

Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación



Trabajo de Diploma

Algoritmos para problemas de secuenciación de tareas en ambientes *online*

Autor:

Jessica Coto Palacio

Tutores:

Dra. Yailen Martínez Jiménez

MSc. Beatriz M. Méndez Hernández

Santa Clara, 2015

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Ingeniería Informática, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del tutor

Firma del jefe del
Laboratorio

Dedicatoria

A mi mamá, mi abuela y mi tía por ser para mí, dignos ejemplos de mujeres
luchadoras y perseverantes.

Agradecimientos

A mi mamuti linda por ser una excelente madre, por apoyarme en todo momento, tolerarme tanto y pelearme poco. A mi wiwi por ser mi segunda madre, mucho más peleona que la primera, pero la quiero igual; gracias por complacerme en todos mis caprichos y malcriarme de vez en cuando. A mi tía hermosa Maray que sin ella no hubiese sido posible todos estos cinco años de universidad, gracias por malcriarme y cumplir mis caprichos y por tener paciencia conmigo; este título de oro te lo dedico a ti. También le agradezco a Hamlet por apoyar a mi tía y a su vez a mí.

A mi tutora Bea gracias por toda la ayuda brindada durante el transcurso de esta tesis y por la preocupación constante aun estando en la distancia durante un tiempo. A Yailen por ser más que mi tutora, mi amiga; gracias por preocuparte por mí y “correr” conmigo cuando ocurría algún problema y sobre todo por dedicarme gran parte de tu tiempo sagrado. A Vladi y Niurka por toda la ayuda brindada.

A mi cuchín (Alejandro) y a toda su gran familia por acogerme tan bien y quererme tanto.

A Rosa Mary como cariñosamente le digo a Rosmery por ser una gran amiga durante estos años y los que nos quedan porque es una amistad para toda la vida, gracias por todas las ayudas que me brindaste que fueron muchas. A Sieiry por todoooo, gracias, por los mensajitos lindos que me hacían llorar en fechas conmemorativas, por brindarme su amistad y ayuda en todo momento, es que talento le sobra jaja. A Mi (Claudia) por estar presente en todo momento, por ser mi confidente y amiga; por tenerme presente y siempre estar guardándome cosas en cada viaje que hacía o en cada oportunidad que tenía jaja. En fin, a todas las amistades que hice durante estos años de Universidad: Grettel (súper buena, pero cuando coge genio...), Naylet (siempre le saca lo bueno de las cosas malas y muy carismática), Ferna con su espaldota jaja y Dayan. También a mis amigas de tantos años Gabriela, Yanelis y Elizabeth.

A todos los profesores que me ayudaron a lo largo de la carrera como María Matilde, Gálvez, Gheisa, entre otros.

Pensamiento

Para ser exitoso no tienes que hacer cosas extraordinarias. Haz cosas ordinarias, extraordinariamente bien.

Resumen

El problema de secuenciación de tareas es usualmente definido como “el problema de acomodar los recursos en el tiempo para realizar un conjunto de tareas”. Algunos de esos problemas ocurren en ambientes *online*, debido a la necesidad de secuenciar con información incompleta, pues en ciertos puntos las decisiones deben ser tomadas sin tener conocimiento sobre eventos futuros.

El uso de agentes y de aprendizaje reforzado combinado con reglas de despacho para dar solución a este tipo de problemas resulta un enfoque poco explorado, de ahí la idea de utilizar dichas técnicas en la solución del mismo, ya que este tipo de aprendizaje es uno de los que está siendo actualmente aplicado en la solución de problemas de secuenciación de forma *offline*. El aprendizaje de los agentes en cada punto de decisión estará determinado por un algoritmo de aprendizaje, y para esto se implementaron dos alternativas, el *Q-Learning* y el *Learning Automata*.

El enfoque de solución a problemas de secuenciación de tareas en ambientes *online* propuesto en esta tesis se basa en un caso de estudio existente en la literatura, y a partir del mismo se definen otros escenarios de producción. Los resultados obtenidos son validados utilizando pruebas estadísticas, a partir de 315 instancias, divididas por escenario y combinación posible de cada algoritmo con varios parámetros reportados en la literatura.

Abstract

Scheduling problems are usually defined as “the problem of allocating resources in time in order to perform a set of tasks”. Some of these problems occur in online environments, due to the need of scheduling with incomplete information, because in certain states decision have to be made without knowledge about future events.

The use of agents and reinforcement learning combined with dispatching rules in order to solve this kind of problems is a poorly explored approach, that is why we decided to use these techniques for its solution, because this type of learning is being widely used in the solution of offline scheduling problems. The learning process of the agents in each decision point will be determined by a learning algorithm, and for this we have two alternatives, Q-Learning and Learning Automata.

The approach to solve online scheduling problems proposed in this thesis is based on a case study from the literature, and starting from this basic idea we define other production scenarios. The results are validated using statistical tests, for this we use 315 instances, divided by scenario and also by possible combination of each algorithm with different parameters reported in the literature.

Tabla de Contenidos

Introducción	1
Capítulo 1: Problemas de secuenciación de tareas en ambientes <i>online</i>	4
1.1 Problemas de Secuenciación de Tareas.....	4
1.2 Notación empleada en los problemas de secuenciación.....	5
1.3 Tipos de soluciones	6
1.4 Tipos de problemas	7
1.5 Ambientes <i>offline</i> y <i>online</i>	7
1.5.1 Ambientes <i>online</i>	8
1.6 Métodos de solución para problemas de secuenciación de tareas	9
1.7 Aprendizaje Reforzado	11
1.7.1 <i>Q-Learning</i>	14
1.7.2 <i>Learning Automata</i>	15
1.7.3 Mecanismos de selección de acciones	16
1.8 Sistemas Multi-Agente.....	18
1.9 Aplicación de RL a problemas de secuenciación	19
1.10 Conclusiones Parciales	20
Capítulo 2: <i>Q-Learning</i> y <i>Learning Automata</i> para problemas de secuenciación de tareas en ambientes <i>online</i>	21
2.1 Descripción del problema	21
2.2 Posibles escenarios.....	22
2.3 Relación entre los algoritmos de aprendizaje y las reglas de despacho.....	24
2.4 Aplicación del algoritmo QL a problemas de secuenciación de tareas <i>online</i>	25
2.4.1 Ejemplo de aplicación del QL	26
2.5 Aplicación del algoritmo LA a problemas de secuenciación de tareas <i>online</i>	28
2.5.1 Ejemplo de aplicación del LA.....	29
2.6 Diagrama de clases de la aplicación.....	31
2.7 Interfaz visual de la relación entre las máquinas en las diferentes etapas	35
2.8 Conclusiones Parciales	37
Capítulo 3: Resultados experimentales.....	38
3.1 Especificaciones del experimento.....	38
3.1.1 Descripción del experimento.....	40
3.2 Marco experimental estadístico	43
3.3 Análisis estadísticos de los resultados.....	44
3.4 Colas de las máquinas	50
3.5 Conclusiones Parciales	52
Conclusiones	53
Recomendaciones	54

Referencias Bibliográficas 55

Anexos 59

 Anexo 1 59

 Anexo 2 60

 Anexo 3 62

 Anexo 4 63

 Anexo 5 65

 Anexo 6 66

Introducción

La secuenciación de tareas es la asignación de entidades (personas, tareas, vehículos, conferencias, exámenes, reuniones, etc.) a un modelo en el espacio de tiempo, de tal manera que las restricciones impuestas sean satisfechas y ciertas metas sean logradas, es decir, 'un plan para ejecutar un trabajo o alcanzar un objetivo, especificando el orden y el tiempo asignado para cada parte'. En los problemas del mundo real, expresar las condiciones que hacen una secuenciación más preferible que otra e incorporar esta información en un sistema automatizado, no resulta una tarea fácil.

Es fácil notar que el problema de la secuenciación de tareas constituye una labor que todos hemos desarrollado en algún momento, quizás sin notarlo. En el día a día van ocurriendo o se tienen planificadas tareas que se desean desarrollar en un tiempo determinado, lo cual conforma un ejemplo más de problemas de secuenciación.

En los procesos de manufactura la secuenciación se define como un proceso de optimización que asigna recursos limitados, específicamente máquinas, en el tiempo entre actividades que se pueden ejecutar en paralelo. Esta asignación tiene que cumplir un conjunto de restricciones que reflejan las relaciones temporales entre las actividades y las limitaciones de capacidad del conjunto de recursos compartidos (Wang & Shen 2007).

Los problemas de secuenciación de tareas se pueden desarrollar en ambientes *offline* o ambientes *online*, donde su principal diferencia radica en que del primero se conoce toda la información necesaria de antemano, como por ejemplo, el tiempo de procesamiento de las operaciones, mientras que del segundo no se conoce cuando acaba un trabajo hasta que termine de ser procesado.

La mayoría de las investigaciones en el área de la secuenciación de tareas se han enfocado en el desarrollo de procedimientos exactos para la generación de una solución base asumiendo que se tiene toda la información necesaria y un ambiente determinístico (Herroelen & Leus 2005). En los últimos años el Aprendizaje Reforzado ha servido de plataforma para la solución a problemas de secuenciación de tareas (Horn, J., Nafpliotis, N. & Goldberg 1994; E. 1999b Mariano, C. & Morales 1999; Mariano, C. & Morales 2000; E. 1999a Mariano, C. & Morales 1999), donde cada agente es entrenado para alcanzar una meta u objetivo específico.

El aprendizaje denota cambios en un sistema que lo hacen capaz de realizar la misma tarea o tareas de forma más eficiente y más efectiva la próxima vez (Simon & Lea

1973). El Aprendizaje Reforzado también es conocido como un *framework* muy popular para el diseño de agentes que interactúan con su ambiente en aras de aprender a resolver una tarea específica a través de interacciones repetidas. En muchas ocasiones resulta más eficiente contar con más de un agente para llevar a cabo la tarea prevista, esto es conocido como sistema multi-agentes, el cual consta de una red de agentes que trabajan juntos para resolver diversos problemas.

El creciente interés en el uso de este tipo de sistema en problemas del mundo real viene dado por su habilidad para resolver problemas que son muy grandes para ser resueltos por un agente centralizado, y también por la habilidad de obtener soluciones donde la experiencia es distribuida, como por ejemplo, problemas de secuenciación en ambientes de manufactura.

Debido a lo antes expuesto surge como **problema de investigación** la no existencia de un algoritmo para problemas de secuenciación de tareas en ambientes *online* usando Aprendizaje Reforzado.

Para dar solución a este problema se plantea como **objetivo general** de esta tesis: Extender la herramienta SMA-JSSP v1.0, la cual optimiza problemas de secuenciación de tareas en ambientes *offline*, para resolver problemas en ambientes *online*.

Este objetivo se desglosa en los siguientes **objetivos específicos**:

- Caracterizar los diferentes algoritmos reportados en la literatura para resolver problemas de secuenciación de tareas en ambientes *online*.
- Implementar los algoritmos *Q-Learning* y *Learning Automata* pertenecientes al Aprendizaje Reforzado para problemas de secuenciación de tareas en ambientes *online*.
- Comparar el comportamiento de los algoritmos implementados.
- Integrar los algoritmos implementados en la herramienta SMA-JSSP v1.0.

Para dar solución al problema de investigación y a los objetivos de este trabajo se dará respuesta a las siguientes **preguntas de investigación**:

- ¿Cuáles son las principales características que deben presentar los algoritmos para problemas de secuenciación *online*?
- ¿Son los algoritmos basados en Aprendizaje Reforzado una buena opción para ambientes *online*?

- ¿Cuál de los algoritmos implementados muestra mejor desempeño ante problemas *online*?

Como **justificación** a esta investigación se plantea que:

En los problemas donde sea necesario generar una secuencia de tareas se puede conocer o no toda la información necesaria (cantidad de trabajos y el tiempo de procesamiento en cada máquina) para generar una secuencia que cumpla con las restricciones impuestas. En la literatura se ha hecho más énfasis en generar secuencias en ambientes *offline* que en ambientes *online*. Un algoritmo para problemas de secuenciación de tareas en ambientes *online* utilizando Aprendizaje Reforzado sería de gran utilidad para afrontar dichos problemas de falta de información y convertiría a SMA-JSSP v1.0 en una herramienta mucho más completa para la construcción de secuenciaciones en ambos tipos de ambiente.

La tesis está estructurada en tres capítulos. En el capítulo uno se realiza un estudio de los problemas de secuenciación de tareas, haciendo énfasis en su notación, los tipos de soluciones que existen y su clasificación de acuerdo a las restricciones que presentan. También se describen los ambientes en que se pueden desarrollar los problemas de secuenciación de tareas y por último de los algoritmos *Q-Learning* y *Learning Automata* pertenecientes al Aprendizaje Reforzado. En el segundo capítulo se plantea un caso de estudio reportado en la literatura y se propone una alternativa de solución diferente, a través de la simulación de escenarios más flexibles y escogidos por el usuario, utilizando interfaces gráficas. Más adelante, en el capítulo tres se realizan las pruebas pertinentes para evaluar el comportamiento de los algoritmos *Q-Learning* y *Learning Automata*, para tres escenarios específicos y con diferentes configuraciones de parámetros en ambos algoritmos. Por último, se plantean las conclusiones y las recomendaciones.

Capítulo 1: Problemas de secuenciación de tareas en ambientes online

Los problemas de secuenciación están presentes en cada contexto donde tiene que realizarse un conjunto dado de tareas y requieren la asignación de los recursos para espacios de tiempo. Esto es algo que usualmente se realiza en la vida diaria, pero cuando las restricciones que deben conocerse aumentan y el número de tareas y recursos crecen, entonces construir un horario que satisfaga todos los requisitos no es tan sencillo.

1.1 Problemas de Secuenciación de Tareas

Los problemas de secuenciación de tareas se reflejan en muchos de los procesos que se llevan a cabo diariamente, donde se hace necesario realizar un conjunto de operaciones en espacios de tiempo determinados y a su vez asignar recursos limitados.

(Zhang 1996) se refiere a la secuenciación de tareas como el proceso de seleccionar planes alternativos y asignar recursos y tiempos a un conjunto de actividades en un plan. Debido a esto, las asignaciones deben obedecer a restricciones que reflejan las relaciones temporales entre las actividades y las limitaciones de los recursos compartidos. Las asignaciones que son llevadas a cabo durante el proceso de secuenciación afectan la optimalidad de una secuencia con relación a criterios como el costo, la tardanza, o el rendimiento en específico.

Generalmente los problemas de secuenciación son clasificados como NP-Complejos (Garey et al. 1976), y el tiempo de cómputo se incrementa exponencialmente de acuerdo al tamaño del problema, convirtiendo a los problemas de secuenciación en ambientes de manufactura en uno de los más complejos de resolver (Shen 2002).

Existen diferentes enfoques para resolver este tipo de problemas, el clásico consiste en construir un modelo matemático, generalmente basado en Programación Mixta en Enteros y luego aplicar un algoritmo de búsqueda como el algoritmo de Ramificación y Acotamiento para encontrar la solución óptima (Méndez et al. 2006). Sin embargo, dicho enfoque presenta sus inconvenientes, a medida que el número de recursos disponibles en el problema se incrementa, o el número de operaciones a planificarse crece, este no será capaz de encontrar la solución óptima en un tiempo computacional razonable (Ruiz et al. 2008). Por consiguiente, los métodos heurísticos se convierten en el centro de atención, pues son capaces de encontrar buenas soluciones de manera eficiente en un tiempo computacional adecuado (Zhang 1996).

En un problema de secuenciación muchos factores pueden influir en el tiempo que toma realizar un conjunto de tareas, entre ellos se puede citar el costo de las operaciones, el orden de las secuencias, entre otros, la Figura 1.1 muestra un esquema de los mismos tomado de (Martínez Jiménez 2012), por supuesto que estos factores no aparecen necesariamente al mismo tiempo.

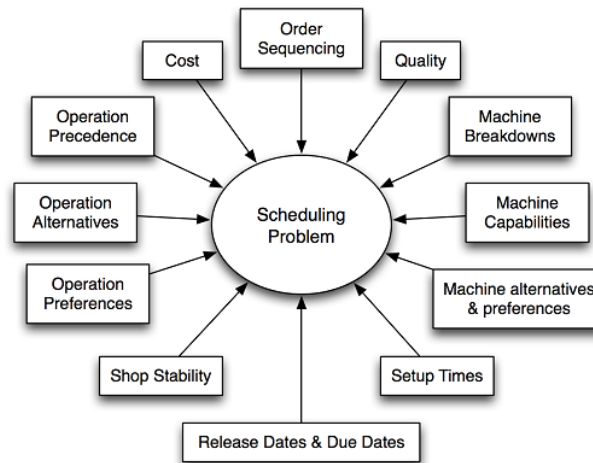


Fig. 1.1 Diferentes factores que pueden influenciar un problema de secuenciación.

Los problemas de secuenciación se pueden clasificar de acuerdo al ambiente de las máquinas, las características de los trabajos y la función objetivo. Esta clasificación se conoce normalmente como $\alpha|\beta|\gamma$ (Graham et al. 1979), donde α representa el ambiente de las máquinas, β las características de los trabajos y γ el criterio a optimizar (Martínez Jiménez 2012).

