sistema (actor).

La regla de decisión final será obtenida mediante método ε-greedy. El crítico entonces evalúa esta regla de decisión y actualiza entonces los pesos del vector $w_d$ correspondiente a las reglas de decisión de las reglas de la base de reglas activada. Este proceso se itera muchas veces y de esta manera el FIS aprende, mejorando su habilidad para identificar las mejores reglas de decisión.

## *3.5 Resumen*

Este capítulo describió las técnicas utilizadas para ejecución de este proyecto, las cuales incluyen: Reforzamiento del Aprendizaje, Sistemas de Inferencia Difuso basado en teorías de Lógica Difusa y por último, los denominados Sistemas de Inferencia Difusos, que como se mencionó anteriormente son una combinación de las dos técnicas anteriores.

La técnica de Reforzamiento del Aprendizaje se utilizará para lograr que el agente aprenda a seleccionar la regla de decisión que minimice la tardanza promedio de los sistemas en estudio. El agente aprenderá a seleccionar acciones óptimas mediante prueba y error. De todos los algoritmos disponibles de Reforzamiento del Aprendizaje, en este proyecto se utilizará específicamente el algoritmo Q-Learning.

La teoría de los sistemas de Inferencia Difuso, los cuales se basan en la teoría de Lógica Difusa, se utilizará para disminuir la gran cantidad de estados del sistema y disminuir por consiguiente el tiempo de computadora. Como existen 3 variables de entrada que definirán el estado del sistema y éstas se pueden combinar de múltiples maneras, a la vez que cada una de ellas puede tomar un sin número de valores, es muy difícil realizar un mapa de estado-acción para definir qué acción seleccionar. Esto consumiría un excesivo tiempo de construcción y programación, debido a la gran cantidad de estados y por lo tanto, también consumiría mucho tiempo de computadora. Por último, los Sistemas de Reforzamiento del Aprendizaje Difuso se detallaron en este capítulo porque a partir de ellos se construyeron los agentes de esta investigación.

En el capítulo cuatro se discute el algoritmo propuesto para programar y controlar a los agentes en esta investigación, el cual aprenderá mediante un algoritmo que se denominó algoritmo Q-

Learning Difuso. Éste es un tipo de sistema de Reforzamiento del Aprendizaje Difuso en cuyo caso específico utiliza el algoritmo Q-learning como técnica de Reforzamiento del Aprendizaje y al cual se le han hecho algunas variaciones para adecuarlo a los sistemas en estudios y a las variables de salida que se esperan obtener.

# CAPITULO 4: Descripción del algoritmo propuesto y características de los sistemas bajo estudio

## *4.1. Introducción*

Este capítulo discute el algoritmo a utilizar en esta investigación para la programación y control de los agentes.

La sección dos de este capítulo describe el algoritmo Q-learning- Difuso que es un Sistema de Reforzamiento del Aprendizaje Difuso que específicamente utiliza el algoritmo Q-learning como técnica RL. En esta sección se especifican las ecuaciones para calcular las funciones de valor de las acciones egoístas (óptimas) y las no egoístas, los errores TD, y la actualización de los pesos de las reglas activadas en cada lapso de tiempo.

La sección tres describe los componentes del algoritmo necesarios para la construcción de los agentes como son las variables de entrada y de salida de los sistemas, las funciones de membresía (número, rangos, y forma) y las etiquetas lingüísticas seleccionadas para cada variable de entrada, las reglas de toma de decisión (o de despacho) seleccionadas y la estructura de la base de reglas del sistema de inferencia difuso.

La sección cuatro discute la estrategia explotación-exploración a utilizar en esta investigación.

## *4.2 Algoritmo Q-learning difuso*

En esta investigación se propone comparar dos sistemas de manufactura de características similares, los cuales serán controlados automáticamente por agentes que utilizan las técnicas de Reforzamiento del Aprendizaje y Lógica Difusa para tomar decisiones y lograr mejorar su rendimiento. Para ello se utilizará el enfoque Reforzamiento del Aprendizaje Difuso que propone [16], pero la técnica de reforzamiento del aprendizaje utilizada será el algoritmo Q-learning. Por tal razón, Jouffe ha denominado a este enfoque Algoritmo Q-learning difuso.

El Algoritmo Q-learning difuso consta de pasos secuenciales, los cuales pueden resumirse de la siguiente manera: un agente percibe un estado del sistema $s_t$ y de acuerdo a ese estado realiza una acción $a_t$, lo cual resulta en un cambio de estado nuevo denominado $s_{t+1}$. La acción realizada fue producto de haber seleccionado una acción de un conjunto de acciones disponibles, cada una con un peso asociado $w_t a_t$ y habiendo seleccionando la acción con el peso más alto de entre las acciones disponibles). Posteriormente, el algoritmo evalúa la acción realizada y entrega una recompensa o penalización al sistema.

En forma detallada, los pasos secuenciales son los siguientes:

1. El agente percibe un estado del sistema ($s_t$), el cual está definido mediante los estados de las variables de entrada.
   a. Fusificación de los valores de las variables de entrada.

2. Inferencia Lógica
   a. Determinar cuales son las reglas de la base de regla activadas.
   b. Calcular el valor Q de la acción egoísta $a_t^i$ de la regla $R_i$ tomada en el estado t, mediante la ecuación 4.1, la cual multiplica los pesos de las reglas activadas con sus verdaderos valores calculados utilizando la T-Norma operador mínimo. Este es el valor Q correspondiente a la acción global $a_t$ resultante de las acciones locales ε-greedy.

$$Q_t(S_t, U_t) = \sum_{R_i \in A_t} W_t^i (a_t^i)\, \alpha_{Ri} \qquad (4.1)$$

   c. Aproximar la función de evaluación óptima del estado actual a partir de la acción óptima máxima (ai) de la regla Ri mediante la ecuación 4.2.

$$Q_t^*(S_{t+1}) = \sum_{Ri \in Ai} \left[ \max_{a \in a(t)} w_t^i(a) \right] \alpha_{Ri} \qquad (4.2)$$

3. Defusificación

   a. Utilizar estrategia de explotación-exploración para seleccionar la regla de decisión. Como se señaló anteriormente, el método utilizado para establecer cuánta explotación y cuánta exploración realizar será el método ε-greedy, donde el (1-ε) % del tiempo se realizará explotación, y el ε % del tiempo se realizará exploración.

      A medida que transcurra el tiempo y el aprendizaje progrese, el valor de ε lo haremos disminuir en el tiempo.

4. El agente recibe una recompensa o penalización por la acción realizada en el entorno.

5. Calcular el error TD mediante la ecuación 4.3, para el estado (t+1) de acuerdo a la recompensa recibida de la acción tomada en el tiempo t. Un error positivo implica que la acción tomada es mucho mejor que la acción conocida hasta ese momento (óptima). Por el contrario, un error negativo implica que la acción es peor que la acción conocida hasta ese momento.

$$Error\ TD = r_{t+1} + \gamma Q_t^*(S_{t+1}) - Q_t(S_t, a_t) \qquad (4.3)$$

6. Actualizar los valores de los pesos (w) de la acción contribuyendo a la acción global basada en el valor del error TD calculado anteriormente. Si el valor del error TD es positivo, entonces los pesos serán incrementados tal que valores más precisos de los pesos de las acciones sean utilizados en el futuro. Por el contrario, si el error TD es negativo, los pesos de las acciones deben ser disminuidos.

La actualización de los pesos se realiza mediante la ecuación 4.4:

$$w_{t+1} = w_t + \alpha_{Ri}(error\ TD) \quad (4.4)$$

## *4.3 Especificaciones de los componentes del algoritmo Q-learning Difuso*

El algoritmo Q-learning difuso descrito en la sección anterior consta de varios componentes, tales como variables de entrada, variables de salida, base de reglas, reglas de decisión (o de despacho), base de reglas, funciones de membresía y etiquetas lingüísticas. Cada uno de los componentes fue elegido en base a la información recolectada de la investigación de Kanetkar [18], en la cual se basa este estudio, los cuales se describen a continuación.

### 4.3.1 Variables de entrada

Las variables de entrada que se consideran en un controlador pueden variar dependiendo de las características del sistema y de la medida de rendimiento de interés. Los resultados obtenidos pueden ser sensibles a la selección de las variables de entrada. En el caso de esta investigación, la medida de rendimiento bajo estudio es la tardanza promedio del sistema y las variables de entrada seleccionadas son los siguientes:

1. Número de piezas en fila
2. Tiempo promedio de espera en fila
3. Tiempo promedio de procesamiento de las piezas que están fila.

El impacto de diferentes opciones de variables de entrada en el rendimiento del sistema, tales como costo por retraso de un tipo de pieza, tiempo de ajuste de la máquina al cambiar de un producto a otro, prioridad de cliente, tiempo de retraso de la pieza, entre otros, quedará para investigaciones futuras.

### 4.3.2 Variables de salida

Como se mencionó en los capítulos anteriores, se desea estudiar la diferencia entre dos sistemas controlados automáticamente mediante Reforzamiento del Aprendizaje y Lógica Difusa. El objetivo de ambos sistemas de manufactura a modelar es minimizar la tardanza promedio de las piezas en el sistema, pero la diferencia primordial entre ellos es la forma de controlarlos, específicamente la manera de calcular la medida de rendimiento Tardanza promedio del sistema. La variable de salida de cada uno de los sistemas es la tardanza promedio de las piezas en el sistema.

a. *Primer sistema:* Cálculo de tardanza promedio estimada para calcular recompensa.
   En el primer sistema de manufactura, las piezas se procesan en cada una de las máquinas de acuerdo a su ruta previamente establecida. En este tipo de sistema, se realizan dos cálculos de tardanza: una tardanza estimada a la salida de cada máquina y una tardanza real que se calcula al momento antes que la pieza deje el sistema.

   Cuando una máquina termina de procesar una pieza, se estima el tiempo restante para que la pieza deje el sistema. Por lo tanto, se estimará los tiempos en fila y los tiempos de procesamiento de las siguientes máquinas de su ruta. Con estos datos, se estimará el tiempo necesario para completar la pieza, al cual se le sumará el tiempo transcurrido hasta la llegada de la pieza al sistema para poder saber en qué momento específico estaría completándose la pieza en particular. Luego, se compara el tiempo estimado para completar la pieza con la fecha de vencimiento para ese tipo de pieza y de esa manera se sabrá si la pieza estaría completándose tarde o temprano. Finalmente se calcula su retraso y tardanza (si es que la hay). En este momento se compara la tardanza promedio estimada con la tardanza promedio calculada al final de la corrida. Si la tardanza estimada es mayor que la real, entonces se le entrega una penalización al agente. Por el contrario, si la tardanza estimada es menor que la real, se le entrega una recompensa.

b. *Segundo sistema:* Cálculo de tardanza promedio para cada máquina individualmente.
   En este sistema se realizarán dos cálculos de tardanza promedio: una tardanza promedio por máquina (individual) y una tardanza promedio total por pieza. La tardanza individual se

calculará al finalizar el procesamiento de una pieza en particular en una determinada máquina. Para ello, se estimarán las fechas de entrega o vencimiento para cada máquina como una proporción de la fecha de entrega total, obtenidos por una corrida de prueba utilizada para calcular el peso relativo para cada máquina. De acuerdo a si la pieza en esa máquina termina su procesamiento a tiempo o tarde con respecto a su fecha de entrega por máquina se le entregará una recompensa o penalización respectivamente.

### 4.3.3 Etiquetas Lingüísticas y Funciones de Membresía

Cada variable de entrada es fusificada mediante funciones de membresía y etiquetas lingüísticas. La selección del número de etiquetas lingüísticas, el rango de las funciones de membresía y la forma de éstas, dependerá del criterio aplicado por el investigador en la resolución de cada problema. Los criterios que se aplican frecuentemente son el conocimiento de un experto, o un ajuste a curvas de acuerdo al comportamiento de la colección de datos.

Las variables de salida obtenidas pueden ser sensibles a la selección. A medida que mayor cantidad de etiquetas lingüísticas son seleccionadas, mayor será la complejidad del sistema y por lo tanto, requerirá mayor tiempo computacional y de programación. Mientras mayor sensibilidad de las variables de salida se requiera, una mayor cantidad de etiquetas lingüísticas se deben seleccionar. Por lo tanto, para un sistema determinado, debe seleccionarse una cantidad de variables lingüísticas de acuerdo a la sensibilidad requerida de los resultados y el nivel de complejidad permitido.

En esta investigación, las etiquetas lingüísticas seleccionadas para las 3 variables de entrada son las siguientes:

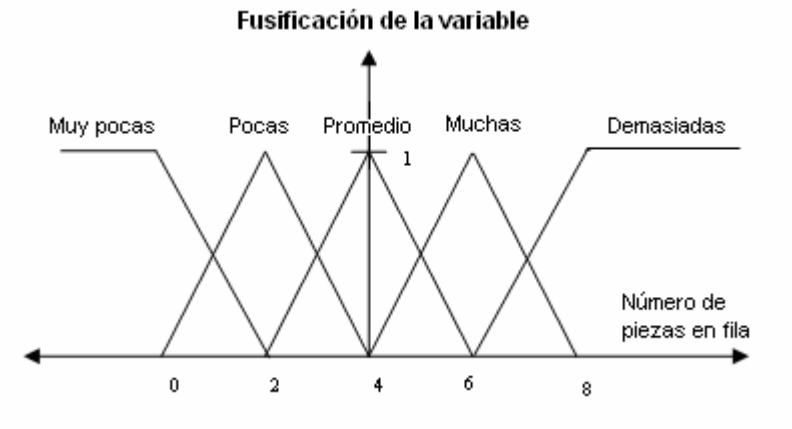
1. Muy Poco
2. Poco
3. Promedio
4. Mucho
5. Demasiado

Existen muchos tipos de funciones de membresía disponibles, tales como funciones gausianas, trapezoidales, triangulares, etc… Para efectos de esta investigación, las funciones de membresía seleccionadas fueron de tipo trapezoidal y triangular. La selección del tipo de función de membresía y su impacto en el rendimiento del sistema es dejado para una investigación futura.

Las funciones de membresía, las etiquetas lingüísticas y sus rangos para cada variable de entrada se presentan en las Figura 3.5 y los rangos de las etiquetas lingüísticas para cada variable de entrada se presentan en la Tabla 3.1.

**Tabla 3. 1.** Rango de las etiquetas lingüísticas para cada variable de entrada

| **Funciones de membresía** | **Trapezoidal** | **Triangular** | **Triangular** | **Triangular** | **Trapezoidal** |
|---|---|---|---|---|---|
| *Variables de entrada /Variable lingüística* | *Muy poco* | *Poco* | *Promedio* | *Mucho* | *Demasiado* |
| Número de piezas en fila | 0-2 | 0-4 | 2-6 | 4-8 | 6 o más |
| Tpo. Promedio de espera en fila | 0-20 | 0-40 | 20-60 | 40-80 | 60 o más |
| Tpo. Promedio de procesamiento de pieza en fila | 0-10 | 0-20 | 10-30 | 20-40 | 30 o más |



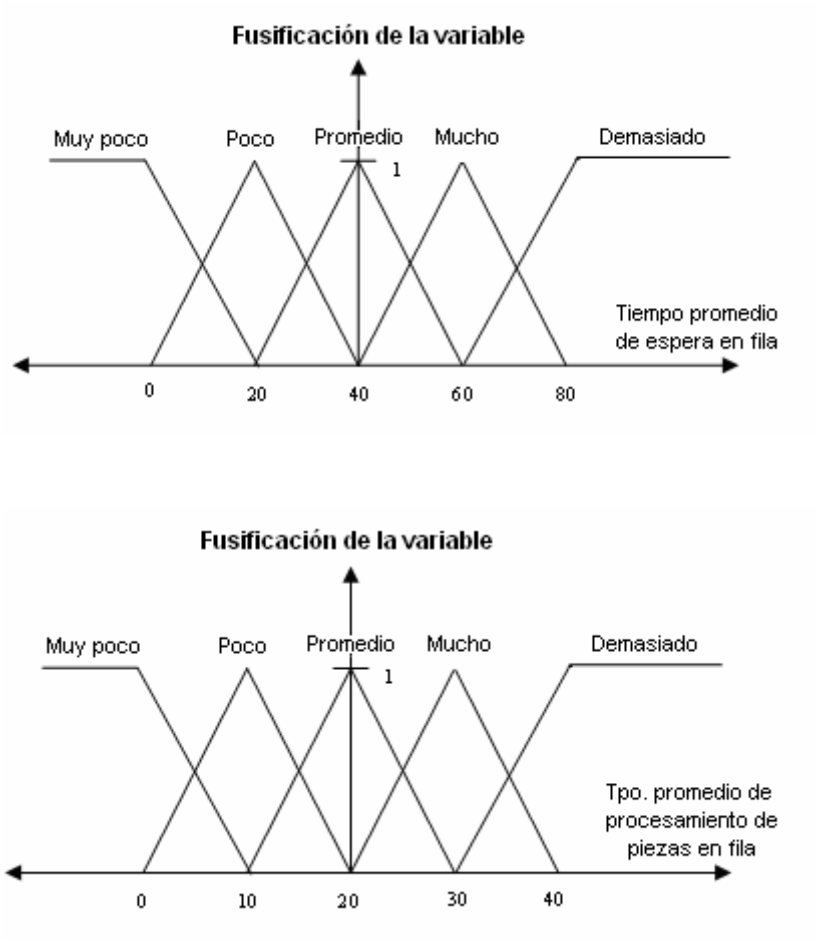



**Figura 3. 5.** Funciones de membresía y etiquetas lingüísticas de las variables de entrada [18]

### 4.3.4 Base de reglas

El número de reglas dentro de la base de reglas es una función del número de variables de entrada (p) y del número de etiquetas lingüísticas ($n_p$) definidas para cada variable de entrada. El número total de reglas en la base de regla (NR) se calcula mediante la ecuación 4.5:

$$NR = \prod_{j=1}^{p} n_j \qquad (4.5)$$

Si el número total de etiquetas lingüísticas es 5 para cada variable de entrada y el número de variables de entrada es 3, por lo tanto, el número total de reglas es:

NR = 5*5*5 = 125

Como se explicó en la sección anterior, la estructura de una regla de decisión es del tipo IF/ELSE-THEN. El componente IF/ELSE corresponde a una combinación de estados de las variables de entrada (un estado de cada variable) y el componente THEN corresponde a un conjunto de acciones, en este caso, reglas de despacho, con sus respectivos pesos. Un ejemplo de una regla de la base de reglas es el siguiente:

IF V1= "Muy Poco", V2= "Poco" y V3= "Mucho",
THEN : Variable de salida = FIFO , w= 1
Variable de salida = SPT , w= 1
Variable de salida = EDD , w= 1,

donde, V1, V2 y V3 son las variables de entrada, "Muy Poco", "Poco" y "Mucho" son etiquetas lingüísticas y, FIFO, SPT y EDD son variables de salida.

El conjunto de acciones asociadas a cada regla en la base de reglas tiene un vector $w_d$ , que contiene los pesos de cada variable de salida, los cuales se actualizan en cada episodio de tiempo aumentando el peso de la acción que mejora la medida de rendimiento. Inicialmente, los pesos asociados a cada variable de salida es el mismo, porque no ha ocurrido aprendizaje hasta el momento y existe la misma probabilidad de seleccionar cualquiera de ellas. El algoritmo de RL actualiza los pesos de las variables de salida a medida que progresa el aprendizaje. Los pesos se calculan como el promedio de los pesos de todas las acciones obtenidas de las reglas activadas.

En la presenta investigación, se requiere que el algoritmo entregue sólo una regla de decisión y no un conjunto de ellas. Para lograr esto, se crearon reglas independientes para cada variable de salida, es decir, un número de reglas con la misma combinación de variables de entrada en el IF, pero con diferentes variables de entrada en el THEN. Por lo tanto, el número total de reglas calculado anteriormente se multiplicará por la cantidad de variables de salida (reglas de despacho seleccionadas).

En concreto, para calcular la cantidad de reglas en la base de reglas se utiliza la ecuación 10 y el resultado se multiplica por la cantidad de variables de salida, que en este caso son 3. Por lo tanto, el número total de reglas en la base de reglas será:

$$NR = 3*(5*5*5) = 375$$

### 4.3.5 Reglas de toma de decisión consideradas (reglas de despacho).

En esta investigación las reglas de decisión o de despacho que se utilizarán en los sistemas son las siguientes:

1. FIFO (Firts in First out): Primero en entrar, primero en salir.
2. SPT (Shortest Processing Time): Menor tiempo de procesamiento
3. EDD (Earliest Due Date): Tiempo de entrega más temprana

Se seleccionaron estas reglas debido a que aritméticamente hablando son fáciles de calcular además de que son las que con mayor frecuencia utilizan los investigadores. Una razón adicional es que en el caso del EDD, afecta directamente la medida de desempeño de interés, la Tardanza Promedio. Por otro lado, con EDD es posible calcular costos debido a retrasos en entrega de productos.

## *4.4. Estrategia de explotación-exploración*

Como se mencionó anteriormente, se utilizará el método ε-greedy para establecer cuánta explotación y cuánta exploración realizar. La estrategia ε-greedy se define mediante 1-ε, siendo (1-ε) el porcentaje del tiempo en el cual se realizará explotación, y ε el porcentaje del tiempo en el cual se realizará exploración. En un comienzo de la simulación se utilizará $\varepsilon = 1$, para cambiar a medida que transcurre el tiempo a $\varepsilon = 0.8$, $\varepsilon = 0.6$, $\varepsilon = 0.4$, $\varepsilon = 0.2$ y finalmente a $\varepsilon = 0$. De esta manera, los porcentajes de explotación-exploración serán de forma secuencial: 0-100%, 20-80%, 40-60%, 60-40%, 80-20%, 100-0% respectivamente.

## *4.5 Resumen*

Este capítulo describió el algoritmo Q-Learning Difuso aplicado a los sistemas en estudio. Se especificaron las ecuaciones que utilizarán los agentes para realizar el aprendizaje mediante Q-learning como también se describió los componentes de este algoritmo. Se especificó las variables de entrada, de salida, base de reglas del sistema de inferencia, las reglas de decisión utilizadas, las funciones de membresía y etiquetas lingüísticas para cada variable de entrada.

En el próximo capítulo se discute todo lo relacionado a la implementación del algoritmo Q-learning Difuso en un sistema de manufactura modelado en Arena™. Se describen los elementos utilizados para el funcionamiento del algoritmo, los que incluyen: los elementos incorporados en el modelo de Arena para realizar interfaces de comunicación entre Arena y Visual Basic de Arena, los distintos códigos de programación en MATLAB™ que forman parte del algoritmo Q-learning Difuso y las interfaces de comunicación entre Visual Basic de Arena™ y Matlab™. En ese capítulo se describe también el funcionamiento del sistema completo en forma secuencial.

# CAPITULO 5: Implementación

## *5.1 Introducción*

En este capítulo se describen los distintos códigos de programación de cada lenguaje utilizado para el control y programación de los agentes, como también el funcionamiento paso a paso de cada uno de ellos cuando ocurre un punto de decisión. Recordemos que se denominó punto de decisión al momento en el cual una máquina termina de procesar una pieza y debe seleccionar la siguiente pieza en su fila para procesarla. Para ello, el agente debe decidir primero cual será la regla de despacho que utilizará para seleccionar la pieza su fila con el objetivo de disminuir la tardanza promedio. Para lograr que las máquinas aprendan a mejorar su rendimiento en la toma de decisiones autónomamente se utilizó el algoritmo Q-learning de Reforzamiento del Aprendizaje en donde a cada máquina se le indica la meta final pero no cómo lograrla.

Para manejar la gran cantidad de estados del sistema máquina y disminuir el tiempo de procesamiento de los datos se utilizó Sistemas de Inferencia Difuso basados en la teoría de Lógica Difusa. El mecanismo que se utilizó para demostrar el impacto de los dos sistemas de control es simulación, del cual se obtendrán y luego analizarán los resultados de los sistemas. Para ello se modeló los sistemas de manufactura utilizando el lenguaje de simulación SIMAN del programado Rockwell Arena$^{TM}$.

En la sección dos se discuten los programas utilizados para el funcionamiento del algoritmo Q-Learning Difuso propuesto, el objetivo de cada uno de los componentes y archivos construidos para cada lenguaje de programación y el método de comunicación entre ellos.

Finalmente, en la sección tres se describe el funcionamiento del algoritmo Q-learning difuso paso a paso.

## *5.2 Programas utilizados en el algoritmo Q-Learning Difuso y sus interfaces de comunicación*

Para la construcción de los agentes programables se utilizó los programas Arena™, Visual Basic de Arena™, Microsoft Excel™y Matlab™. Ëstos interactúan en todo momento durante la corrida de simulación intercambiando información.

### 5.2.1 Rockwell Software Arena™

Los sistemas se modelaron utilizando el programa Arena™, que es la plataforma principal desde donde todo el sistema opera y la cual genera los llamados a los otros programas. Dentro del modelo de Arena™ se colocaron siete bloques ”VBA”, que son los que contienen programación en Visual Basic de Arena™, la cual se ejecuta cuando una entidad pasa por ellos. Los bloques utilizados podemos clasificarlos según su funcionalidad en 3 tipos: bloques de selección de regla, bloques de asignación de recompensas y bloques de cálculo de tardanza. El objetivo de cada bloque se explica a continuación.

Los bloques VBA de selección de regla contienen la programación necesaria para seleccionar una regla de despacho y actualizarla en la máquina de la estación en la que se encuentra al momento en que ocurre un punto de decisión. Se utilizaron 3 de estos bloques, uno en cada estación del modelo, justo antes del bloque “Release”, el cual representa la liberación del recurso que estaba ocupado. Al llegar una pieza a cualquiera de estos bloques “VBA”, se activa inmediatamente la programación contenida en Visual Basic de Arena™. Esta programación involucra lo siguiente:

1. Leer la regla de despacho (política) que se utilizó para seleccionar la pieza que actualmente está ocupando el recurso.
2. Percibir el estado del entorno del sistema, representado por las variables de entrada y almacenar estos valores en un archivo de Excel™.
3. Llamar al algoritmo Q-learning difuso programado en Matlab™, cuyo código es el encargado de seleccionar la mejor regla de despacho para el estado percibido.
4. Almacenar en un archivo de Excel™ la regla seleccionada por el algoritmo.

5. Leer la regla de despacho almacenada en Excel™ y actualizarla en la máquina para seleccionar la próxima pieza (lo cual se hace a través del Elemento "Queues" en el Experimento de Arena™)
6. Guardar toda la información obtenida en archivos de Excel™.

En el caso de los bloques "VBA" de tipo asignación de recompensas, se colocaron 3 de éstos en el modelo, uno en cada estación. En el momento en que una entidad ha terminado de ser procesada por el recurso en una determinada máquina y la entidad lo libera mediante el bloque "Release", ésta pasa inmediatamente por uno de estos bloques, activando la programación de visual Basic contenida en ellos. Este tipo de bloques contienen la programación necesaria para entregar una recompensa, elemento importante en algoritmos de aprendizaje. La recompensa entregada dependerá de la efectividad en seleccionar una regla de despacho que logre disminuir la tardanza promedio del sistema. Es decir, si la regla de despacho seleccionada para remover una pieza de la fila de una máquina en un punto de decisión logra disminuir la tardanza promedio del sistema, el agente entregará una recompensa (reforzamiento positivo) y en caso contrario, entregará una penalización (reforzamiento negativo).

Finalmente se utilizó un bloque "VBA" de tipo cálculo de tardanzas en la estación de salida del sistema, antes de hacer el "Dispose" de las entidades. La programación contenida en este bloque tiene la tarea de almacenar en un archivo de Excel™ para posterior análisis los siguientes datos: tardanza promedio del sistema, tardanza promedio en forma individual para cada tipo de pieza y tiempo de reloj exacto en el cual pasa la entidad por el bloque.

### 5.2.2 Microsoft Excel™

Para almacenar los datos recopilados y posteriormente analizar los resultados obtenidos en cada episodio de la corrida completa se utilizaron archivos de Excel™, los cuales, además de un método de almacenamiento, sirvieron también como un medio de comunicación entre Visual Basic, Matlab™ y Arena™.

Se utilizaron cinco archivos, los cuales se crean al comienzo de la simulación y se mantienen abiertos y activos durante toda la corrida, debido a que se está traspasando información entre los distintos lenguajes de programación. A continuación se detalla el objetivo de cada archivo.

1. Inputs.xls. Este archivo tiene el objetivo de almacenar los valores de las variables de entrada que determinan el estado del sistema cuando ocurre un punto de decisión. Cuando una entidad llega al bloque "VBA" activa la programación de Visual Basic de Arena™ y ésta almacena los valores de las variables de estado en este archivo, para que luego sean leídas por Matlab™ para realizar la fusificación de los valores.