1.2 Notación empleada en los problemas de secuenciación

En la definición de un problema de secuenciación se debe tener claridad en cuanto a la notación utilizada, a continuación se muestra una lista de la notación que será usada a lo largo de esta tesis:

m	Número de máquinas
n	Número de trabajos
M_i	Máquina i , donde $i = \{1, \dots, m\}$
J_j	Trabajo j , donde $j = \{1, \dots, n\}$
C_j	Tiempo de terminación del trabajo j
P_j	Tiempo de procesamiento del trabajo j

o_{ij}	La i -ésima operación del trabajo j
s_{ij}	Tiempo de inicio de la operación i , trabajo j
c_{ij}	Tiempo de terminación de la operación i , trabajo j
p_{ij}	Tiempo de procesamiento de la operación i , trabajo j
r_j	Fecha de inicio del trabajo j
d_j	Fecha de vencimiento del trabajo j
C_{max}	El <i>makespan</i> , máximo C_j entre todos los trabajos, $\max\{C_1, \dots, C_n\}$

La Figura 1.2 muestra un ejemplo de un diagrama de tiempo de la operación o_{ij} donde se resume la notación antes mencionada.

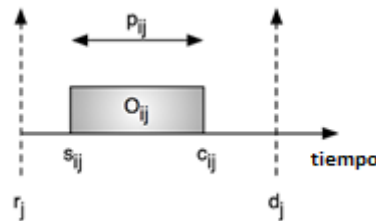


Fig. 1.2 Diagrama de tiempos de una operación

1.3 Tipos de soluciones

Según lo reportado en la literatura, cualquier solución factible puede clasificarse en:

1. Sin-retardo: ningún recurso se mantiene inactivo si hay al menos una operación que pueda ser procesada.
2. Semi-activa: la única opción para que una operación comience antes es cambiar el orden de procesamiento en un recurso.
3. Activa: si no es posible construir otra solución a través de cambios en el orden de procesamiento en los recursos, con al menos una operación terminando más temprano y ninguna terminando más tarde.

1.4 Tipos de problemas

Cuando un trabajo tiene un número fijo de operaciones que requieren diferentes máquinas se dice que se está en presencia de un problema tipo 'shop', y de acuerdo a las restricciones que presente se puede clasificar en:

- *Open Shop*: Existen '*m*' máquinas y cada trabajo tiene que ser procesado en cada una de ellas. No existen restricciones de orden, lo que implica que cada trabajo puede pasar por las máquinas siguiendo rutas distintas.
- *Job Shop*: Los trabajos pasan por cada máquina sólo una vez, existen restricciones de orden, sin embargo el orden no es el mismo para todos los trabajos.
- *Flow Shop*: Todos los trabajos pasan por todas las máquinas en el mismo orden, es decir, la primera tarea de todos los trabajos se debe ejecutar en la máquina 1, la segunda tarea en la máquina 2, y así sucesivamente.
- *Job Shop_Flexible*: Es una generalización del *Job Shop* y el ambiente de máquinas no relacionadas. Cada operación puede procesarse por cualquier máquina de un conjunto, dichas máquinas pueden diferir en velocidad.
- *Flow Shop Híbrido*: Extensión del *Flow Shop* donde las máquinas tienen tiempos de preparación y se produce en lotes (más de un trabajo a la vez) (Martínez Jiménez 2012).

1.5 Ambientes *offline* y *online*

Los problemas de secuenciación de tareas se pueden desarrollar en dos tipos de ambientes: *offline* u *online*; Se utiliza uno u otro dependiendo del conocimiento que se tiene de la información para resolver dicho problema.

Cuando se resuelve un problema de secuenciación *offline* se asume que toda la información se conoce de antemano: el número de trabajos, el número de máquinas, el número de operaciones por trabajo, el tiempo de procesamiento de las operaciones, las restricciones de precedencia y las restricciones del problema (las cuales varían en dependencia del tipo de problema a resolver). Por otro lado, si el problema es *online*, entonces la información no está completa, pues los tiempos de procesamiento de las operaciones pertenecientes a las distintas órdenes son modelados como variables aleatorias. Esto significa que el tiempo de procesamiento de un trabajo completo no se conoce hasta que el mismo no se ha ejecutado totalmente. Esta investigación se centra en los problemas de secuenciación *online*.

1.5.1 Ambientes *online*

Los algoritmos de secuenciación *online* empezaron a ser investigados por los años 60, pero un estudio detenido y extensivo sólo se lleva a cabo hace 15 años, después de introducirse formalmente el concepto de análisis competitivo. Actualmente existe una gran variedad de literatura en el área de la secuenciación *online*.

Durante los últimos 10 años la secuenciación *online* ha recibido un gran interés por parte de la comunidad científica (Albers 1997; Bartal et al. 1995; Karger et al. 1996; Shmoys et al. 1995). Las características de los problemas investigados se basan en modelos diversos (máquinas idénticas, relacionadas o inconexas), formatos de procesamiento diferente (la planificación preventiva o poco preventiva) y diferentes funciones objetivos (*makespan*, *flowtime*, *tardiness*, etc).

En problemas de secuenciación *online* una secuencia de trabajos $\sigma = J_1, J_2, \dots, J_n$ tiene que procesarse en un número determinado de máquinas. Los trabajos arriban al sistema uno a uno, y cuando un nuevo trabajo llega tiene que ser inmediatamente despachado hacia una de las máquinas, sin tener conocimiento sobre trabajos futuros. El objetivo es optimizar una función objetivo dada.

Muchos algoritmos *online* para diferentes problemas de secuenciación se han propuesto y evaluado con análisis competitivo. Sin embargo, no es usual encontrar una evaluación experimental de los algoritmos.

En (Albers & Schröder 2001) se presenta un estudio experimental de algoritmos para uno de los principales problemas de secuenciación *online*. A este problema se le conoce como el problema de Graham y se ha investigado desde un punto de vista teórico (Albers 1997; Bartal et al. 1995; Karger et al. 1996). En el problema de Graham, una secuencia de trabajos tiene que ser secuenciada en m máquinas paralelas idénticas.

Cada vez que un trabajo nuevo J_t , $1 \leq t \leq n$ arriba, su tiempo de procesamiento P_t es conocido de antemano. Cada trabajo tiene que asignarse inmediatamente a una de las máquinas, sin conocimiento de trabajos futuros, con el propósito de optimizar la función objetivo definida. A menudo en estos ambientes no se permiten las preferencias (*preemption*) ya que los altos requisitos de memoria de los trabajos hacen que la preferencia sea sumamente cara. Los tiempos de ejecución de los trabajos son conocidos al menos de forma aproximada, ya que los usuarios están usualmente obligados a dar una estimación para los requisitos de CPU de sus trabajos. El objetivo de minimizar el *makespan* se traduce en obtener una alta utilización de las máquinas.

Además de su relevancia práctica, el problema de Graham es importante porque es la raíz de muchas variantes problemáticas donde, por ejemplo, la preferencia es permitida, existen restricciones de precedencia entre los trabajos, o las máquinas funcionan a velocidades diferentes.

En (Mao et al. 1995) se estudia el problema de secuenciación de una sola máquina, donde se definen dos algoritmos *online*: *First-Come-First-Served* (FCFS) y *Shortest-Available- Job-First* (SAJF). En ambos algoritmos se mantiene una cola que contiene todos los trabajos que arriban pero no se han ejecutado. En FCFS, los trabajos en la cola están listados de forma no decreciente teniendo en cuenta las fechas de liberación r_j (los trabajos con el mismo r_j son ordenados de la misma forma teniendo en cuenta el tiempo de procesamiento P_j), mientras que en SAJF, los trabajos en la cola están ordenados teniendo en cuenta solo el P_j . Cuando la máquina se desocupa después de completar la ejecución de un trabajo, el primer trabajo en la cola es asignado a la máquina para su ejecución. Cuando un trabajo nuevo llega, se inserta en su posición correspondiente en la cola. FCFS y SAJF son *online* ya que se toman decisiones solo basadas en la información sobre la cola disponible. También son conservadores, lo cual quiere decir que la máquina no está nunca desocupada cuando hay trabajos en la cola. Aunque estos dos algoritmos son comúnmente usados, pocos resultados analíticos están disponibles.

1.6 Métodos de solución para problemas de secuenciación de tareas

La mayoría de los problemas de secuenciación de tareas son NP-Complejos y su tamaño es generalmente grande, es por esto que los métodos de optimización exactos fallan en proveer soluciones óptimas en un tiempo razonable y comienzan a utilizarse más los métodos heurísticos.

Uno de los enfoques básicos más utilizados en la solución de problemas de secuenciación son las reglas de despacho, debido a su baja complejidad y facilidad de implementación. Una regla de despacho es una estrategia de secuenciación a través de la cual se le asigna una prioridad a cada trabajo que está esperando para ser ejecutado en una máquina específica. Cuando una máquina está disponible, una regla basada en prioridades inspecciona los trabajos en espera y el que tiene la prioridad más alta se selecciona para procesarse (Nhu Binh HO 2005).

Con el paso de los años numerosos investigadores han propuesto una gran cantidad de reglas de despacho (Blackstone et al. 1982; Haupt 1989; Ramasesh 1990). Sin embargo, se demostró que no existe ninguna regla que funcione adecuadamente para

todos los criterios importantes (funciones objetivo), por ejemplo el *flowtime* y la tardanza.

La elección de la regla de despacho a utilizar depende del objetivo a optimizarse. En general, se ha observado que las reglas basadas en tiempo de procesamiento funcionan mejor bajo condiciones donde hay alta carga de trabajo, mientras las reglas basadas en la fecha de vencimiento funcionan mejor cuando la carga es baja (Conway 1965; Rochette & Sadowski 1976; Blackstone et al. 1982).

Las reglas de despacho pueden clasificarse de varias formas (Haupt 1989). Una de estas clasificaciones es la siguiente:

1. Reglas basadas en tiempo de procesamiento.
2. Reglas basadas en fechas de vencimiento.
3. Combinación de reglas.
4. Reglas que no son basadas en tiempo de procesamiento ni en fecha de vencimiento.

En (Türsel & Uzunoğlu Koçer 2006) los autores proponen un trabajo que se centra en un modelo que generaliza la secuenciación de tareas estocástica y *online*. Se asume que los trabajos llegan de forma *online*, y una vez que entran al sistema se revela su tiempo de terminación estimado, pero el tiempo de procesamiento real permanece desconocido hasta que el trabajo ha sido completado. El problema generalizado es importante debido a que puede utilizarse para modelar problemas prácticos, ya que existen muchas situaciones de la vida real que pueden considerarse estocásticas y *online*. En este artículo los autores modelan el problema, proporcionan literatura relacionada con el tema e identifican áreas de aplicación.

Como parte de la revisión bibliográfica propuesta por los autores, se dice que las metaheurísticas son ampliamente utilizadas para resolver problemas de optimización combinatoria, sin embargo, en problemas de secuenciación estocásticos y *online*, la identificación del espacio de búsqueda se hace difícil por la naturaleza del problema, aunque esto no ha limitado a los investigadores en proponer alternativas de su uso.

En (Wang et al. 2000) se presenta una aplicación para la secuenciación de tareas en ambientes *online* utilizando algoritmos genéticos, la cual resuelve un problema del mundo real relacionado con las industrias del polímero. El estudio está enfocado en la codificación genética y cómo mantener la factibilidad de los individuos durante la evolución. En la planta se producen dos tipos de polietileno expandible de varios

tamaños. La parte reactiva de la planta es manejada en lotes (por grupos) mientras que el procesamiento final se realiza de forma continua.

Un enfoque de solución de problemas de secuenciación de tareas a través de Aprendizaje Reforzado fue expuesto en (Glaubius et al. 2010). En este artículo los autores utilizan el aprendizaje reforzado para integrar la estimación del modelo y de la política. Una pregunta importante es cuánta experiencia es necesaria antes de que se pueda confiar en las políticas aprendidas. La solución a esta pregunta se busca derivando una cota aproximadamente correcta (*Probably Approximately Correct*, PAC) de la complejidad al obtener una política cercana a la óptima. Según los autores, éste es el primer acercamiento utilizando Aprendizaje Reforzado en problemas con estados infinitos y costos ilimitados.

1.7 Aprendizaje Reforzado

El Aprendizaje Reforzado o RL (terminología en inglés) se asocia con aprender qué hacer (cómo asociar situaciones con acciones) de forma tal que se maximice una señal numérica de recompensa. Es el problema encarado por un agente que debe aprender comportamientos a través de interacciones a 'prueba y error' con un ambiente dinámico. Un agente es un sistema computacional situado en algún ambiente, que es capaz de actuar de manera autónoma y flexible en dicho ambiente en aras de cumplir con su objetivo (Jennings et al. 1998).

Cada vez que el agente realiza una acción en su ambiente, el ambiente provee una recompensa o una penalización resultante de la acción tomada. Por ejemplo, al entrenar a un agente para jugar un juego, el ambiente podría proveer una recompensa positiva cuando el juego es ganado, una negativa cuando está perdido y cero en otros casos.

Al aprendiz no se le dice qué acciones tomar, sino que debe descubrir qué acciones proyectan la mayor recompensa al seleccionarlas. En los casos más interesantes y desafiantes las acciones afectan no solo a la recompensa inmediata sino también a las situaciones siguientes y por tanto las recompensas futuras. Estas dos características, búsqueda a 'prueba y error' y recompensa retardada son dos de las características más significativas que distinguen el aprendizaje reforzado.

En el paradigma estándar del RL, un agente está conectado con su ambiente vía percepción y acción, como se muestra en la Figura 1.5. En cada interacción el agente percibe el estado actual 's' del ambiente y selecciona una acción 'a' para cambiar el estado en que se encuentra. Esta transformación genera una señal 'r' que es recibida

por el agente, cuya tarea es aprender una política de selección de acciones en cada estado para recibir la mayor recompensa acumulada a largo plazo (Zhang 1996).

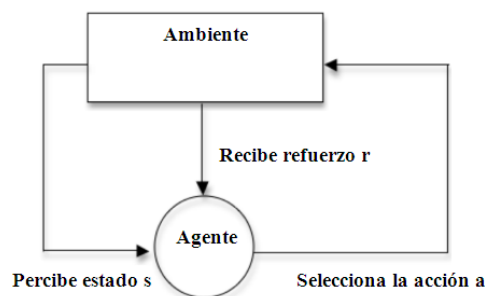


Fig. 1.3 Modelo estándar del aprendizaje reforzado

El modelo básico de RL consiste en:

- Un conjunto de estados del ambiente S
- Un conjunto de acciones A
- Un conjunto de “recompensas” en R
- Una función de transición T

Uno de los desafíos del RL es el intercambio entre la exploración y la explotación. Para obtener una recompensa alta, un agente RL debe preferir acciones que ha probado en el pasado y ha encontrado que son efectivas a la hora de producir recompensa. Pero para descubrir tales acciones, tiene que probar otras que no ha seleccionado antes. El agente tiene que explotar lo que sabe para obtener gratificaciones o recompensas, pero también tiene que explorar para hacer mejor selección de acciones en el futuro. El dilema es que ni exploración ni la explotación pueden perseguirse exclusivamente sin fallar en la tarea. Por eso, el agente debe analizar las acciones disponibles y progresivamente favorecer aquellas que parecen ser mejores.

Más allá del agente y del ambiente, se pueden identificar cuatro subelementos de un sistema de RL: la política, la función de recompensa, la función de valor y opcionalmente el modelo del ambiente.

La política define la forma del comportamiento del agente en un tiempo dado (Sutton & Barto 1998). Es decir, es la probabilidad de seleccionar una acción a en el estado s en un tiempo t . En cada instante t , el agente percibe su estado $s_t \in S$ y el conjunto de posibles acciones $A(s_t)$. Se elige una acción $a \in A(s_t)$ y recibe del entorno el nuevo estado s_{t+1} y una recompensa r_{t+1} , esto significa que el agente implementa un mapeo de los estados a las probabilidades de selección de cada acción posible. La

política del agente se denota π_t , donde $\pi_t(s, a)$ es la probabilidad de que $a_t = a$ si $s_t = s$.

La función de recompensa define la meta en un problema de RL. Es decir, se asigna a cada par estado-acción del ambiente, una recompensa, lo que indica la conveniencia intrínseca de ese estado. El objetivo de un agente de RL es maximizar la recompensa total de lo que recibe a largo plazo. La función de recompensa define qué acontecimientos son buenos y cuales malos para el agente (Sutton & Barto 1998).

Mientras que la función de recompensa indica qué es bueno en sentido "inmediato", la función de valor representa qué es bueno a largo plazo. En términos generales, el valor de un estado es la cantidad total de recompensa que un agente puede planear acumular en el futuro, a partir de ese estado (Sutton & Barto 1998).

El cuarto y último elemento de algunos sistemas de aprendizaje reforzado es el modelo del ambiente, el cual imita el comportamiento del ambiente. Por ejemplo, dado una condición y una acción, el modelo podría predecir el siguiente estado y la siguiente recompensa resultante. Los modelos son usados para la planificación, lo cual quiere decir cualquier forma de decidir el curso de la acción considerando las posibles situaciones futuras antes de que se presenten (Sutton & Barto 1998).

Los agentes inteligentes no tienen que ser diseñados necesariamente por RL, también pueden seguir otros paradigmas como la planificación y el aprendizaje supervisado.

Los algoritmos de planificación pueden dar marcha atrás o "deshacer" las transiciones de estado que entran en estados no deseados, mientras que el RL está diseñado para aplicarse a las situaciones en las que no existe un modelo de acción lo suficientemente manejable y donde normalmente no se pueden deshacer las transiciones de estado. En consecuencia, un agente en el paradigma de RL debe explorar activamente su entorno para observar los efectos de sus acciones.