2. Recompensas.xls. En este archivo se almacena la recompensa obtenida por la efectividad de la selección de la regla de despacho recién seleccionada. Al terminar de procesar una pieza en una máquina, la entidad procesada entra al bloque "VBA" y activa la programación de Visual Basic de Arena™, la cual calcula la recompensa que debe entregarse dependiendo si la regla utilizada contribuye a disminuir la tardanza promedio, en cuyo caso es un reforzamiento positivo, o si la aumenta, donde el reforzamiento es negativo. El reforzamiento asignado es almacenado en este archivo para luego ser leído por Matlab™.

3. Regla.xls. En este archivo se almacena la regla de despacho que es seleccionada mediante el algoritmo Q-learning Difuso, es decir, a partir de la función de Matlab™ algoritmo_qlearning.m que se describe en la siguiente subsección. La regla de despacho seleccionada es posteriormente leída de este archivo por Visual Basic de Arena™ para actualizar la regla de despacho a utilizar en el elemento Queues del experimento en Arena™ para seleccionar la siguiente pieza en fila.

4. Tardanzas.xls. Este archivo almacena la tardanza promedio calculada en la estación "Salida" del modelo de Arena™. Antes que una entidad abandone el sistema pasa por el último bloque "VBA", el cual calcula la tardanza promedio del sistema, la tardanza promedio para cada tipo de pieza y el tiempo en el cual la entidad pasa por el bloque y los almacena en este archivo para posterior análisis.

5. Todos los Datos.xls. Este archivo guarda toda la información recopilada en cada episodio de tiempo para cada una de las máquinas. Este archivo divide los datos en las tres máquinas y para cada una de ellas almacena los valores de las variables de entrada, la regla utilizada, la tardanza promedio y la recompensa obtenida.

### 5.2.3 Matlab™

Como se mencionó en la sección anterior, el algoritmo Q-learning Difuso fue programado en Matlab™, el cual fue divido en cuatro diferentes funciones que poseen el mismo nombre que los archivos que las contienen. Estos archivos se describen a continuación.

1. Variables_almacenadas.m. Este archivo contiene algunas de las variables utilizadas en el algoritmo que necesitan inicializarse. A lo largo de la corrida estas variables cambian de valor y estos nuevos valores son almacenados en el archivo con el mismo nombre, pero con extensión .mat. Cada vez que inicialice la corrida de simulación, se debe llamar a esta función previamente.

2. Fusificacion.m. Este archivo lee los estados de las variables de entradas almacenadas previamente por Visual Basic de Arena™ en el archivo "Inputs.xls" de Excel™.

A los valores de las variables de entrada se le asigna un índice que representa la pertenencia a una determinada etiqueta lingüística, siendo 0 si no hay pertenencia y entre 1 y 5 si hay pertenencia a los conjuntos. El número es indicativo de la pertenencia a una etiqueta en particular, por ejemplo el 3 indica la pertenencia a la etiqueta 3. Las pertenencias a los conjuntos son almacenados en variables tipo arreglo de tamaño 1x5 (una para cada variable de entrada) que contiene la pertenencia a cada etiqueta lingüística. Estas variables son enviadas por referencia a la función "algoritmo_qlearning.m". La última línea del código de la función "fusificacion.m" hace un llamado a la función "algoritmo_qlearning.m", utilizando como argumento los vectores con las pertenencias a cada etiqueta lingüística de cada variable de entrada.

3. Algoritmo_q-learning.m. Este archivo es el encargado de seleccionar la regla de despacho y de actualizar los pesos de las reglas activadas en el episodio de tiempo anterior. El código de esta

función tiene dos etapas. La primera es evaluar los resultados del episodio anterior (t) y la segunda es seleccionar la regla de despacho para la corrida actual (t+1).

La primera etapa del algoritmo comienza leyendo la recompensa asignada por el resultado obtenido al seleccionar la regla de despacho en el episodio anterior. Con esta recompensa se calcula el error TD mediante la ecuación 8 del capítulo 4. Se calculan también los valores alfa del episodio anterior mediante la función "obtener_regla_alfa.m". Finalmente, con el error TD y los valores alfa se actualizan los valores de los pesos w de las reglas activadas de la base de reglas en la corrida anterior mediante la ecuación 9 del capítulo 4. Cada regla de la base de reglas tiene asociada un peso w, que es el indicador de cuán buena es la selección de una regla despacho en particular. En un comienzo todas las reglas se inicializan con un peso igual a 0.5 y se van actualizando a medida que la corrida de simulación avanza y se va explorando la selección de distintas reglas de despacho.

La segunda etapa del algoritmo corresponde al episodio actual, en el cual el algoritmo tiene que seleccionar dos acciones, la óptima (ó acción egoísta) y la que no es óptima (no egoísta), independientemente de la estrategia de explotación-exploración que se esté utilizando en ese momento. Inicialmente el algoritmo selecciona la acción no egoísta, que en este caso se selecciona aleatoriamente entre las tres reglas de despacho disponibles. Luego, el algoritmo debe seleccionar la acción óptima, es decir, la regla de despacho que hasta ese momento es la que refleja un mayor reforzamiento por la acción realizada en episodios anteriores. Ésta se encuentra seleccionando entre las reglas de la base de reglas activadas en el episodio de tiempo actual, la que posea el valor de pesos w más alto. Cuando ya se han seleccionado las reglas egoísta y no egoísta, es necesario calcular el valor Q de cada acción. Para ello, hay que encontrar el valor de los pesos de las reglas activadas por la combinación de valores de las variables de entrada y el valor alfa para cada regla activada, el cual se calcula mediante la función "obtener_regla_alfa.m". Los valores Q se calculan mediante las ecuaciones 6 y 7 del capítulo 4.

Finalmente, para seleccionar una de las reglas de despacho, se debe utilizar la estrategia de explotación-exploración. Para seleccionar la estrategia se utilizó el método ε-greedy. Recordemos

que este método establece realizar explotación con una probabilidad de un (1-ε) % de las veces y realizar exploración un ε %.

En un comienzo ε variará desde 1 hasta 0, decreciendo en montos de 0.2. El cambio de estrategia será determinado por la cantidad de veces que sea llamado el algoritmo para reflejar el tiempo. Con la estrategia de explotación-exploración se selecciona la regla final, que se almacena en el archivo "Reglas.xls" de Excel™. En el primer episodio se seleccionará la acción no egoísta, debido a que se hará 0% explotación y 100% exploración, posteriormente variará entre explotación y exploración dependiendo de las probabilidades y en el episodio final se seleccionará la acción egoísta debido a que se hará 100% explotación y 0% exploración.

4. Obtener_regla_alfa.m. Esta función es la encargada de calcular el verdadero valor alfa de las distintas combinaciones de pertenencias a las etiquetas lingüísticas de cada variable de entrada. Esta función es llamada por la función "algoritmo-qlearning.m" entregándole como argumentos tres vectores correspondientes a cada variable de entrada, que contienen la pertenencia de los valores reales de las variables de entrada a cada una de las etiquetas lingüística. También se le entregan como argumentos índices con los cuales se calculan las distintas combinaciones de estas pertenencias a conjuntos.

Con los valores de pertenencia a cada una de las etiquetas lingüísticas y los índices entregados, se calcula el verdadero valor alfa para esa combinación en particular en cada una de las variables de entrada. Finalmente, se calcula el valor alfa final utilizando la T-Norma producto, valor que es devuelto a la función "algoritmo_qlearning.m".

## *5.3 Funcionamiento del algoritmo Q-learning Difuso propuesto*

Como se mencionó en la sección anterior, el algoritmo q-learning difuso opera mediante la comunicación entre el modelo construido en Arena™ y Visual Basic de Arena™, Matlab™ y Excel™. Hemos denominado punto de decisión al momento en el cual una máquina termina de procesar una pieza, ésta queda desocupada y tiene que seleccionar la próxima pieza a procesar

entre las piezas que esperan en su fila. La siguiente pieza será seleccionada mediante una regla de despacho que es entregada por el algoritmo Q-learning Difuso que hemos propuesto. Todo el procedimiento comienza desde el modelo de Arena™. Cuando ocurre un punto de decisión en una máquina, una entidad entra al bloque de VBA en el modelo de simulación construido en Arena™, lo cual activa el código de programación escrito en Visual Basic de Arena™ contenida en éste. La secuencia de acciones de cada lenguaje cuando ocurre un punto de decisión se presenta en la Figura 5.1 y es explicada paso a paso a continuación.

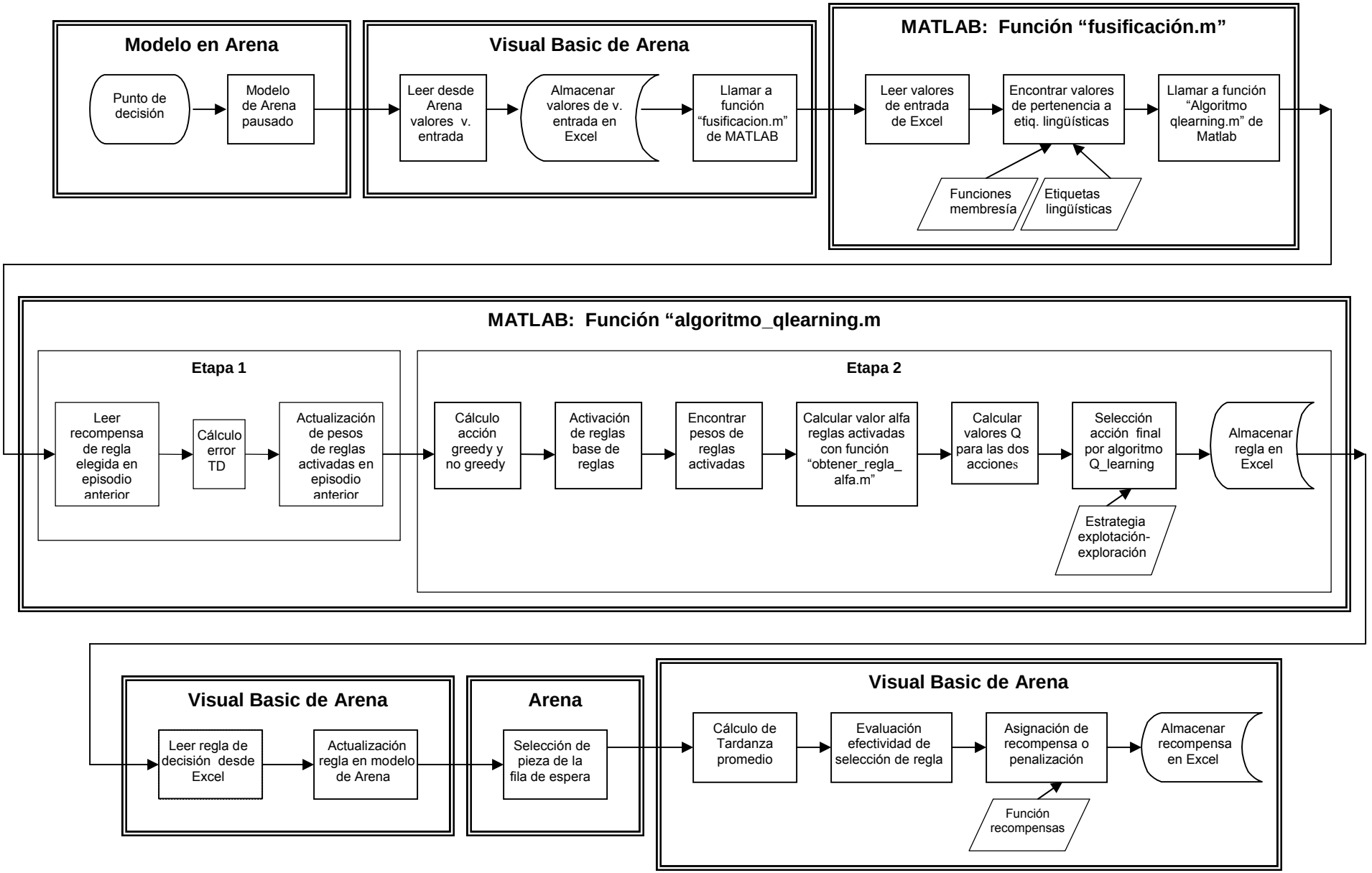



**Figura 5. 1.** Procedimiento a utilizar cuando ocurre un punto de decisión en Arena™

**1. Desde Visual Basic de Arena™**

Visual Basic lee desde el modelo de Arena™ las variables de entrada de la máquina que activó el bloque VBA, las cuales son almacenadas los archivos de Excel™ "Inputs.xls" y en "Todos los Datos.xls".

Posteriormente, llama al algoritmo Q-learning difuso desde Visual Basic el cual está escrito en Matlab™, ejecutando la función "fusificacion.m". Esta función es la encargada de leer los valores de las variables de entrada desde el archivo "Inputs.xls" y de transformarlos en valores difusos mediante las funciones de membresía y etiquetas lingüísticas. La Figura 5.2 presenta los pasos que sigue el algoritmo Q-learning difuso, desde el momento en el cual se llama a Matlab™ desde Visual Basic. La última línea de esta función llama a la función "algoritmo_qlearning.m" de Matlab™, traspasándole los valores de las variables de entrada fusificadas para cada etiqueta lingüística en forma de argumentos.

**2. Desde Matlab™**

La función "algoritmo_qlearning.m" es la encargada de seleccionar la regla de despacho para que la máquina pueda seleccionar la siguiente pieza de su fila de espera. Como explicó en la subsección anterior, esta función consta de dos etapas. La primera etapa es la encargada de leer la recompensa entregada en el episodio de tiempo anterior y con ésta calcular el error TD.

Luego, se actualizan los pesos de las reglas de la base de reglas activadas en el episodio anterior. Si la acción seleccionada en el episodio anterior fue buena, el error TD es positivo, y el peso de la regla incrementa, y si la acción fue desfavorable, el error TD es negativo y es peso de la regla decrece.

La segunda etapa es la encargada de seleccionar la regla de despacho. Para ello se seleccionan las reglas no egoísta y la egoísta. Recordemos que la regla no egoísta en esta investigación se selecciona en forma aleatoria. La regla egoísta se selecciona entre las variables de salida que presenta el valor de los pesos w más alto entre las reglas de la base de reglas activadas en el episodio de tiempo actual.

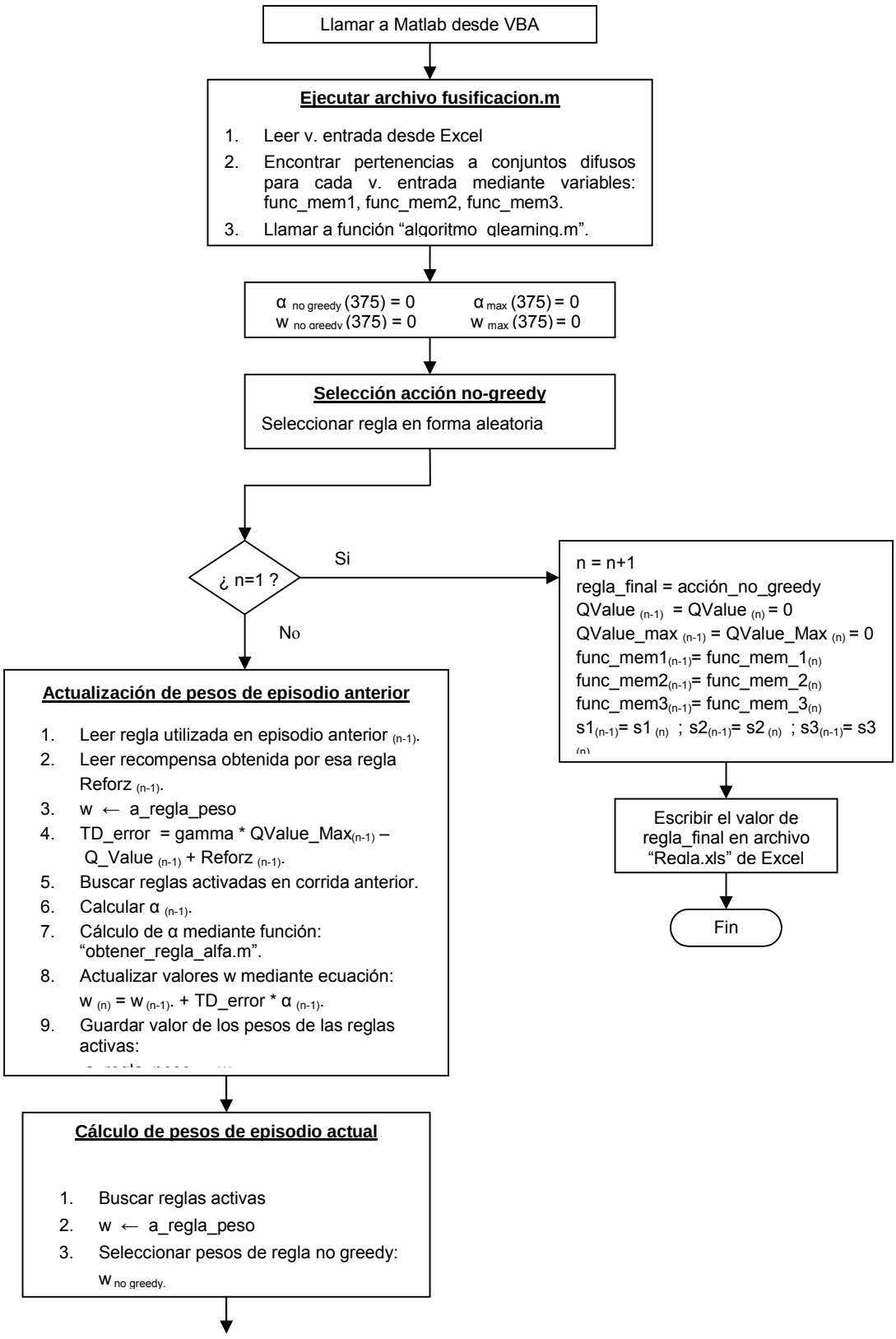



**Figura 5. 2.** Pasos a seguir por el algoritmo Q-learning difuso escrito en Matlab™ (primera parte)**.**

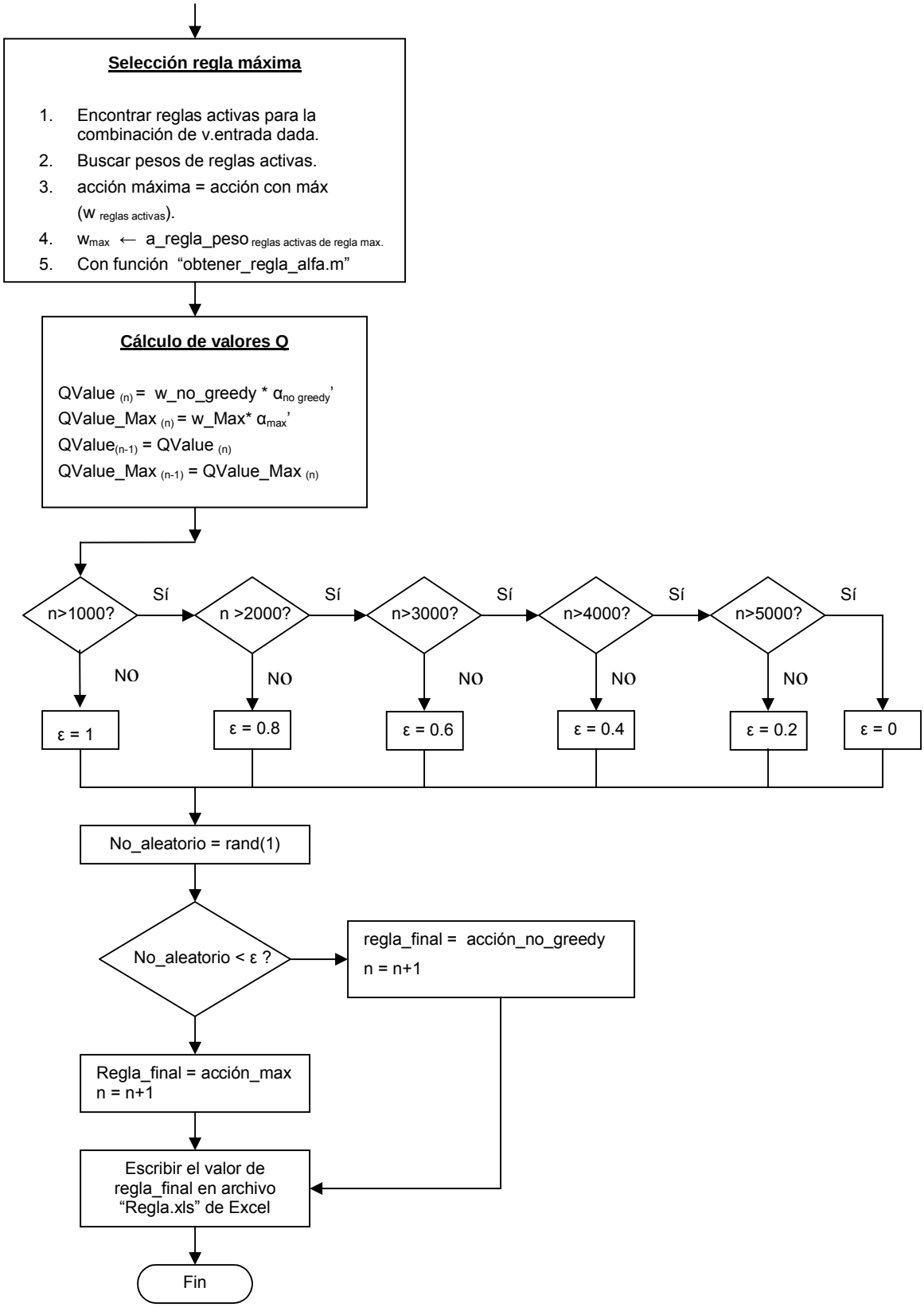



**Figura 5. 3.** Pasos a seguir por el algoritmo Q-learning difuso escrito en Matlab™ (continuación).

Como se explicó en la sección anterior, cuando se ha encontrado tanto la regla egoísta como la no egoísta, se debe encontrar los pesos de las reglas activadas correspondientes a esas variables de salida (regla de despacho seleccionada). Además, se debe calcular los valores alfa de cada combinación de pertenencias a las etiquetas lingüísticas. Para ello, la función "algoritmo_qlearning.m" ejecuta la función "obtener_regla_alfa.m", que es la encargada de calcular el valor alfa de una combinación de pertenencias a etiquetas lingüísticas de las variables de entrada. Con los valores de los pesos w de las reglas activadas y los valores alfa de las reglas activadas se calculan los valores Q para la regla egoísta y la no egoísta, mediante las ecuaciones 6 y 7 del capítulo 4. Finalmente, mediante la estrategia de explotación-exploración se selecciona una regla de despacho entre las reglas egoísta y no egoísta respectivamente. La regla de despacho seleccionada por la función algoritmo q_learning es almacenada en el archivo "Reglas.xls" de Excel™.

### 3. Visual Basic de Arena™

El control vuelve nuevamente a Visual Basic. Después de haber ejecutado a las funciones de Matlab™, se lee la regla almacenada recientemente por Matlab™ en el archivo "Reglas.xls" de Excel™.

La regla de despacho leída es primero almacenada en el archivo "Todos los Datos.xls" y luego es actualizada en el modelo de Arena™ mediante el elemento Queues del Experimento. Los cambios realizados de acuerdo a la regla de despacho seleccionada son los siguientes:

- Si la regla de despacho seleccionada es FIFO, sólo se actualiza el nombre de la regla, en este caso "FIFO", en el "Ranking Criterion". De esta forma, la regla seleccionará las piezas en orden de llegada a la fila.
- Si la regla de despacho seleccionada es SPT, se actualiza en el "Ranking Criterion" el criterio "Low Value First " y donde dice "Rule Expression" se coloca el atributo del tiempo de procesamiento de las piezas en esa máquina. Si se trata de la máquina 1, colocará el atributo Tpo_proces_1. De esta forma, este criterio seleccionará las piezas de la fila de acuerdo al menor tiempo de procesamiento.
- Si la regla de despacho seleccionada es EDD, se actualiza en el "Ranking Criterion" el criterio "Low Value First " y donde dice "Rule Expresión" se coloca el atributo del "due

date". De esta forma, este criterio seleccionará las piezas de la fila de acuerdo al menor "due date".

**4. Arena™**

El control vuelve al modelo de Arena™. Como ya la regla de despacho ha sido actualizada, la máquina puede seleccionar la siguiente pieza a procesar. La pieza es seleccionada, procesada y al salir la pieza de la máquina se activa el bloque de VBA encargado de asignar las recompensas. El modelo de Arena™ queda nuevamente pausado y el control es pasado a Visual Basic de Arena™.

**5. Visual Basic de Arena™**

En este punto existen diferencias en cuanto a cómo se calculan las recompensas en los dos sistemas que se desea comparar. Se explicarán ambas a continuación.

- *En el sistema que posee sólo un agente para controlar todo el sistema*

En el bloque VBA de recompensas, se calcula el tiempo estimado que le tomaría a la pieza salir del sistema, considerando el tiempo transcurrido desde que ésta entró al sistema y el tiempo transcurrido desde el comienzo de la corrida hasta que la pieza entró al sistema. El valor estimado se compara con el la fecha de entrega de la pieza y con respecto a éste se evalúa si la pieza está atrasada o está a tiempo. Si la pieza está atrasada, la tardanza es positiva y el reforzamiento que se le entrega es negativo. Si la pieza está a tiempo, la tardanza es igual a cero y se le entrega un reforzamiento positivo. El reforzamiento entregado es almacenado en los archivos de Excel™ "Recompensas.xls" y "Todos los Datos.xls". El control es pasado nuevamente al modelo de Arena™, donde la simulación sigue su curso hasta que ocurra otro punto de decisión.

- *En el sistema que es controlado por un agente para cada máquina.*

Se ha hecho un estimado de la fecha de entrega por pieza en cada una de las máquinas, es decir, la fecha de entrega por pieza se han divido en forma proporcional al tiempo del sistema para cada estación. Esto quiere decir, se hizo una corrida de prueba con la política FIFO de forma estática y se midió el tiempo en el sistema para cada tipo de pieza y el tiempo que le tomaba a cada pieza pasar por cada una de las estaciones. El tiempo que le toma a un tipo de pieza determinado pasar

por una estación con respecto al tiempo que se espera que la pieza esté en el sistema debe ser proporcional al tiempo de entrega por estación con respecto al tiempo de entrega final para el sistema.

La tardanza por máquina se calcula en el modelo de Arena™ para cada pieza para efectos de análisis de datos, y también se calcula en el código del bloque VBA que se encuentra a la salida de cada una de las máquinas para efectos de otorgar la recompensa. Se evalúa si la pieza está tardía (tardanza sea mayor a cero) o si la pieza está a tiempo (tardanza igual a cero). Ésta se compara con la tardanza promedio calculada hasta ese momento y si la tardanza promedio en esa máquina en particular disminuyó, debido a la selección de la regla de despacho, se otorga un reforzamiento positivo, y en caso contrario, se otorga un reforzamiento negativo. El reforzamiento asignado se almacena en los archivos de Excel™ "Recompensas.xls" y "Todos los Datos.xls" y el control es pasado nuevamente al modelo de Arena™, el cual continúa con la corrida de simulación hasta que ocurra otro punto de decisión.

## *5.4. Resumen*

Este capítulo discutió los programas utilizados y los distintos códigos de programación utilizados para lograr el funcionamiento integral del algoritmo Q-learning difuso y por lo tanto, el control de los agentes. Además, se describió los elementos necesarios incorporados en el modelo de Arena™ para lograr la comunicación entre el modelo y Visual Basic, como también se describió el objetivo de cada uno de los archivos para cada lenguaje involucrados en el funcionamiento del algoritmo y su mecanismo de comunicación.

En el siguiente capítulo se discutirán los experimentos realizados en cada uno de los sistemas en estudio, la cantidad de réplicas, los tiempos de corrida, los tiempos entre llegadas y de procesamiento, las rutas de procesamiento de las piezas, entre otros. Finalmente, se presentarán los resultados de cada uno de los experimentos y se realizará un análisis de los resultados obtenidos.

# CAPITULO 6: Experimentación y análisis de resultados

## *6.1 Introducción*

Para estudiar el comportamiento de los dos sistemas bajo estudio, se llevaron a cabo 3 experimentos. Datos razonables fueron seleccionados arbitrariamente, pero lo más similares posibles a los datos utilizados en la investigación de Kanetkar [18], en la cual está basada este trabajo.

El primer experimento es la simulación de la celda de manufactura controlada por sólo un agente, el cual contiene la información de todo lo que ocurre en el sistema. El segundo experimento es la simulación de la celda de manufactura, que es controlada por tres agentes, donde cada agente contiene sólo la información correspondiente a la máquina a la cual pertenece y a su entorno, y las decisiones que toma sólo afectan a la máquina en particular. Para evaluar la efectividad de cada uno de los sistemas, se realizó un tercer experimento, en el cuál se hicieron corridas de simulación utilizando reglas estáticas, es decir, la mima regla para todas las máquinas y la misma regla durante toda la corrida de simulación. Las reglas estáticas utilizadas fueron FIFO, SPT y EDD. Finalmente, se compararán los resultados de aplicar el algoritmo Q-learning Difuso mediante un agente con los resultados de aplicar el algoritmo Q-learning Difuso mediante tres agentes, es decir, uno para cada máquina.

La sección dos de este capítulo discute en detalle la experimentación realizada para cada tipo de sistema simulado y los resultados obtenidos.

La sección tres presenta el análisis de los resultados obtenidos mediante la experimentación realizada, en donde se utilizó comparación mediante análisis estadístico a través de intervalos de confianza y prueba pareada t y comparación mediante percentilas.

## *6.2 Experimentación*

Para llevar a cabo la experimentación se definieron valores iniciales necesarios para correr el modelo de simulación para los dos sistemas. Éstos son tiempos de corrida, unidades tiempo, tiempo entre llegadas de piezas al sistema, tiempos de procesamiento, secuencia de procesamiento, entre otros.

Cada sistema de manufactura constaba de 3 tipos de pieza y 3 máquinas. Cada tipo de pieza tenía su propia ruta de procesamiento y cada una de ellas debía ser procesada por todas las máquinas. En cada máquina había una fila dedicada con capacidad de 10 piezas. Todas las piezas llegaban inicialmente a una misma fila de capacidad infinita, las cuales se iban removiendo a medida que las filas dedicadas de cada máquina se iban desocupando, tomado en cuenta la ruta del tipo de pieza. Como las filas locales poseían capacidad finita, se creó una fila a la salida de cada máquina. Las piezas de las filas de salida eran removidas a medida que había espacio en la fila dedicada de la máquina a la cual le correspondía ir según su ruta de procesamiento. Las piezas llegaban a la fila local de una máquina ya fuese desde la fila del sistema o desde la fila de salida de otra máquina. La capacidad de procesamiento de cada máquina era de una pieza y el tiempo de procesamiento de cada pieza en cada máquina se comportaba de acuerdo a una distribución de probabilidad triangular, pero para el tiempo de procesamiento sólo se tomó en cuenta la moda.

Debido a que el algoritmo aprendía a prueba y error, necesitaba obtener gran cantidad de información, la cual se obtenía de la cantidad de piezas procesadas. Para ello se hicieron corridas de simulación de datos de un total de 604,800 segundos, es decir 168 horas. Esta cantidad de tiempo fue suficiente para observar el comportamiento de los datos cuando se compara políticas estáticas con la política utilizada por el algoritmo Q-learning de un agente, lo cual se explicará en la próxima subsección. Los tiempos entre llegadas se definieron mediante una distribución de probabilidad exponencial con media de 90 segundos y las piezas llegaban en lotes de 5 piezas.

Como se explicó en los capítulos anteriores, la celda de manufactura producía 3 tipos de producto, donde cada tipo de producto era procesado en todas las máquinas de acuerdo a su ruta. Para poder generar 3 tipos diferentes de piezas se utilizó la función “Discrete” de Arena™, donde

la proporción de piezas tipo 1 es de 40 %, de tipo 2 es de 25% y de tipo 3 es de 35%. A cada tipo de pieza se le asignó una fecha de entrega (due date), la cual estaba definida por la siguiente expresión:

Fecha de entrega = Tiempo de llegada + (Tpo en el sistema + 5% Tpo. En el sistema) (6. 1)

Se le asignó un 5% adicional al tiempo en el sistema para simular un factor se seguridad con el objetivo de darle un rango un poco mayor a la pieza para que pueda completar su procesamiento, debido a que se está trabajando con un valor promedio de tiempo en el sistema entregado por el reporte de Arena™. Los tiempos de procesamiento en cada una de las máquinas y la secuencia de procesamiento para cada tipo de pieza se presentan en la Tabla 6.1.

**Tabla 6. 1.** Secuencia y tiempo de procesamiento de las piezas [s]

| Tipo de pieza | Rutas y tiempos de procesamiento |
|---|---|
| 1 | M1(15) - M3 (10) - M2 (10) |
| 2 | M2 (5) - M3 (10) - M1 (10) |
| 3 | M2 (20) - M1 (10) - M3 (25) |

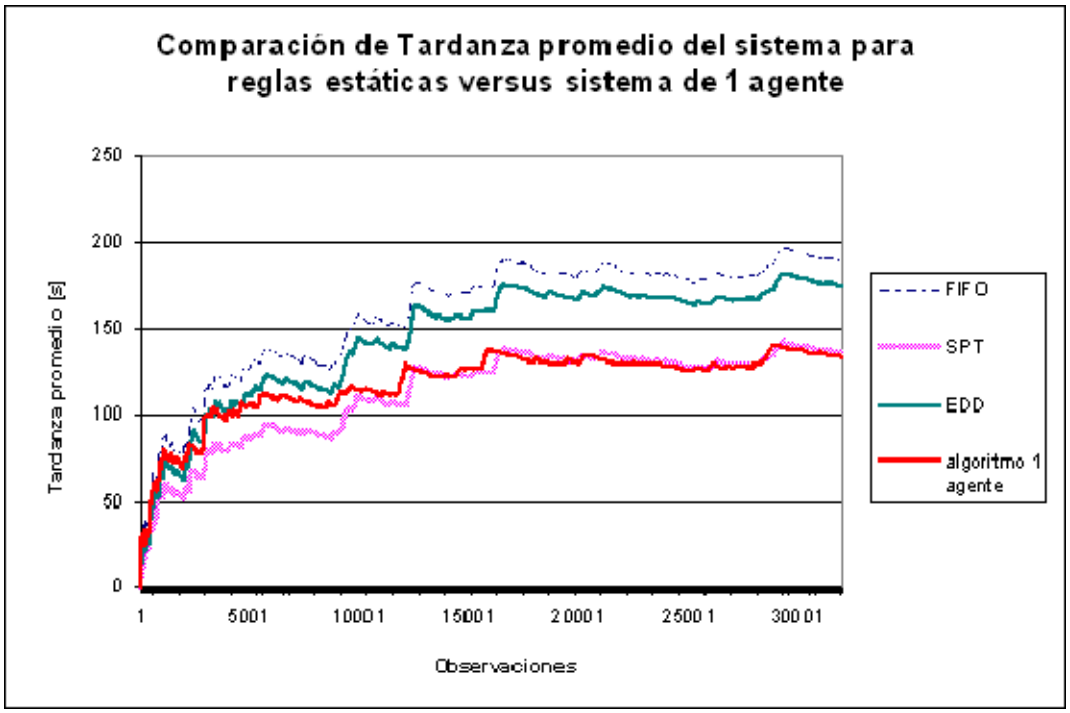
### 6.2.1 Sistema controlado por sólo un agente

El objetivo del primer experimento era simular una celda de manufactura compuesta por tres máquinas, las cuales eran controladas por un agente computacional, quien es el que toma las decisiones en cada punto de decisión y quien interactuaba con el entorno de cada una de las máquinas. La meta del agente era minimizar la tardanza promedio del sistema utilizando el algoritmo Q-learning Difuso propuesto por [18]. El algoritmo seleccionaba la regla de despacho que era utilizada para escoger la siguiente pieza a ser procesada en la máquina que llamó al algoritmo.

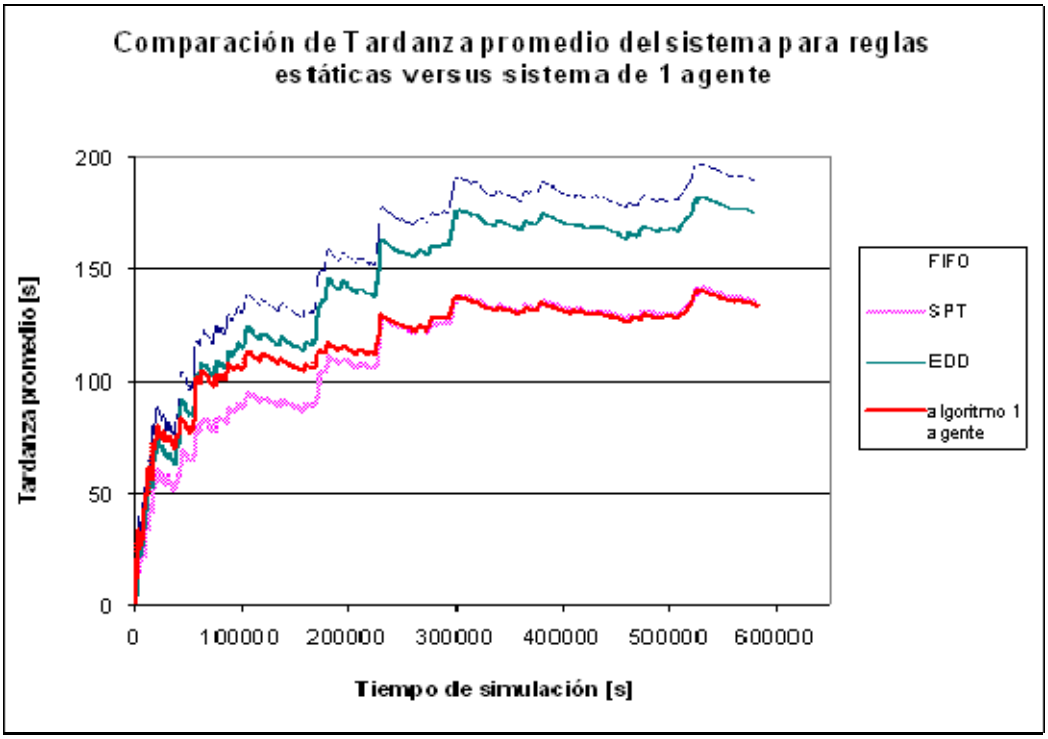
El modelo de simulación para este sistema fue construido utilizando el software Rockwell Arena™. El mismo se presenta en el Apéndice A. En este modelo, el agente mantiene la

información de todo lo que ocurre en el sistema, es decir, el estado del entorno de cada una de las máquinas en todo momento, la regla de despacho aplicada para cada pieza que es seleccionada para procesar, el reforzamiento entregado por el agente que representa la efectividad en la selección de la regla, las rutas de procesamiento de cada una de las piezas, ente otros. Debido a la complejidad del algoritmo Q-learning difuso, a la gran cantidad de tiempo de computadora que toma realizar una sola réplica, a la cantidad de programas involucrados en la ejecución de este modelo y principalmente a que los sistemas son de tipo no terminal, se decidió realizar una sola corrida de simulación de muy larga duración en vez de realizar réplicas. Para comparar la efectividad del algoritmo propuesto por Kanetkar [18], se comparó un enfoque estático con un enfoque dinámico. En un enfoque basado en reglas estáticas, una regla de despacho es usada en una máquina a través de todo el tiempo de simulación.

Por otro lado, en un enfoque dinámico, reglas óptimas están cambiando a lo largo del tiempo. Para este caso, los datos de la tardanza promedio del sistema obtenidos por el algoritmo Q-learning difuso se compararon con los resultados de tardanza promedio obtenidos al utilizar reglas estáticas. Se hizo una corrida de simulación para cada regla estática: FIFO, SPT y EDD. Los resultados obtenidos por las reglas estáticas y por el algoritmo Q-learning difuso se presentan en la Figura 6.1, la cual muestra la tardanza promedio de cada política para cada entidad, las cuales son calculadas por SIMAN en el modelo de simulación mediante un bloque VBA colocado antes del bloque “Dispose”, quien es el encargado de simular el instante en el cual una entidad abandona el sistema.




6.1 (a) Reglas estáticas y algoritmo Q-learning difuso de un agente versus observaciones




6.1 (b) Reglas estáticas y algoritmo Q-learning difuso de un agente versus tiempo de simulación.

**Figura 6. 1**. Tardanza promedio del sistema obtenida al utilizar reglas estáticas y Algoritmo Q-learning Difuso de un agente.

La Figura 6.1 muestra la tardanza promedio del sistema calculada cuando cada entidad sale del sistema, desde el tiempo cero, es decir, cuando el sistema se encuentra vacío. En un comienzo de la simulación, la tardanza promedio de las primeras piezas es igual a cero, pero a medida que las máquinas y las filas se van ocupando, la tardanza promedio del sistema comienza a aumentar. En un comienzo este aumento es considerable y la variabilidad entre los datos también, pero luego, a medida que el sistema se estabiliza, la variación entre datos consecutivos es cada vez menor.

En la figura es posible observar que en un comienzo el algoritmo Q-learning difuso de un agente presenta un comportamiento de la tardanza promedio muy similar a a todas las políticas estáticas. A medida que el algoritmo comienza a aprender, cambiando constantemente sus políticas de despacho y a medida que va utilizando diferentes estrategias de explotación-exploración, va superando en eficiencia al resto de las políticas de despacho, hasta que llega el momento en que la tardanza promedio del sistema que logra el algoritmo está por debajo del la política SPT, que es la que presenta la menor tardanza promedio del sistema entre las políticas estáticas.

Cuando el algoritmo Q-learning difuso de un agente ha llegado a su máximo aprendizaje, es decir, con una estrategia explotación-exploración 100-0%, presenta una tardanza promedio inferior al método que utiliza reglas de despacho estáticas.

Como la simulación comienza cuando el sistema se encuentra vacío y no existe una condición natural de terminar, se definió el sistema como uno de tipo no terminal. En este tipo de sistema, los primeros datos obtenidos de la medida de rendimiento, Tardanza promedio del sistema, pertenecen a las primeras piezas que llegan a un sistema, el cual se encuentra no congestionado, con máquinas y filas que se encuentran vacías. Las primeras piezas llegan al sistema y se moverán rápidamente por el sistema y la medida de rendimiento tenderá a ser a ser baja o nula en la primera parte de la simulación, lo cual generará una influencia en el promedio de la tardanza total haciéndola más baja, lo que hará presumir que la medida de rendimiento promedio es mejor que lo que realmente es. A este período se le conoce como período de transición.

Es necesario eliminar el período de transición para evitar que los resultados estén sesgados. Para ello, se utilizó la técnica “Moving Average” con un parámetro k=1000 para observar visualmente

en una gráfica el momento en el cual la medida de rendimiento comienza a presentar un estado más estable, el cual señala el término del período de transición.

La Figura 6.2 presenta los datos obtenidos al utilizar la técnica moving average a la medida de rendimiento Tardanza promedio del sistema para el caso del sistema de una agente. Como se observa en la figura, el período de transición termina alrededor de la observación 3170 y en el tiempo de simulación 60,020 segundos, donde se truncó la corrida.

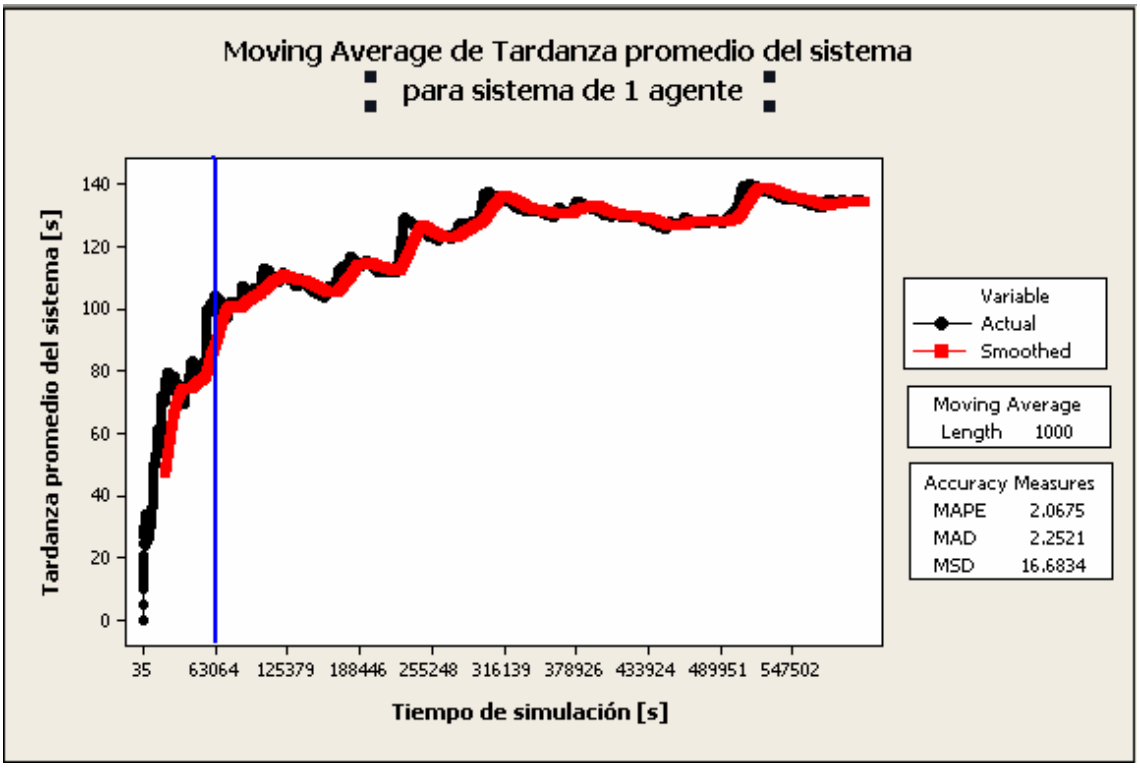
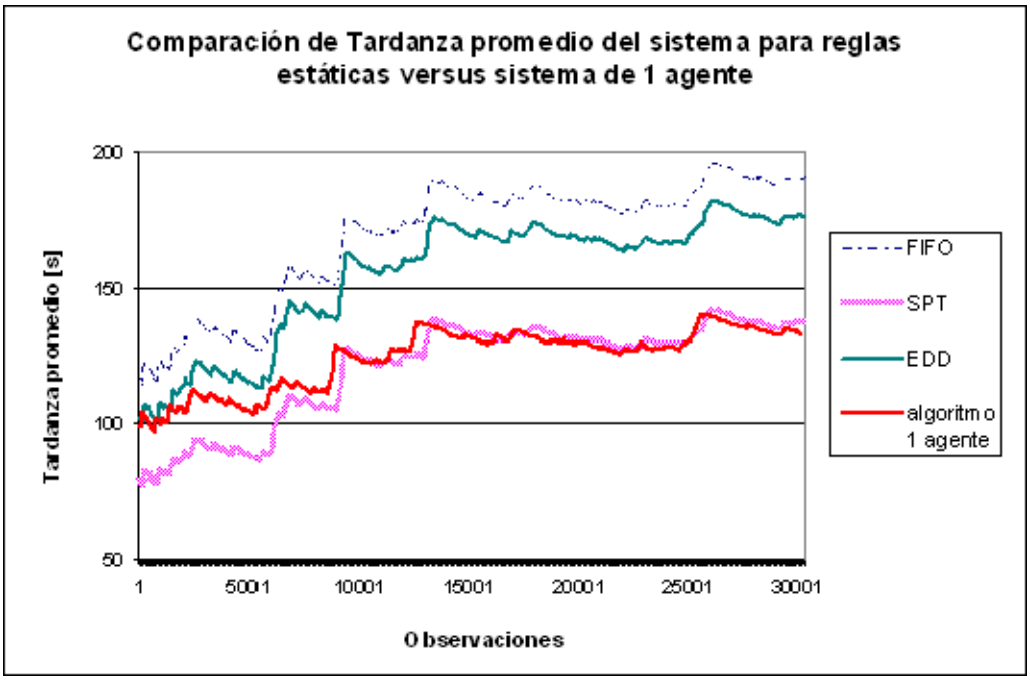
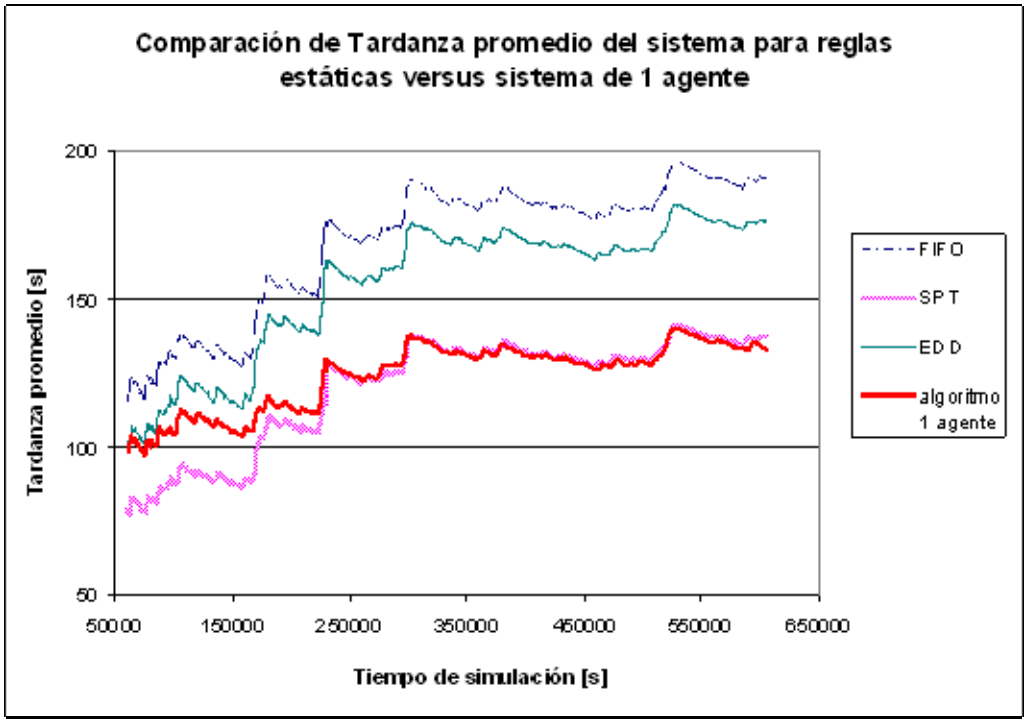



**Figura 6. 2.** Moving Average de Tardanza promedio del sistema para un parámetro k=1000.

Las Figuras 6.3 (a) y 6.3 (b) presentan los resultados del algoritmo Q-learning difuso y de las reglas estáticas FIFO, SPT y EDD por observaciones y a lo largo del tiempo de simulación respectivamente, una vez se ha eliminado el tiempo de transición.




6.3 (a). Reglas estáticas y Algoritmo Q-learning difuso un agente versus Observaciones




6.3 (b). Reglas estáticas y algoritmo Q-learning difuso un agente versus tiempo de simulación

**Figura 6. 3.** Tardanza promedio del sistema calculada utilizando reglas estáticas y algoritmo Q-learning Difuso de un agente.

En las Figuras 6.1 y 6.3 es posible observar que al comienzo de la simulación los datos de tardanza promedio del sistema de la regla estática SPT, presentan la menor tardanza promedio del sistema entre todas las reglas estáticas, son menores que las obtenidas mediante el algoritmo Q-learning difuso. A medida que transcurre el aprendizaje del algoritmo la tardanza promedio del sistema que presenta el algoritmo pasa ha ser similar a la de la regla estática SPT.

Es importante indicar que a través de estas figuras hemos aprendido que si tuvieramos que escoger entre untilizar un agente o utilizar la regla estática de SPT para controlar el sistema, debemos escoger utilizar la regla estática ya que la misma no implica nínguna inversión como lo requiere el agente.

### 6.2.2 Sistema controlado por tres agentes

Este sistema presenta similares características que el sistema controlado por un agente. La diferencia radica en el momento en el cual se realizan los cálculos de tardanza promedio y cómo se asignan las recompensas.

Recordemos que en el sistema controlado por un agente, la tardanza se calcula antes de que la entidad salga del sistema en la estación salida de acuerdo a las fechas de entrega de cada tiempo de pieza, y las recompensas se entregan a la salida de cada máquina, pero de acuerdo a un estimado del tiempo que demoraría la pieza en salir del sistema.

En el sistema controlado por tres agentes, es decir, un agente para cada máquina, se calcula la tardanza promedio por estación, y se entrega una recompensa por estación. Para saber si la pieza está atrasada en una estación en particular, debimos asignar fechas de entrega por máquina como una proporción del tiempo del sistema por máquina para un determinado tipo de pieza. Cuando la pieza ya ha completado su procesamiento en todas las máquinas se dirige a la estación de salida donde se determina finalmente la tardanza promedio del sistema

Es este sistema la celda de manufactura es controlada por tres agentes, donde cada uno de ellos posee sólo la información correspondiente a la estación en donde se encuentra, es decir, los

agentes no comparten información como en el sistema controlado por un agente, por lo cual se esperaría que el tiempo de aprendizaje para cada agente sea aún más lento. . El modelo de simulación para este sistema construido utilizando el programa SIMAN se presenta en el Apéndice B.

En los datos de tardanza promedio del sistema para el sistema controlado por tres agentes también fue necesario eliminar el período de transición, el cual se hizo mediante Moving Average con parámetro k=1000. El gráfico de moving average versus el tiempo de simulación para la tardanza promedio del sistema se presenta en la Figura 6.4.

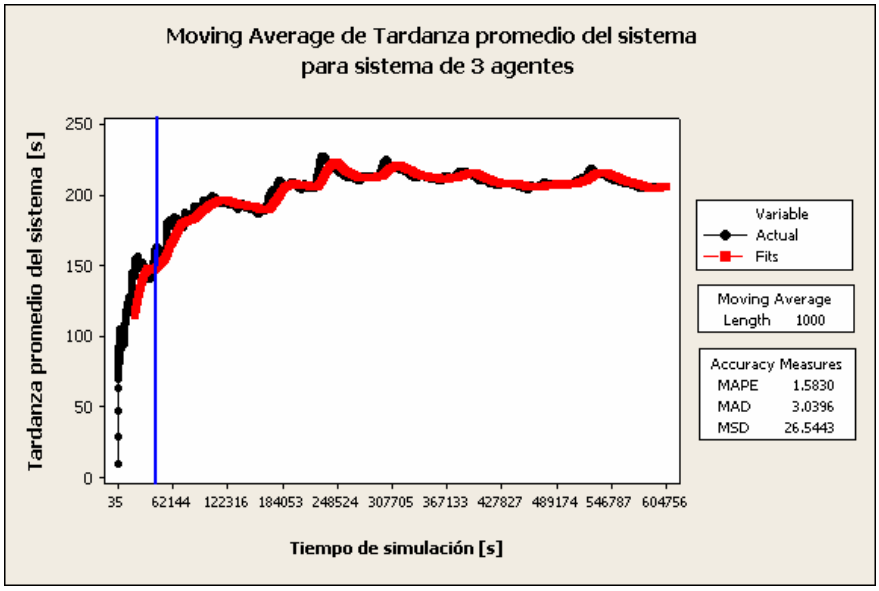
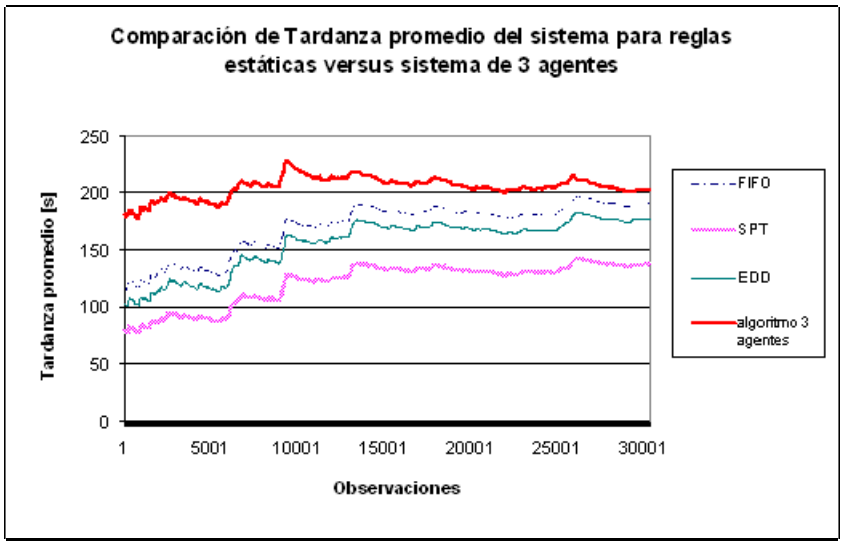



**Figura 6. 4**. Moving Average de Tardanza promedio del sistema para algoritmo de tres agentes para k=1000.

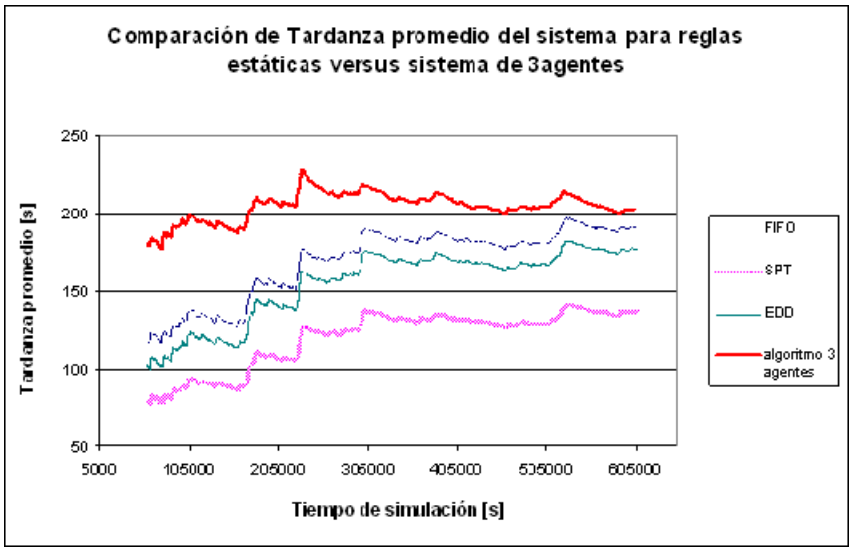
Sin tomar en cuenta la corrida del sistema de un agente, se habría truncado los datos hasta la observación 2525 y en el tiempo de simulación 46,842 [s], pero, como para el caso de un agente se truncaron 3170 datos, también se truncó la misma cantidad en la corrida de datos de este sistema, el cual ocurre a los 58,668.7 segundos.