El aprendizaje supervisado tiene como objetivo inducir una política general de los ejemplos de entrenamiento. Por el contrario, RL no precisa un conocimiento previo de las decisiones correctas e incorrectas. RL puede aplicarse a situaciones en que las recompensas son escasas, por ejemplo, las recompensas se pueden asociar sólo con ciertos estados. En tales casos, puede ser imposible asociar una etiqueta de correcta o incorrecta sobre una decisión particular sin hacer referencia a las decisiones posteriores del agente, lo cual hace no factible el uso del aprendizaje supervisado (Martínez Jiménez 2012).

1.7.1 Q-Learning

Uno de los adelantos más importantes en el aprendizaje reforzado fue el de (Watkins & Dayan 1992) con el desarrollo del algoritmo *Q-Learning* (QL). Dicho algoritmo se basa en aprender una función acción-valor que brinda la utilidad esperada de tomar una acción determinada en un estado específico. El centro del algoritmo es una simple actualización de valores, cada par (s, a) tiene un Q-valor asociado, cuando la acción 'a' es seleccionada mientras el agente está en el estado 's', el Q-valor para ese par estado-acción se actualiza basado en la recompensa recibida por el agente al tomar la acción. También se tiene en cuenta el mejor Q-valor para el próximo estado 's'. La regla de actualización completa es la siguiente:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (1.1)$$

En la expresión anterior, $\alpha \in [0; 1]$ representa la velocidad del aprendizaje y r la recompensa o penalización resultante de ejecutar la acción a en el estado s . La velocidad de aprendizaje α determina el grado por el cual el valor anterior es actualizado. Normalmente se utiliza un valor pequeño para la velocidad de aprendizaje, por ejemplo, $\alpha = 0.1$. El factor de descuento (parámetro γ) toma un valor entre 0 y 1, si está cercano a 0 entonces el agente tiende a considerar sólo la recompensa inmediata, si está cercano a 1 el agente considerará la recompensa futura como más importante.

El pseudocódigo del algoritmo se muestra en la Figura 1.4.

```

Inicializar  $Q(s, a)$  arbitrariamente
Repetir (por cada episodio)
  Inicializar  $s$ 
  Repetir (para cada paso del episodio)
    Escoger  $a$  de  $s$  usando una política derivada de  $Q$  (e.g., e-greedy)
    Tomar acción  $a$ , recompensa  $r$ , estado futuro  $s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ 
  Hasta que  $s$  sea terminal

```

Fig. 1.4 Algoritmo *Q-Learning*

El algoritmo *Q-Learning* es usado por los agentes para aprender de la experiencia, donde cada episodio es equivalente a una sesión de entrenamiento. En cada iteración el agente explora el ambiente y obtiene señales numéricas hasta alcanzar el estado objetivo. El propósito del entrenamiento es incrementar el conocimiento del agente, representado a través de los Q-valores. A mayor entrenamiento, mejores serán los valores que el agente puede utilizar para comportarse de una forma más óptima.

1.7.2 Learning Automata

Learning Automata (LA) representa un enfoque de iteración por política que fue introducido en los años cincuenta (Bush & Mosteller 1955). Algunos años más tarde, en los años sesenta, comenzó a usarse en investigaciones en el área de las ingenierías (Tsetlin 1962).

Un LA toma decisiones de forma adaptable manteniendo una distribución de probabilidad sobre sus acciones. Formalmente puede definirse como (A, B, p, U) , donde $A = \{a_1, \dots, a_r\}$ es el conjunto de posibles acciones que el autómata puede ejecutar, p es la distribución de probabilidad de las acciones, B es el conjunto de respuestas del ambiente (valores entre 0 y 1) y U es el esquema de aprendizaje utilizado para actualizar p .

La Figura 1.5 muestra un agente de tipo LA en su ambiente. En cada intervalo de tiempo selecciona una de las acciones posibles según su distribución de probabilidad. Después de llevar a cabo la acción seleccionada i , su probabilidad p_i es actualizada basada en la recompensa $r \in \{0,1\}$, vea Ecuación 1.2. Las demás probabilidades p_j (para todas las acciones $j \neq i$) se ajustan de forma que la suma de todas las probabilidades sea 1 ($\sum_i p_i = 1$), vea Ecuación 1.3. Este algoritmo se basa en la simple idea de que cada vez que la acción seleccionada resulte en una respuesta favorable, la probabilidad de esa acción aumenta, de lo contrario disminuye.

$$p_i \leftarrow p_i + \alpha r(1 - p_i) - \beta(1 - r)p_i \quad (1.2)$$

$$p_j \leftarrow p_j - \alpha r p_j + \beta(1 - r) \left(\frac{1}{n-1} - p_j \right), \quad \forall j \neq i \quad (1.3)$$

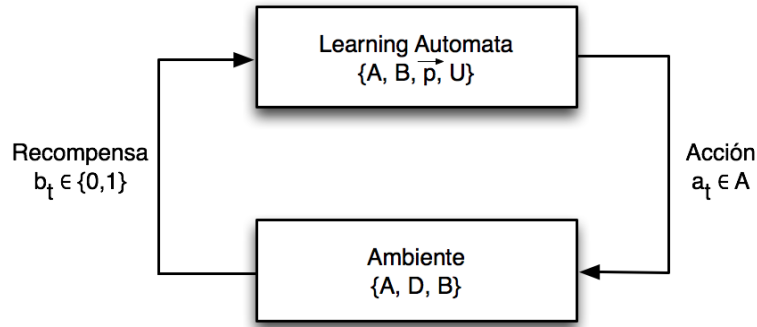


Fig. 1.5 *Learning Automata* en su ambiente

Los parámetros α y β ($\alpha, \beta \in [0,1]$) son los factores de aprendizaje de la recompensa y la penalización respectivamente.

En la literatura se definen tres esquemas comunes de actualización basados en los valores de α y β :

- Lineal Reward-Inaction (L_{R-I}) para $\beta = 0$,
- Lineal Reward-Penalty (L_{R-P}) para $\alpha = \beta$,
- Lineal Reward-e-Penalty ($L_{R-\epsilon P}$) para $\beta \ll \alpha$

1.7.3 Mecanismos de selección de acciones

Uno de los desafíos mencionados anteriormente que aparece en el aprendizaje reforzado es el balance entre la exploración y la explotación. El control correcto del balance entre estos procesos es importante para construir un método eficiente de aprendizaje. Se explicarán de forma breve tres métodos de selección de acción comúnmente usados, *greedy*, ϵ -*greedy* y *softmax*.

Si el agente decide escoger la mejor entre las posibles acciones, podemos decir que sigue una estrategia de selección de acción de tipo *greedy*. Sin embargo, escoger siempre la mejor acción puede conducir a un desempeño subóptimo, dependiendo de la varianza de las recompensas de las acciones.

Una alternativa para este comportamiento *greedy* es seguir la estrategia de selección ϵ -*greedy*. Este método de selección de acción instruye al agente para escoger la mejor acción la mayoría de las veces, pero en algunas ocasiones escoger una acción al azar (con igual probabilidad para cada acción posible a en el estado actual s). El valor de ϵ determina la probabilidad de elección de una acción aleatoria. Aunque ϵ -*greedy* es una estrategia efectiva y popular para balancear la explotación y exploración en el aprendizaje reforzado, un inconveniente es que cuando explora escoge igualmente entre todas las posibles acciones (Sutton & Barto 1998). Lo que quiere decir que

podría escoger entre la que parece ser la peor y la que parece ser la mejor acción con la misma probabilidad.

Una opción para lidiar con esto es variar las probabilidades de las acciones como una función de sus valores estimados, lo cual hace la estrategia de selección de acción *softmax*. La acción *greedy* todavía tendrá la probabilidad más alta, pero las demás son ordenadas por rango según sus estimaciones de valor. Lo que significa que la probabilidad de seleccionar la acción a entre m acciones posibles viene dada por la distribución Boltzmann (vea Ecuación 1.4), la cual constituye la distribución más comúnmente utilizada cuándo se usa este mecanismo de selección de acciones.

$$\Pr(a) = \frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^m e^{Q_t(b)/\tau}} \quad (1.4)$$

En esta ecuación τ es un parámetro positivo llamado temperatura, el cual controla qué tan codiciosamente el agente se comportará, m representa el número de acciones disponibles, y $Q_t(a)$ la estimación de la acción a en el momento t . Las altas temperaturas hacen que todas las acciones sean (casi) equi-probables. Bajas temperaturas causan una mayor diferencia en la probabilidad de selección para las acciones que difieren en su valor estimado, en otras palabras, valores inferiores de temperatura harán que el agente actúe más codiciosamente.

Es posible notar las similitudes entre la estrategia de selección de acción *softmax* y la estrategia ϵ -*greedy*, ambos métodos tienen un parámetro único que definir, pero no queda claro cuál puede ser mejor, ya que esto puede depender de la tarea que está siendo solucionada. También es posible para ambos métodos disminuir el valor de sus parámetros con el paso del tiempo (ϵ y τ), esto significa que los agentes explorarán más al principio de la fase de aprendizaje, y actuarán más codiciosamente (sacando provecho de su conocimiento) en el fin.

Varios experimentos se han desarrollados para determinar cuál estrategia de selección de acción es mejor (Auer et al. 2002). Los experimentos demostraron que diferentes parámetros para las estrategias de exploración proveen resultados muy diversos. Poca exploración lleva a resultados subóptimos al final del proceso de aprendizaje, mientras que demasiada conduce a un mal desempeño durante el proceso de aprendizaje y tiempos largos para el mismo. La mejor opción depende del problema y según el estudio bibliográfico realizado, en problemas de secuenciación de tareas la estrategia más utilizada es la ϵ -*greedy*, de ahí que sea la escogida para implementar en esta tesis.

1.8 Sistemas Multi-Agente

De acuerdo con (Russell & Norvig 2003) un agente puede ser mirado como un individuo capaz de percibir su ambiente a través de sensores y actuando sobre ese ambiente a través de acciones. Los agentes habitan algún ambiente dentro del cual tienen que luchar por cumplir a cabalidad alguna tarea específica. Para hacer eso, secuencialmente deben llevar a cabo decisiones sobre las acciones que deben ejecutar después, deben tratar de adaptar sus políticas de selección de acciones para comportarse óptimamente, y estas decisiones pueden ser requeridas para razonar acerca de la existencia de otros agentes con los cuales cooperar o competir.

Un Sistema Multi-Agente (SMA) es una red de agentes que trabajan juntos para resolver problemas que están más allá de las capacidades individuales o el conocimiento de cada agente (Jennings et al. 1998).

Según (Ferber 1999) un sistema multi-agente es un sistema de inteligencia artificial distribuido que encarna un número de agentes autónomos para lograr metas comunes.

Los agentes autónomos deberían tener las siguientes propiedades (Wooldridge 1999):

- i. la autonomía: Los agentes encapsulan algunos estados de su ambiente, y hacen decisiones acerca de qué hacer basado en estas condiciones;
- ii. la reactividad: Los agentes pueden percibir su ambiente y responder a los cambios que ocurren en él;
- iii. proactividad: Los agentes pueden exhibir comportamiento dirigidos a la meta tomando la iniciativa;
- iv. la habilidad social: Los agentes interactúan con otros agentes por un lenguaje de comunicación del agente, y tienen la habilidad para involucrarse en actividades sociales para lograr sus metas colectivas.

La razón más importante para usar SMA al diseñar un sistema es que algunos dominios la requieren, en particular, si hay organizaciones o personas con metas diferentes (posiblemente estando en conflicto).

El campo de los SMA se centra en procesos descentralizados, ya que cada agente del sistema tiene su propia percepción (pueden estar en ubicaciones distintas y ser responsables de diferentes partes del sistema), control (diferente experiencia) y forma de actuar (diferentes acciones potenciales) (Kaminka 2004).

Existen dos posibles escenarios cuando se trabaja con múltiples agentes, pueden trabajar juntos tratando de alcanzar un objetivo común, o pueden tener sus propios objetivos que entran en conflicto, lo que significa que podemos tener ya sea

aprendices independientes o en conjunto. Cuando se utilizan aprendices en conjunto se asume que las acciones que toman los otros agentes pueden ser observadas. Los aprendices independientes no necesitan observar las acciones tomadas por los otros agentes (Martínez Jiménez 2012).

1.9 Aplicación de RL a problemas de secuenciación

Uno de los trabajos existentes sobre la aplicación de enfoques basados en RL para resolver problemas de secuenciación fue presentado en (Zhang & Dietterich 1995) y extendido un año después (Zhang 1996). Este trabajo se centró en aplicaciones para la NASA, donde el problema consistía en planificar varias tareas que debían ser ejecutadas para instalar y probar las cargas que son ubicadas en el espacio de carga de los transbordadores. En dicho estudio cada misión se consideraba un trabajo, el cual tiene una fecha de vencimiento y está compuesto por un conjunto de tareas parcialmente ordenadas con restricciones de recursos. El objetivo es planificar dicho conjunto de tareas para satisfacer un conjunto de restricciones temporales y a la misma vez minimizar el tiempo total de duración de la ejecución.

En (Gabel and Riedmiller, 2007) y (Gabel, 2009), los autores sugieren y analizan la aplicación de RL para resolver problemas de secuenciación de tipo Job Shop. En estos trabajos se demuestra que interpretar y resolver este tipo de escenarios a través de sistemas multi-agentes y RL es beneficioso para obtener soluciones cercanas a las óptimas y puede muy bien competir con enfoques de solución alternativos.

El software SMA-JSSP versión 1.0 proporciona un sistema para el procesamiento de instancias de problemas de secuenciación de tareas tipo Job Shop. Permite al usuario obtener la solución óptima de una instancia dada y además le brinda la solución robusta de la secuenciación obtenida de acuerdo a dos métodos basados en tiempo extra, Protección Temporal y Máquinas Críticas. Además esta herramienta, que está implementada en JAVA, permite al usuario generarle a la secuenciación obtenida un conjunto de perturbaciones que pueden ser entradas manualmente o a través de un archivo con un determinado formato.

Dicha aplicación implementa técnicas en ambientes *offline*, por lo cual resulta atractivo e interesante extender esta herramienta a ambientes *online* donde no se cuente con una instancia dada por el usuario, simplemente se vayan generando trabajos y sus tiempos correspondientes de forma aleatoria.

1.10 Conclusiones Parciales

- Los problemas de secuenciación de tareas en ambientes tanto *offline* como *online* son muy importantes y utilizados en los procesos de manufacturas, ellos se definen como el proceso de optimización que asigna recursos limitados en el tiempo entre actividades que se pueden ejecutar en paralelo, aunque se han presentado mayores resultados en ambientes *offline*.
- El RL resulta un enfoque natural para resolver problemas de secuenciación de tareas, donde la utilización de sistemas multi-agente se presenta como el centro de atención.
- Los algoritmos QL y LA presentan dos enfoques distintos para entrenar un agente en la toma de decisiones durante el proceso de secuenciación de tareas. Muestran distintos comportamientos que deben ser adaptados a ambientes *online* con el fin de evaluar sus desempeños.

Capítulo 2: Q-Learning y Learning Automata para problemas de secuenciación de tareas en ambientes *online*

En este capítulo se introduce un problema de secuenciación que está inspirado en un escenario de la vida real, y que constituye un ejemplo de problema de secuenciación de tareas en ambientes *online*. Se proponen además tres escenarios, incluyendo el construido a partir del problema original, los cuales son propuestos para una evaluación del comportamiento de los algoritmos. También se explica en este capítulo la aplicación de los algoritmos QL y LA a problemas de secuenciación de tareas en ambientes *online*, particularmente para problemas similares al caso de estudio. Conjuntamente con lo mencionado anteriormente, se propone además una herramienta de fácil uso para resolver la problemática planteada, así como una breve descripción de su modo de utilización.

2.1 Descripción del problema

En un problema de secuenciación *online* quien toma las decisiones no conoce por adelantado cuántos trabajos tienen que ser procesados, ni que tiempo demora en ser procesado cada trabajo. La toma de decisiones tiene en cuenta la existencia de un trabajo solo cuando este llega al sistema y su tiempo de ejecución se conoce cuando el trabajo ha sido completado (Pinedo 2008).

Un ejemplo real de problemas de secuenciación de tareas en ambientes *online* fue expuesto en (Peeters 2008), el cual se basa en la planta química de producción de (Castillo & Roberts 2001). En esta planta se producen dos tipos de productos y se cuenta con una serie de una o más etapas procesadoras con unidades paralelas de procesamiento (máquinas) en cada etapa. El orden de llegada de cada producto y el tiempo de permanencia en cada máquina es generado con una distribución exponencial con determinada media.

En cada etapa se colocan agentes, los cuales escogen en cada punto de decisión la máquina o recurso que procesará el trabajo actual siguiendo un algoritmo de Aprendizaje Reforzado. Cada agente debe seleccionar cuál es la mejor elección entre todas las disponibles, el proceso de decisión tiene en cuenta dos máquinas como mínimo, y para esto el agente se basa en la experiencia adquirida a través del proceso de aprendizaje. Cada trabajo debe ejecutarse por un número específico de recursos, el cual depende de la cantidad de etapas impuestas inicialmente. En el caso de ser dos etapas cada trabajo deberá pasar por dos máquinas y en el caso de ser tres, cada uno

pasará por tres, en ambos casos el número de máquinas capaz de procesar un mismo trabajo por etapas es una.