Para comparar la efectividad del algoritmo propuesto de tres agentes, se comparó un enfoque estático con un enfoque dinámico. Los datos de tardanza promedio del sistema obtenidos por el algoritmo Q-learning difuso de tres agentes, se compararon con los resultados de tardanza

promedio obtenidos al utilizar las reglas estáticas FIFO, SPT y EDD. Los resultados obtenidos para un tiempo de corrida de simulación de 168 horas se presentan en la Figura 6.5 (a) y 6.5(b).




6.5 (a) Reglas estáticas versus algoritmo de tres agentes por observaciones




6.5 (b) Reglas estáticas versus algoritmo de tres agentes por tiempo de simulación

**Figura 6. 5.** Tardanza promedio del sistema utilizando políticas de despacho estáticas y algoritmo Q-learning difuso de tres agentes.

La Figura 6.5 muestra la tardanza promedio del sistema utilizando reglas estátitcas y el algoritmo Q-learning disfuso de tres agentes una vez se ha eliminado el período de transición. Los

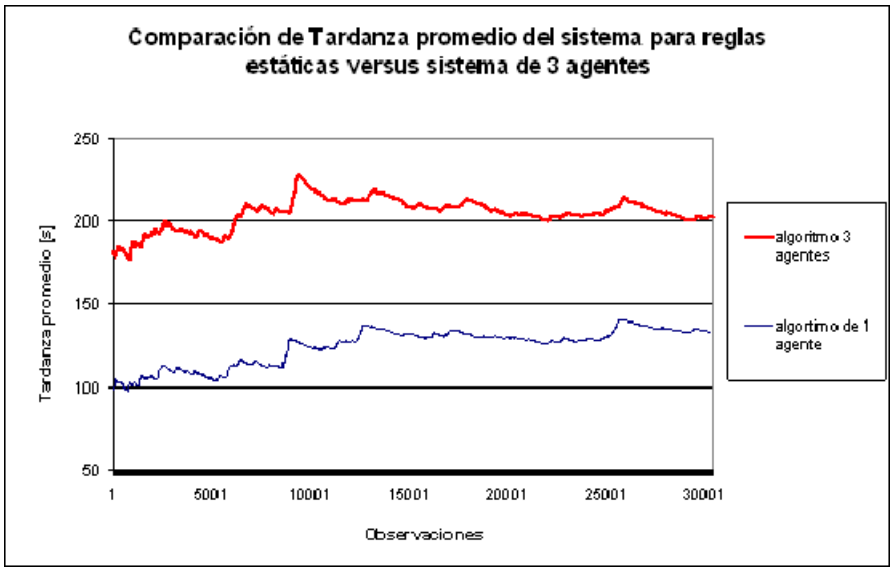
resultados obtenidos del experimento presentado en la figura 6.5 muestran que en un comienzo de la corrida de simulación el algoritmo de tres agentes presenta una tardanza promedio del sistema mucho mayor a las tardanzas promedios que se obtuvieron de la aplicación de las reglas estáticas. A medida que el aprendizaje del algoritmo transcurrió, la tardanza promedio del sistema siempre fue mayor a la tardanza promedio de las reglas estáticas.

Es importante indicar que a través de estas figuras hemos aprendido que si tuvieramos que escoger entre untilizar un agente para cada estación de trabajo versus utilizar la regla estática de SPT para controlar el sistema, debemos escoger utilizar la regla estática ya que la misma no implica nínguna inversión como lo requieren los agentes.

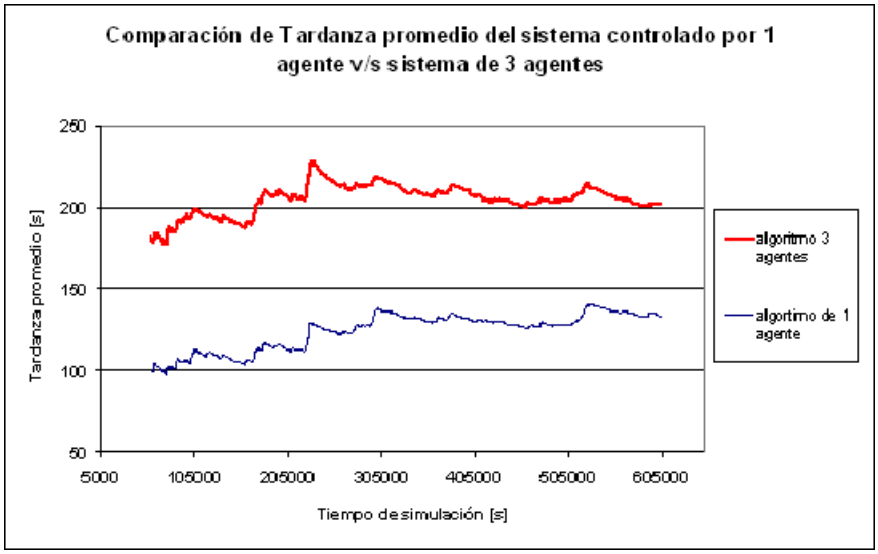
### 6.2.3 Comparación de sistema de resultados obtenidos con Algoritmo Q-learning Difuso de un agente con Q-learning Difuso de tres agentes

Una vez se ha mostrado el comportamiento del algoritmo Q-learning difuso de tres agentes en comparación a la utilización de reglas estáticas, es posible ahora comparar los algoritmos de un agente y tres agentes para evaluar si existe diferencia entre ellos.

Las Figuras 6.6 (a) y 6.6 (b) muestran la tardanza promedio del sistema obtenida al utilizar el algoritmo Q-learning difuso de un agente para selección de políticas de despacho versus el algoritmo Q-learning difuso de tres agentes. La figura 6.6 (a) presenta las tardanzas promedio del sistema versus las observaciones que generaron el cálculo de la tardanza y la figura 6.6 (b) presentan las tardanzas promedio del sistema versus el tiempo de simulación en el cual fue calculada la estadística. En las figuras se presentan los datos truncados, es decir, ya se le ha eliminado la etapa de transición.




6.6 a) Tardanza promedio del sistema de un agente y 3 agentes por observaciones




6.6 (b) Tardanza promedio del sistema de un agente y 3 agentes por tiempo de simulación.

**Figura 6. 6.** Comparación de Tardanza promedio obtenida al utilizar algoritmo Q-learning difuso de un agente versus de tres agentes.

En las Figuras 6.6 (a) y 6.6 (b) se puede observar que existe una gran diferencia entre los valores obtenido por ambos algoritmos. A los largo del tiempo de simulación, la tardanze promedio para tres agentes es mayor a la tardanza promedio para un agente. Lo que nos indica que de utilizar agentes para controlar el sistema, el número de agentes a utilizar debe ser uno.

## *6.3 Análisis estadístico*

Como el propósito del estudio es comparar el rendimiento del sistema controlado por un agente y el sistema controlado por tres agentes, se utilizó técnicas estadísticas además de las gráficas de tendencia presentadas anteriormente. Una de las técnicas a utilizar es construir un intervalo de confianza para cada sistema y finalmente realizar un análisis estadístico t-pareado para construir un intervalo de confianza para la diferencia entre las observaciones de cada sistema. Otra de las técnicas a utilizar es la comparación de los datos individuales pertenencientes a las percentilas del 95, 90, 85, ….., hasta la del 5.

### *6.3.1 Intervalos de confianza*

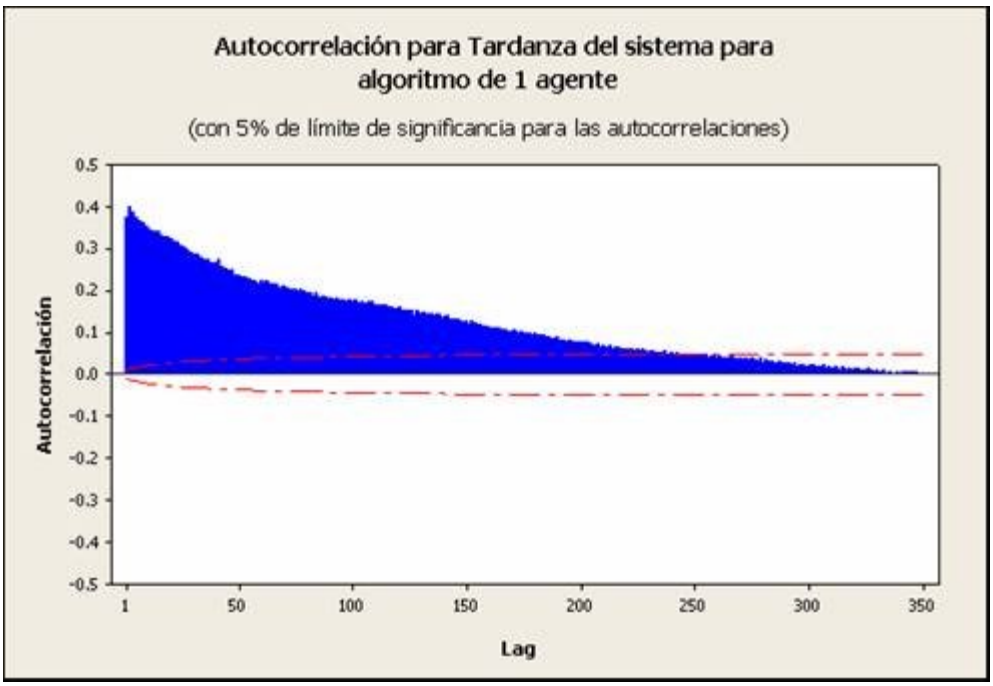
Como se mencionó anteriormente, los sistemas son de tipo no terminales, lo cual nos presenta dos grandes problemas al momento de realizar el análisis de los datos. Primero, presentan el problema de la etapa de transición, por lo que se debe eliminar la etapa inicial para poder estudiar el comportamiento de la variable bajo estudio en un estado relativamente estable, lo cual ya se hizo en la sección anterior. Segundo, este tipo de sistemas presentan el problema de tener datos altamente correlacionados, lo cuál complica al momento de realizar un intervalo de confianza, donde el requisito es que los datos deben ser independientes y además presentar comportamiento cercano a la distribución normal.

Para construir intervalos de confianza de sistemas de tipo no terminales existen diversas técnicas, entre las cuales se encuentras: réplicas, suma de covarianzas y observaciones en lotes. En esta investigación se utiliza el método de lotes, propuesta por Welch [37], debido a que es el método más práctico para interpretar los resultados. En vez de utilizar réplicas de simulación, se hacen corridas de simulación muy muy largas y luego se divide el grupo de observaciones en sub-secuencias de datos. La secuencia de datos obtenidos para la tardanza promedio del sistema utilizando cada algoritmo será dividido en grupos, con el objetivo de que la media de cada lote sea independiente de la media del siguiente grupo y se logre eliminar la correlación de los datos. Cada grupo de datos será tratado como si fuese una réplica independiente del sistema.

Para utilizar esta técnica es necesario eliminar primero la etapa de transición mediante la técnica de “Moving Average”. Posteriormente, se divide la cantidad de observaciones en lotes de igual tamaño. De esta forma, la media de dos lotes adyacentes será aproximadamente independiente, incluso aunque la última observación del lote j esté correlacionada con la primera observación del lote j+1 [26].

El tamaño adecuado de lote se determina mediante el análisis de autocorrelación, el cual grafica un correlograma. El tamaño de lote adecuado será 10 veces el lag que presente una correlación lo más cercana a cero, y se encuentre dentro de los límites de significancia especificada en el análisis, generalmente 5% [27]. Se construyó un gráfico de correlación de la tardanza promedio del sistema para el sistema controlado por el algoritmo Q-learning difuso de un agente y otro para el de tres agentes.

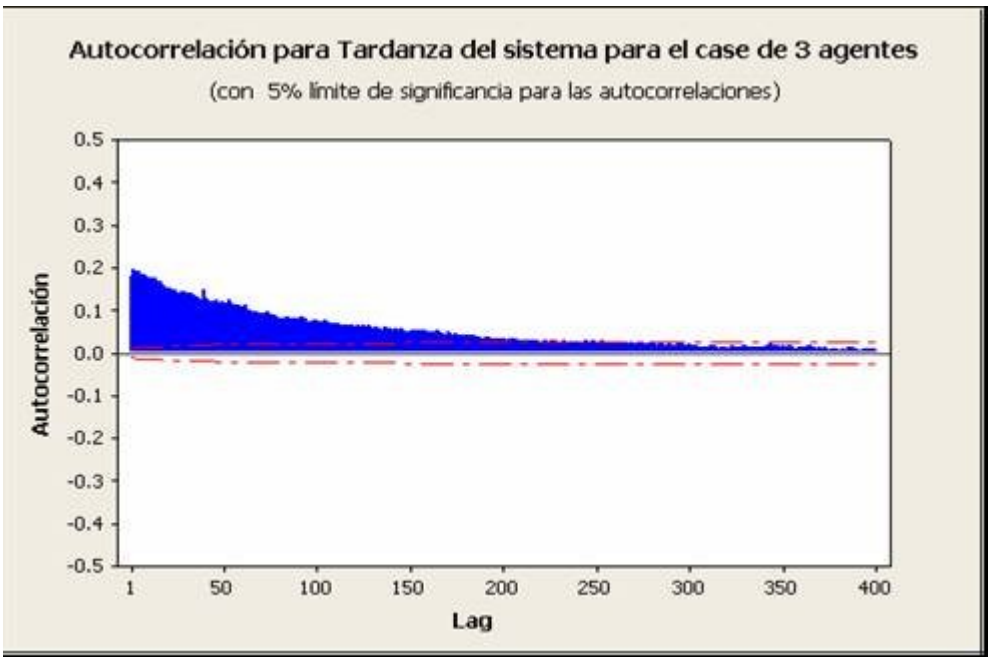
En el caso del algoritmo de un agente, se seleccionó un máximo número de lags igual a 350. La Figura 6.7 presenta el gráfico de correlación para el algoritmo de un agente para un máximo de 350 lags, el cual fue generado por Minitab. Se puede observar que la correlación es relativamente baja a partir del lag 260, a partir de los cuales, la correlación se encuentra dentro de los límites de significancia representados por la línea discontinua. Para obtener un mejor resultado en cuanto a datos menos correlacionados, se seleccionó una correlación igual a la mitad del valor donde el límite de significancia era igual a la correlación, es decir, 0.025. En la gráfica se consigue una correlación cercana a 0.025 entre los lags 290 y 291. Por lo tanto, se seleccionó el lag 290, donde un razonable tamaño de lote mínimo es 10 veces el número total de lags, es decir, lotes de tamaño 2900 observaciones.




| Lag | Correl |
|---|---|
| 1 | 0.372850 |
| 2 | 0.398688 |
| 3 | 0.387679 |
| . | . |
| . | . |
| 285 | 0.029052 |
| 286 | 0.028665 |
| 287 | 0.032310 |
| 288 | 0.028052 |
| 289 | 0.025983 |
| 290 | 0.027890 |
| 291 | 0.021977 |
| 292 | 0.026342 |
| 293 | 0.022594 |
| 294 | 0.027959 |
| 295 | 0.020338 |
| . | . |
| . | . |
| 348 | 0.002572 |
| 349 | 0.003414 |
| 350 | 0.001112 |

**Figura 6. 7.** Autocorrelación de tardanza promedio del sistema para sistema controlado por algoritmo Q-learning difuso de un agente.

Para el caso del algoritmo de tres agentes, también se realizó un análisis de autocorrelación, el cuál se presenta en la Figura 6.8. En la figura se puede observar que la correlación es relativamente baja a partir del lag 270, a partir de los cuales, la correlación se encuentra dentro de los límites de significancia. La correlación de aproximadamente 0.025 se presenta entre los lags 219 y 220. Se seleccionó una cantidad de lags de 2200, con lo cual, un número razonable de observaciones por lote sería de 10 veces la cantidad de lags, lo que implica 2200 observaciones por lote. Dado que para el sistema de un agente se pueden formar 10 grupos de 2900 observaciones y uno más con las obseraciones restantes, y como en el caso de tres agentes se cuenta con alrededor de 30,000 datos, entonces se utilizará una cantidad de datos de 2900 por lote, para construir intervalos de confianza con la misma cantidad de datos. Hay que notar que mientras más cantidad de datos se incluyan menor será la correlación de los datos, así que si se utiliza un lag mayor la correlación entre los lotes será menor.




| Lag | Correl. |
|---|---|
| 1 | 0.180376 |
| 2 | 0.195708 |
| 3 | 0.190923 |
| . | . |
| . | . |
| 216 | 0.030530 |
| 217 | 0.028408 |
| 218 | 0.029634 |
| 219 | 0.023356 |
| 220 | 0.024875 |
| 221 | 0.025046 |
| 222 | 0.030639 |
| 223 | 0.030707 |
| 224 | 0.023450 |
| 225 | 0.021433 |
| . | . |
| . | . |
| 398 | 0.005611 |
| 399 | 0.009565 |
| 400 | 0.006249 |

**Figura 6. 8.** Autocorrelación de tardanza promedio del sistema para sistema controlado por algoritmo Q-learning difuso de tres agentes.

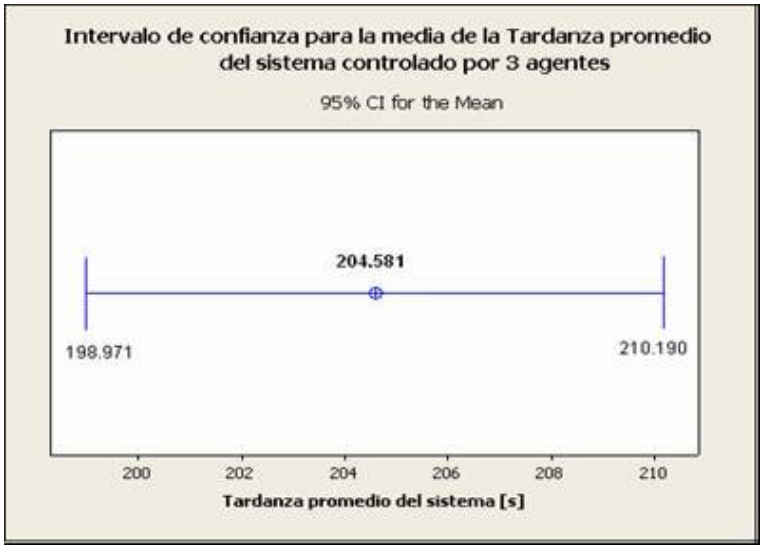
Ya habiendo determinado el tamaño de lote para cada sistema, se calcula el promedio de cada lote, los cuales se presentan en la Tabla 6.2.

**Tabla 6. 2.** Datos de tardanza promedio del sistema por lote

| 1 AGENTE | | 3 AGENTES | |
|---|---|---|---|
| **lote** | **promedio** | **lote** | **promedio** |
| 1 | 104.71 | 1 | 188.4245911 |
| 2 | 107.56 | 2 | 192.1917314 |
| 3 | 113.57 | 3 | 204.8401245 |
| 4 | 124.71 | 4 | 216.3354241 |
| 5 | 132.73 | 5 | 214.3483469 |
| 6 | 131.56 | 6 | 208.7811465 |
| 7 | 130.82 | 7 | 208.189383 |
| 8 | 127.85 | 8 | 202.9237971 |
| 9 | 131.44 | 9 | 206.1783176 |
| 10 | 136.00 | 10 | 206.5964749 |
| 11 | 133.82 | 11 | 201.5792375 |

Dado que los promedios de los lotes son independientes (concluído a través del los correlogramas) y distribuídos normalmente (por el Teorema del Límite Central), se procede a construir los intervalos de confianza del promedio de la tardanza promedio del sistema para cada uno de los sistemas. Los intervalos de confianza para la media de la tardanza promedio del sistema para los algoritmos de un agente y tres agentes se presentan en la Figura 6.9 (a) y 6.9 (b).


6.9 (a). Sistema controlado por un agente




6.9 (b). Sistema controlado por tres agentes

**Figura 6. 9.** Intervalo de confianza para tardanza promedio del sistema para sistema de un agente y tres agentes.

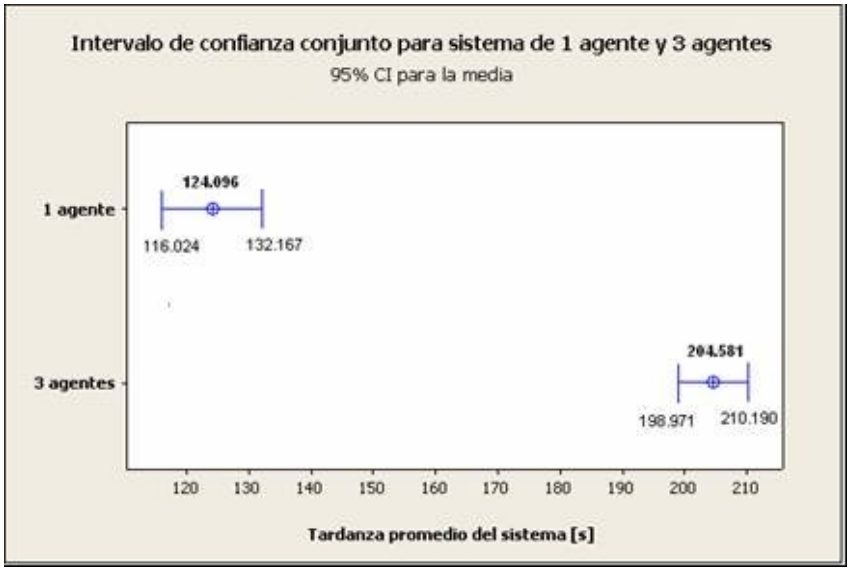



**Figura 6. 10.** Intervalos de confianza para sistemas controlados por un agente y tres agentes.

Habiendo ya construido los intervalos de confianza para cada sistema, se realiza una prueba pareada t para encontrar el intervalo de confianza de la diferencia entre las medias de los dos sistemas. Aunque muchos otros métodos pueden ser usados para comparar dos sistemas, esta técnica no debe asumir que las varianzas de los sistemas son iguales. Sólo se debe cumplir con que el grupo de observaciones sean independientes, tanto para el caso del sistema controlado por un agente como para el sistema de tres agentes. La Prueba Pareada se realizó en Minitab y los resultados se presentan a continuación:

Hipótesis

- $H_0$ : $\mu_1 - \mu_2 = 0$
- $H_1$ : $\mu_1 - \mu_2 \neq 0$

**Paired T-Test and CI: Lotes_1_agente, Lotes_3_agentes**

```
                 N      Mean   StDev  SE Mean
Lotes_1_agente  11   124.979  11.098    3.346
Lotes_3_agentes  11   204.581   8.350    2.518
Difference       11  -79.6017  7.7663   2.3416
95% CI for mean difference: (-84.8191, -74.3842)
T-Test of mean difference = 0 (vs not = 0): T-Value = -33.99  P-Value =
0.000
```

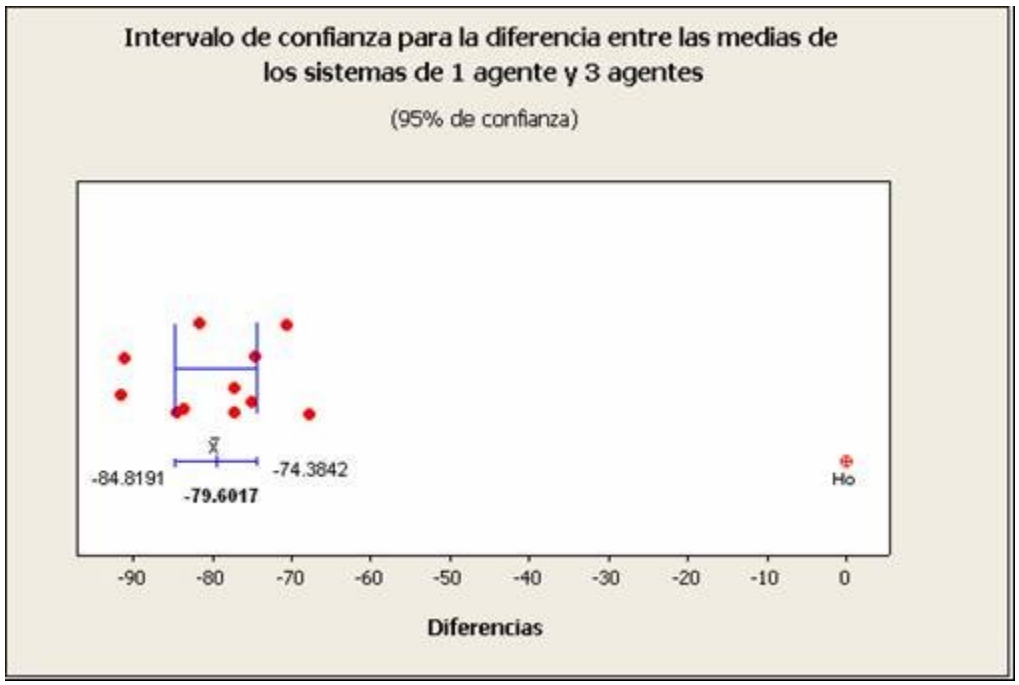



**Figura 6. 11.** Intervalo de confianza para la diferencia entre las medias de los sistemas controlados por un agente y tres agentes.

La Figura 6.11 presenta el intervalo de confianza para la diferencia entre las medias de la tardanza promedio del sistema para el algoritmo de un agente y el de tres agentes. Al analizar la figura y el análisis estadístico se puede deducir que existe suficiente evidencia estadística para rechazar Ho, la hipótesis nula, la cual establece que las medias son iguales, debido que Ho cae fuera del intervalo de confianza de las diferencias, además que el P-value = 0.

Luego de evaluar los intervalos de confianza, los resultados de la Prueba Pareada y el Intervalo de confianza de la diferencia entre ambos sistemas, se concluye que los sistemas de un agente y tres agentes no son iguales y que presentan diferencias significativas en cuanto a la tardanza promedio del sistema.

### *6.3.2 Método de percentilas*

Luego de realizar el análisis para lotes de observaciones utilizando intervalos de confianza, se quiso complementar el análisis con una comparación de datos individuales, para la cual se utilizó el método de percentilas. Para ello se ordenaron los datos de tardanza promedio del sistema para

los sistemas de un agente y tres agentes y se calculó las percentilas desde el 95 al 5. La tabla presenta los datos indivuales de cada uno de los sistemas para cada percentila.

**Tabla 6. 3.** Datos de tardanza promedio del sistema para distintas percentilas de los sistemas de uno y tres agentes.

| Percentila | Un agente | Tres Agentes |
|---|---|---|
| 95 | 75.28 | 148.83 |
| 90 | 99.7 | 179.44 |
| 85 | 104.95 | 189.31 |
| 80 | 107.07 | 192.37 |
| 75 | 110.82 | 195.14 |
| 70 | 112.78 | 200.79 |
| 65 | 116.12 | 202.2 |
| 60 | 124.22 | 203.16 |
| 55 | 127.18 | 203.86 |
| 50 | 127.78 | 204.49 |
| 45 | 128.3 | 205.5 |
| 40 | 129.38 | 206.58 |
| 35 | 130.15 | 207.6 |
| 30 | 130.92 | 208.6 |
| 25 | 131.92 | 209.56 |
| 20 | 133.12 | 211.03 |
| 15 | 134.31 | 212.55 |
| 10 | 135.49 | 213.45 |
| 5 | 137.05 | 216.29 |

En la tabla 6.3 es posible obsevar que los datos de tardanza promedio del sistema para el sistema controlado por un agente presentan una gran difrenencia con respecto a los datos de tardanza promedio del sistema presentados por el sistema de tres agentes en cada una de las percentilas. De estos datos es posible concluir que para el tiempo de simulación utilizado, el sistema controlado por un agente es muy superior en rendimiento que el sistema controlado por tres agentes, dado que presenta tardanzas promedio del sistema bastante menores.

## *6.4 Resumen*

En este capítulo se presentaron los resultados de tardanza promedio del sistema, los cuales se obtuvieron mediante la simulación de los sistemas de manufactura controlados por un agente y tres agentes.

Se presentaron gráficos de tendencia de comportamiento para los dos sistemas con respecto al comportamiento de un sistema que utiliza políticas de despacho estáticas, para luego comparar el rendimiento sólo de los sistemas de un agente y tres agentes.

Se presentaron también dos análisis de resultados, uno mediante construcción de intervalos de confianza y prueba pareada t para determinar si existe diferencia significativa en la utilización de los sistemas de un agente y tres agentes. También se realizó un análisis de percentilas para comparar los datos de tardanza promedio del sistema en forma individual para cada percentila.

En el siguiente capítulo se presentarán los hallazgos encontrados, las conclusiones de la investigación y se entregarán algunas recomendaciones.

# 7. CONCLUSIONES

## *7.1 Introducción*

Este capítulo contiene las conclusiones, los hallazgos encontrados y las recomendaciones relacionadas a la ejecución de este proyecto. Las conclusiones están basadas en los objetivos que se definieron, los hallazgos están relacionados sólo al caso de estudio bajo estudio y las recomendaciones están relacionadas a oportunidades futuras de investigación.

## *7.2 Hallazgos*

A partir de la información obtenida de las corridas de simulación para el caso de estudio fue posible aprender que utilizar un agente para controlar todo el sistema presenta tardanzas promedio del sistema menores a las que presenta un sistema utilizando la regla estática, SPT. Esta información nos reveló que no se justifica invertir en un controlador para tomar decisiones ya que utilizando la política SPT, se puede alcanzar la menorr tardanza promedio.

Hay que mencionar que la medida de rendimiento bajo estudio es la tardanza promedio del sistema y no la tardanza de individual de cada pieza, lo cual significa que se obtiene una tardanza promedio cada vez que sale una pieza del sistema (esto es el promedio de todas las piezas que han salido hasta el momento del cálculo). Sería necesario evaluar los sistemas mediante la tardanza del sistema pieza a pieza y no como promedio para determinar si es conveniente invertir en la utilización de un agente computacional.

Para el caso del sistema controlado por tres agentes, para el tiempo de simulación utilizado, la tardanza promedio del sistema siempre es mayor a la tardanza promedio para las reglas estáticas, además es mayor que la tardanza promedio para el caso del sistema controlado por un agente. El algoritmo de un agente mantiene constantemente la información de lo que ocurre en cada una de las máquinas, las variables de entrada de cada una de las máquinas, las recompensas entregadas,

la contribución a la disminución en la tardanza promedio, los pesos actualizados de las reglas difusas activadas en cada punto de decisión, las políticas de despacho entregadas en cada máquina, entre otros. Por tal razón, la información de todas las máquinas contribuye a que el agente reuna más información y que tome más decisiones en la misma cantidad de tiempo que le toma al algoritmo de tres agentes. Aquí se encontró que en el algoritmo de un agente se comparte información, lo cual contribuye a un aprendizaje más rápido en términos de tiempo de simulación y que en el caso del sistema controlado por tres agentes, cada agente aprende de la experiencia de su propia máquina, lo que dilata su aprendizaje.

## *7.3 Conclusiones*

Las conclusiones son presentadas en términos de aplicación práctica de tiempo simulación en computadora para los sistemas bajo estudio.

Para el caso de estudio, concluímos que la mejor alternativa para controlar el sistema es la regla estática, SPT. La utilización de esta regla resultó en la menor tardanza promedio.

Mediante las gráficas y los análisis estadísticos de los resultados obtenidos del capítulo 6 concluímos que para el tiempo de simulación utilizado, existe diferencia significativa entre controlar un sistema de manufactura mediante un agente y tres agentes que utilizan las técnicas de reforzamiento del aprendizaje y lógica difusa. Como en la investigación se trabaja un caso de estudio, es difícil extrapolar las conclusiones a un caso más general y concluir que siempre será mejor utilizar un agente en vez de tres agentes. Solo es posible concluir que para este caso es mejor utilizar un agente en vez de tres para controlar un sistema, debido a que con él se obtiene una mayor eficiencia en la disminución de la tardanza promedio del sistema.

## *7.4 Recomendaciones*

En esta investigación de realizó un caso de estudio y los hallazgos y conclusiones aplican solo para este caso. Es por esta razon que sería recomedable ampliar este estudio y realizar uno que

permita concluir de modo general si el utilizar un agente para controlar la toma de decisiones de las estaciones de trabajo de un sistema de manufactura realmente es más eficiente que el utilizar un agente por estación de trabajo dentro del sistema. En este estudio se podría:

- incorporar un análisis de sensibilidad para estudiar si variando la cantidad de máquinas, la razón de llegada de las piezas al sistema y los tiempos de servicio de las máquinas se obtiene nuevamente que utilizar un agente resulta en tardanzas promedio del sistema menores que las de tres agentes;
- utilizar diferentes técnicas para determinar la estrategia de explotación-exploración a utilizar y evaluar si los resultados cambian al cambiar la estrategia;
- utilizar diferentes formas, cantidad y rangos de etiquetas lingüísticas, distintos número de máquinas, distinta medida de rendimiento, otra técnica de reforzamiento del aprendizaje o combinación de técnicas, entre otras; y, finalmente
- utilizar otras variables de entrada y de salida o mayor número de ellas.

Sería recomendable también estudiar diferentes maneras de asignar recompensas, de modo tal de asegurarse que la recompensa que se entrega en una máquina se deba sólo a la política que se realizó en la estación que se le entregó y no al resultado en conjunto de la política recién realizada más las acciones realizadas en las máquinas anteriores.

Se recomienda también estudiar la posibilidad de utilizar otro tipo de programa de simulación que no sea Arena™ debido a la limitación que ofrecec este programa en cuanto a la licencia, como también utilizar otro programa para construir rl algoritmo Q-learning difuso que no sea Matlab™, el cuál consume mucho espacio de memoria computacional.

Finalmente, se recomienda utilizar computadoras con más de un procesador o microcomputadoras y una gran capacidad de memoria para poder realizar corridas de simulación más largas en menor cantidad de tiempo.

# REFERENCIAS


1. V.A. Armentano and D.S.Yamashita, "Tabu search for scheduling on identical parallel machines to minimize mean tardiness," *Journal of Intelligent Manufacturing,* Vol 11, pp. 453-460, 2000.
2. A.G. Barto, "Reinforcement Learning," University of Massachusetts, 2003.
3. Y.B. Cambolat and E. Gundogar.. "Fuzzy priority rule for job shop scheduling," *Journal of Intelligent Manufacturing*, Vol 15, pp. 527-533, 2004.
4. H. Cho and R.A. Wysk, "Intelligent workstation controller for computer integrated manufacturing: Problems and models," *Journal of Manufacturing Systems,Vol* 14, (4), pp. 252-263, 1995.
5. H. Cho and R.A. Wysk , "A robust adaptative scheduler for an intelligent workstation controller," *International journal of production research*, Vol 31 (4), pp. 771-789, 1993.
6. P. Cichosz, "Truncating Temporal Difference: On the Efficient Implementation of TD(λ) for Reinforcement learning," *Journal of Artificial Inteligente Research, Vol 2*, pp. 287-318, 1995.
7. R. Crites and A. Barto. "Improving Elevator Performance Using Reinforcement Learning," *Proc. 1996, Advances in Neural Information Processing Systems Conf, 1996.*
8. Y. Duan, and X. Xu, "Fuzzy Reinforcement Learning and its Application in Robot Navigation". *Proc. 2005 Machine Learning and Cybernetics.* Fourth International Conference, Guangzho. Institute of Artificial Intelligence and Robotics, Northeastern University, Shenyang, China.
9. Ghahramani, Z. (2004). Unsupervised Learning. University College London, UK.
10. R. Girshick. (2002) "Topics in Machine Learning: Backgammon Players". Available: (http://www.cs.brandeis.edu/~cs113/classprojects/~rossgir/cs113/backgammon.html).
11. W. González, (2004). "Fuzzy Logic, Fuzzy Sets". Available: http://www.answermath.com/Panels/fuzzy/esp-fuzzy1.htm
12. Gu, D., Hu, H., "Reinforcement learning of fuzzy logic controller for quadruped walking robots," *Proc. 2002 15th IFAC World Congr.*, Barcelona, 21-26 July.

13. D. Gu, H. Hu and L. Spacek. "Learning Fuzzy Logic Controller for Reactive Robot Behaviours". Department of Computer Science, University of Essex. Wivenhoe Park, Colchester, UK, 2002.
14. M. Harmon and S. Harmon, "Reinforcement Learning: A tutorial", 2000.
15. J. Hong and V. Prabhu, "Distributed Reinforcement Learning Control for Batch Sequencing and Sizing in Just-In-Time Manufacturing Systems". *Applied Intelligence*, Vol 20, pp.71–87.
16. L. Jouffe, "Fuzzy Inference System Learning by Reinforcement Methods". *IEEE Transactions on systems, Man, And Cybernetics—Part C: Applications And Reviews*, Vol. 28, (3), 1998.
17. L.P Kaelbling, M.L. Littman, and A.W. Moore, "Reinforcement Learning: A survey". *Journal of Artificial Intelligence Research* Vol 4, (1996).
18. R. Kanetkar, "Fuzzy Logic Based Reinforcement Learning Aprroach for Dynamic Job Shop Scheduling". M.S. Thesis. Dept. Syst. and Ind. Eng., University of Arizona, Arizona, 2003 .
19. W.D Kelton, R. Sadowski and D. Sturrock. *Simulation with Arena*. MacGraw-Hill Professional, 2003.
20. S. Kim, J. Woo, S. Park, B. Jung and H. Cho. "Making simulation relevant in business: integrated development of nonlinear process planning and simulation-based shop floor control" in *Winter Simulation Conf., 2002,* SESSION: Business process reengineering. pp. 1465 – 1468.
21. R.M. Kretchmar (2003). "Reinforcement learning algorithms for homogenous multi-agent systems". *Workshop on Agent and Swarm Programming,* WASP, 2003.
22. J. Lenz. (2000), "Reinforcement Learning and the Temporal Difference Algorithm". Available: (http://www.tdx.cesca.es/tesis_upc/available/tdx-0207105- 105056//04Rpp04de11.pdf).
23. J.A. López. (2001). "Lógica Difusa". Available: http://members.tripod.com/jesus_alfonso_lopez/FuzzyIntro.html.
24. J.M. Mendel and R.W. McLaren. "Reinforcement learning control and pattern recognition systems". In J. M. Mendel, & K. S. Fu (Eds.), Adaptive, learning and pattern recognition systems: Theory and applications, pp. 287-318. New York: Academic Press.
25. M. Minsky, "Neural Nets and the Brain Model Problem," Ph.D. dissertation, Dept. Math., Princeton University, Princeton, NJ. , 1954.

26. D. Pegden, R. Shannon and R. Sadowski, “Introduction to Simulation Using SIMAN”. MacGraw-Hill, Inc., 1991.
27. Schmeiser, B. “Batch Size Effects in the Analysis of Simulation Output”. *Operations Research*, Vol 30, pp. 556-568, 1982.
28. R. Scott, “The importance of Manufacturing: Key to recovery in the state and the nation”. *Economic Policy Institute*. EPI Briefing Paper. Briefing Paper # 221, 2008.
29. W. Shen, D.H. Norrie, D.H. and J-PA. Barthès, “Multi-agent systems for concurrent intelligent design and manufacturing”. London: Taylor & Francis, 2000.
30. J. Shin and H. Cho*,* **“**Rapid development of a distributed shop floor control system from an XML model-based control software specification”. *International Journal of Production Research,* Vol 44, (2) pp. 329, 2006.
31. M. Shin and M. Jung,. **“**Bid generation and evaluation for MANPro-based real time scheduling”. *International Journal of Production Research, Vol* 43, (18), pp. 3821, 2005.
32. R.S. Sutton and A.G. “Reinforcement Learning: An Introduction”. MIT Press, Cambridge, Masachussets, London, England, 1998.
33. A.M. Turing, “Computing Machinery and Intelligence,” Mind, Vol 59, pp. 433-460, 1950. Available: http://www.loebner.net/Prizef/TuringArticle.html.
34. Y. Wang and J. Usher. “Application of reinforcement learning for agent-based production scheduling”, *Engineering Applications of Artificial Intelligence,* 18, pp. 73–82, 2005.
35. Y. Wang and J. Usher, “Learning policies for single machine job dispatching,” *Robotics and Computer-Integrated Manufacturing,* Vol 20, pgs. 553–562, (2004).
36. C. J. C. H. Watkins, “Learning from delayed rewards*”,* Ph.D. thesis, Cambridge University, Cambridge, England, 1989.
37. P.D. Welch. “The Statistical Analysis of Simulation Results”. *Computer performance Modeling Handbook*, edited by S.S. Laverberg, Academic Press, 1983.
38. G.R. Yi, J. Shin, H.Cho and K.J. Kim, “Quality-oriented shop floor control system for large-scale manufacturing processes: Functional framework and experimental results”. *Journal of Manufacturing Systems*, Vol 21, (3), pp.187-199, 2002.
39. Zadeh, L., Klir, G., Yuan, B. (1996). *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh*. World Scientific Publishing Company.