Las velocidades de procesamiento de las máquinas también son desconocidas, solo se conoce la distribución con la que trabajan, sin embargo, cuando se termina el procesamiento de un trabajo, el programador tiene acceso al tiempo exacto de duración de la tarea.

2.2 Posibles escenarios

En las Figuras 2.1, 2.2 y 2.3 se muestran posibles alternativas de producción por las cuales cada tipo de producto puede pasar hasta salir del sistema una vez se hayan procesados completamente. Estas alternativas o escenarios tienen diferentes números de etapas y/o máquinas y constituyen extensiones del ejemplo mencionado en el epígrafe anterior.

Las máquinas paralelas pueden manipular el mismo tipo de trabajo, pero pueden diferir en la velocidad. La elección posible en cada máquina paralela está representada mediante flechas en las figuras mencionadas anteriormente. Todas las máquinas cuentan con una cola siguiendo una política FIFO (*First In First Out*), es decir, cada trabajo es ejecutado en el orden en que llegan a las colas y ningún trabajo en procesamiento puede interrumpirse para darle paso a otro (no se permite la preferencia).

La Figura 2.1 representa el escenario de una planta química con dos etapas. En la primera etapa para ambos tipos de producto P_1 y P_2 existen dos máquinas paralelas capaces de procesarlos. En la segunda etapa del procesamiento también hay dos máquinas paralelas en cada punto de decisión.

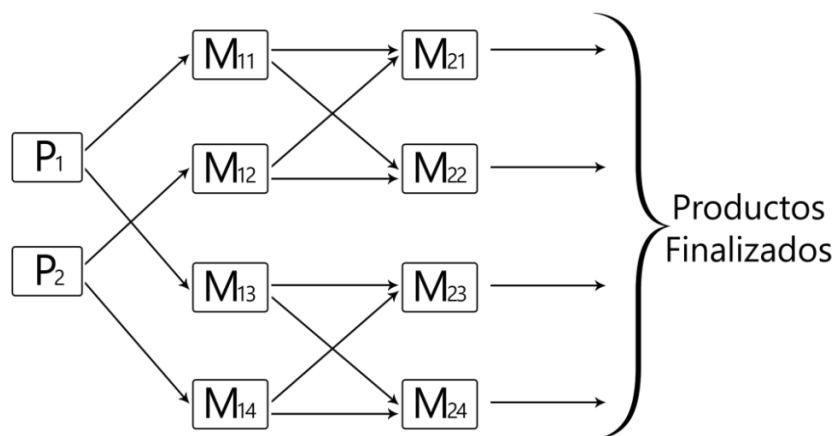


Fig. 2.1 Ejemplo de planta química de producción con dos etapas de procesamiento. En cada etapa existen cuatro máquinas paralelas.

La planta química de la Figura 2.2 cuenta con tres etapas. En la primera etapa, igual que en el caso anterior, para ambos tipos de producto, P_1 y P_2 , existen dos máquinas paralelas distintas, y en la segunda y tercera etapa también hay dos máquinas paralelas en cada punto de decisión.

En el tercer caso (Figura 2.3) se presentan dos etapas con seis máquinas cada una, donde a diferencia de los ejemplos anteriores, los agentes deben escoger la mejor opción de procesamiento en algunos casos entre tres máquinas paralelas.

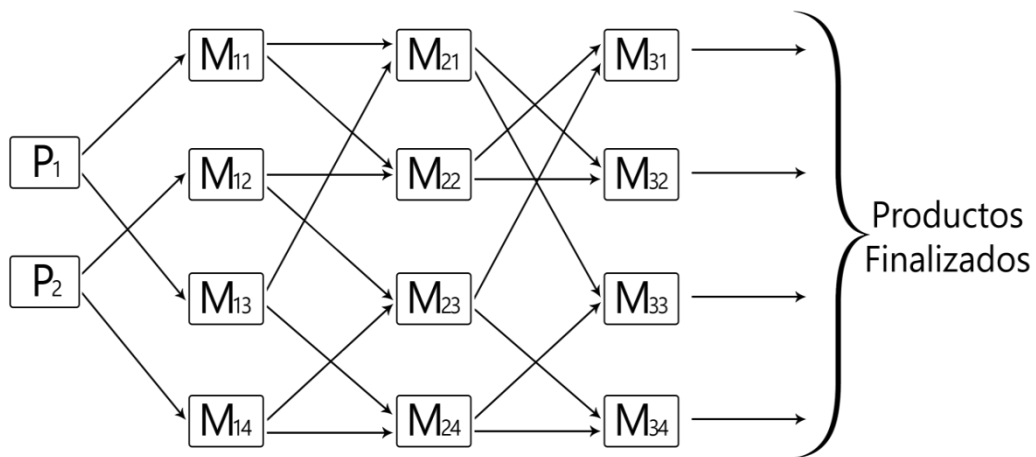


Fig. 2.2 Ejemplo de planta química de producción con tres etapas de procesamiento. En cada etapa existen cuatro máquinas paralelas.

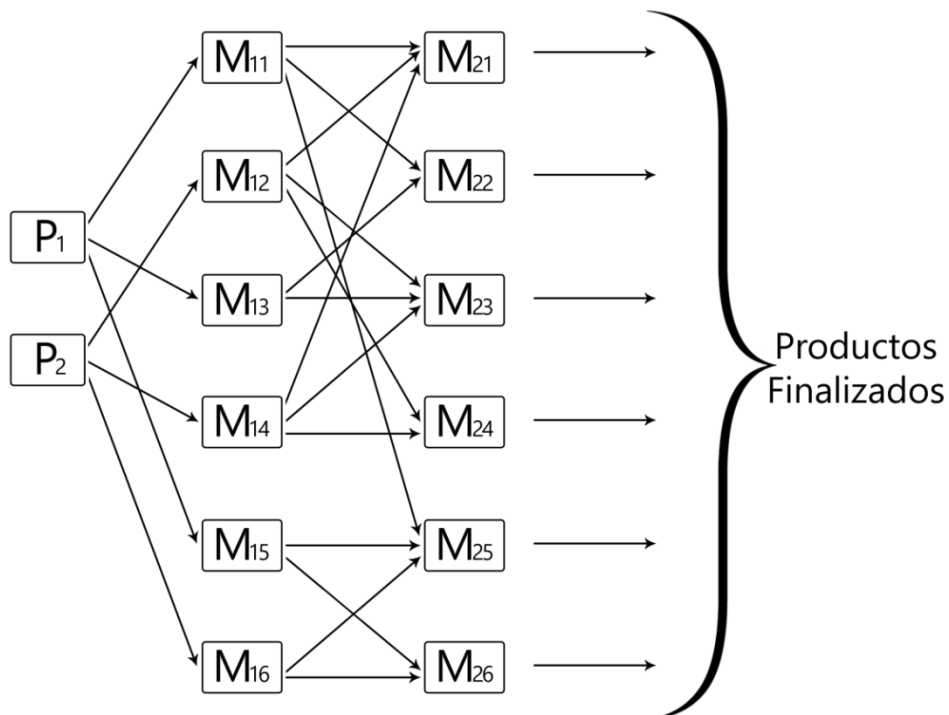


Fig. 2.3 Ejemplo de planta química de producción con dos etapas de procesamiento. En cada etapa existen seis máquinas paralelas.

Cada escenario es simulado durante un tiempo específico. El reloj de la simulación siempre se va a mover hasta el siguiente evento a ocurrir en el sistema. Los posibles eventos pueden constituir el arribo de un nuevo producto al sistema o la culminación del procesamiento de un trabajo en una de las máquinas.

2.3 Relación entre los algoritmos de aprendizaje y las reglas de despacho

Se ha probado que las reglas de despacho obtienen buenas soluciones por si solas, pero no siempre óptimas. Una variante sería, por ejemplo, combinarlas con algoritmos de aprendizaje, utilizándolas en la función de recompensa para indicar la calidad de la acción escogida por un agente.

Algunas de las reglas de despacho más utilizadas son:

- *Shortest Processing Time (SPT)*: Se le asigna mayor prioridad a la operación en cola con el menor tiempo de procesamiento.
- *First In First Out (FIFO)*: La operación que llegó primero a la cola tiene mayor prioridad.
- *Most Work Remaining (MWKR)*: La prioridad más alta se le asigna a la operación que pertenece al trabajo que tiene el mayor tiempo de procesamiento aún por ejecutarse.
- *Earliest Due Date (EDD)*: El trabajo que tiene la fecha de vencimiento más cercana se procesa primero.
- *Weighted Shortest Expected Processing Time (WSEPT)*: Se procesa primero el trabajo que tiene el tiempo esperado de terminación ponderado más pequeño.

La regla SPT es un ejemplo de regla basada en tiempo de procesamiento, estas reglas ignoran la información sobre la fecha de vencimiento de los trabajos. Se ha demostrado que la regla SPT minimiza la media del *flowtime* y también se ha observado que tiene un buen desempeño cuando el objetivo a optimizar es la tardanza bajo condiciones de alta carga a procesar (Conway 1965; Rochette & Sadowski 1976; Blackstone et al. 1982; Haupt 1989).

Un ejemplo de una regla basada en fecha de vencimiento es EDD. En general, las reglas basadas en fechas de vencimiento dan buenos resultados bajo condiciones ligeras de carga, y su desempeño se deteriora en caso contrario (Ramasesh 1990). Las reglas combinadas hacen uso de las reglas de despacho mencionadas anteriormente, y asignan las prioridades utilizando ambas informaciones, las fechas de vencimiento y el tiempo de procesamiento. Ejemplos de este tipo son: la regla *Least Slack*, *Critical Ratio*, etc. (Blackstone et al. 1982). Las reglas que no caen en ninguna

de estas categorías asignan los trabajos en dependencia de las condiciones del ambiente de trabajo en vez de en las características de los mismos, un ejemplo de este tipo de regla es la denominada WINQ (Haupt 1989).

Cuando se utiliza esta regla los trabajos se ordenan de acuerdo a una estimación del tiempo que deben esperar antes de que el procesamiento en la próxima máquina pueda comenzar. Esta estimación incluye el tiempo necesitado por una máquina M_i para terminar su trabajo actual más la suma de los tiempos de procesamiento de todos los trabajos esperando actualmente para ser procesados por M_i . El trabajo que tenga la menor de estas sumas tiene la prioridad más alta.

Según lo revisado en la literatura, no existe un buen algoritmo de solución de problemas de secuenciación de tareas *online*, estocásticos y multi-etapa al mismo tiempo. En el caso de una etapa existe una muy buena heurística: *Weighted Shortest Expected Processing Time* (WSEPT) (Megow et al. 2006).

Esta heurística trabaja de la siguiente manera: las órdenes o trabajos llegan con el paso del tiempo y deben procesarse por una de las máquinas paralelas existentes que se encuentre desocupada. El objetivo es reducir el tiempo ponderado total de terminación ($\sum_j w_j C_j$, para todo trabajo j). Cada vez que una orden llega, la regla WSEPT envía el trabajo a la máquina que se espera que la termine de procesar primero. Con este fin, la regla encuesta cada máquina para saber su makespan actual esperado (incluyendo el trabajo nuevo). Si por ejemplo, todos los trabajos tienen igual tiempo esperado de procesamiento, y cada máquina tiene la misma velocidad promedio, el makespan esperado es el largo de la cola (incluyendo el trabajo actualmente procesado, si existe).

2.4 Aplicación del algoritmo QL a problemas de secuenciación de tareas *online*

En el capítulo anterior se abordaron, entre otras cosas, distintos aspectos generales sobre el algoritmo QL, vale recordar que es una técnica de RL que trabaja con una función acción-valor, la cual ofrece una utilidad esperada de tomar una acción determinada en un estado dado y luego seguir una política óptima. Una ventaja de QL es que puede comparar la utilidad esperada de las acciones disponibles sin requerir un modelo del ambiente. En cada iteración del algoritmo se actualiza un valor llamado Q-valor para cada par estado-acción (s, a) , lo cual constituye el centro del algoritmo. Cuando la acción a es seleccionada por el agente que se encuentra en el estado s el Q-valor para ese par estado-acción es actualizado en base a la recompensa otorgada por la acción, y el mejor Q-valor para el subsecuente estado s , aunque también se

tienen en cuenta el Q-valor anterior, la velocidad de aprendizaje α y el factor de descuento γ ; los dos últimos constituyen parámetros fijos para cada actualización del Q-valor.

Como fue mencionado anteriormente, los agentes necesitan un balance entre la explotación y la exploración. El método de selección ε -greedy, instruye al agente a seguir la política π gran parte del tiempo, pero en ocasiones, el agente elige una acción de forma aleatoria (con igual probabilidad para cada posible acción en el estado actual). La probabilidad ε determina cuando usar una acción aleatoria, esto provee equilibrio entre explotación y exploración.

Cuando se aplica el algoritmo QL a problemas de secuenciación de tareas, se deben tener en cuenta elementos importantes que ya han sido aludidos con anterioridad, pero que serán mejor explicados a continuación.

Estados: en este caso existirá un estado en lo que se llama punto de decisión, que no es más que cada lugar donde haya que decidir la próxima máquina que ejecutará cada trabajo a procesar por el sistema.

Acciones: conjunto de posibles elementos a escoger desde el estado actual, en el problema de secuenciación de tareas *online* que se aborda en esta tesis, seleccionar una acción equivale a seleccionar una máquina para procesar el trabajo actual.

Q-Valores: arreglo de pares estado–acción que refleja, para cada estado, cuan buena resulta cada una de las posibles acciones que se pueden tomar. Este valor se actualiza cada vez que se selecciona una acción de acuerdo a la ecuación 1.1.

Recompensa: valor recibido por el agente como señal de retroalimentación una vez que ha seleccionado una acción desde un estado determinado y que indica la calidad de la misma.

Estrategia de selección de acciones: Para seleccionar las acciones la estrategia que se usa es la ε -greedy, pues el uso de la misma proporciona una mayor exploración del espacio de soluciones. El parámetro ε , como se indica en la literatura, debe ser un valor pequeño, idealmente 0.1, lo que indica que el 10% de las veces se explora y el 90% restante se explota, es decir, se escoge la mejor de las opciones existentes en el momento de la decisión.

2.4.1 Ejemplo de aplicación del QL

Para ilustrar mejor el funcionamiento del algoritmo QL en ambientes *online* se utiliza el escenario descrito en la Figura 2.1 del epígrafe anterior.

Como fue mencionado anteriormente, en cada punto de decisión de cada etapa se colocó un agente, cuya tarea consiste en seleccionar acciones basándose en la experiencia acumulada. De forma general se colocaron seis agentes en el sistema. En la primera etapa se ubicaron dos agentes, uno para escoger entre las máquinas M_{11} y M_{13} en caso de que sea un trabajo tipo P_1 y el otro para decidir entre las máquinas M_{12} y M_{14} en el caso del arribo al sistema de un trabajo tipo P_2 . En la segunda etapa se insertaron cuatro agentes, uno en cada punto de salida de cada máquina de la etapa 1, por ejemplo, luego de que un trabajo culmine su tiempo de procesamiento en la máquina M_{11} , el agente encargado en la segunda etapa debe seleccionar entre las máquinas M_{21} y M_{22} la mejor opción de procesamiento, con el objetivo de que el tiempo que pase dicho trabajo en el sistema sea el mínimo posible.

En cada selección de acciones de los agentes se genera un número aleatorio, el cual es comparado con el valor de ε , si es menor que dicho valor el agente seleccionará una acción al azar, si esto no ocurre así, la acción escogida será la mejor máquina posible, es decir, la que tenga el mayor Q-valor. Si varias máquinas tienen el mismo y el mejor Q-valor, se seleccionará la que tenga menos trabajos en cola esperando por ser procesados.

Otro aspecto importante durante la ejecución del algoritmo es la recompensa que recibe cada agente al seleccionar una acción, si la opción escogida es la mejor recibirá una recompensa, sino una penalización. En este caso se analizaron tres maneras distintas de recompensar, la primera tiene en cuenta el tiempo que estuvo el trabajo procesándose en la máquina, en la segunda se consideró el tamaño de la cola y en la tercera el tamaño de la cola multiplicado por la velocidad de las máquinas. La tercera forma de recompensa fue la implementada ya que muestra en mejor medida cuál máquina será la indicada para llevar a cabo el procesamiento del trabajo pues presenta un equilibrio entre posible velocidad de procesamiento y posible tiempo de espera en cola. La penalización viene dada por la disminución del Q-valor al recibir poca recompensa.

Cuando se quiere utilizar, por ejemplo, el tamaño de la cola para valorar cuan buena es una acción (cuán buena es la máquina seleccionada para procesar el trabajo actual), se debe tener en cuenta que mientras menor sea la cola mejor es la máquina escogida. Esto implica que a la hora de enviar la retroalimentación al agente no se puede dar como recompensa el tamaño de la cola, pues el agente intenta maximizar la recompensa que recibe durante el proceso de selección de acciones, por tanto se

debe proporcionar un valor ajustado que indique de forma proporcional la calidad de la acción, y es por esto que se divide 1 entre el tamaño de la cola.

El punto cumbre del algoritmo es la actualización de los Q-valores que están contenidos en arreglos con dimensiones proporcionales a la cantidad de máquinas que maneja cada agente. En el caso del ejemplo, cada agente cuenta con un arreglo de Q-valores de tamaño dos, lo que implica que van a decidir entre dos Q-valores posibles.

Luego de conocidos los aspectos importantes del algoritmo solamente queda sustituir en la fórmula de actualización del algoritmo (ecuación 1.1) los distintos valores de los parámetros cada vez que un trabajo termine de ser procesado en una máquina.

Un ejemplo de la sustitución de los valores en la regla de actualización puede ser de la siguiente forma. Supongamos que arribó al sistema el trabajo número 4 que constituye un producto tipo 1 y el agente encargado de su distribución en la primera etapa lo ubicó en la máquina M_{11} , la cual tiene una velocidad de 30, luego de terminado su procesamiento el agente actualiza el Q-valor correspondiente. Una posible combinación de valores de sustitución para la ecuación 1.1 puede ser:

- Q-valor anterior correspondiente a la máquina M_{11} : $Q(s, a) = 0.2$
- Velocidad de aprendizaje: $\alpha = 0.1$
- Factor de descuento: $\gamma = 0.8$
- Q-valor del estado futuro: $\max Q(s', a') = 0.235$
- Tamaño de la cola de la máquina M_{11} : 2
- Recompensa: $r = \frac{1}{(30*2)} = 0.0166$

El Q-valor resultante con dichos valores sería:

$$Q(s, a) \leftarrow 0.2 + 0.1 [0.0166 + 0.8 * 0.235 - 0.2]$$

$$Q(s, a) \leftarrow 0.20046$$

Una vez culminada la actualización del Q-valor con los parámetros seleccionados se puede ver que la memoria del agente para el par estado-acción en cuestión almacena ahora un valor ligeramente más grande, lo cual quiere decir que en próximas iteraciones tomar la acción a desde el estado s puede ser más favorable.

2.5 Aplicación del algoritmo LA a problemas de secuenciación de tareas online

Como fue expuesto en el capítulo anterior, los LA son dispositivos diseñados para la toma de decisiones en ambientes estocásticos desconocidos, su principal

característica es que son adaptables al ambiente en el cual se desarrollan, mejorando su rendimiento mediante un proceso de aprendizaje. Este algoritmo trabaja sobre probabilidades en cada punto de decisión y cuenta con tres esquemas de actualización que fueron mencionados en el capítulo anterior.

El esquema lineal de recompensa-penalización (L_{R-P}) es quizás el más antiguo de la psicología matemática (Bush & Mosteller 1955). Las propiedades de este esquema han sido ampliamente estudiadas por investigadores del área (Viswanathan & Narendra 1973; Fu & McMurtry 1966). En este esquema se le da el mismo peso a la recompensa y a la penalización.

El esquema lineal de recompensa-en-acción (L_{R-I}) es otro esquema lineal simple que puede ser derivado por medio de una modificación al esquema general L_{R-P} . Sin embargo, este presenta un comportamiento asintótico totalmente diferente al del esquema L_{R-P} . La idea básica de este esquema radica en no modificar las probabilidades cuando se obtenga una respuesta no favorable de parte del ambiente. En contraparte, si se presenta una respuesta favorable, las probabilidades son modificadas de la misma manera que en el esquema L_{R-P} . El esquema (L_{R-I}) fue considerado por primera vez en la psicología matemática por (Norman 1968), pero se introduce posteriormente de manera independiente a la literatura de ingeniería por (Shapiro & Narendra 1969).

Por último, en el esquema lineal de recompensa- ϵ -penalización ($L_{R-\epsilon P}$) se considera la recompensa como parte fundamental, dándole menor peso a la penalización, esto se logra utilizando un valor de α considerablemente mayor que el valor de β . Si la acción escogida resulta ser favorable (dígase $r = 1$) su probabilidad aumentará considerablemente y de lo contrario, si esta misma acción hubiese tenido recompensa no favorable ($r = 0$), la probabilidad se verá disminuida pero no con la misma magnitud con que se incrementa.

2.5.1 Ejemplo de aplicación del LA

El ejemplo descrito anteriormente sobre la planta química de producción (Figura 2.1) será el utilizado para describir en detalle el funcionamiento del algoritmo LA.

La selección de acciones en cada punto de decisión está basada en probabilidades, por tanto, cada agente contiene un conjunto de probabilidades de tamaño igual a la cantidad de posibles acciones a escoger, que en este caso serían dos, las cuales se relacionan con cada una de las máquinas. Todas las probabilidades son inicializadas a partes iguales de manera tal que la suma total por agente sea igual a uno, a medida

que el algoritmo va siendo ejecutado las probabilidades son actualizadas teniendo en cuenta la acción seleccionada y la recompensa obtenida.

A cada acción tomada se le asocia una recompensa que viene dada por la expresión:

$$r = \begin{cases} 0 & \text{si } F > F_{avg} \\ 1 & \text{en otro caso} \end{cases}, \quad (2.1)$$

donde F es el tiempo que pasa un trabajo en el sistema. Es decir, si el tiempo que paso el trabajo procesándose más el tiempo que pasó en cola es mayor que el tiempo promedio de todos los trabajos que han pasado por el sistema (tiempo total de trabajos procesados más el total de tiempo en cola de todos los trabajos entre la cantidad de trabajos finalizados), entonces la recompensa es equivalente a 0 o mejor dicho, se penaliza con 0, en caso contrario se recompensa con 1.

Durante el proceso de simulación, una vez que ocurra el evento de terminación de un trabajo, se actualizan todas las probabilidades de las acciones asociadas a los agentes que se encargaron de distribuir el procesamiento del trabajo en las distintas etapas. Esta tarea es llevada a cabo en cada etapa y en la última etapa se realiza una actualización especial, por ejemplo, si el trabajo ya terminó de procesarse y se encuentra en la etapa 2, las probabilidades de la primera etapa sufren modificaciones, con el objetivo de valorar mejor la decisión del agente midiendo la trayectoria total del trabajo. Cuando existen más de dos etapas (Figura 2.2) igualmente se actualizan las probabilidades de todas las etapas una vez que se llegue a la última.

Según la acción escogida por el agente su probabilidad es actualizada mediante la ecuación 1.2 expuesta en el capítulo 1 y las probabilidades de las demás acciones no escogidas son recalculadas utilizando la ecuación 1.3. Si a la llegada al sistema de un trabajo tipo P_1 el agente encargado de la selección de la mejor máquina para su procesamiento escoge la máquina M_{11} como mejor opción, su probabilidad es actualizada al finalizar su procesamiento a través de la ecuación 1.2 teniendo en cuenta la recompensa obtenida (0 o 1). La probabilidad de la máquina M_{13} también sufre modificaciones pero mediante la ecuación 1.3. Si la recompensa que se obtuvo fue de 1 entonces la probabilidad de la máquina M_{11} va a aumentar en dependencia del esquema de actualización y a su vez del valor asignado a α , y la probabilidad de la máquina M_{13} disminuirá. En caso de una recompensa igual a 0 ocurrirá lo contrario, excepto en el esquema actualización L_{R-I} donde las probabilidades se mantendrán iguales.

A continuación se muestra un ejemplo de actualización de las probabilidades para el caso en el que arriba al sistema un determinado trabajo, dígame el número 6 de tipo P_1 , y el agente encargado de su manejo en la etapa 1 lo ubica en la máquina M_{11} , la cual tiene una velocidad de 30, una vez acabado su procesamiento el agente actualiza todas las probabilidades relacionadas con sus máquinas. Las posibles combinaciones de parámetros de sustitución para las fórmulas asociadas con las probabilidades (Ecuación 1.2 y Ecuación 1.3) pueden ser:

- Factor de aprendizaje de la recompensa: $\alpha = 0.1$
- Factor de aprendizaje de la penalización: $\beta = 0.1$
- Recompensa sobre la acción: $r = 1$

Acción escogida (M_{11})

- Probabilidad anterior: $p_i = 0.7$

$$p_i \leftarrow 0.7 + 0.1 * 1 * (1 - 0.7) - 0.1 * (1 - 1) * 0.7$$

$$p_i \leftarrow 0.73$$

Acción no escogida (M_{13})

- Probabilidad anterior: $p_j = 0.3$

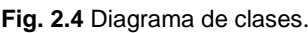
$$p_j \leftarrow 0.3 - 0.1 * 1 * 0.3 + 0.1 * (1 - 1) * \left(\frac{1}{2 - 1} - 0.3 \right)$$

$$p_j \leftarrow 0.27$$

Al ser $\alpha = \beta = 0.1$ estamos en presencia de un esquema de actualización *reward-penalty*, es decir, se le da el mismo peso a la recompensa que a la penalización. En este caso se puede ver que la acción recibe una recompensa favorable ($r = 1$), por tanto la probabilidad se ve incrementada en 0.03 y por consiguiente la probabilidad de la acción no escogida se decrementa en igual medida.

2.6 Diagrama de clases de la aplicación

La aplicación se implementó en el lenguaje de programación orientado a objetos Java. La Figura 2.4 muestra un diagrama de clases en el cual se recogen las principales clases utilizadas para resolver el problema en cuestión y las relaciones entre ellas. El software cuenta con un conjunto de cinco clases encargadas del procesamiento del algoritmo y con varias clases visuales encargadas de recoger y mostrar la información necesaria.



32

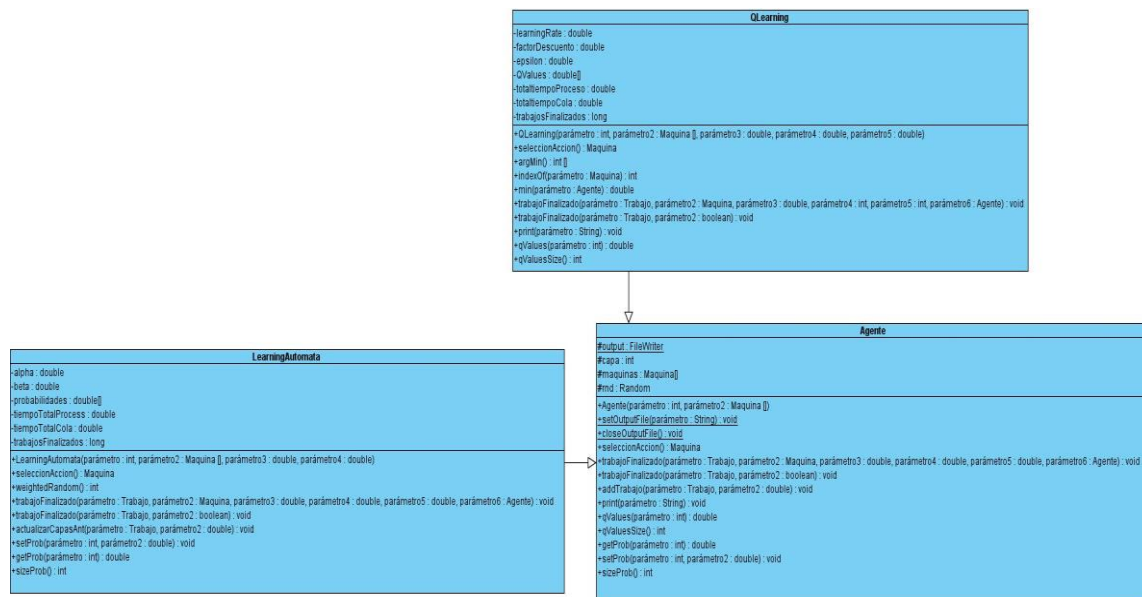


Fig. 2.5 Diagrama de clases. Relación Agente-QLearning-LearningAutomata.

La clase “Agente” también está relacionada con la clase “Maquina” mediante dos atributos, uno en cada clase (Figura 2.6). En el caso de la clase “Agente” se creó un arreglo que indica las máquinas que maneja cada uno de los agentes en cada punto de decisión y la clase “Maquina” contiene un atributo con el agente encargado de la decisión en la siguiente etapa, si existe. La clase “Trabajo” también está vinculada con “Agente” (Figura 2.6) mediante un arreglo de agentes, lo que significa que se va a almacenar, por cada etapa por la que pasa el trabajo, cuál fue el agente encargado de su distribución.

En la clase “AmbientesOnline” (Figura 2.7) existe un conjunto de instancias representativas de la clase “Agente” con el objetivo de crear y manejar los agentes asociados a cada etapa, la cantidad de atributos utilizados varían según la cantidad de etapas y máquinas por etapas. También se definen tres arreglos de tipo “Maquina”, conteniendo cada uno los diferentes elementos distintivos de las máquinas de cada etapa, en caso de ser dos etapas los arreglos utilizados serían dos. Luego de ser recogidos todos los parámetros necesarios para la simulación y la corrida de los algoritmos en las clases visuales, se instancia esta clase para construir todos los objetos necesarios mediante el constructor de “AmbientesOnline”. Esta clase contiene el método “simulacion” que, como su nombre lo indica, es el que lleva el control de la simulación y sus respectivos eventos, los cuales consisten en la creación y finalización del procesamiento de los trabajos siguiendo una distribución exponencial.

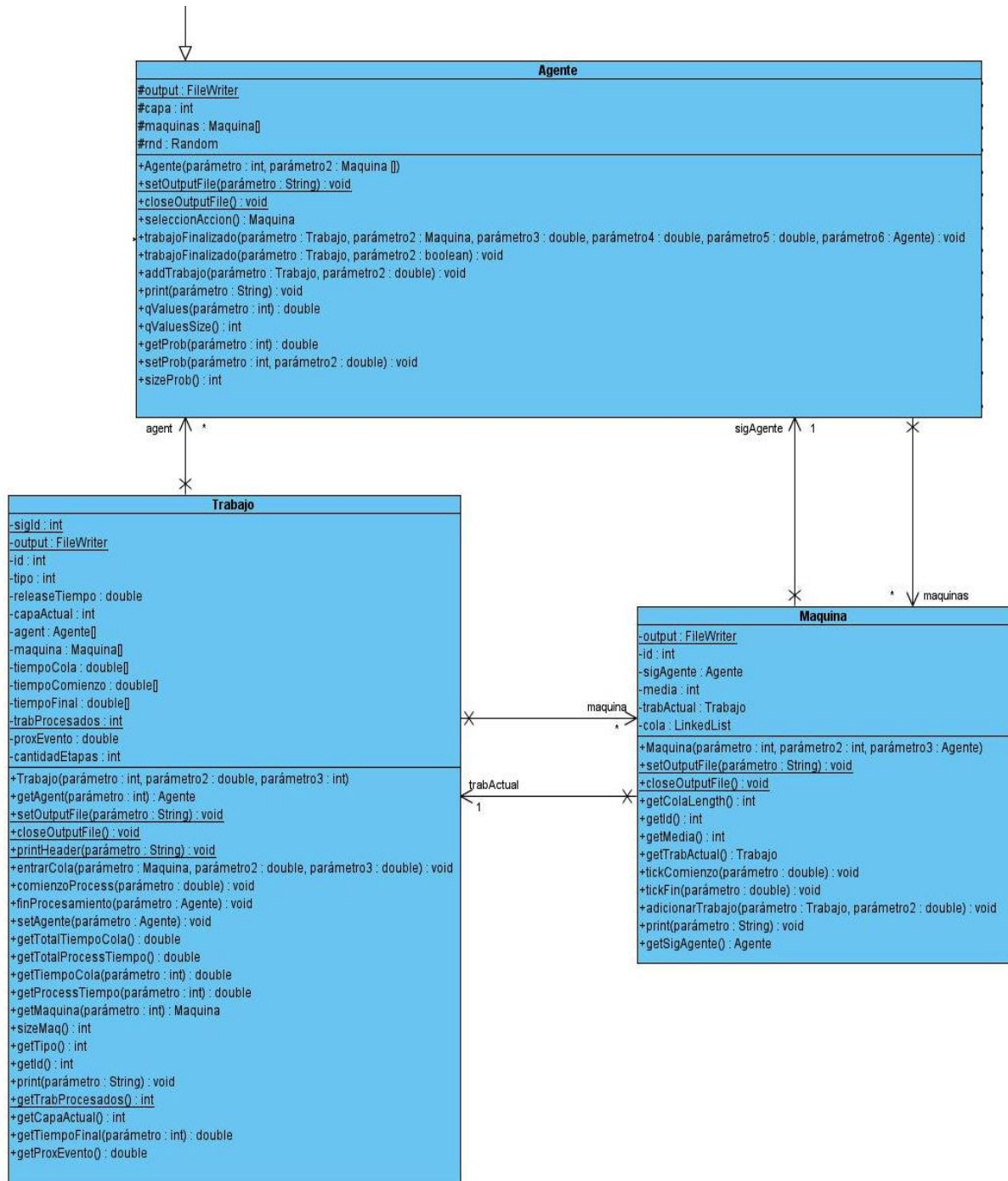
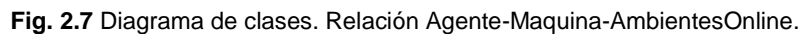


Fig. 2.6 Diagrama de clases. Relación Agente-Trabajo-Maquina.



El software implementado con el objetivo de visualizar los resultados obtenidos por los algoritmos QL y LA, se compone primeramente por un conjunto de ventanas encargadas de recoger la información necesaria para el procesamiento de dichos algoritmos. En este epígrafe se muestran dichas ventanas, y se detallan los pasos a seguir por el usuario para diseñar el escenario que describe el problema de secuenciación de tareas que desea resolver.