# APÉNDICE A.

# Modelo de Arena™ para sistema de manufactura controlado por un agente que utiliza Algoritmo Q-learning difuso.

## *A.1. Elementos del experimento de Arena™*

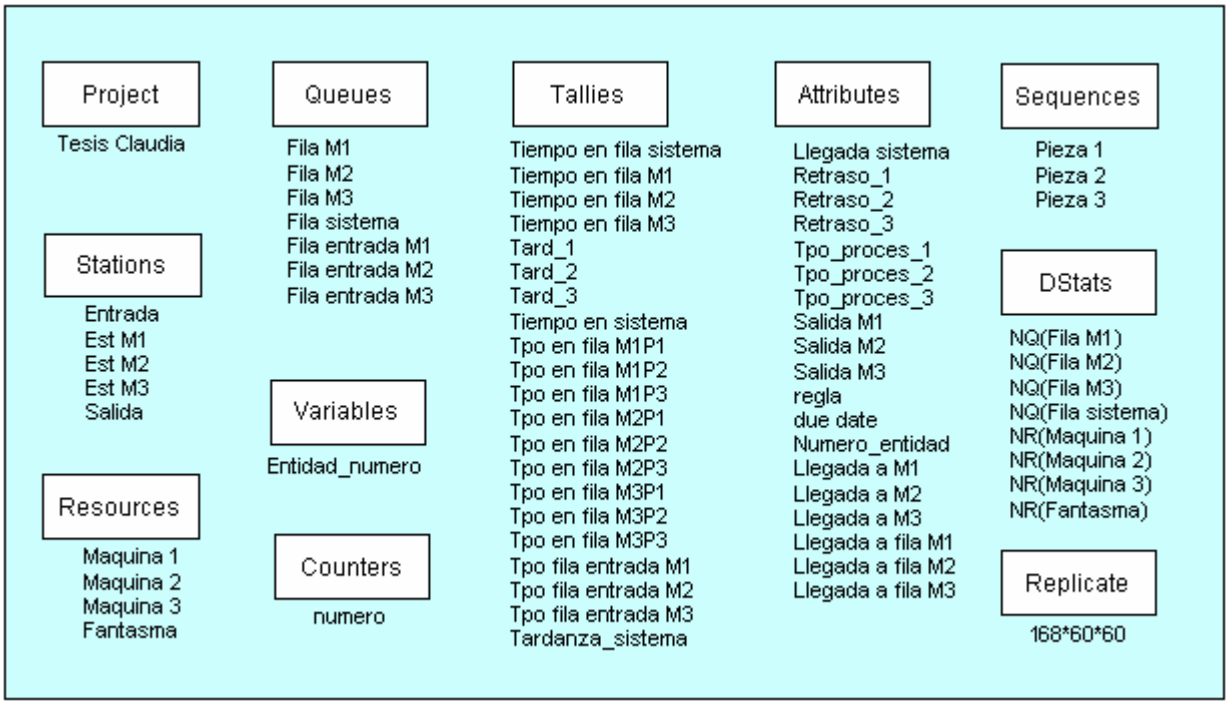



**Figura A. 1**. Experimento del Modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

## *A.2. Estación Entrada*

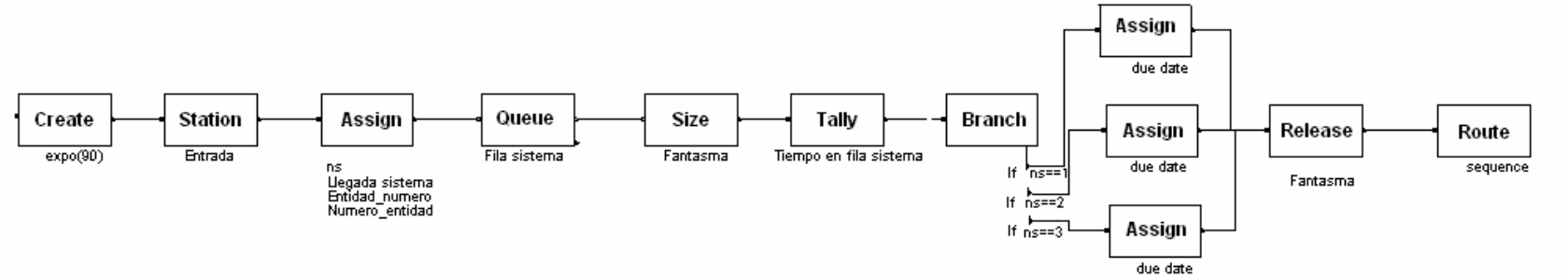



**Figura A. 2.** Estación entrada del modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

## *A.3. Estación Máquina 1.*

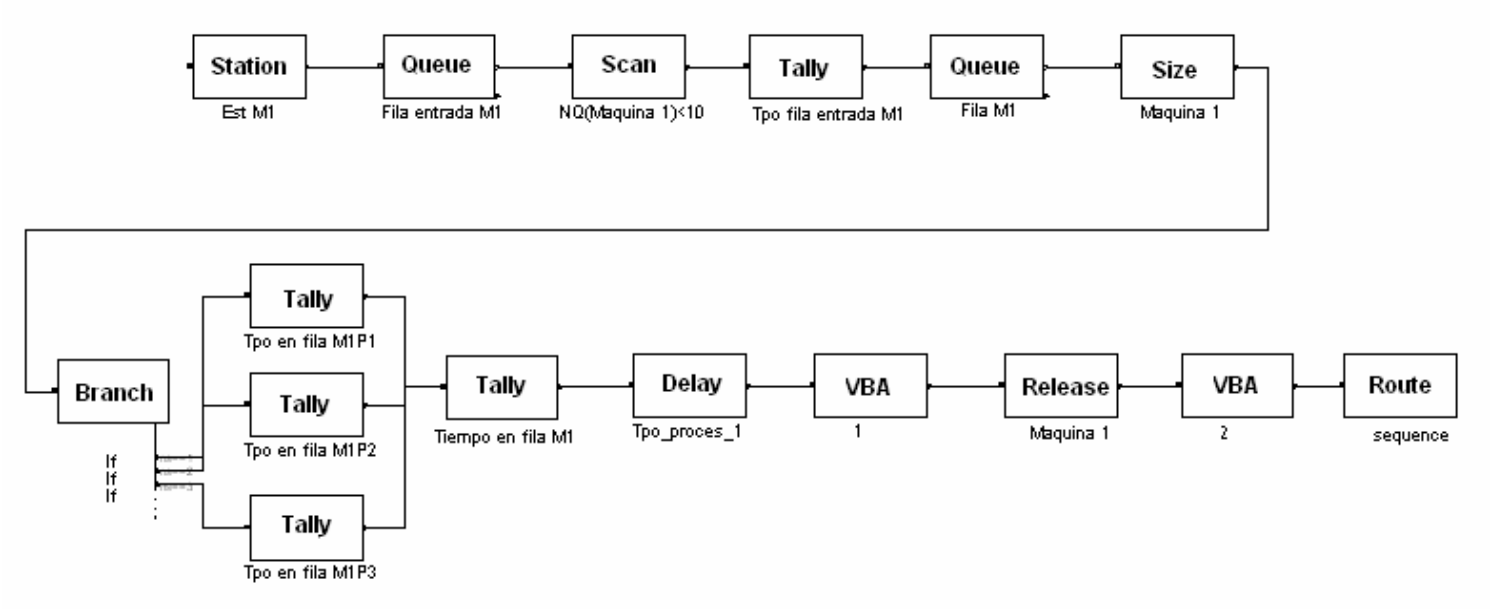



**Figura A. 3.** Estación de Máquina 1 del modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

## *A.4. Estación Máquina 2.*

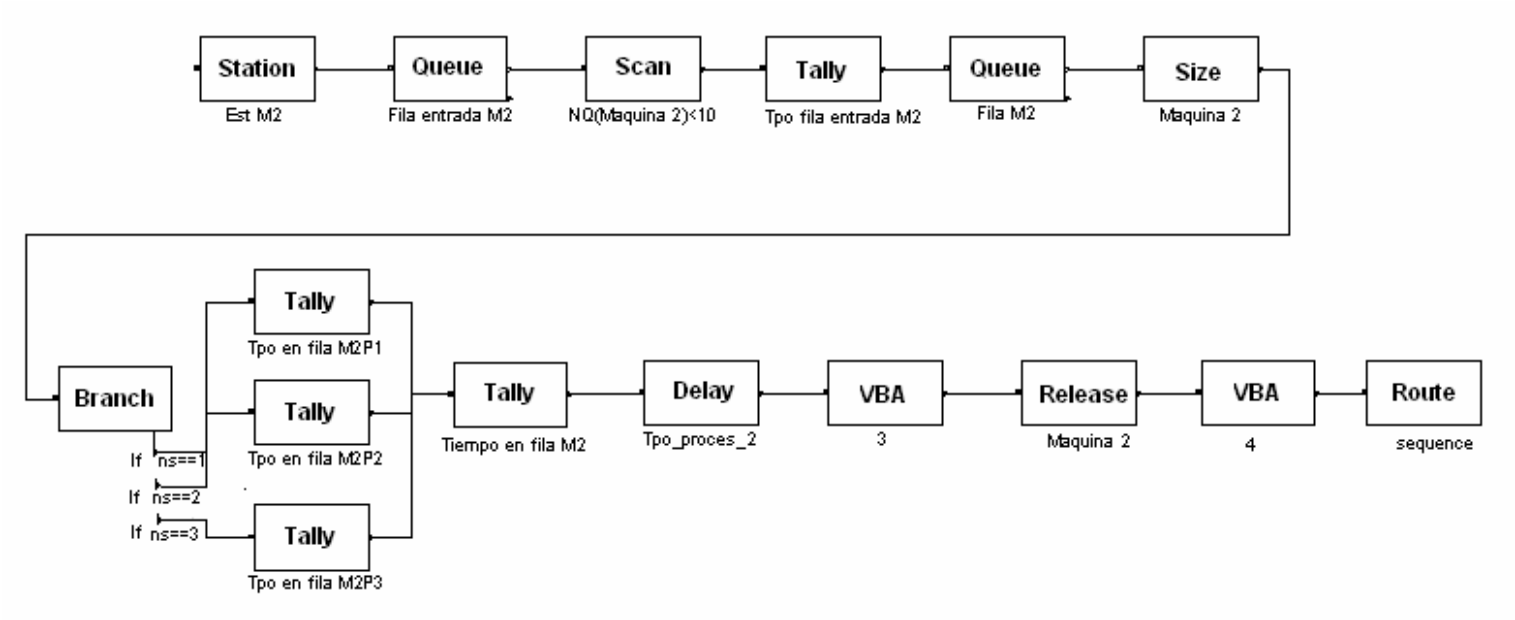



**Figura A. 4.** Estación de Máquina 2 del modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

## *A.5. Estación Máquina 3*

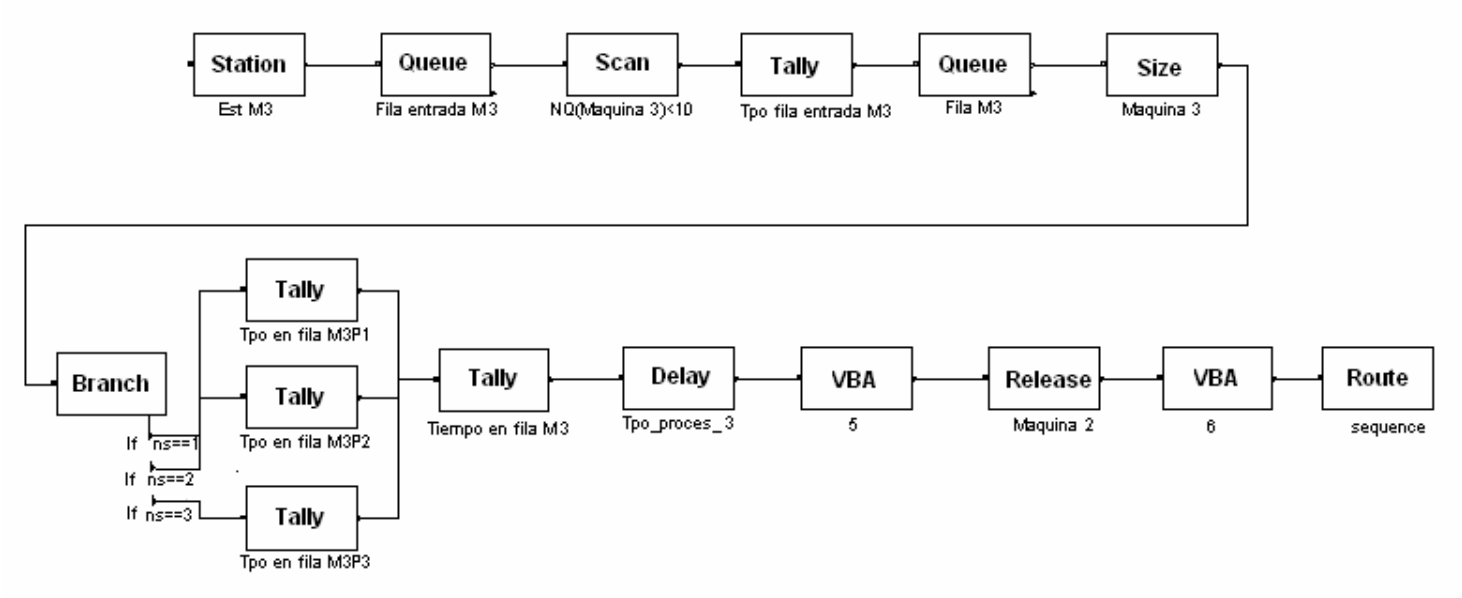



**Figura A. 5.** Estación de Máquina 3 del modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

## *A.6. Estación Salida*

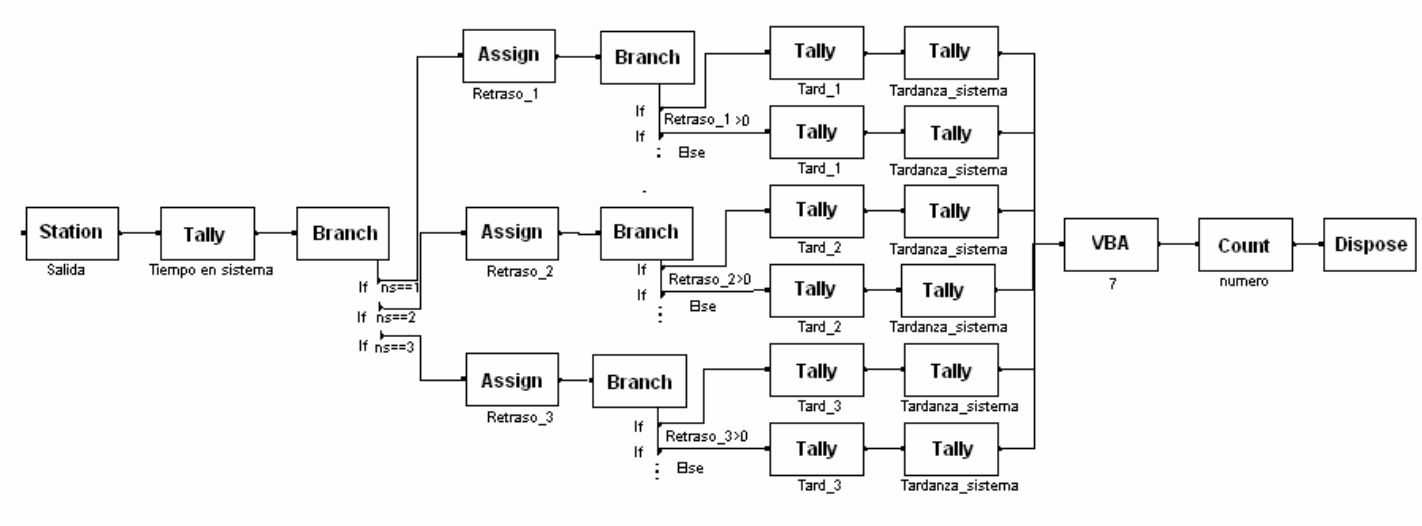



**Figura A. 6.** Estación Salida del modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

# APENDICE B.

# Modelo de Arena™ para sistema de manufactura controlado por tres agentes que utilizan Algoritmo Q-learning difuso.

## *B.1. Elementos del experimento de Arena™*

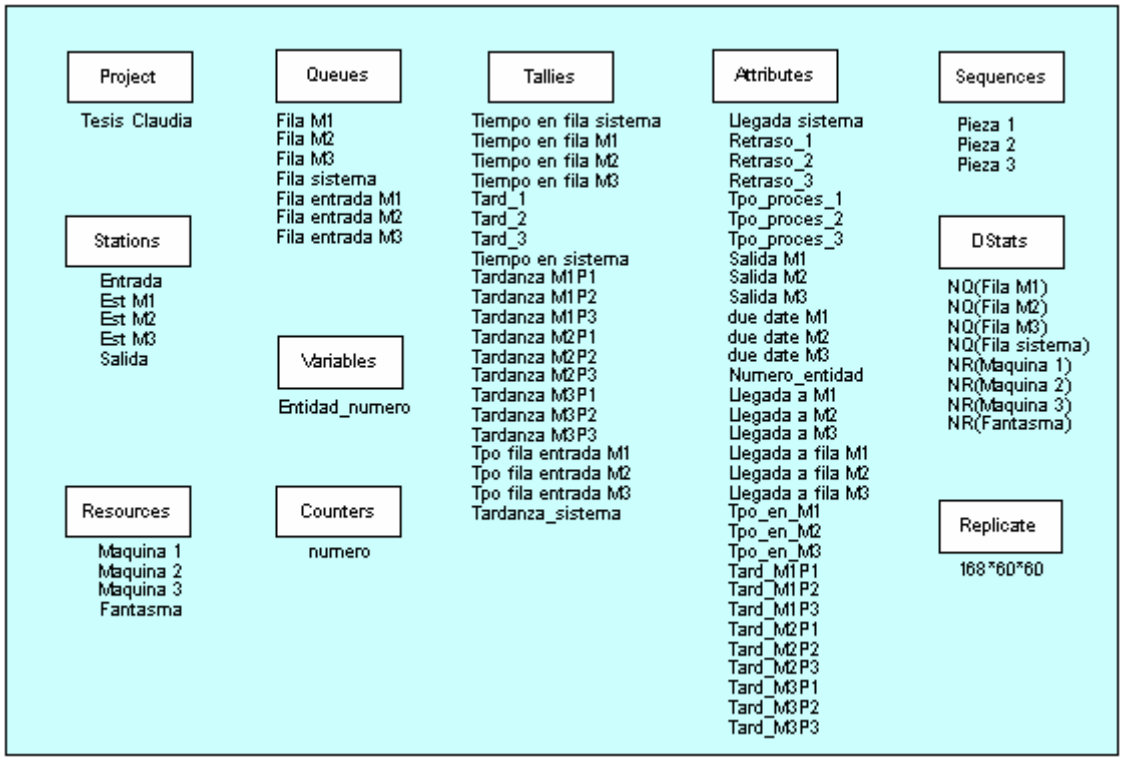



**Figura B. 1.** Experimento del Modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

## *B.2. Estación Entrada.*

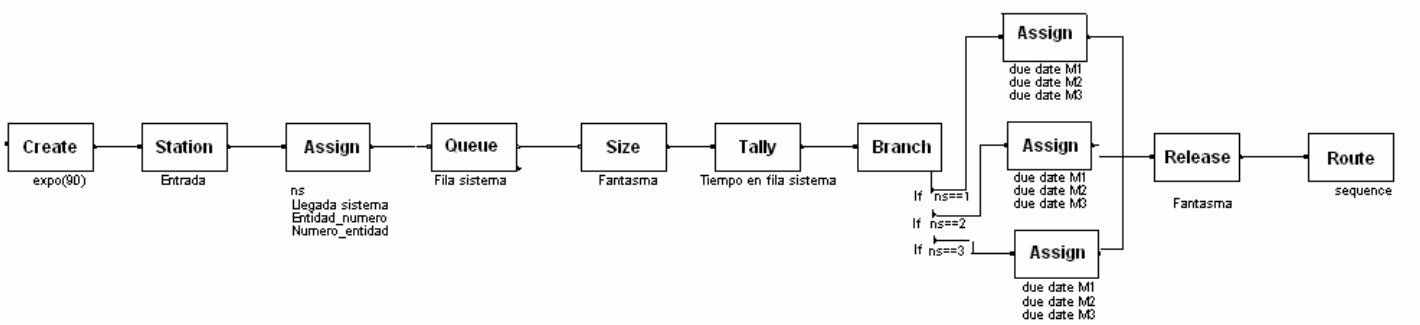



**Figura B. 2.** Estación Entrada del modelo de Arena™ para el sistema controlado por tres agentes utilizando algoritmo Q-learning.

## *B.3. Estación Máquina 1.*

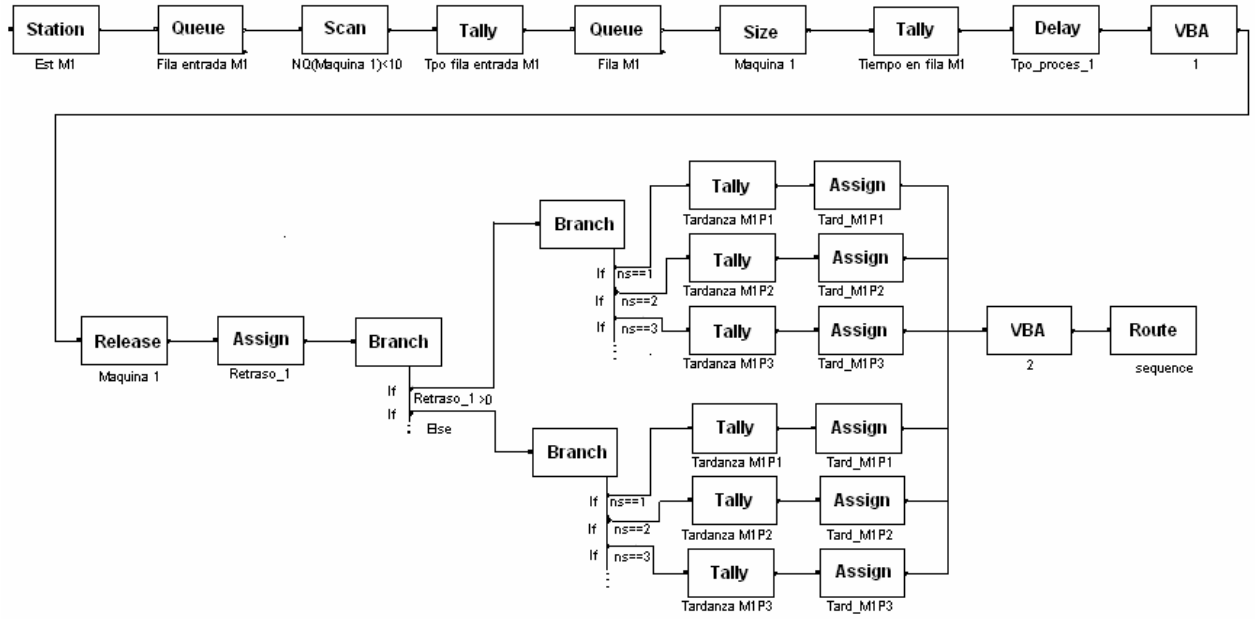



**Figura B. 3.** Estación de Máquina 1 del modelo de Arena™ para el sistema controlado por tres agentes utilizando algoritmo Q-learning.

## *B.4. Estación Máquina 2.*

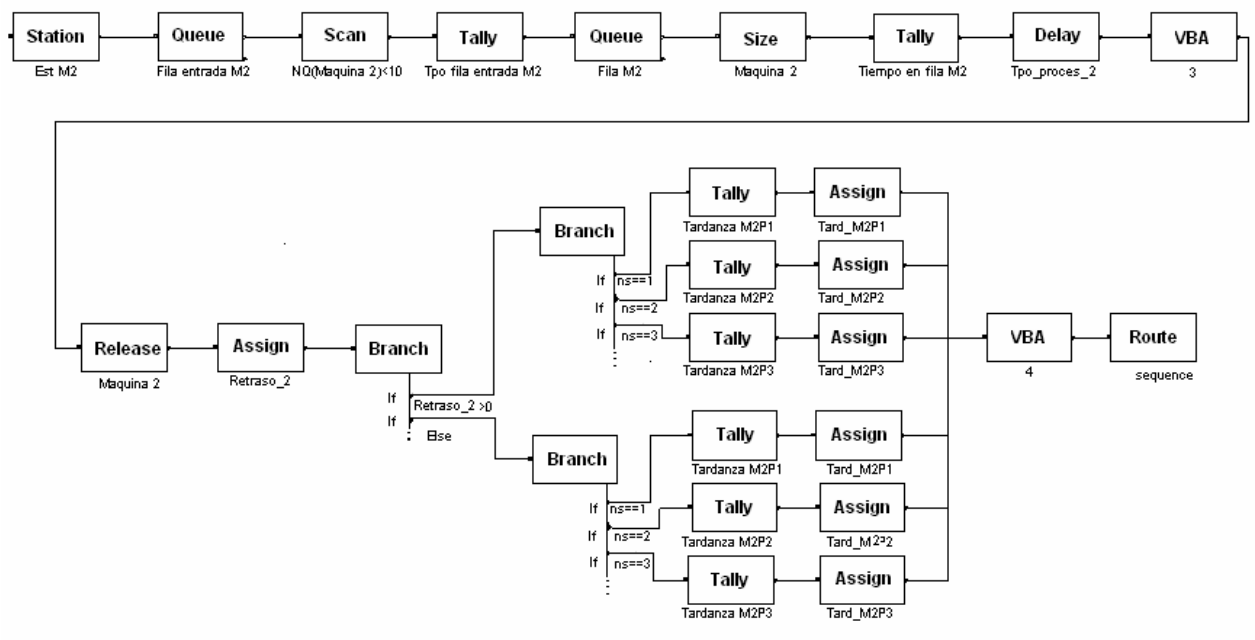



**Figura B. 4.** Estación de Máquina 2 del modelo de Arena™ para el sistema controlado por tres agentes utilizando algoritmo Q-learning.

## *B.5. Estación Máquina 3.*

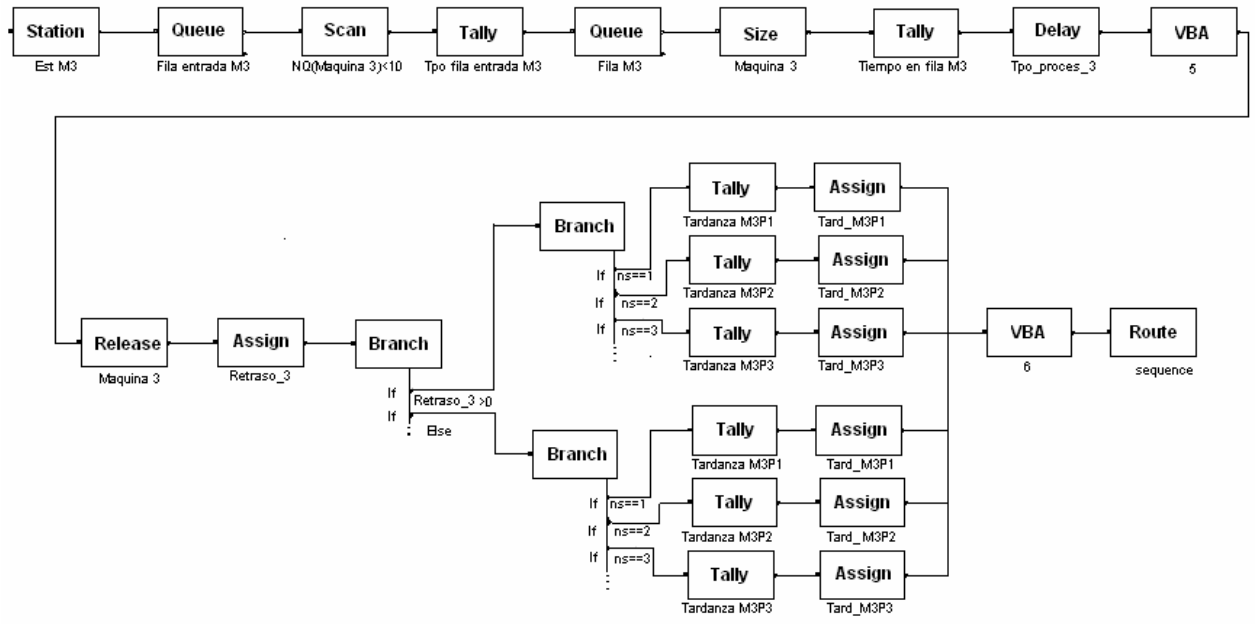



**Figura B. 5**. Estación de Máquina 3 del modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

## *B.6. Estación Salida.*

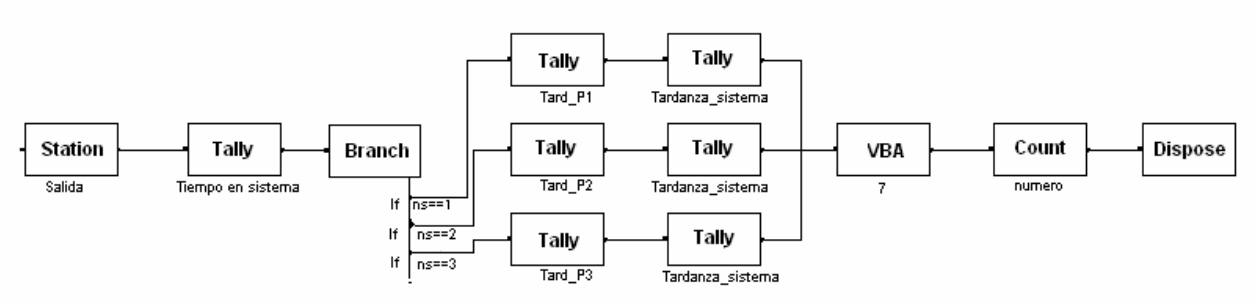



**Figura B. 6.** Estación Salida del modelo de Arena™ para el sistema controlado por un agente utilizando algoritmo Q-learning.

# APÉNDICE C. Programación de interface en VBA

```
'DECLARACIÓN DE VARIABLES

'Variables objeto globales
  Dim g_Model As Arena.Model
  Dim g_SIMAN As Arena.SIMAN
  Dim ModuloQueues As Arena.Module
  Dim oMatlab As Object

 'Variables de Excel
  Dim g_ExcelApp As Excel.Application
    Dim g_Workbook_1 As Excel.Workbook
  Dim g_Workbook_2 As Excel.Workbook
  Dim g_Workbook_3 As Excel.Workbook
  Dim g_Workbook_4 As Excel.Workbook
  Dim g_Worksheet_1 As Excel.Worksheet
  Dim g_Worksheet_2 As Excel.Worksheet
  Dim g_Worksheet_3 As Excel.Worksheet
  Dim g_Worksheet_4 As Excel.Worksheet

  'Variables del programa
  Dim Proxfila As Integer
  Dim g_Tpoespera_q1 As Long
  Dim g_Tpoespera_q2 As Long
  Dim g_Tpoespera_q3 As Long
   Dim g_Npiezas_q1 As Long
  Dim g_Npiezas_q2 As Long
  Dim g_Npiezas_q3 As Long
  Dim g_TpoProces_en_q1 As Long
  Dim g_TpoProces_en_q2 As Long
  Dim g_TpoProces_en_q3 As Long
  Dim g_utilizacion_M1 As Long
  Dim g_utilizacion_M2 As Long
  Dim g_utilizacion_M3 As Long
```

```
Dim g_Llegada_sistema As Long
Dim g_Num_Entidad As Long
Dim g_reglausada As Long
Dim g_Salida_M1 As Long
Dim g_Salida_M2 As Long
Dim g_Salida_M3 As Long
Dim g_Tard_1 As Long
Dim g_Tard_2 As Long
Dim g_Tard_3 As Long
Dim g_Tard_sistema As Long
Dim g_Fila1 As Long
Dim g_Fila2 As Long
Dim g_Fila3 As Long
Dim Tard_P1 As Double
Dim Tard_P2 As Double
Dim Tard_P3 As Double
Dim Retraso_P1 As Double
Dim Retraso_P2 As Double
Dim Retraso_P3 As Double

Dim Tard_total_P1 As Double
Dim Tard_total_P2 As Double
Dim Tard_total_P3 As Double
Dim Num_Entidad As Integer
Dim indiceFilas As Long
Dim Delta_Tard As Double
Dim ReforzaM As Double
Dim num_maq As Integer
Dim regla_utilizada As Integer
Dim llamarmatlab As String

'índices de las filas de los archivos de Excel
Dim i1 As Integer, i2 As Integer, i3 As Integer
Dim j1 As Integer, j2 As Integer, j3 As Integer
Dim m1 As Integer, m2 As Integer, m3 As Integer
Dim r1 As Integer, r2 As Integer, r3 As Integer
Dim t1 As Integer, t2 As Integer, t3 As Integer
```

```
  Dim rg1 As Integer, rg2 As Integer, rg3 As Integer
  Dim m As Integer, n As Long, p As Integer
  Dim Col As Integer, j As Integer, Fila As Integer
  Dim t As Integer, ValorQ As Double, ValorQmax As Double, x As Long

'índices de los tallies de las filas
Dim g_Tpo_fila_M1P1 As Double, g_Tpo_fila_M1P2 As Double, g_Tpo_fila_M1P3 As Double
Dim g_Tpo_fila_M2P1 As Double, g_Tpo_fila_M2P2 As Double, g_Tpo_fila_M2P3 As Double
Dim g_Tpo_fila_M3P1 As Double, g_Tpo_fila_M3P2 As Double, g_Tpo_fila_M3P3 As Double
Dim g_Tpo_fila_entrada_M1 As Double
Dim g_Tpo_fila_entrada_M2 As Double
Dim g_Tpo_fila_entrada_M3 As Double
Dim Tpo_proces_M1P1 As Double, Tpo_proces_M1P2 As Double, Tpo_proces_M1P3 As Double
Dim Tpo_proces_M2P1 As Double, Tpo_proces_M2P2 As Double, Tpo_proces_M2P3 As Double
Dim Tpo_proces_M3P1 As Double, Tpo_proces_M3P2 As Double, Tpo_proces_M3P3 As Double

'Due Dates
Dim DD1 As Double, DD2 As Double, DD3 As Double

'Para calcular valores de los tallies
Dim Tpo_fila_entrada_M1 As Double, Tpo_fila_entrada_M2 As Double, Tpo_fila_entrada_M3 As Double
Dim Tpo_fila_M1P1 As Double, Tpo_fila_M1P2 As Double, Tpo_fila_M1P3 As Double
Dim Tpo_fila_M2P1 As Double, Tpo_fila_M2P2 As Double, Tpo_fila_M2P3 As Double
Dim Tpo_fila_M3P1 As Double, Tpo_fila_M3P2 As Double, Tpo_fila_M3P3 As Double

Private Sub ModelLogic_RunBeginSimulation()

 'INICIO PROGRAMACION

 m1 = 3
 m2 = 3
 m3 = 3
 j1 = 3
 j2 = 3
 j3 = 3
 t1 = 2
 t2 = 2
```

```
 t3 = 2
 t = 2
 rg1 = 3
 rg2 = 3
 rg3 = 3
 x = 2

 'Asignar la variable global g_SIMAN
 Set g_SIMAN = ThisDocument.Model.SIMAN

 'Asignación de variables Tpos de procesamiento y Due Dates
Tpo_proces_M1P1 = 15
Tpo_proces_M1P2 = 10
Tpo_proces_M1P3 = 10
Tpo_proces_M2P1 = 10
Tpo_proces_M2P2 = 5
Tpo_proces_M2P3 = 20
Tpo_proces_M3P1 = 10
Tpo_proces_M3P2 = 10
Tpo_proces_M3P3 = 25
DD1 = 95
DD2 = 85
DD3 = 115

 'Asignar los índices de la variables y atributos de Siman que se utilizan
 ' en la lógica del modelo a las variables globales

  'Tiempo en fila  (para colocar dato en inputs)
  g_Tpoespera_q1 = g_SIMAN.SymbolNumber("Tiempo en fila M1")
  g_Tpoespera_q2 = g_SIMAN.SymbolNumber("Tiempo en fila M2")
  g_Tpoespera_q3 = g_SIMAN.SymbolNumber("Tiempo en fila M3")

  'Tpo de procesamiento total (suma)de las piezas que están en fila
  g_TpoProces_en_q1 = g_SIMAN.SymbolNumber("Tpo_proces_1")
  g_TpoProces_en_q2 = g_SIMAN.SymbolNumber("Tpo_proces_2")
  g_TpoProces_en_q3 = g_SIMAN.SymbolNumber("Tpo_proces_3")
```

```
' Tiempo de salida de las máquinas
  g_Salida_M1 = g_SIMAN.SymbolNumber("Salida M1")
  g_Salida_M2 = g_SIMAN.SymbolNumber("Salida M2")
  g_Salida_M3 = g_SIMAN.SymbolNumber("Salida M3")

  'Utilización de las máquinas (ocupado/desocupado)
  g_utilizacion_M1 = g_SIMAN.SymbolNumber("Maquina 1")
  g_utilizacion_M2 = g_SIMAN.SymbolNumber("Maquina 2")
  g_utilizacion_M3 = g_SIMAN.SymbolNumber("Maquina 3")

  'Tardanzas por pieza (cálculo real al final de la corrida)
  g_Tard_1 = g_SIMAN.SymbolNumber("Tard_1")
  g_Tard_2 = g_SIMAN.SymbolNumber("Tard_2")
  g_Tard_3 = g_SIMAN.SymbolNumber("Tard_3")
  g_Tard_sistema = g_SIMAN.SymbolNumber("Tardanza_sistema")

  'Para encontrar fila y poder calcular # de piezas en esa fila
  g_Fila1 = g_SIMAN.SymbolNumber("Fila M1")
  g_Fila2 = g_SIMAN.SymbolNumber("Fila M2")
  g_Fila3 = g_SIMAN.SymbolNumber("Fila M3")

  'Tiempo en fila entrada a máquinas (para calcular recompensas)
  g_Tpo_fila_entrada_M1 = g_SIMAN.SymbolNumber("Tpo fila entrada M1")
  g_Tpo_fila_entrada_M2 = g_SIMAN.SymbolNumber("Tpo fila entrada M2")
  g_Tpo_fila_entrada_M3 = g_SIMAN.SymbolNumber("Tpo fila entrada M3")

  'Tiempo en fila de máquinas por tipo de pieza (para calcular recompensas)
  g_Tpo_fila_M1P1 = g_SIMAN.SymbolNumber("Tpo en fila M1P1")
  g_Tpo_fila_M1P2 = g_SIMAN.SymbolNumber("Tpo en fila M1P2")
  g_Tpo_fila_M1P3 = g_SIMAN.SymbolNumber("Tpo en fila M1P3")
  g_Tpo_fila_M2P1 = g_SIMAN.SymbolNumber("Tpo en fila M2P1")
  g_Tpo_fila_M2P2 = g_SIMAN.SymbolNumber("Tpo en fila M2P2")
  g_Tpo_fila_M2P3 = g_SIMAN.SymbolNumber("Tpo en fila M2P3")
  g_Tpo_fila_M3P1 = g_SIMAN.SymbolNumber("Tpo en fila M3P1")
  g_Tpo_fila_M3P2 = g_SIMAN.SymbolNumber("Tpo en fila M3P2")
  g_Tpo_fila_M3P3 = g_SIMAN.SymbolNumber("Tpo en fila M3P3")
```

```
'Otros datos
  g_Llegada_sistema = g_SIMAN.SymbolNumber("Llegada sistema")
  g_Num_Entidad = g_SIMAN.SymbolNumber("Numero_entidad")
  g_reglausada = g_SIMAN.SymbolNumber("regla")

'Encontrar el módulo Queues de Elementos mediante sus índices
indiceFilas = ThisDocument.Model.Modules.Find(smFindTag, "Filas")
Set ModuloQueues = ThisDocument.Model.Modules.Item(indiceFilas)

'INICIAR EXCEL Y CREAR NUEVOS ARCHIVOS

'Crear una aplicación de Excel
    Set g_ExcelApp = New Excel.Application
    g_ExcelApp.Visible = True
    g_ExcelApp.SheetsInNewWorkbook = 1

'Crear archivo "Todos Los Datos"
    Set g_Workbook_1 = g_ExcelApp.Workbooks.Add
    'Para que no pregunte si sobreescribe se agregó la siguiente línea
    g_ExcelApp.DisplayAlerts = False
    g_Workbook_1.SaveAs ThisDocument.Model.Path & "Todos los Datos.xls"
 'Crear archivo "Inputs"
    Set g_Workbook_2 = g_ExcelApp.Workbooks.Add
    'Para que no pregunte si sobreescribe agregamos la siguiente línea
    g_ExcelApp.DisplayAlerts = False
    g_Workbook_2.SaveAs ThisDocument.Model.Path & "Inputs.xls"
 'Crear archivo "Recompensas"
    Set g_Workbook_3 = g_ExcelApp.Workbooks.Add
    'Para que no pregunte si sobreescribe agregamos la siguiente línea
    g_ExcelApp.DisplayAlerts = False
    g_Workbook_3.SaveAs ThisDocument.Model.Path & "Recompensas.xls"
 'Crear archivo "Tardanzas"
    Set g_Workbook_4 = g_ExcelApp.Workbooks.Add
    'Para que no pregunte si sobreescribe agregamos la siguiente línea
    g_ExcelApp.DisplayAlerts = False
    g_Workbook_4.SaveAs ThisDocument.Model.Path & "Tardanzas.xls"
```

```
'FORMATO DE LOS ARCHIVOS

'Dar formato al archivo "Todos los Datos"
g_ExcelApp.Workbooks("Todos los Datos.xls").Activate
With g_ExcelApp.Workbooks("Todos los Datos.xls").Worksheets("Sheet1")
    .Activate
  'Nombres de las columnas
  For j = 1 To 3
    Col = (12 * (j - 1)) + 1
   .Cells(1, Col + 3) = "MAQUINA " & j
   .Cells(2, Col) = "# Entidad"
   .Cells(2, Col + 1) = "No. piezas en fila"
   .Cells(2, Col + 2) = "Tpo. en fila"
   .Cells(2, Col + 3) = "Tpo. proces. piezas en fila"
   .Cells(2, Col + 4) = "# Entidad"
   .Cells(2, Col + 5) = " Regla utilizada"
   .Cells(2, Col + 6) = "n"
   .Cells(2, Col + 7) = "Epsilon"
   .Cells(2, Col + 8) = "Tipo de Pieza"
   .Cells(2, Col + 9) = "Tardanza estimada"
   .Cells(2, Col + 10) = "Recompensa"
  Next j
End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save

'Dar formato a archivo "Inputs" y agregarle nombre a las columnas
g_ExcelApp.Workbooks("Inputs.xls").Activate
With g_ExcelApp.Workbooks("Inputs.xls").Worksheets("Sheet1")    .Activate
    'Nombre de las columnas
    .Cells(1, 1) = "Tpo.espera en fila"
    .Cells(1, 2) = "No. piezas en fila"
    .Cells(1, 3) = "Tpo. proces. piezas en fila"
End With
g_ExcelApp.Workbooks("Inputs.xls").Save

'Dar formato a archivo "Recompensas" y agregarle nombre a las columnas
g_ExcelApp.Workbooks("Recompensas.xls").Activate
```

```
With g_ExcelApp.Workbooks("Recompensas.xls").Worksheets("Sheet1")
    .Activate
    'Nombre de las columnas
    .Cells(1, 1) = "Máquina"
    .Cells(1, 2) = "Tipo de Pieza"
    .Cells(1, 3) = "Regla_utilizada"
    .Cells(1, 4) = "Reforzamiento"
    For m = 1 To 3
       For n = 1 To 3
           For p = 1 To 3
               Fila = (3 ^ 2 * (m - 1) + 3 * (n - 1) + (p - 1)) + 2
               .Cells(Fila, 1) = m
               .Cells(Fila, 2) = n
               .Cells(Fila, 3) = p
               .Cells(Fila, 4) = 0
           Next p
        Next n
    Next m
 .Cells(35, 2) = 1
End With
g_ExcelApp.Workbooks("Recompensas.xls").Save
'Dar formato a archivo "Tardanzas" y agregarle nombre a las columnas
g_ExcelApp.Workbooks("Tardanzas.xls").Activate
Set g_Worksheet_4 = g_ExcelApp.Workbooks("Tardanzas.xls").Worksheets("Sheet1")
With g_Worksheet_4
    .Activate
    'Nombre de las columnas
    .Cells(1, 1) = "Tard. P1"
    .Cells(1, 2) = "Tard. P2"
    .Cells(1, 3) = "Tard. P3"
    .Cells(1, 4) = "Tard. sistema%"
    .Cells(1, 5) = "Tiempo actual"
    .Cells(1, 7) = "QValue"
    .Cells(1, 8) = "QValue_Max"
End With
g_ExcelApp.Workbooks("Tardanzas.xls").Save
End Sub
```

```
Private Sub VBA_Block_1_Fire()
'VBA de INPUTS MAQUINA 1

'Declaración de Variables
  Dim Espera_q1 As Double
  Dim PiezasM1 As Long
  Dim TpoProces_q1 As Double
  Dim regla_1 As Integer
  Dim Tpo_Proces_pieza_en_q1 As Double
  Dim utiliz_M1 As Double
  Dim Epsilon As Double
  Dim n As Double

'Obtener los valores de Tallys y Atributos
  Espera_q1 = g_SIMAN.TallyAverage(g_Tpoespera_q1)
  PiezasM1 = g_SIMAN.QueueNumberOfEntities(g_Fila1)
  TpoProces_q1 = g_SIMAN.QueuedEntityAttributesSum(g_Fila1, g_TpoProces_en_q1)
  Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
  Tpo_Proces_pieza_en_q1 = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_TpoProces_en_q1, 0, 0)
  utiliz_M1 = g_SIMAN.ResourceInstantaneousUtilization(g_utilizacion_M1)

'Para leer regla utilizada grabada en Excel
If (ModuloQueues.Data("Ranking(1)") = "FIFO") Then
    regla_utilizada = 1
ElseIf (ModuloQueues.Data("Ranking(1)") = "LVF") Then
    If (ModuloQueues.Data("RankExp(1)") = "Tpo_proces_1") Then
        regla_utilizada = 2
    ElseIf (ModuloQueues.Data("RankExp(1)") = "due date") Then
        regla_utilizada = 3
    End If
End If

' Asignar la regla leída al atributo regla
g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_reglausada) = regla_utilizada
```

```
'ESCRIBIR DATOS DE LA CORRIDA EN LOS ARCHIVOS DE EXCEL

'Escribir datos en archivo "Todos los Datos"
g_ExcelApp.Workbooks("Todos los Datos").Activate
 With g_ExcelApp.Workbooks("Todos los Datos").Worksheets("Sheet1")
    .Cells(m1, 1) = Num_Entidad
    .Cells(m1, 3) = Espera_q1
    If PiezasM1 = 0 Then  'cuando pasa la pieza y no hay nada en fila
        If utiliz_M1 = 1 Then   'recurso ocupado
           .Cells(m1, 2) = (PiezasM1 + 1)
           .Cells(m1, 4) = Tpo_Proces_pieza_en_q1
        Else 'máquina desocupada y no hay fila
           .Cells(m1, 2) = PiezasM1
           .Cells(m1, 4) = 0
        End If
    Else 'Si hay piezas en fila cuando llega la entidad activa
        .Cells(m1, 2) = (PiezasM1)
        .Cells(m1, 4) = (TpoProces_q1 + Tpo_Proces_pieza_en_q1) / (PiezasM1)
    End If
   m1 = m1 + 1
End With
 g_ExcelApp.Workbooks("Todos los Datos").Save

'Escribir datos en archivo "Inputs"
g_ExcelApp.Workbooks("Inputs").Activate
  With g_ExcelApp.Workbooks("Inputs").Worksheets("Sheet1")
    .Cells(2, 1) = Espera_q1
    If PiezasM1 = 0 Then  'cuando pasa la pieza y no hay nada en fila
        If utiliz_M1 = 1 Then   'recurso ocupado
           .Cells(2, 2) = (PiezasM1 + 1)
           .Cells(2, 3) = Tpo_Proces_pieza_en_q1
        Else 'máquina desocupada y no hay fila
           .Cells(2, 2) = PiezasM1
           .Cells(2, 3) = 0
        End If
    Else 'Si hay piezas en fila cuando llega la entidad activa
        .Cells(2, 2) = (PiezasM1)
```

```
        .Cells(2, 3) = (TpoProces_q1 + Tpo_Proces_pieza_en_q1) / (PiezasM1)
    End If
  End With
g_ExcelApp.Workbooks("Inputs").Save


'Escribir num_maq y Tipo_pieza en Excel para que lo lea MATLAB
With g_ExcelApp.Workbooks("Recompensas.xls").Worksheets("Sheet1")
    .Activate
    .Cells(35, 1) = 1            'número de la máquina
End With
g_ExcelApp.Workbooks("Recompensas.xls").Save

'LLAMAR ALGORITMO Q_LEARNING DE MATLAB
Set oMatlab = CreateObject("MatLab.Application")
oMatlab.Execute ("enableservice('automationserver',true)")
llamarmatlab = oMatlab.Execute("cd C:\Program Files\MATLAB\R2006b\work")
llamarmatlab = oMatlab.Execute("fusificacion")
Set oMatlab = Nothing

'LEYENDO REGLA DE DECISIÓN DESDE EL ACHIVO "REGLA"
g_ExcelApp.Workbooks.Open ("C:\Temp\un agente\Regla.xls")
With g_ExcelApp.Workbooks("Regla.xls").Worksheets("Sheet1")
    regla_1 = .Cells(1, 1)
    Epsilon = .Cells(1, 2)
    n = .Cells(1, 3)
    ValorQ = .Cells(1, 4)
    ValorQmax = .Cells(1, 5)
End With
g_ExcelApp.Workbooks("Regla").Close

'ESCRIBIR REGLA EN ARCHIVO "Todos los Datos.xls"
g_ExcelApp.Workbooks("Todos los Datos.xls").Activate
With g_ExcelApp.Workbooks("Todos los Datos").Worksheets("Sheet1")
    .Activate
    .Cells(rg1, 6) = regla_1
    .Cells(rg1, 7) = n
```

```
    .Cells(rg1, 8) = Epsilon
    rg1 = rg1 + 1
End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save

'ASIGNAR AL MODULO DE ELEMENTOS QUEUES DE LA FILA 1 LA NUEVA REGLA DE DECISIÓN
If regla_1 = 1 Then              'Regla FIFO
    ModuloQueues.Data("Ranking(1)") = "FIFO"
ElseIf regla_1 = 2 Then          'Regla SPT
    ModuloQueues.Data("Ranking(1)") = "LVF"
    ModuloQueues.Data("RankExp(1)") = "Tpo_proces_1"
ElseIf regla_1 = 3 Then          'Regla EDD
    ModuloQueues.Data("Ranking(1)") = "LVF"
    ModuloQueues.Data("RankExp(1)") = "due date"
End If
ThisDocument.Model.Save

'Guardar los datos de los valores QValue y QValue_Max en archivo de Excel
g_ExcelApp.Workbooks("Tardanzas.xls").Activate
With g_ExcelApp.Workbooks("Tardanzas").Worksheets("Sheet1")
    .Activate
    .Cells(x, 7) = ValorQ
    .Cells(x, 8) = ValorQmax
    x = x + 1
End With
g_ExcelApp.Workbooks("Tardanzas.xls").Save
End Sub


Private Sub VBA_Block_2_Fire()
'BLOQUE VBA DE RECOMPENSAS EN MAQUINA 1

'Declaración de variables de SIMAN
Dim Tipo_Pieza As Integer
Dim Llegada_sist As Double
Dim Tpo_estimado_sistema As Double
Dim Tpo_salida_M1 As Double
```

```
Dim a As Integer, b As Integer, c As Integer, k As Integer

'Inicialización de variables de prueba
Tpo_estimado_sistema = 0
Retraso_P1 = 0
Retraso_P2 = 0
Retraso_P3 = 0
Tard_total_P1 = 0   'Tardanza estimada
Tard_total_P2 = 0   'Tardanza estimada
Tard_total_P3 = 0   'Tardanza estimada
Delta_Tard = 0
ReforzaM = 0

'OBTENER EL VALOR DEL ATRIBUTO DE LA ENTIDAD ACTUAL
Tipo_Pieza = g_SIMAN.EntitySequenceAttribute(g_SIMAN.ActiveEntity)
Llegada_sist = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_Llegada_sistema)
Tpo_salida_M1 = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_Salida_M1)
Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
regla_utilizada = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_reglausada)
'Tpo en filas
Tpo_fila_entrada_M1 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M1)
Tpo_fila_entrada_M2 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M2)
Tpo_fila_entrada_M3 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M3)
Tpo_fila_M1P1 = g_SIMAN.TallyAverage(g_Tpo_fila_M1P1)
Tpo_fila_M1P2 = g_SIMAN.TallyAverage(g_Tpo_fila_M1P2)
Tpo_fila_M1P3 = g_SIMAN.TallyAverage(g_Tpo_fila_M1P3)
'Tardanzas promedio reales medidas hasta ese momento
Tard_P1 = g_SIMAN.TallyAverage(g_Tard_1)
Tard_P2 = g_SIMAN.TallyAverage(g_Tard_2)
Tard_P3 = g_SIMAN.TallyAverage(g_Tard_3)


'CALCULO DE TARDANZA ESTIMADA POR TIPO DE PIEZA

'Pieza 1
If Tipo_Pieza = 1 Then
```

```
   Tpo_estimado_sistema = (Tpo_salida_M1 - Llegada_sist) + (Tpo_fila_entrada_M3 + Tpo_fila_M3P1 +
Tpo_proces_M3P1) + (Tpo_fila_entrada_M2 + Tpo_fila_M2P1 + Tpo_proces_M2P1)
   Retraso_P1 = Tpo_estimado_sistema - DD1
   If Retraso_P1 > 0 Then
      Tard_total_P1 = Retraso_P1
   Else
      Tard_total_P1 = 0
   End If
   Delta_Tard = (Tard_total_P1 - Tard_P1)

'Pieza 2
ElseIf Tipo_Pieza = 2 Then
   Tpo_estimado_sistema = (Tpo_salida_M1 - Llegada_sist)
   Retraso_P2 = Tpo_estimado_sistema - DD2
   If Retraso_P2 > 0 Then
      Tard_total_P2 = Retraso_P2
   Else
      Tard_total_P2 = 0
   End If
      Delta_Tard = (Tard_total_P2 - Tard_P2)

'Pieza 3
ElseIf Tipo_Pieza = 3 Then
   Tpo_estimado_sistema = (Tpo_salida_M1 - Llegada_sist) + (Tpo_fila_entrada_M3 + Tpo_fila_M3P3 +
Tpo_proces_M3P3)
   Retraso_P3 = Tpo_estimado_sistema - DD3
   If Retraso_P3 > 0 Then
      Tard_total_P3 = Retraso_P3
   Else
      Tard_total_P3 = 0
   End If
   Delta_Tard = (Tard_total_P3 - Tard_P3)
End If

'ASIGNACIÓN DE RECOMPENSAS
If (Delta_Tard < 0) Then
    ReforzaM = 10
```

```
ElseIf (Delta_Tard = 0) Then
    ReforzaM = 1
ElseIf (Delta_Tard > 0 And Delta_Tard <= 1) Then
    ReforzaM = -1
ElseIf (Delta_Tard > 1 And Delta_Tard <= 2) Then
    ReforzaM = -2
ElseIf (Delta_Tard > 2 And Delta_Tard <= 3) Then
    ReforzaM = -3
ElseIf (Delta_Tard > 3 And Delta_Tard <= 4) Then
    ReforzaM = -4
ElseIf (Delta_Tard > 4 And Delta_Tard <= 5) Then
    ReforzaM = -5
ElseIf (Delta_Tard > 5 And Delta_Tard <= 6) Then
    ReforzaM = -6
ElseIf (Delta_Tard > 6 And Delta_Tard <= 7) Then
    ReforzaM = -7
ElseIf (Delta_Tard > 7 And Delta_Tard <= 8) Then
    ReforzaM = -8
ElseIf (Delta_Tard > 8 And Delta_Tard <= 9) Then
    ReforzaM = -9
ElseIf (Delta_Tard > 9 And Delta_Tard <= 10) Then
    ReforzaM = -10
ElseIf (Delta_Tard > 10) Then
    ReforzaM = -20
End If

'ESCRIBIR DATOS EN ARCHIVO "RECOMPENSAS"
num_maq = 1
g_ExcelApp.Workbooks("Recompensas.xls").Activate
With g_ExcelApp.Workbooks("Recompensas.xls").Worksheets("Sheet1")
    For a = 1 To 3
        If (num_maq = a) Then
            For b = 1 To 3
                If (Tipo_Pieza = b) Then
                    For c = 1 To 3
                        If (regla_utilizada = c) Then
                            Fila = (3 ^ 2 * (a - 1) + 3 * (b - 1) + (c - 1)) + 2
```

```
                        .Cells(Fila, 4) = ReforzaM
                    End If
                Next c
            End If
        Next b
    End If
Next a
.Cells(35, 2) = Tipo_Pieza
End With
g_ExcelApp.Workbooks("Recompensas.xls").Save

'Escribir datos en archivo "Todos los Datos"
g_ExcelApp.Workbooks("Todos los Datos.xls").Activate
With g_ExcelApp.Workbooks("Todos los Datos.xls").Worksheets("Sheet1")
    .Cells(j1, 5) = Num_Entidad
    .Cells(j1, 9) = Tipo_Pieza
    If Tipo_Pieza = 1 Then
    .Cells(j1, 10) = Tard_total_P1
    ElseIf Tipo_Pieza = 2 Then
    .Cells(j1, 10) = Tard_total_P2
    ElseIf Tipo_Pieza = 3 Then
    .Cells(j1, 10) = Tard_total_P3
    End If
    .Cells(j1, 11) = ReforzaM
    j1 = j1 + 1
End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save
End Sub


Private Sub VBA_Block_3_Fire()
'VBA de INPUTS MAQUINA 2

'Declaración de Variables
  Dim Espera_q2 As Double
  Dim PiezasM2 As Long
  Dim TpoProces_q2 As Double
```

```
  Dim regla_2 As Integer
  Dim Tpo_Proces_pieza_en_q2 As Double
  Dim utiliz_M2 As Double
  Dim num_seq_M2 As Integer
  Dim Epsilon As Double
  Dim n As Double

'Obtener valores de los Tallys y Atributos
  Espera_q2 = g_SIMAN.TallyAverage(g_Tpoespera_q2)
  PiezasM2 = g_SIMAN.QueueNumberOfEntities(g_Fila2)
  TpoProces_q2 = g_SIMAN.QueuedEntityAttributesSum(g_Fila2, g_TpoProces_en_q2)
  Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
  Tpo_Proces_pieza_en_q2 = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_TpoProces_en_q2, 0, 0)
  utiliz_M2 = g_SIMAN.ResourceInstantaneousUtilization(g_utilizacion_M2)

'Para leer regla utilizada para seleccionar la pieza
If (ModuloQueues.Data("Ranking(2)") = "FIFO") Then
    regla_utilizada = 1
ElseIf (ModuloQueues.Data("Ranking(2)") = "LVF") Then
    If (ModuloQueues.Data("RankExp(2)") = "Tpo_proces_2") Then
        regla_utilizada = 2
    ElseIf (ModuloQueues.Data("RankExp(2)") = "due date") Then
        regla_utilizada = 3
    End If
End If


' Asignar la regla leída al atributo regla
g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_reglausada) = regla_utilizada
'ESCRIBIR LOS DATOS DE LA CORRIDA EN LOS ARCHIVOS DE EXCEL

'Escribir datos en archivo "Todos los Datos"
g_ExcelApp.Workbooks("Todos los Datos").Activate
 With g_ExcelApp.Workbooks("Todos los Datos").Worksheets("Sheet1")
    .Cells(m2, 13) = Num_Entidad
    .Cells(m2, 15) = Espera_q2
    If PiezasM2 = 0 Then  'cuando pasa la pieza y no hay nada en fila
```

```
        If utiliz_M2 = 1 Then   'recurso ocupado
           .Cells(m2, 14) = (PiezasM2 + 1)
           .Cells(m2, 16) = Tpo_Proces_pieza_en_q2
        Else 'máquina desocupada y no hay fila
           .Cells(m2, 14) = PiezasM2
           .Cells(m2, 16) = 0
        End If
    Else 'Si hay piezas en fila cuando llega la entidad activa
        .Cells(m2, 14) = (PiezasM2)
        .Cells(m2, 16) = (TpoProces_q2 + Tpo_Proces_pieza_en_q2) / (PiezasM2)
    End If
    m2 = m2 + 1
  End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save

'Escribir datos en archivo "Inputs"
g_ExcelApp.Workbooks("Inputs").Activate
  With g_ExcelApp.Workbooks("Inputs").Worksheets("Sheet1")
     .Cells(2, 1) = Espera_q2
    If PiezasM2 = 0 Then  'cuando pasa la pieza y no hay nada en fila
        If utiliz_M2 = 1 Then   'recurso ocupado
           .Cells(2, 2) = (PiezasM2 + 1)
           .Cells(2, 3) = Tpo_Proces_pieza_en_q2
        Else 'máquina desocupada y no hay fila
           .Cells(2, 2) = PiezasM2
           .Cells(2, 3) = 0
        End If
    Else 'Si hay piezas en fila cuando llega la entidad activa
        .Cells(2, 2) = (PiezasM2)
        .Cells(2, 3) = (TpoProces_q2 + Tpo_Proces_pieza_en_q2) / (PiezasM2)
    End If
  End With
g_ExcelApp.Workbooks("Inputs.xls").Save

'Escribir num_maq y Tipo_pieza en Excel para que lo lea MATLAB
With g_ExcelApp.Workbooks("Recompensas.xls").Worksheets("Sheet1")
    .Activate
```

```
    .Cells(35, 1) = 2            'número de la máquina
End With
g_ExcelApp.Workbooks("Recompensas.xls").Save

'LLAMAR ALGORITMO Q_LEARNING DE MATLAB
Set oMatlab = CreateObject("MatLab.Application")
oMatlab.Execute ("enableservice('automationserver',true)")
llamarmatlab = oMatlab.Execute("cd C:\Program Files\MATLAB\R2006b\work")
llamarmatlab = oMatlab.Execute("fusificacion")
Set oMatlab = Nothing

'LEYENDO REGLA DE DECISIÓN DESDE EL ACHIVO "REGLAS"
g_ExcelApp.Workbooks.Open ("C:\Temp\un agente\Regla.xls")
With g_ExcelApp.Workbooks("Regla.xls").Worksheets("Sheet1")
    regla_2 = .Cells(1, 1)
    Epsilon = .Cells(1, 2)
          n = .Cells(1, 3)
     ValorQ = .Cells(1, 4)
    ValorQmax = .Cells(1, 5)
End With
g_ExcelApp.Workbooks("Regla").Close

'ESCRIBIR REGLA EN ARCHIVO "TODOS LOS DATOS.XLS"
g_ExcelApp.Workbooks("Todos los Datos.xls").Activate
With g_ExcelApp.Workbooks("Todos los Datos").Worksheets("Sheet1")
    .Activate
    .Cells(rg2, 18) = regla_2
    .Cells(rg2, 19) = n
    .Cells(rg2, 20) = Epsilon
    rg2 = rg2 + 1
End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save

'ASIGNAR AL MODULO DE ELEMENTOS QUEUES DE LA FILA 2 LA NUEVA REGLA DE DECISIÓN
If regla_2 = 1 Then              'Regla FIFO
    ModuloQueues.Data("Ranking(2)") = "FIFO"
ElseIf regla_2 = 2 Then          'Regla SPT
```

```
    ModuloQueues.Data("Ranking(2)") = "LVF"
    ModuloQueues.Data("RankExp(2)") = "Tpo_proces_2"
ElseIf regla_2 = 3 Then          'Regla EDD
    ModuloQueues.Data("Ranking(2)") = "LVF"
    ModuloQueues.Data("RankExp(2)") = "due date"
End If
ThisDocument.Model.Save

'Guardar los datos de los valores QValue y QValue_Max en archivo Tardanzas
g_ExcelApp.Workbooks("Tardanzas.xls").Activate
With g_ExcelApp.Workbooks("Tardanzas").Worksheets("Sheet1")
    .Activate
    .Cells(x, 7) = ValorQ
    .Cells(x, 8) = ValorQmax
    x = x + 1
End With
g_ExcelApp.Workbooks("Tardanzas.xls").Save
End Sub


Private Sub VBA_Block_4_Fire()
'BLOQUE VBA DE RECOMPENSAS EN MAQUINA 2

'Declaración de variables de SIMAN
Dim Tipo_Pieza As Integer
Dim Llegada_sist As Double
Dim Tpo_estimado_sistema As Double
Dim Tpo_salida_M2 As Double
Dim a As Integer, b As Integer, c As Integer, k As Integer

'Inicialización de variables de prueba
Tpo_estimado_sistema = 0
Retraso_P1 = 0
Retraso_P2 = 0
Retraso_P3 = 0
Tard_total_P1 = 0
Tard_total_P2 = 0
```

```
Tard_total_P3 = 0
Delta_Tard = 0
ReforzaM = 0

'Obtener el valor del atributo de la entidad actual
Tipo_Pieza = g_SIMAN.EntitySequenceAttribute(g_SIMAN.ActiveEntity)
Llegada_sist = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_Llegada_sistema)
Tpo_salida_M2 = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_Salida_M2)
Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
regla_utilizada = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_reglausada)
'Tardanzas promedio reales medidas hasta ese momento
Tard_P1 = g_SIMAN.TallyAverage(g_Tard_1)
Tard_P2 = g_SIMAN.TallyAverage(g_Tard_2)
Tard_P3 = g_SIMAN.TallyAverage(g_Tard_3)
Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
Tpo_fila_entrada_M1 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M1)
Tpo_fila_entrada_M2 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M2)
Tpo_fila_entrada_M3 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M3)
Tpo_fila_M2P1 = g_SIMAN.TallyAverage(g_Tpo_fila_M2P1)
Tpo_fila_M2P2 = g_SIMAN.TallyAverage(g_Tpo_fila_M2P2)
Tpo_fila_M2P3 = g_SIMAN.TallyAverage(g_Tpo_fila_M2P3)

'CALCULO DE TARDANZA ESTIMADA POR TIPO DE PIEZA

'Pieza 1
If Tipo_Pieza = 1 Then
   Tpo_estimado_sistema = (Tpo_salida_M2 - Llegada_sist)
   Retraso_P1 = Tpo_estimado_sistema - DD1
   If Retraso_P1 > 0 Then
      Tard_total_P1 = Retraso_P1
   Else
      Tard_total_P1 = 0
   End If
   Delta_Tard = (Tard_total_P1 - Tard_P1)

'Pieza 2
ElseIf Tipo_Pieza = 2 Then
```

```
   Tpo_estimado_sistema = (Tpo_salida_M2 - Llegada_sist) + (Tpo_fila_entrada_M3 + Tpo_fila_M3P2 +
Tpo_proces_M3P2) + (Tpo_fila_entrada_M1 + Tpo_fila_M1P2 + Tpo_proces_M1P2)
   Retraso_P2 = Tpo_estimado_sistema - DD2
   If Retraso_P2 > 0 Then
      Tard_total_P2 = Retraso_P2
   Else
      Tard_total_P2 = 0
   End If
      Delta_Tard = (Tard_P2 - Tard_total_P2)

'Pieza 3
ElseIf Tipo_Pieza = 3 Then
   Tpo_estimado_sistema = (Tpo_salida_M2 - Llegada_sist) + (Tpo_fila_entrada_M1 + Tpo_fila_M1P3 +
Tpo_proces_M1P3) + (Tpo_fila_entrada_M3 + Tpo_fila_M3P3 + Tpo_proces_M3P3)
   Retraso_P3 = Tpo_estimado_sistema - DD3
   If Retraso_P3 > 0 Then
      Tard_total_P3 = Retraso_P3
   Else
      Tard_total_P3 = 0
   End If
   Delta_Tard = (Tard_P3 - Tard_total_P3)
End If

'ASIGNACIÓN DE RECOMPENSAS
If (Delta_Tard < 0) Then
    ReforzaM = 10
ElseIf (Delta_Tard = 0) Then
    ReforzaM = 1
ElseIf (Delta_Tard > 0 And Delta_Tard <= 1) Then
    ReforzaM = -1
ElseIf (Delta_Tard > 1 And Delta_Tard <= 2) Then
    ReforzaM = -2
ElseIf (Delta_Tard > 2 And Delta_Tard <= 3) Then
    ReforzaM = -3
ElseIf (Delta_Tard > 3 And Delta_Tard <= 4) Then
    ReforzaM = -4
ElseIf (Delta_Tard > 4 And Delta_Tard <= 5) Then
```

```
    ReforzaM = -5
ElseIf (Delta_Tard > 5 And Delta_Tard <= 6) Then
    ReforzaM = -6
ElseIf (Delta_Tard > 6 And Delta_Tard <= 7) Then
    ReforzaM = -7
ElseIf (Delta_Tard > 7 And Delta_Tard <= 8) Then
    ReforzaM = -8
ElseIf (Delta_Tard > 8 And Delta_Tard <= 9) Then
    ReforzaM = -9
ElseIf (Delta_Tard > 9 And Delta_Tard <= 10) Then
    ReforzaM = -10
ElseIf (Delta_Tard > 10) Then
    ReforzaM = -20
End If


'ESCRIBIR DATOS EN ARCHIVO "RECOMPENSAS"
num_maq = 2
g_ExcelApp.Workbooks("Recompensas.xls").Activate
With g_ExcelApp.Workbooks("Recompensas.xls").Worksheets("Sheet1")
    For a = 1 To 3
        If (num_maq = a) Then
            For b = 1 To 3
                If (Tipo_Pieza = b) Then
                    For c = 1 To 3
                        If (regla_utilizada = c) Then
                            Fila = (3 ^ 2 * (a - 1) + 3 * (b - 1) + (c - 1)) + 2
                            .Cells(Fila, 4) = ReforzaM
                        End If
                    Next c
                End If
            Next b
        End If
    Next a
    .Cells(35, 2) = Tipo_Pieza
End With
g_ExcelApp.Workbooks("Recompensas.xls").Save
```

```
'Escribir datos en archivo "Todos los Datos"
g_ExcelApp.Workbooks("Todos los Datos.xls").Activate
With g_ExcelApp.Workbooks("Todos los Datos.xls").Worksheets("Sheet1")
    .Cells(j2, 17) = Num_Entidad
    .Cells(j2, 21) = Tipo_Pieza
    If Tipo_Pieza = 1 Then
    .Cells(j2, 22) = Tard_total_P1
    ElseIf Tipo_Pieza = 2 Then
    .Cells(j2, 22) = Tard_total_P2
    ElseIf Tipo_Pieza = 3 Then
    .Cells(j2, 22) = Tard_total_P3
    End If
    .Cells(j2, 23) = ReforzaM
    j2 = j2 + 1
 End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save
End Sub


Private Sub VBA_Block_5_Fire()
'VBA de INPUTS MAQUINA 3

'Declaración de Variables
  Dim Espera_q3 As Double
  Dim PiezasM3 As Long
  Dim TpoProces_q3 As Double
  Dim regla_3 As Integer
  Dim Tpo_Proces_pieza_en_q3  As Double
  Dim utiliz_M3 As Double
  Dim num_seq_M3 As Integer
  Dim Epsilon As Double
  Dim n As Double

'Obtener los valores de los Tallys y de los atributos
  Espera_q3 = g_SIMAN.TallyAverage(g_Tpoespera_q3)
  PiezasM3 = g_SIMAN.QueueNumberOfEntities(g_Fila3)
```

```
  TpoProces_q3 = g_SIMAN.QueuedEntityAttributesSum(g_Fila3, g_TpoProces_en_q3)
  Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
  Tpo_Proces_pieza_en_q3 = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_TpoProces_en_q3, 0, 0)
  utiliz_M3 = g_SIMAN.ResourceInstantaneousUtilization(g_utilizacion_M3)

'Para leer regla utilizada para seleccionar la pieza
If (ModuloQueues.Data("Ranking(3)") = "FIFO") Then
    regla_utilizada = 1
ElseIf (ModuloQueues.Data("Ranking(3)") = "LVF") Then
    If (ModuloQueues.Data("RankExp(3)") = "Tpo_proces_3") Then
        regla_utilizada = 2
    ElseIf (ModuloQueues.Data("RankExp(3)") = "due date") Then
        regla_utilizada = 3
    End If
End If

' Asignar la regla leída al atributo regla
g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_reglausada) = regla_utilizada


'ESCRIBIR LOS DATOS DE LA CORRIDA EN LOS ARCHIVOS DE EXCEL

'Escribir datos en archivo "Todos los Datos"
g_ExcelApp.Workbooks("Todos los Datos").Activate
 With g_ExcelApp.Workbooks("Todos los Datos").Worksheets("Sheet1")
    .Cells(m3, 25) = Num_Entidad
    .Cells(m3, 27) = Espera_q3
    If PiezasM3 = 0 Then  'cuando pasa la pieza y no hay nada en fila
        If utiliz_M3 = 1 Then   'recurso ocupado
           .Cells(m3, 26) = (PiezasM3 + 1)
           .Cells(m3, 28) = Tpo_Proces_pieza_en_q3
        Else 'máquina desocupada y no hay fila
           .Cells(m3, 26) = PiezasM3
           .Cells(m3, 28) = 0
        End If
    Else 'Si hay piezas en fila cuando llega la entidad activa
        .Cells(m3, 26) = (PiezasM3)
```

```
        .Cells(m3, 28) = (TpoProces_q3 + Tpo_Proces_pieza_en_q3) / (PiezasM3)
    End If
    m3 = m3 + 1
 End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save

'Escribir datos en archivo "Inputs"
g_ExcelApp.Workbooks("Inputs").Activate
  With g_ExcelApp.Workbooks("Inputs").Worksheets("Sheet1")
    .Cells(2, 1) = Espera_q3
    If PiezasM3 = 0 Then  'cuando pasa la pieza y no hay nada en fila
        If utiliz_M3 = 1 Then   'recurso ocupado
           .Cells(2, 2) = (PiezasM3 + 1)
           .Cells(2, 3) = Tpo_Proces_pieza_en_q3
        Else 'máquina desocupada y no hay fila
           .Cells(2, 2) = PiezasM3
           .Cells(2, 3) = 0
        End If
    Else 'Si hay piezas en fila cuando llega la entidad activa
        .Cells(2, 2) = (PiezasM3)
        .Cells(2, 3) = (TpoProces_q3 + Tpo_Proces_pieza_en_q3) / (PiezasM3)
    End If
End With
  g_ExcelApp.Workbooks("Inputs.xls").Save

'Escribir num_maq y Tipo_pieza en Excel para que lo lea MATLAB
With g_ExcelApp.Workbooks("Recompensas.xls").Worksheets("Sheet1")
    .Activate
    .Cells(35, 1) = 3           'número de la máquina
End With
g_ExcelApp.Workbooks("Recompensas.xls").Save

'LLAMAR ALGORITMO Q_LEARNING DE MATLAB
Set oMatlab = CreateObject("MatLab.Application")
oMatlab.Execute ("enableservice('automationserver',true)")
llamarmatlab = oMatlab.Execute("cd C:\Program Files\MATLAB\R2006b\work")
llamarmatlab = oMatlab.Execute("fusificacion")
```

```
Set oMatlab = Nothing

'LEYENDO REGLA DE DECISIÓN Y OTROS DATOS DESDE EL ACHIVO "REGLAS"
g_ExcelApp.Workbooks.Open ("C:\Temp\un agente\Regla.xls")
With g_ExcelApp.Workbooks("Regla.xls").Worksheets("Sheet1")
    regla_3 = .Cells(1, 1)
    Epsilon = .Cells(1, 2)
          n = .Cells(1, 3)
     ValorQ = .Cells(1, 4)
  ValorQmax = .Cells(1, 5)
End With
g_ExcelApp.Workbooks("Regla").Close

'ESCRIBIR REGLA EN ARCHIVO "TODOS LOS DATOS.XLS"
g_ExcelApp.Workbooks("Todos los Datos.xls").Activate
With g_ExcelApp.Workbooks("Todos los Datos").Worksheets("Sheet1")
    .Activate
    .Cells(rg3, 30) = regla_3
    .Cells(rg3, 31) = n
    .Cells(rg3, 32) = Epsilon
    rg3 = rg3 + 1
End With

'ASIGNAR AL MODULO DE ELEMENTOS QUEUES DE LA FILA 3 LA NUEVA REGLA DE DECISIÓN
If regla_3 = 1 Then              'Regla FIFO
    ModuloQueues.Data("Ranking(3)") = "FIFO"
ElseIf regla_3 = 2 Then          'Regla SPT
    ModuloQueues.Data("Ranking(3)") = "LVF"
    ModuloQueues.Data("RankExp(3)") = "Tpo_proces_3"
ElseIf regla_3 = 3 Then          'Regla EDD
    ModuloQueues.Data("Ranking(3)") = "LVF"
    ModuloQueues.Data("RankExp(3)") = "due date"
End If
ThisDocument.Model.Save

'Guardar los datos de los valores QValue y QValue_Max en archivo de Excel
g_ExcelApp.Workbooks("Tardanzas.xls").Activate
```

```
With g_ExcelApp.Workbooks("Tardanzas").Worksheets("Sheet1")
    .Activate
    .Cells(x, 7) = ValorQ
    .Cells(x, 8) = ValorQmax
    x = x + 1
End With
End Sub


Private Sub VBA_Block_6_Fire()
'BLOQUE VBA DE RECOMPENSAS EN MAQUINA 3

'Declaración de variables de SIMAN
Dim Tipo_Pieza As Integer
Dim Llegada_sist As Double
Dim Tpo_estimado_sistema As Double
Dim Tpo_salida_M3 As Double
Dim a As Integer, b As Integer, c As Integer, k As Integer

'Inicialización de variables de prueba
Tpo_estimado_sistema = 0
Retraso_P1 = 0
Retraso_P2 = 0
Retraso_P3 = 0
Tard_total_P1 = 0
Tard_total_P2 = 0
Tard_total_P3 = 0
Delta_Tard = 0
ReforzaM = 0
'Obtener el valor del atributo de la entidad actual
Tipo_Pieza = g_SIMAN.EntitySequenceAttribute(g_SIMAN.ActiveEntity)
Llegada_sist = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_Llegada_sistema)
Tpo_salida_M3 = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_Salida_M3)
Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
regla_utilizada = g_SIMAN.EntityAttribute(g_SIMAN.ActiveEntity, g_reglausada)

'Tardanzas promedio reales medidas hasta ese momento
```

```
Tard_P1 = g_SIMAN.TallyAverage(g_Tard_1)
Tard_P2 = g_SIMAN.TallyAverage(g_Tard_2)
Tard_P3 = g_SIMAN.TallyAverage(g_Tard_3)
Num_Entidad = g_SIMAN.AttributeValue(g_SIMAN.ActiveEntity, g_Num_Entidad, 0, 0)
'Tpo en filas
Tpo_fila_entrada_M1 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M1)
Tpo_fila_entrada_M2 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M2)
Tpo_fila_entrada_M3 = g_SIMAN.TallyAverage(g_Tpo_fila_entrada_M3)
Tpo_fila_M3P1 = g_SIMAN.TallyAverage(g_Tpo_fila_M3P1)
Tpo_fila_M3P2 = g_SIMAN.TallyAverage(g_Tpo_fila_M3P2)
Tpo_fila_M3P3 = g_SIMAN.TallyAverage(g_Tpo_fila_M3P3)

'CALCULO DE TARDANZA ESTIMADA POR TIPO DE PIEZA

'Pieza 1
If Tipo_Pieza = 1 Then
   Tpo_estimado_sistema = (Tpo_salida_M3 - Llegada_sist) + (Tpo_fila_entrada_M2 + Tpo_fila_M2P1 +
Tpo_proces_M2P1)
   Retraso_P1 = Tpo_estimado_sistema - DD1
   If Retraso_P1 > 0 Then
      Tard_total_P1 = Retraso_P1
   Else
      Tard_total_P1 = 0
   End If
   Delta_Tard = (Tard_total_P1 - Tard_P1)

'Pieza 2
ElseIf Tipo_Pieza = 2 Then
   Tpo_estimado_sistema = (Tpo_salida_M3 - Llegada_sist) + (Tpo_fila_entrada_M1 + Tpo_fila_M1P2 +
Tpo_proces_M1P2)
   Retraso_P2 = Tpo_estimado_sistema - DD2
   If Retraso_P2 > 0 Then
      Tard_total_P2 = Retraso_P2
   Else
      Tard_total_P2 = 0
   End If
      Delta_Tard = (Tard_total_P2 - Tard_P2)
```

```
'Pieza 3
ElseIf Tipo_Pieza = 3 Then
   Tpo_estimado_sistema = (Tpo_salida_M3 - Llegada_sist)
   Retraso_P3 = Tpo_estimado_sistema - DD3
   If Retraso_P3 > 0 Then
      Tard_total_P3 = Retraso_P3
   Else
      Tard_total_P3 = 0
   End If
   Delta_Tard = (Tard_total_P3 - Tard_P3)
End If

'ASIGNACIÓN DE RECOMPENSAS
If (Delta_Tard < 0) Then
    ReforzaM = 10
ElseIf (Delta_Tard = 0) Then
    ReforzaM = 1
ElseIf (Delta_Tard > 0 And Delta_Tard <= 1) Then
    ReforzaM = -1
ElseIf (Delta_Tard > 1 And Delta_Tard <= 2) Then
    ReforzaM = -2
ElseIf (Delta_Tard > 2 And Delta_Tard <= 3) Then
    ReforzaM = -3
ElseIf (Delta_Tard > 3 And Delta_Tard <= 4) Then
    ReforzaM = -4
ElseIf (Delta_Tard > 4 And Delta_Tard <= 5) Then
    ReforzaM = -5
ElseIf (Delta_Tard > 5 And Delta_Tard <= 6) Then
    ReforzaM = -6
ElseIf (Delta_Tard > 6 And Delta_Tard <= 7) Then
    ReforzaM = -7
ElseIf (Delta_Tard > 7 And Delta_Tard <= 8) Then
    ReforzaM = -8
ElseIf (Delta_Tard > 8 And Delta_Tard <= 9) Then
    ReforzaM = -9
ElseIf (Delta_Tard > 9 And Delta_Tard <= 10) Then
```

```
    ReforzaM = -10
ElseIf (Delta_Tard > 10) Then
    ReforzaM = -20
End If

'ESCRIBIR DATOS EN ARCHIVO "RECOMPENSAS"
num_maq = 3
g_ExcelApp.Workbooks("Recompensas.xls").Activate
With g_ExcelApp.Workbooks("Recompensas.xls").Worksheets("Sheet1")
    For a = 1 To 3
        If (num_maq = a) Then
            For b = 1 To 3
                If (Tipo_Pieza = b) Then
                    For c = 1 To 3
                        If (regla_utilizada = c) Then
                            Fila = (3 ^ 2 * (a - 1) + 3 * (b - 1) + (c - 1)) + 2
                            .Cells(Fila, 4) = ReforzaM
                        End If
                    Next c
                End If
            Next b
        End If
    Next a
    .Cells(35, 2) = Tipo_Pieza
End With
g_ExcelApp.Workbooks("Recompensas.xls").Save

'Escribir datos en archivo "Todos los Datos"
g_ExcelApp.Workbooks("Todos los Datos.xls").Activate
 With g_ExcelApp.Workbooks("Todos los Datos.xls").Worksheets("Sheet1")
    .Cells(j3, 29) = Num_Entidad
    .Cells(j3, 33) = Tipo_Pieza
    If Tipo_Pieza = 1 Then
    .Cells(j3, 34) = Tard_total_P1
    ElseIf Tipo_Pieza = 2 Then
    .Cells(j3, 34) = Tard_total_P2
    ElseIf Tipo_Pieza = 3 Then
```

```
    .Cells(j3, 34) = Tard_total_P3
    End If
    .Cells(j3, 35) = ReforzaM
    j3 = j3 + 1
 End With
g_ExcelApp.Workbooks("Todos los Datos.xls").Save
End Sub


Private Sub VBA_Block_7_Fire()
'BLOQUE VBA PARA ESCRIBIR LAS TARDANZAS

Dim Tard_P1 As Double
Dim Tard_P2 As Double
Dim Tard_P3 As Double
Dim Tard_sist As Double
Dim Tipo_Pieza As Integer
Dim Tpo_actual As Double
'Obtener los valores de las variables de SIMAN
Tard_P1 = g_SIMAN.TallyAverage(g_Tard_1)
Tard_P2 = g_SIMAN.TallyAverage(g_Tard_2)
Tard_P3 = g_SIMAN.TallyAverage(g_Tard_3)
Tard_sist = g_SIMAN.TallyAverage(g_Tard_sistema)
Tipo_Pieza = g_SIMAN.EntitySequenceAttribute(g_SIMAN.ActiveEntity)
Tpo_actual = g_SIMAN.RunCurrentTime

'Escribir en archivo "Tardanzas" los valores de la corrida
g_ExcelApp.Workbooks("Tardanzas.xls").Activate
  With g_ExcelApp.Workbooks("Tardanzas.xls").Worksheets("Sheet1")
      If Tipo_Pieza = 1 Then
        .Cells(t1, 1) = Tard_P1
        t1 = t1 + 1
      ElseIf Tipo_Pieza = 2 Then
        .Cells(t2, 2) = Tard_P2
        t2 = t2 + 1
      ElseIf Tipo_Pieza = 3 Then
        .Cells(t3, 3) = Tard_P3
```

```
        t3 = t3 + 1
      End If
      .Cells(t, 4) = Tard_sist
      .Cells(t, 5) = Tpo_actual
      t = t + 1
  End With
g_ExcelApp.Workbooks("Tardanzas.xls").Save
End Sub
```