35

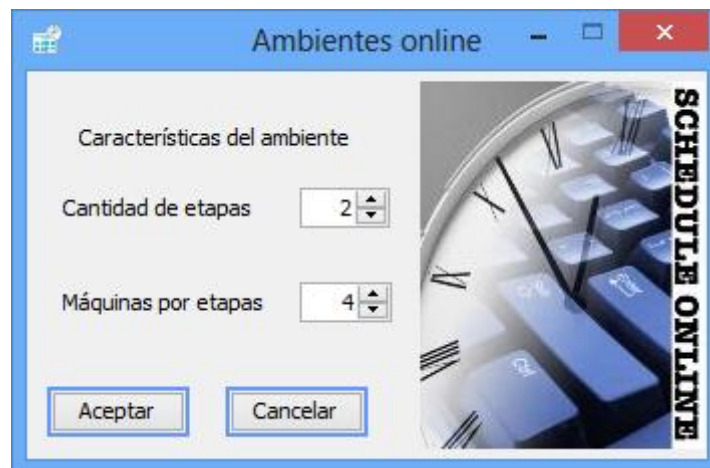


Fig. 2.8 Ventana principal de la aplicación.

Una vez escogidos los datos en la Figura 2.8 y presionando el botón “Aceptar”, el software muestra la Figura 2.9. En esta ventana se deben señalar, por tipo de producto, las máquinas que guardan relación con ellos, es decir, se debe escoger cuáles máquinas son las encargadas de procesar cada tipo de producto en la etapa 1. Si no se seleccionan al menos dos máquinas para cada producto, la aplicación mostrará una notificación como la de la Figura 2.11.



Fig. 2.9 Relaciones entre las máquinas y los productos. Etapa 1.

El siguiente paso es seleccionar la relación de cada máquina de la etapa 1 con las máquinas de la etapa 2 (Figura 2.10). Luego se seleccionan las relaciones de la etapa 3 (si existe) (Figura 2.10), de manera similar a la etapa 2. Cada máquina se debe relacionar al menos con dos máquinas de la siguiente etapa, en caso contrario se mostrará un cartel similar al de la Figura 2.11.

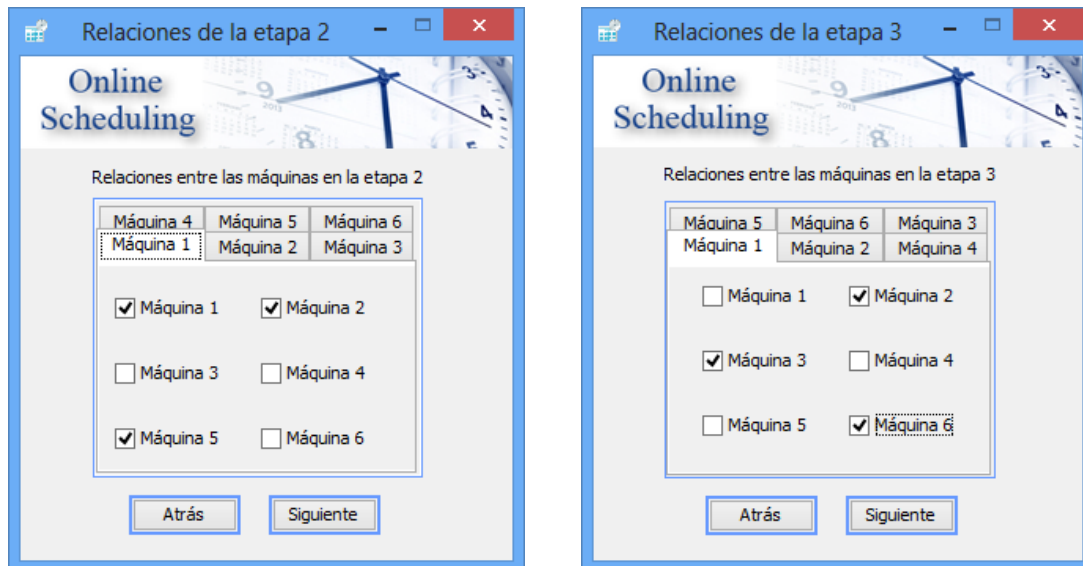


Fig. 2.10 Relaciones entre las máquinas de la etapa 1 y la etapa 2 (izquierda), y las máquinas de la etapa 2 y la etapa 3 (derecha).

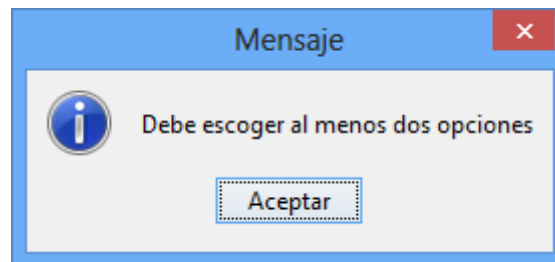


Fig. 2.11 Ventana de notificación.

Una vez diseñado el escenario de trabajo se puede proceder a dar solución al problema modelado a través de cualquiera de los algoritmos implementados. Los pasos a seguir para ejecutar la simulación se brindan en detalle en el capítulo 3.

2.8 Conclusiones Parciales

Se implementaron los algoritmos QL y LA, como métodos para el entrenamiento de los agentes encargados de la distribución de los trabajos por las máquinas.

La aplicación diseñada facilita una mejor interacción con el usuario a partir de interfaces visuales de fácil acceso y comprensión, en la cual se introducen los datos necesarios para el procesamiento y se muestran además los resultados obtenidos mediante tablas y gráficos.

Capítulo 3: Resultados experimentales

En este capítulo se detallan las características de la simulación, especificándose las combinaciones de parámetros a utilizar por los algoritmos de aprendizaje. Los resultados obtenidos por los mismos, divididos por algoritmo y escenario, se comparan teniendo en cuenta la cantidad total de trabajos que se procesan en cada variante con respecto al total generado, así como la longitud de las colas a lo largo de la simulación. Para la validación se aplicaron las pruebas estadísticas de *Friedman* y *Wilcoxon* a un total de 315 instancias o corridas.

3.1 Especificaciones del experimento

Para medir el rendimiento de los algoritmos implementados se utilizaron los tres escenarios mencionados en el epígrafe 2.2.

En el funcionamiento de los algoritmos intervienen diversos parámetros que guían la ejecución del método y que pueden tomar diferentes valores, sin embargo para la realización del experimento se tomaron las configuraciones siguientes:

Para los parámetros de la simulación:

- Tiempo de la simulación: 400 horas = 24000 minutos
- Media del fin de procesamiento en cada máquina: velocidad de cada máquina (recordar que esta media se utiliza como parámetro de la distribución exponencial, según se explicó en la descripción de los escenarios en el capítulo anterior)
- Velocidades de las máquinas: $\{M_{11} \dots M_{1n}, M_{21} \dots M_{2n}, M_{31} \dots M_{3n}\}$
 - Escenario 1: {30, 50, 100, 100, 100, 50, 30, 50}
 - Escenario 2: {30, 50, 100, 100, 100, 50, 30, 50, 60, 90, 40, 80}
 - Escenario 3: {30, 50, 100, 100, 80, 60, 100, 50, 30, 50, 100, 80}
- Media del arribo de los productos tipo 1 y tipo 2: 45

Para los parámetros del algoritmo QL:

- Velocidad de aprendizaje: $\alpha = 0.1$
- Factor de descuento: $\gamma = 0.8, \gamma = 0.9$
- Épsilon: $\epsilon = 0.1, \epsilon = 0.2$

Para los parámetros del algoritmo LA:

- Lineal Reward-Inaction (L_{R-I}) $\beta = 0, \alpha = 0.1$

- Lineal Reward-Penalty (L_{R-P}) $\alpha = \beta = 0.1$
- Lineal Reward- ϵ -Penalty ($L_{R-\epsilon P}$) $\beta = 0.1$, $\alpha = 0.3$

En la aplicación implementada luego de configurar el ambiente de trabajo, se muestra una ventana (Figura 3.1), la cual es la encargada de recoger toda la información necesaria para llevar a cabo la simulación del escenario escogido. Para el caso de los parámetros de los algoritmos la aplicación muestra seguidamente otra ventana como la mostrada en la Figura 3.2. Luego de editar todos los datos necesarios, el software se encuentra listo para realizar la simulación de ambos algoritmos.

The 'Simulación' window contains the following parameters:

- Parámetros de la simulación**
 - Tiempo de la simulación (min): 25.000
- Velocidades de las máquinas**
 - Etapa 1**
 - Máquina 1: 30
 - Máquina 2: 50
 - Máquina 3: 100
 - Máquina 4: 100
 - Máquina 5: 80
 - Máquina 6: 50
 - Etapa 2**
 - Máquina 1: 100
 - Máquina 2: 50
 - Máquina 3: 30
 - Máquina 4: 50
 - Máquina 5: 100
 - Máquina 6: 60
 - Etapa 3**
 - Máquina 1: 60
 - Máquina 2: 90
 - Máquina 3: 40
 - Máquina 4: 80
 - Máquina 5: 50
 - Máquina 6: 80
- Tiempo promedio de creación de los trabajos**
 - Productos tipo 1: 45
 - Productos tipo 2: 45
- Buttons: Atrás, Siguiente

Fig. 3.1 Parámetros para la simulación de los algoritmos.

The 'Aprendizaje Reforzado' window contains the following parameters:

- Parámetros del Aprendizaje Reforzado**
 - Q-Learning**
 - Factor de descuento: 0.8
 - Factor de aprendizaje: 0.1
 - Epsilon: 0.2
 - Buttons: Simular, Ver estado de las colas, Trabajos procesados
 - Learning Automata**
 - Alpha (α): 0.1
 - Beta (β): 0.1
 - Buttons: Simular, Ver estado de las colas, Trabajos procesados
- Buttons: Atrás, Cerrar

Fig. 3.2 Parámetros del Aprendizaje Reforzado.

3.1.1 Descripción del experimento

Los resultados para el estudio posterior de los algoritmos se obtuvieron a partir de 15 corridas realizadas a cada combinación posible de parámetros, en los distintos escenarios. De cada corrida se reporta un conjunto de datos necesarios, primeramente se obtiene la cantidad de trabajos generados, luego, de ese total se analiza cuántos fueron realmente procesados y de este número cuántos correspondieron a productos tipo 1 y cuántos a productos tipo 2. Por último, se consideró el porcentaje que representan los trabajos procesados de los trabajos generados. La aplicación muestra además, por cada corrida, un gráfico de línea reflejando el tamaño de las colas en varios instantes de tiempo. Los datos del algoritmo LA para $\alpha = 0.1$ y $\beta = 0.1$ correspondiente a los escenarios 1, 2 y 3 se muestran en las tablas 3.1, 3.2 y 3.3 respectivamente. Las demás tablas fueron ubicadas en los anexos de manera tal que solamente quedara en este epígrafe una pequeña muestra de los datos obtenidos.

Tabla 3.1 Resultados del algoritmo LA con $\alpha = 0.1$ y $\beta = 0.1$ para el escenario 1.

Escenario 1				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1163	1119	555	564	96,21668
1154	1097	546	551	95,06066
1160	1112	575	537	95,86207
1105	1039	505	534	94,02715
1113	1090	549	541	97,93351
1140	1091	540	551	95,70175
1155	1089	565	524	94,28571
1078	1037	499	538	96,19666
1124	1047	516	531	93,14947
1039	1006	509	497	96,82387
1073	1044	545	499	97,297295
1090	1028	537	491	94,31193
1121	1077	560	517	96,074936
1110	1078	563	515	97,11712
1156	1106	550	556	95,67474

Tabla 3.2 Resultados del algoritmo LA con $\alpha = 0.1$ y $\beta = 0.1$ para el escenario 2.

Escenario 2				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1106	1062	519	543	96,0217
1046	1027	533	494	98,183556
1110	1067	518	549	96,12613
1101	1074	533	541	97,54768
1092	1072	511	561	98,168495
1086	1078	518	560	99,26335
1127	1082	529	553	96,007095
1138	1081	542	539	94,99121
1139	1083	553	530	95,083405
1110	1088	529	559	98,01802
1095	1067	525	542	97,442924
1089	1066	543	523	97,88797
1107	1067	560	507	96,38663
1172	1104	545	559	94,19795
1085	1052	536	516	96,95853

Tabla 3.3 Resultados del algoritmo LA con $\alpha = 0.1$ y $\beta = 0.1$ para el escenario 3.

Escenario 3				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1086	1071	517	554	98,61878
1075	1069	532	537	99,441864
1102	1092	571	521	99,09256
1056	1042	507	535	98,67424
1166	1155	589	566	99,0566
1147	1128	547	581	98,343506
1105	1090	537	553	98,64253
1099	1083	539	544	98,54413
1088	1073	511	562	98,62132
1110	1099	577	522	99,00901
1091	1079	551	528	98,90009
1039	1030	510	520	99,13378
1165	1149	570	579	98,62661
1063	1049	545	504	98,682976
1207	1185	582	603	98,1773

Una vez obtenidos todos los resultados pertinentes, se calculó por cada escenario y posible combinación de parámetros de cada algoritmo, el porcentaje promedio de los trabajos procesados, expresados en la Tabla 3.4. Por ejemplo, en la segunda fila y primera columna se encuentran, divididos por escenario, los porcentajes arrojados por el algoritmo LA con $\alpha = 0.3$ y $\beta = 0.1$ y en la quinta fila los correspondientes al algoritmo QL con parámetros $\gamma = 0.9$, $\alpha = 0.1$ y $\epsilon = 0.2$. Aunque cada corrida de los algoritmos es independiente porque no genera la misma cantidad de trabajos, se consideró tomar este valor resumen como una manera inicial de medir el desempeño de los algoritmos.

En el primer escenario, considerado como el básico y el más sencillo, el LA con $\alpha = 0.3$ y $\beta = 0.1$ arroja mejores resultados que los demás, mientras que en el segundo y el tercero el QL muestra un mejor comportamiento con $\gamma = 0.9$, $\alpha = 0.1$, $\epsilon = 0.1$ y $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.2$ respectivamente. En todos los escenarios cuando se utiliza el esquema de actualización L_{R-1} , el comportamiento obtenido no es bueno, ya que no existe una penalización sobre las malas acciones. El comportamiento en los escenarios 2 y 3 difiere del escenario 1, quizás debido a que el algoritmo QL utiliza valores más flexibles en el momento de asignar las recompensas, pues los parámetros asociados a dicha asignación están vinculados a la simulación y pueden variar con mayor facilidad, contrariamente al algoritmo LA que otorga como recompensa, en este caso, los números 0 o 1 dependiendo de ciertos valores especificados en el epígrafe 2.4.

Tabla 3.4 Promedio de trabajos procesados por algoritmo y escenario.

Esc.	Escenario 1	Escenario 2	Escenario 3
Alg.	% promedio de trabajos procesados	% promedio de trabajos procesados	% promedio de trabajos procesados
LA-0.3-0.1	95,8417163	96,22178	98,6019931
LA-0.1-0.1	95,7155701	96,81898	98,7710197
LA-0.1-0.0	76,3279984	85,74716	95,2315647
QL-0.9-0.1-0.2	92,8891818	98,39792	99,19530067
QL-0.8-0.1-0.1	92,320968	98,32991	99,1764605
QL-0.8-0.1-0.2	91,8365638	98,15031	99,3217521
QL-0.9-0.1-0.1	91,4412211	98,42221	99,256839

3.2 Marco experimental estadístico

La determinación de si la diferencia es significativa o no entre dos o varios algoritmos, puede determinarse mediante la aplicación de distintas técnicas estadísticas. Las pruebas son clasificadas generalmente en dos grupos:

- **Pruebas paramétricas:** estas pruebas son las más empleadas. Cuantifican la asociación o independencia entre una variable cuantitativa y una categórica. Exigen además ciertos requisitos previos para su aplicación: la distribución Normal de la variable cuantitativa en los grupos que se comparan, la homogeneidad de varianzas en las poblaciones de las que proceden los grupos y un tamaño de muestra superior a 30. Su incumplimiento implica que se debe recurrir a pruebas estadísticas no paramétricas (Sheskin 2006).
- **Pruebas no paramétricas:** estas pruebas no requieren asumir normalidad de la población y en su mayoría se basan en el ordenamiento de los datos. Ejemplos de estas pruebas son la prueba de *Wilcoxon* (para comparar los resultados de dos algoritmos) y la prueba de *Friedman* (para comparaciones de varios algoritmos) (Sheskin 2006).

Los datos obtenidos para la realización de los experimentos no presentan distribución normal ni varianza homogénea, por lo que no se pueden aplicar pruebas paramétricas, de ahí que surja la necesidad de aplicar pruebas no paramétricas a nuestros algoritmos.

Dentro de las pruebas no paramétricas se utilizaron dos test importantes: primero el test de *Friedman* para comparar los algoritmos con sus posibles combinaciones de parámetros, divididos por escenario y por último el test de *Wilcoxon* para realizar pruebas dos a dos, las cuales serán explicadas con mayor detenimiento en epígrafes posteriores. A continuación se muestra una breve descripción de las pruebas mencionadas anteriormente.

Prueba de *Friedman*: esta prueba puede utilizarse en aquellas situaciones en las que se seleccionan n grupos de k elementos de forma que los elementos de cada grupo sean lo más parecidos posible entre sí, y a cada uno de los elementos del grupo se le aplica uno de entre k "tratamientos", o bien cuando a cada uno de los elementos de una muestra de tamaño n se le aplican los k "tratamientos". La hipótesis nula que se contrasta es que las respuestas asociadas a cada uno de los "tratamientos" tienen la misma distribución de probabilidad o distribuciones con la misma mediana, frente a la hipótesis alternativa de que por lo menos la distribución de una de las respuestas difiere de las demás (Friedman 1940).