# APENDICE D. Programación del algoritmo Q-learning Difuso en MATLAB

**1. Variables_almacenadas.m**

```
%Valores iniciales de las variables para la primera corrida
 %variables globales
  no_inputs=3; % N° de variables de inputs
  no_etiq=5; % N° etiquetas lingüísticas
  no_outputs=3;  % N° de variables de outputs
  no_reglas=(no_etiq^no_inputs)*no_outputs;   % Calcular N° de reglas

 for i=1:no_reglas
     a_regla_peso(i)=0.5;
 end

QValue_ant=0;
QValue_Max_ant=0;
func_mem1_ant(1:no_etiq)=0;
func_mem2_ant(1:no_etiq)=0;
func_mem3_ant(1:no_etiq)=0;
regla_final=1;
s1_ant=0;
s2_ant=0;
s3_ant=0;
Epsilon=1;
n=1;
q=1;
p=1;
double(n);
double(q);
double(p);
vector=rand(1,25000);
vector_epsilon=rand(1,25000);

save('variables_almacenadas_1','a_regla_peso','QValue_ant','QValue_Max_
ant','QValue_Max','func_mem1_ant','func_mem2_ant','func_mem3_ant','regl
a_final','s1_ant','s2_ant',
's3_ant','Epsilon','n','q','vector','vector_epsilon','p');
```

**2. Fusificacion.m**

```
% Pertenencia a conjuntos difusos

function fusificacion

% obtener los valores de las variables de inputs desde archivo de Excel
```

```
B=xlsread('C:\Temp\un agente\Inputs.xls','A2:C2');

% Clasificación de las variables de inputs
s1=B(1,1);
s2=B(1,2);
s3=B(1,3);

%Inicialización de los arreglos
func_mem1(5)=0;
func_mem2(5)=0;
func_mem3(5)=0;

% Pertenencia a conjuntos difusos

% Para input "Tiempo promedio de espera en fila"
if (s1>=0 & s1<=20)     func_mem1(1)=1;
end
if (s1>=0 & s1<=40)     func_mem1(2)=2;
end
if (s1>=20 & s1<=60)    func_mem1(3)=3;
end
if (s1>=40 & s1<=80)    func_mem1(4)=4;
end
if (s1>=60)             func_mem1(5)=5;
end


% Para input "N° de piezas en fila"
if (s2>=0 & s2<=2)      func_mem2(1)=1;
end
if (s2>=0 & s2<=4)      func_mem2(2)=2;
end
if (s2>=2 & s2<=6)      func_mem2(3)=3;
end
if (s2>=4 & s2<=8)      func_mem2(4)=4;
end
if (s2>=6)              func_mem2(5)=5;
end


% Para input "Tiempo promedio de procesamiento de piezas en fila"
if (s3>=0 & s3<=10)    func_mem3(1)=1;
end
if (s3>=0 & s3<=20)    func_mem3(2)=2;
end
if (s3>=10 & s3<=30)   func_mem3(3)=3;
end
if (s3>=20 & s3<=40)   func_mem3(4)=4;
end
if (s3>=30)            func_mem3(5)=5;
end
```

```
% Llamar algoritmo Q-learning en archivo "algoritmo_qlearning.mat"
algoritmo_qlearning(func_mem1,func_mem2,func_mem3,s1,s2,s3)
```

**3. Algoritmo_qlearning.m**

```
function algoritmo_qlearning(func_mem1,func_mem2,func_mem3,s1,s2,s3)


% INICIALIZACIÓN DE VARIABLES

  %variables globales
  no_inputs=3; % N° de variables de inputs
  no_etiq=5; % N° etiquetas lingüísticas
  no_outputs=3;  % N° de variables de outputs
  no_reglas=(no_etiq^no_inputs)*no_outputs;   % Calcular N° de reglas

  alfa_no_greedy(1:no_reglas)=0;
  alfa_max(1:no_reglas)=0;
  gamma=0.8;
  w_Max(1:no_reglas)=0;
  w_no_greedy(1:no_reglas)=0;
  w_actual(1:no_reglas)=0;


  %SELECCION DE ACCIÓN NO GREEDY (en forma aleatoria)
  load                                                    ('C:\Program
Files\MATLAB\R2006b\work\variables_almacenadas_1.mat','a_regla_peso','Q
Value_ant','QValue_Max_ant','func_mem1_ant','func_mem2_ant','func_mem3_
ant','regla_final','s1_ant','s2_ant','s3_ant','n','Epsilon','q','vector
','vector_epsilon','p');

  aleatorio= vector(q);
  q=q+1;
  probabilidad=1/no_outputs;
  probab_acum=0;

  for i=1:no_outputs
      if  (aleatorio  >=  probab_acum    &    aleatorio  <(probab_acum  +
probabilidad))
          accion_no_greedy = i;
      end
      probab_acum = probab_acum + probabilidad;
  end

  save                                                    ('C:\Program
Files\MATLAB\R2006b\work\variables_almacenadas_1.mat','a_regla_peso','Q
Value_ant','QValue_Max_ant','func_mem1_ant','func_mem2_ant','func_mem3_
ant','regla_final','s1_ant','s2_ant','s3_ant','n','Epsilon','q','vector
','vector_epsilon','p');
```

```
% PARA LA PRIMERA CORRIDA
 load                                                    ('C:\Program
Files\MATLAB\R2006b\work\variables_almacenadas_1.mat','a_regla_peso','Q
Value_ant','QValue_Max_ant','func_mem1_ant','func_mem2_ant','func_mem3_
ant','regla_final','s1_ant','s2_ant','s3_ant','n','Epsilon','q','vector
','vector_epsilon','p');

if (n==1)
  QValue_ant=0;
  QValue_Max_ant=0;
  regla_final=accion_no_greedy;
  func_mem1_ant=func_mem1;
  func_mem2_ant=func_mem2;
  func_mem3_ant=func_mem3;
  s1_ant=s1;
  s2_ant=s2;
  s3_ant=s3;

  n=n+1;

  save                                                    ('C:\Program
Files\MATLAB\R2006b\work\variables_almacenadas_1.mat','a_regla_peso','Q
Value_ant','QValue_Max_ant','func_mem1_ant','func_mem2_ant','func_mem3_
ant','regla_final','s1_ant','s2_ant','s3_ant','n','Epsilon','q','vector
','vector_epsilon','p');


  % GUARDAR REGLA SELECCIONADA EN ARCHIVO DE EXCEL
  C=[regla_final Epsilon n QValue QValue_Max]
  comprobar=xlswrite('C:\Temp\un agente\Regla.xls',C,1);




else     % PARA CORRIDAS SIGUIENTES  (n>1)

    % LLamar archivo con valores de la las variables de la corrida
anterior
  load                                                    ('C:\Program
Files\MATLAB\R2006b\work\variables_almacenadas_1.mat','a_regla_peso','Q
Value_ant','QValue_Max_ant','func_mem1_ant','func_mem2_ant','func_mem3_
ant','regla_final','s1_ant','s2_ant','s3_ant','n','Epsilon','q','vector
','vector_epsilon','p');

  % Respaldar valores de los pesos de las reglas
  for i=1:no_reglas
     w(i)=a_regla_peso(i);
  end


  %Leer el número de la máquina dese donde se está llamando y el tipo
de pieza de la última entidad que recibió la recompensa en esa máquina
```

```
  F=xlsread('C:\Temp\un agente\Recompensas.xls',1,'A35:B35');
  num_maq=F(1,1);
  num_seq=F(1,2);   %Tipo de pieza


  % Leer recompensa desde archivo "Recompensas.xls" dependiendo de los
  % datos anteriores
  for i=1:3
     if(num_maq==i)
        for j=1:3
           if (num_seq==j)
              for m=1:3
                 if(regla_final==m)
                    Fila=(3^2*(i-1)+ 3*(j-1)+(m-1))+2;
                    A               =               xlsread('C:\Temp\un
agente\Recompensas.xls',1,'A2:D28');
                    recompensa = A(Fila,4);
                 end
              end
           end
        end
     end
  end


  % Calcular TD error de la corrida anterior
  TD_error=gamma*QValue_Max_ant-QValue_ant + recompensa;


  %ACTUALIZACIÓN DE LOS PESOS DE LA REGLA SELECCIONADA EN LA CORRIDA
ANTERIOR

  alfa_anterior(1:no_reglas)=0;
  for i=1:no_etiq
      k1=func_mem1_ant(i);
      if(k1>0)
              for j=1:no_etiq
                  k2=func_mem2_ant(j);
                  if(k2>0)
                     for m=1:no_etiq
                         k3=func_mem3_ant(m);
                         if(k3>0)
                            reglas_activas                                =
1+((no_etiq^2)*no_outputs)*(k1-1)+(no_etiq*no_outputs)*(k2-
1)+(no_outputs)*(k3-1);

                            for pos=1:no_outputs
                                if regla_final==pos
                                   alfa_anterior(reglas_activas+(pos-
1))=obtener_regla_alfa(s1_ant, s2_ant,s3_ant, k1,k2,k3);
```

```
                                    w(reglas_activas+(pos-
1))=w(reglas_activas+(pos-1))+
TD_error*alfa_anterior(reglas_activas+(pos-1));
                                end
                            end
                        end
                    end
                end
            end
  end

 %  Guardar  los  valores  delos  pesos  actualizados  en  variable
a_regla_peso
  for i=1:no_reglas
          a_regla_peso(i)=w(i);
  end


  % CÁLCULO DE LOS PESOS Y VALORES Q DE LA CORRIDA ACTUAL (w(t+1) y
Q(t+1))
  for i=1:no_etiq
        k1=func_mem1(i);
        if(k1>0)
          for j=1:no_etiq
             k2=func_mem2(j);
             if(k2>0)
                for m=1:no_etiq
                   k3=func_mem3(m);
                   if(k3>0)
                     reglas_activas  =  1+((no_etiq^2)*no_outputs)*(k1-
1)+(no_etiq*no_outputs)*(k2-1)+(no_outputs)*(k3-1);

                     %Traspasar  los  pesos  de  las  reglas  activadas  a
vector w_actual
                     for pos=1:no_outputs
                        w_actual(reglas_activas+(pos-
1))=a_regla_peso(reglas_activas+(pos-1));
                     end

                     % Cálculo de w_no_greedy
                     for pos=1:no_outputs
                        if accion_no_greedy==pos
                           w_no_greedy(reglas_activas+(pos-
1))=w_actual(reglas_activas+(pos-1));
                           alfa_no_greedy(reglas_activas+(pos-
1))=obtener_regla_alfa(s1,s2,s3,k1,k2,k3);
                        end
                     end

                   end
                end
```

```
              end
          end
        end
  end


  % CÁLCULO DE REGLA REGLA ÓPTIMA (con w Máximo)

  [valor_accion_max,posicion]=max(w_actual);
  residuo=mod(posicion,3);
  if     residuo==1   accion_max=1;
  elseif residuo==2   accion_max=2;
  elseif residuo==0   accion_max=3;
  end


  %  CÁLCULO DE VECTOR w_Max Y ALFA_MAX
  for i=1:no_etiq
     k1=func_mem1(i);
     if(k1>0)
        for j=1:no_etiq
           k2=func_mem2(j);
           if(k2>0)
              for m=1:no_etiq
                 k3=func_mem3(m);
                 if(k3>0)
                    reglas_activas   =   1+((no_etiq^2)*no_outputs)*(k1-
1)+(no_etiq*no_outputs)*(k2-1)+(no_outputs)*(k3-1);

                    % Obtener vector de pesos de acción óptima
                    for pos=1:no_outputs
                       if accion_max==pos
                          w_Max(reglas_activas+(pos-
1))=a_regla_peso(reglas_activas+(pos-1));
                          alfa_max(reglas_activas+(pos-
1))=obtener_regla_alfa(s1,s2,s3,k1,k2,k3);
                       end
                    end
                 end
              end
           end
        end
     end
  end

%  CÁLCULO DE VALORES q
  QValue = w_no_greedy * alfa_no_greedy';
  QValue_Max= w_Max * alfa_max';

  QValue_ant = QValue;
  QValue_Max_ant = QValue_Max;
  func_mem1_ant = func_mem1;
```

```
  func_mem2_ant = func_mem2;
  func_mem3_ant = func_mem3;
  s1_ant = s1;
  s2_ant = s2;
  s3_ant = s3;

  %Modificar el valor de Epsilon
  if (n>=1 & n<1000)         Epsilon=1
  elseif (n>=1000 & n<2000)  Epsilon=0.8
  elseif (n>=2000 & n<3000)  Epsilon=0.6
  elseif (n>=3000 & n<4000)  Epsilon=0.4
  elseif (n>=4000 & n<5000) Epsilon=0.2
  elseif (n>=5000)          Epsilon=0
  end




  % Determinar la acción final con estrategia explotación-exploración
  % (1-Epsilon)
  no_aleatorio = vector_epsilon(p);

  if no_aleatorio <= Epsilon
     regla_final=accion_no_greedy;
  else  % (no_aleatorio<(1-Epsilon))
     regla_final = accion_max;
  end


  n=n+1;
  p=p+1;

 %Grabar variables para próxima corrida
   save                                                    ('C:\Program
Files\MATLAB\R2006b\work\variables_almacenadas_1.mat','a_regla_peso','Q
Value_ant','QValue_Max_ant','func_mem1_ant','func_mem2_ant','func_mem3_
ant','regla_final','s1_ant','s2_ant','s3_ant','n','Epsilon','q','vector
','vector_epsilon','p');


 % GUARDAR REGLA SELECCIONADA EN ARCHIVO DE EXCEL

  C=[regla_final Epsilon n QValue QValue_Max]
  comprobar=xlswrite('C:\Temp\un agente\Regla.xls',C,1);


end  % (del n>1)
```

**4. Obtener_regla_alfa.m**

```
%Función para obtener los grados de membresía de cada input a cada
conjunto %difuso, y por lo tanto se obtiene la regla Alfa como
resultado final

function [Regla_alfa]=obtener_regla_alfa(s1,s2,s3,k1,k2,k3)

% Regla alfa de s1
if (k1==1 | k1==2)      s1_eval = s1;
end
if (k1==3)              s1_eval = s1-20;
end
if (k1==4)              s1_eval = s1-40;
end
if (k1==5)              s1_eval = s1-60;
end

if (k1==2 | k1==3 | k1==4)
    if (s1_eval<20)         alfa_s1 = (s1_eval/20);
    elseif (s1_eval>20)     alfa_s1 = (s1_eval/20) -1;
    elseif (s1_eval==20)    alfa_s1 = 1;
    end
elseif (k1==1)              alfa_s1 = 1 - (s1_eval/20);
elseif (k1==5)
    if (s1_eval >20)        alfa_s1=1;
    else                    alfa_s1 = (s1_eval/20);
    end
end


% Regla alfa de s2
if (k2==1 | k2==2)      s2_eval = s2;
end
if (k2==3)              s2_eval = s2-2;
end
if (k2==4)              s2_eval = s2-4;
end
if (k2==5)              s2_eval = s2-6;
end

if (k2==2 | k2==3 | k2==4)
    if (s2_eval<2)         alfa_s2 = (s2_eval/2);
    elseif (s2_eval>2)     alfa_s2 = (s2_eval/2) -1;
    elseif (s2_eval==2)    alfa_s2 = 1;
    end
elseif (k2==1)              alfa_s2 = 1 - (s2_eval/2);
elseif (k2==5)
    if (s2_eval >2)       alfa_s2=1;
    else                    alfa_s2 = (s2_eval/2);
    end
end
```

```
% Regla alfa para s3
if (k3==1 | k3==2)      s3_eval = s3;
end
if (k3==3)              s3_eval = s3-10;
end
if (k3==4)              s3_eval = s3-20;
end
if (k3==5)              s3_eval = s3-30;
end

if (k3==2 | k3==3 | k3==4)
    if (s3_eval<10)          alfa_s3 = (s3_eval/10);
    elseif (s3_eval>10)      alfa_s3 = (s3_eval/10) -1;
    elseif (s3_eval==10)     alfa_s3 = 1;
    end
elseif (k3==1)               alfa_s3 = 1 - (s3_eval/10);
elseif (k3==5)
    if (s3_eval > 10)       alfa_s3=1;
    else                     alfa_s3 = (s3_eval/10);
    end
end


% Si se utiliza T-norma producto
Regla_alfa = alfa_s1*alfa_s2*alfa_s3;
```