Prueba de Suma de Rangos de Wilcoxon: cuando se trata de variables medibles en por lo menos una escala ordinal y pueden suponerse poblaciones continuas la prueba no paramétrica más potente es la de *Wilcoxon*. La hipótesis nula del contraste postula que las muestras proceden de poblaciones con la misma distribución de probabilidad; la hipótesis alternativa establece que hay diferencias respecto a la tendencia central de las poblaciones y puede ser direccional o no.

La significación asintótica en ambas pruebas es comparada con el valor 0.05, lo cual indica que si la significación es mayor que dicho valor se acepta la hipótesis nula y en caso contrario se acepta la hipótesis alternativa.

3.3 Análisis estadísticos de los resultados

Las tablas 3.5, 3.8 y 3.11 muestran los datos de los escenarios 1, 2 y 3 respectivamente, a los cuales se les aplicaron las pruebas estadísticas. Dichos datos están conformados por los trabajos generados menos los trabajos procesados, para cada combinación de algoritmo con parámetros posibles. Los trabajos generados y los trabajos procesados fueron tomados de las tablas especificadas en el epígrafe 3.1.1. Los algoritmos representados en cada columna se interpretan de la siguiente manera:

- LA01: $\alpha = 0.1$, $\beta = 0.1$
- LA02: $\alpha = 0.1$, $\beta = 0.0$
- LA03: $\alpha = 0.3$, $\beta = 0.1$
- QL01: $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.2$
- QL02: $\gamma = 0.9$, $\alpha = 0.1$, $\epsilon = 0.2$
- QL03: $\gamma = 0.9$, $\alpha = 0.1$, $\epsilon = 0.1$
- QL04: $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.1$

A cada escenario se le aplicaron pruebas no paramétricas, específicamente el test de *Friedman* para una primera valoración de las diferencias significativas entre los algoritmos, luego de comprobar dichas diferencias se identificó el mejor algoritmo como el de menor rango promedio, pues esto indica que el algoritmo fue capaz de procesar la mayor cantidad de trabajos posibles comparado con los demás, y por último se realizó el test de *Wilcoxon* dos a dos. La prueba de *Wilcoxon* dos a dos consiste en comparar, en este caso, el mejor de los algoritmos obtenidos mediante la prueba de *Friedman* con los demás, pero de dos en dos, obteniendo en cada caso una tabla con la significación asintótica, demostrando las diferencias o semejanzas entre ambos algoritmos.

En el primer escenario (tabla 3.5) la significación asintótica equivale a $0.000 < 0.05$ (tabla 3.6b) por lo que se rechaza la hipótesis nula y se acepta la hipótesis alternativa, concluyendo que existen diferencias significativas entre al menos dos de los algoritmos. Luego de analizar los rangos promedios en la tabla 3.6a se ratifica que el algoritmo LA con $\alpha = 0.3$ y $\beta = 0.1$ arroja mejores resultados que los demás, aunque el LA con $\alpha = 0.1$ y $\beta = 0.1$ muestra resultados iguales, se decidió tomar el primero como el mejor ya que en el epígrafe 3.1.1 también fue considerado el más eficiente.

Tabla 3.5 Datos de las pruebas estadísticas del escenario 1.

Escenario 1						
LA01	LA02	LA03	QL01	QL02	QL03	QL04
44	260	33	52	111	130	65
57	172	33	101	76	73	23
48	170	34	95	89	99	132
66	343	36	69	64	87	109
23	264	73	85	122	91	85
49	366	77	101	57	105	76
66	295	33	30	100	89	88
41	236	47	96	75	94	130
77	330	59	85	62	109	83
33	245	27	131	36	73	75
29	161	59	57	105	51	71
62	321	32	118	25	100	100
44	261	43	117	78	150	64
32	217	52	102	79	59	56
50	325	24	142	104	116	122

Tabla 3.6 Prueba de *Friedman* escenario 1. a) Rangos promedio, b) Estadísticos de contraste.

	Rango promedio
LA01	1,87
LA02	7,00
LA03	1,87
QL01	4,57
QL02	3,67
QL03	4,77
QL04	4,27

a)

N	15
Chi-cuadrado	61,840
gl	6
Sig. asintót.	,000

b)

La comparación del escenario 1 mediante la prueba de *Wilcoxon* dos a dos, se realizó teniendo en cuenta el LA con $\alpha = 0.1$ y $\beta = 0.1$ en cada caso, comparándolo así con cada algoritmo posible (tabla 3.7a). La significación con respecto a LA02, QL01, QL02,

QL03 y QL04 es menor que 0.05, por lo que existen diferencias significativas entre el algoritmo LA03 y los mencionados anteriormente, contrariamente ocurre con el LA01 donde es aceptada la hipótesis nula, comprobando así que no existen diferencias entre estos dos algoritmos (tabla 3.7b).

Tabla 3.7a Prueba de los rangos con signo de *Wilcoxon* del escenario 1.

		N	Rango promedio	Suma de rangos
LA03 - LA01	R ⁻ (LA03 < LA01)	8(a)	9,06	72,50
	R ⁺ (LA03 > LA01)	7(b)	6,79	47,50
	Empates (LA03 = LA01)	0(c)		
	Total	15		
LA03 - LA02	R ⁻ (LA03 < LA02)	15(d)	8,00	120,00
	R ⁺ (LA03 > LA02)	0(e)	,00	,00
	Empates (LA03 = LA02)	0(f)		
	Total	15		
QL01 - LA03	R ⁻ (QL01 < LA03)	2(g)	2,00	4,00
	R ⁺ (QL01 > LA03)	13(h)	8,92	116,00
	Empates (QL01 = LA03)	0(i)		
	Total	15		
QL02 - LA03	R ⁻ (QL02 < LA03)	3(j)	2,67	8,00
	R ⁺ (QL02 > LA03)	12(k)	9,33	112,00
	Empates (QL02 = LA03)	0(l)		
	Total	15		
QL03 - LA03	R ⁻ (QL03 < LA03)	0(m)	,00	,00
	R ⁺ (QL03 > LA03)	15(n)	8,00	120,00
	Empates (QL03 = LA03)	0(o)		
	Total	15		
QL04 - LA03	R ⁻ (QL04 < LA03)	1(p)	2,00	2,00
	R ⁺ (QL04 > LA03)	14(q)	8,43	118,00
	Empates (QL04 = LA03)	0(r)		
	Total	15		

Tabla 3.7b Prueba de los rangos con signo de *Wilcoxon* del escenario 1. Estadísticos de contraste.

	LA03-LA01	LA03-LA02	QL01-LA03	QL02-LA03	QL03-LA03	QL04-LA03
Z	-,710(a)	-3,408(a)	-3,181(b)	-2,954(b)	-3,411(b)	-3,295(b)
Sig. asintót. (bilateral)	,477	,001	,001	,003	,001	,001

a Basado en los rangos positivos.

b Basado en los rangos negativos.

La significación asintótica del segundo y tercer escenario (Tabla 3.9b y Tabla 3.12b) corresponde a $0.000 < 0.05$, lo que implica que no se cumple la hipótesis nula y que se acepta la hipótesis alternativa, por lo que existen diferencias significativas entre al menos dos de los algoritmos.

Un análisis llevado a cabo de los rangos promedios de la tabla 3.9a dio como resultados que el algoritmo QL con $\gamma = 0.9$, $\alpha = 0.1$, $\epsilon = 0.1$ es el que muestra mejores resultados respecto a los demás. En las pruebas realizadas en el epígrafe 3.1.1 también se consideró este algoritmo como el más eficiente para este escenario.

Tabla 3.8 Datos de las pruebas estadísticas del escenario 2.

Escenario 2						
LA01	LA02	LA03	QL01	QL02	QL03	QL04
44	130	30	26	17	19	24
19	186	46	20	15	14	16
43	257	33	26	14	23	17
27	134	24	28	24	20	19
20	92	44	20	7	11	19
8	192	43	20	23	16	17
45	141	37	14	10	31	17
57	184	33	21	16	16	40
56	153	47	13	25	22	21
22	96	50	16	11	12	33
28	193	41	16	10	18	12
23	122	72	31	25	21	15
40	85	34	21	22	8	6
68	228	63	18	6	21	10
33	202	38	16	40	10	14

Tabla 3.9 Prueba de *Friedman* escenario 2. a) Rangos promedio, b) Estadísticos de contraste.

	Rango promedio
LA01	4,83
LA02	7,00
LA03	5,23
QL01	3,57
QL02	2,47
QL03	2,37
QL04	2,53

a)

N	15
Chi-cuadrado	59,914
gl	6
Sig. asintót.	,000

b)

La prueba de *Wilcoxon* dos a dos para el escenario 2, se efectuó tomando el QL03 como el algoritmo común para cada comparación (tabla 3.10a). La significación asintótica con respecto a LA01, LA02 y LA03 es menor que 0.05, por lo que existen diferencias significativas entre estos algoritmos (tabla 3.10b). En los algoritmos QL01, QL02 y QL04 no sucede igual, pues en estos casos se acepta la hipótesis nula debido

a que los valores de las significaciones son mayores que 0.05, comprobando así que no existen diferencias significativas entre estos algoritmos (tabla 3.10b).

Tabla 3.10a Prueba de los rangos con signo de *Wilcoxon* del escenario 2.

		N	Rango promedio	Suma de rangos
QL03 - LA01	R ⁻ (QL03 < LA01)	14(a)	8,29	116,00
	R ⁺ (QL03 > LA01)	1(b)	4,00	4,00
	Empates (QL03 = LA01)	0(c)		
	Total	15		
QL03 - LA02	R ⁻ (QL03 < LA02)	15(d)	8,00	120,00
	R ⁺ (QL03 > LA02)	0(e)	,00	,00
	Empates (QL03 = LA02)	0(f)		
	Total	15		
QL03 - LA03	R ⁻ (QL03 < LA03)	15(g)	8,00	120,00
	R ⁺ (QL03 > LA03)	0(h)	,00	,00
	Empates (QL03 = LA03)	0(i)		
	Total	15		
QL03 - QL01	R ⁻ (QL03 < QL01)	11(j)	8,18	90,00
	R ⁺ (QL03 > QL01)	4(k)	7,50	30,00
	Empates (QL03 = QL01)	0(l)		
	Total	15		
QL03 - QL02	R ⁻ (QL03 < QL02)	7(m)	7,21	50,50
	R ⁺ (QL03 > QL02)	7(n)	7,79	54,50
	Empates (QL03 = QL02)	1(o)		
	Total	15		
QL04 - QL03	R ⁻ (QL04 < QL03)	8(p)	7,56	60,50
	R ⁺ (QL04 > QL03)	7(q)	8,50	59,50
	Empates (QL04 = QL03)	0(r)		
	Total	15		

Tabla 3.10b Prueba de los rangos con signo de *Wilcoxon* del escenario 2. Estadísticos de contraste.

	QL03-LA01	QL03-LA02	QL03-LA03	QL03-QL01	QL03-QL02	QL04-QL03
Z	-3,181(a)	-3,408(a)	-3,408(a)	-1,705(a)	-,126(b)	-,028(a)
Sig. asintót (bilateral)	,001	,001	,001	,088	,900	,977

a Basado en los rangos positivos.

b Basado en los rangos negativos.

En el caso de los rangos promedios del escenario 3 (tabla 3.12a), el que arroja mejores resultados es el algoritmo QL con $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.2$, pues contiene un rango promedio de 2.47, lo que constituye el menor de todos. Igualmente, este algoritmo según las pruebas realizadas en el epígrafe 3.1.1 resultó ser el más eficiente para este escenario.

Tabla 3.11 Datos de las pruebas estadísticas del escenario 3.

Escenario 3						
LA01	LA02	LA03	QL01	QL02	QL03	QL04
15	49	10	6	6	10	6
6	43	16	4	10	8	10
10	82	7	10	7	6	14
14	8	13	6	12	8	8
11	47	19	5	3	8	9
19	49	8	9	11	9	6
15	52	16	4	3	8	9
16	22	8	11	9	7	7
15	40	11	12	15	13	11
11	128	21	7	10	9	12
12	58	14	10	10	10	11
9	48	22	6	12	8	8
16	66	25	5	7	6	13
14	54	8	11	9	8	8
22	49	34	8	10	6	6

Tabla 3.12 Prueba de *Friedman* escenario 3. a) Rangos promedio, b) Estadísticos de contraste.

	Rango promedio
LA01	5,07
LA02	6,73
LA03	4,63
QL01	2,47
QL02	3,43
QL03	2,57
QL04	3,10

a)

N	15
Chi-cuadrado	48,251
gl	6
Sig. asintót.	,000

b)

En la comparación de los resultados de los algoritmos en el escenario 3 mediante la prueba de *Wilcoxon* dos a dos, se tuvo en cuenta como algoritmo principal y centro de comparación el QL01 en cada caso (tabla 3.13a). Con respecto al algoritmo LA, incluyendo todas las combinaciones de parámetros posibles, el QL01 muestra mejores resultados y diferencias significativas, pues las significaciones asintóticas de dichas comparaciones se encuentran por debajo de 0.05. De forma contraria ocurre con el QL02, QL03 y QL04 donde es aceptada la hipótesis nula, comprobando así que no existen diferencias significativas entre estos algoritmos (tabla 3.13b).

Tabla 3.13a Prueba de los rangos con signo de *Wilcoxon* del escenario 3.

		N	Rango promedio	Suma de rangos
QL01 - LA01	R ⁻ (QL01 < LA01)	14(a)	7,50	105,00
	R ⁺ (QL01 > LA01)	0(b)	,00	,00
	Empates (QL01 = LA01)	1(c)		
	Total	15		
QL01 - LA02	R ⁻ (QL01 < LA02)	15(d)	8,00	120,00
	R ⁺ (QL01 > LA02)	0(e)	,00	,00
	Empates (QL01 = LA02)	0(f)		
	Total	15		
QL01 - LA03	R ⁻ (QL01 < LA03)	10(g)	10,50	105,00
	R ⁺ (QL01 > LA03)	5(h)	3,00	15,00
	Empates (QL01 = LA03)	0(i)		
	Total	15		
QL02 - QL01	R ⁻ (QL02 < QL01)	5(j)	4,70	23,50
	R ⁺ (QL02 > QL01)	8(k)	8,44	67,50
	Empates (QL02 = QL01)	2(l)		
	Total	15		
QL03 - QL01	R ⁻ (QL03 < QL01)	4(m)	8,50	34,00
	R ⁺ (QL03 > QL01)	9(n)	6,33	57,00
	Empates (QL03 = QL01)	2(o)		
	Total	15		
QL04 - QL01	R ⁻ (QL04 < QL01)	5(p)	5,50	27,50
	R ⁺ (QL04 > QL01)	9(q)	8,61	77,50
	Empates (QL04 = QL01)	1(r)		
	Total	15		

Tabla 3.13b Prueba de los rangos con signo de *Wilcoxon* del escenario 3. Estadísticos de contraste.

	QL01-LA01	QL01-LA02	QL01-LA03	QL02-QL01	QL03-QL01	QL04-QL01
Z	-3,301(a)	-3,410(a)	-2,560(a)	-1,558(b)	-,812(b)	-1,574(b)
Sig. asintót (bilateral)	,001	,001	,010	,119	,417	,116

a Basado en los rangos positivos.

b Basado en los rangos negativos.

3.4 Colas de las máquinas

Una comparación factible además de las ya vistas en los epígrafes anteriores, puede realizarse desde el punto de vista de las colas de las máquinas en varios instantes de tiempo. Por cada corrida de los algoritmos, la aplicación es capaz de mostrar un gráfico de línea con las colas de las máquinas, las cuales fueron comparadas de manera tal que el mejor algoritmo mostrara un buen balance de la cantidad de trabajos en cola en cada una de las máquinas.

En los epígrafes 3.1 y 3.3 se demostró que para el escenario 1, el algoritmo LA con $\alpha = 0.3$ y $\beta = 0.1$ muestra mejores resultados. Este algoritmo fue comparado, por ejemplo, con el QL con $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.2$, teniendo en cuenta las colas de las máquinas. Se puede apreciar en la Figura 3.4 que el algoritmo LA presenta un mejor balance de las colas en las máquinas que el algoritmo QL (Figura 3.3), pues en este último las colas de las máquinas 5 y 6 muestran un mayor crecimiento respecto a las demás.

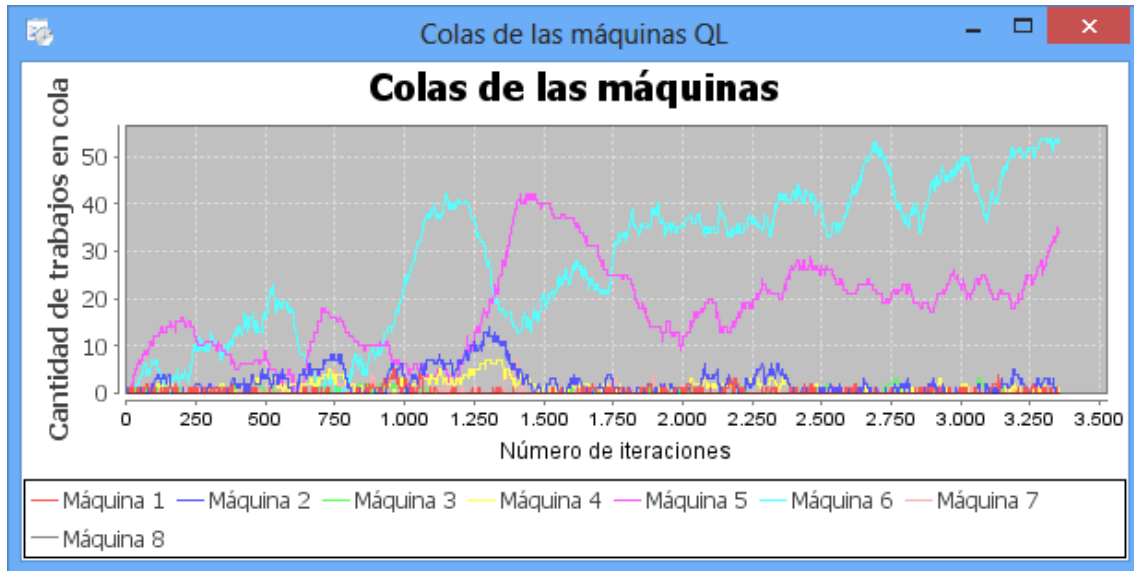


Fig. 3.3 Colas de las máquinas del algoritmo QL con $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.2$ en el escenario 1.

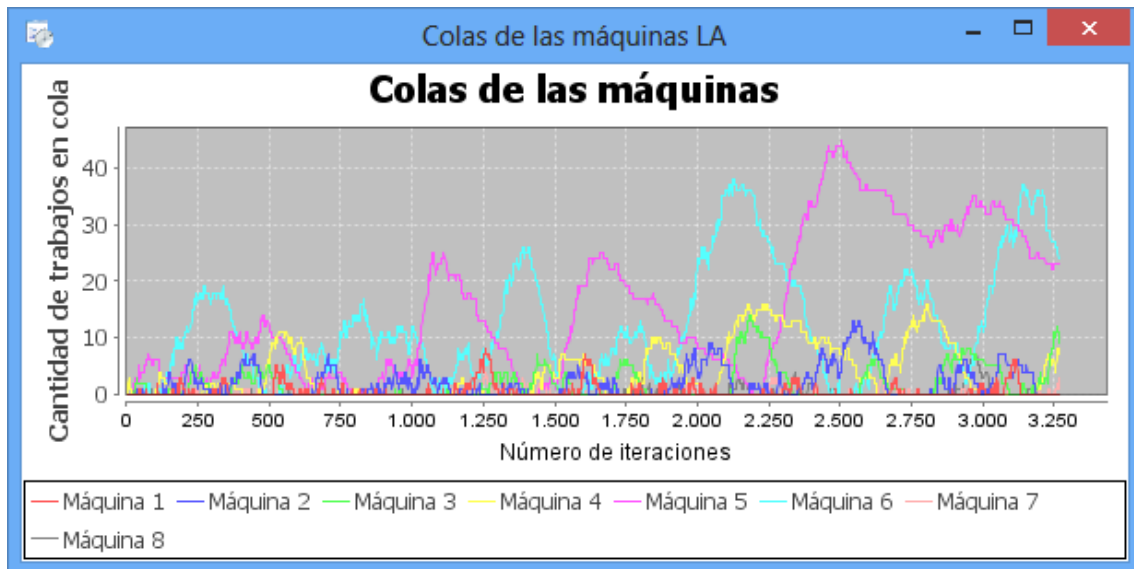


Fig. 3.4 Colas de las máquinas del algoritmo LA con $\alpha = 0.3$, $\beta = 0.1$ en el escenario 1.

Para los escenarios 2 y 3 se realizó la misma operación, pero teniendo en cuenta el algoritmo QL con $\gamma = 0.9$, $\alpha = 0.1$, $\epsilon = 0.1$ y con $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.2$ respectivamente, dichos algoritmos fueron comparados con el LA, teniendo en cuenta varias combinaciones de parámetros. Las colas en las máquinas en ambos casos

muestran resultados similares al del escenario 1, es decir, el mejor algoritmo de cada escenario según lo demostrado en los epígrafes 3.1 y 3.3, muestra un menor número y mejor balance de elementos en cola, reafirmando así la veracidad de las pruebas realizadas en epígrafes anteriores. De acuerdo a los gráficos de las colas de las máquinas se puede decir también que las máquinas 5 y 6 son las que asumen más carga de trabajo, esto pasa porque en la primera etapa las máquinas 1 y 2 son las más rápidas y esto implica que se envíe una mayor cantidad de trabajos a las máquinas 5 y 6.

3.5 Conclusiones Parciales

En este capítulo se realizaron diversas pruebas con el objetivo de comparar el comportamiento de los diferentes algoritmos implementados en esta investigación, teniendo en cuenta tres escenarios y varias configuraciones de parámetros para ambos algoritmos. Como conclusión se puede decir que para escenarios pequeños el algoritmo LA obtiene mejores resultados, pero cuando la complejidad aumenta, es decir, existen más etapas y/o más máquinas por etapas, entonces el QL muestra un mejor comportamiento.

Las estadísticas obtenidas de las colas del sistema ayudan a la toma de decisiones por parte de la empresa debido a que con estos gráficos es posible identificar las máquinas con mayor carga de trabajo y así la empresa podría tener en cuenta esta información a la hora de planificar sus mantenimientos y compras.

Conclusiones

- De la caracterización de los diferentes algoritmos reportados en la literatura para resolver problemas de secuenciación de tareas en ambientes *online*, se concluye que el Aprendizaje Reforzado no ha sido muy utilizado en la solución de este tipo de problema, pero si se visualiza con potencialidades para obtener buenos resultados.
- Se implementaron los algoritmos *Q-Learning* y *Learning Automata* para dar solución a problemas de secuenciación de tareas en ambientes *online*, definiéndose los estados, acciones y funciones de recompensa correspondientes para cada algoritmo.
- Se realizó una comparación entre los dos algoritmos implementados, de la cual se puede concluir que para escenarios pequeños el algoritmo LA obtiene mejores resultados, pero cuando la complejidad aumenta, es decir, cuando existen más etapas y/o más máquinas por etapa, entonces el QL muestra un mejor comportamiento.
- Se adicionaron a la herramienta SMA-JSSP v1.0 los algoritmos implementados extendiéndola a ambientes *online* los cuales se validaron mediante los escenarios expuestos.

Recomendaciones

- Definir otras funciones de recompensa para el algoritmo *Learning Automata* que tengan en cuenta el tamaño de las colas y no solamente el tiempo de procesamiento.
- Valorar la inclusión de otros objetivos en el proceso de construcción de soluciones.
- Incorporar una técnica reactiva para incrementar la robustez de las soluciones.
- Incorporar la funcionalidad de proponer, en base a la complejidad del escenario que se defina, la mejor combinación de parámetros para el aprendizaje.

Referencias Bibliográficas

- Albers, S., 1997. Better bounds for online scheduling. *In Proc. 29th Annual ACM Symposium on Theory of Computing*, pp.130–139.
- Albers, S. & Schröder, B., 2001. An Experimental Study of Online Scheduling Algorithms. *Lecture Notes in Computer Science*, 1982, pp.11–22.
- Auer, P., Cesa-Bianchi, N. & Fischer, P., 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47, pp.235–256.
- Bartal, Y. et al., 1995. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51, pp.359–366.
- Blackstone, J.H., Phillips, D.T. & Hogg, G.L., 1982. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, 20, pp.27–45.
- Bush, F. & Mosteller, R.R., 1955. *Stochastic Models for Learning.*, Wiley.
- Castillo, I. & Roberts, C.A., 2001. Real-time control/scheduling for multi-purpose batch plants. *Computers & industrial engineering*, 41(2), pp.211–225.
- Conway, R.W., 1965. Priority dispatching and job lateness in a job shop. *Journal of Industrial Engineering*, 16, pp.228–237.
- Ferber, J., 1999. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. *Addison-Wesley, Reading*.
- Friedman, M., 1940. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *Ann. Math. Statist.*, 11(1), pp.86–92. Available at: <http://dx.doi.org/10.1214/aoms/1177731944>.
- Fu, K.S. & McMurtry, G.J., 1966. A study of stochastic automata as a model for learning and adaptive controllers. *IEEE Trans. Automatic Control AC-11*, pp.379 – 387.
- Garey, M.R., Johnson, D.S. & Sethi, R., 1976. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1(2), pp.117–129.
- Glaubius, R. et al., 2010. *Real-Time Scheduling via Reinforcement Learning*. Washington University in St. Louis.
- Graham, R.L. et al., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5, pp.287–326.

- Haupt, R., 1989. A survey of priority rule-based scheduling. *OR Spektrum*, 11, pp.3–16.
- Herroelen, W. & Leus, R., 2005. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2), pp.289–306.
- Horn, J., Nafpliotis, N. & Goldberg, D.E., 1994. A niched Pareto genetic algorithm for multiobjective optimization. In *IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 82–87.
- Jennings, N.R., Sycara, K. & Wooldridge, M., 1998. A roadmap of agent research and development. *Autonomous Agents and Multi-agent Systems*, 1, pp.7–38.
- Kaminka, G.A., 2004. Multi-Agent Systems. *Encyclopedia of Human-Computer Interaction*, p.8.
- Karger, D.R., Phillips, S.J. & Torng, E., 1996. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20, pp.400–430.
- Mao, W., Kincaid, R.K. & Rifkin, A., 1995. On-line Algorithms for a Single Machine Scheduling Problem. *The Impact of Emerging Technologies on Computer Science and Operations Research*, pp.157–172.
- Mariano, C. & Morales, E., 2000. A new distributed reinforcement learning algorithm for multiple objective optimization problems. *Springer*, (Advances in Artificial Intelligence).
- Mariano, C. & Morales, E. 1999a, 1999. MOAQ a Distributed Reinforcement Learning Algorithm for the Solution of Multiple Objectives Optimization Problems. In *Memorias del Segundo Encuentro Nacional de Computación ENC99*. pp. 12–15.
- Mariano, C. & Morales, E. 1999b, 1999. MOAQ: An Ant-Q algorithm for multiple objective optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference, 1999b*. pp. 894–901.
- Martínez Jiménez, Y., 2012. *A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems*. PhD PhD, Vrije Universiteit Brussel.
- Megow, N., Uetz, M. & Vredeveld, T., 2006. Models and Algorithms for Stochastic Online Scheduling. *Mathematics of Operations Research*, 31(3), pp.513–525.
- Méndez, C.A. et al., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & chemical engineering*, 30(6-7), pp.913–946.

- Nhu Binh HO, J.C.T.A.Y., 2005. Evolving Dispatching Rules for solving the Flexible Job-Shop Problem. *The 2005 IEEE Congress on Evolutionary Computation.*, 3, pp.2848–2855.
- Norman, M.F., 1968. Mathematical learning theory. *Mathematics of the Decision Sciences* 2, pp.283–313.
- Peeters, M., 2008. *Solving Multi-Agent Sequential Decision Problems Using Learning Automata*. Vrije Universiteit Brussel.
- Pinedo, M., 2008. Scheduling: theory, algorithms, and systems, Third Edition. *Springer Verlag*.
- Ramasesh, R., 1990. Dynamic job shop scheduling: A survey of simulation research. *Omega*, 18, pp.43–57.
- Rochette, R. & Sadowski, R.P., 1976. A statistical comparison of the performance of simple dispatching rules for a particular set of job shops. *International Journal of Production Research*, 14, pp.63–75.
- Ruiz, R., Serifoğlu, F. & Urlings, T., 2008. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35(4), pp.1151–1175.
- Russell, S. & Norvig, P., 2003. *Artificial Intelligence: A Modern Approach (2nd Edition)*, Prentice Hall.
- Shapiro, I.J. & Narendra, K.S., 1969. Use of stochastic automata for parameter self- optimization with multi-modal performance criteria. *IEEE Trans. Syst. Sci. and Cybern.* SSC-5, pp.352–360.
- Shen, W., 2002. Distributed Manufacturing Scheduling Using Intelligent Agents. *IEEE Intelligent Systems*, 17, pp.88–94.
- Sheskin, D., 2006. *Handbook of parametric and nonparametric statistical procedures*, Chapman & Hall: London/West Palm Beach.
- Shmoys, D., Wein, J. & Williamson, D.P., 1995. Scheduling parallel machines on-line. *SIAM Journal on Computing*, 24, pp.1313–1331.
- Simon, H.A. & Lea, G., 1973. Problem solving and rule induction: A unified view. *Knowledge and Cognition*, pp.105–128.
- Sutton, R.S. & Barto, A.G., 1998. *Reinforcement Learning: An Introduction*, The MIT Press.
- Tsetlin, M., 1962. On the behavior of finite automata in random media. *Automation and Remote Control*, 22, pp.1210–1219.

- Türsel, D. & Uzunoğlu Koçer, U., 2006. Stochastic Online Scheduling. In *International Conference on Mathematical and Statistical Modeling in Honor of Enrique Castillo*.
- Viswanathan, R. & Narendra, K.S., 1973. Stochastic automata models with application to learning systems. *IEEE Trans. Syst., Man and Cybern. SMC-3*, pp.107–111.
- Wang, K. et al., 2000. *A genetic algorithm for online-scheduling of a multiproduct polymer batch plant*. University of Dortmund.
- Wang, L. & Shen, W., 2007. Process Planning and Scheduling for Distributed Manufacturing L. Wang & W. Shen, eds. *Springer*, p.429.
- Watkins, C.J.C.H. & Dayan, P., 1992. Technical note -- Q-learning. *Machine learning*, 8(3), pp.279–292.
- Wooldridge, M., 1999. Intelligent Agents. In G. Weiss, ed. *Multiagent Systems*. The MIT Press, p. 51.
- Zhang, W., 1996. *Reinforcement Learning for Job Shop Scheduling*. Oregon State University.
- Zhang, W. & Dietterich, T.G., 1995. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, pp. 1114–1120.

Anexos

Anexo 1

Resultados del algoritmo LA con $\alpha = 0.1$ y $\beta = 0.0$ para los escenarios 1, 2 y 3.

Escenario 1				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1103	843	422	421	76,427925
1104	932	461	471	84,42029
1092	922	506	416	84,432236
1190	847	436	411	71,17647
1098	834	469	365	75,95628
1107	741	467	274	66,93767
1151	856	421	435	74,37011
1111	875	448	427	78,75787
1136	806	434	372	70,95071
1134	889	432	457	78,395065
1084	923	452	471	85,1476
1074	753	409	344	70,11173
1103	842	426	416	76,337265
1105	888	436	452	80,36199
1126	801	388	413	71,136765

Escenario 2				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1107	977	467	510	88,256546
1083	897	448	449	82,825485
1166	909	368	541	77,95883
1113	979	462	517	87,960464
1110	1018	539	479	91,71171
1131	939	470	469	83,02387
1134	993	503	490	87,56614
1130	946	521	425	83,71681
1113	960	484	476	86,25337
1074	978	515	463	91,061455
1148	955	511	444	83,188156
1071	949	514	435	88,60878
1115	1030	466	564	92,37668
1112	884	487	397	79,49641
1135	933	487	446	82,202644

Escenario 3				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1046	997	508	489	95,31549
1047	1004	496	508	95,89303
1113	1031	494	537	92,63252
1099	1091	520	571	99,272064
1113	1066	513	553	95,777176
1155	1106	551	555	95,757576
1096	1044	500	544	95,25548
1181	1159	578	581	98,13717
1060	1020	505	515	96,22642
1198	1070	542	528	89,31553
1085	1027	522	505	94,65438
1077	1029	517	512	95,543175
1161	1095	594	501	94,31525
1062	1008	493	515	94,91525
1080	1031	489	542	95,46296

Anexo 2

Resultados del algoritmo LA con $\alpha = 0.3$ y $\beta = 0.1$ para los escenarios 1, 2 y 3.

Escenario 1				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1085	1052	515	537	96,95853
1119	1085	542	543	96,96157
1101	1065	533	532	96,73025
1123	1050	537	513	93,49956
1109	1032	504	528	93,05681
1099	1066	523	543	96,99727
1116	1069	516	553	95,78853
1118	1059	536	523	94,72272
1066	1039	481	558	97,46717
1139	1080	565	515	94,820015
1102	1070	557	513	97,09619
1118	1075	526	549	96,15385
1068	1016	523	493	95,13109
1054	1030	532	498	97,72296
1040	983	486	497	94,51923

Escenario 2				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1102	1072	509	563	97,27768
1142	1096	534	562	95,97198
1090	1057	515	542	96,97248
1124	1100	567	533	97,86477
1166	1122	568	554	96,22642
1106	1063	507	556	96,112114
1135	1098	574	524	96,74009
1038	1005	494	511	96,82081
1191	1144	573	571	96,053734
1120	1070	518	552	95,53571
1146	1105	553	552	96,42234
1142	1070	528	542	93,695274
1050	1016	469	547	96,7619
1086	1023	511	512	94,1989
1142	1104	537	567	96,6725

Escenario 3				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1124	1114	554	560	99,11032
1105	1089	545	544	98,55203
1115	1108	560	548	99,3722
1104	1091	541	550	98,822464
1140	1121	581	540	98,333336
1073	1065	527	538	99,254425
1102	1086	534	552	98,548096
1138	1130	563	567	99,29701
1069	1058	522	536	98,971
1143	1122	526	596	98,16273
1154	1140	583	557	98,78683
1063	1041	502	539	97,93039
1148	1123	580	543	97,822296
1080	1072	500	572	99,25926
1065	1031	529	502	96,80751

Anexo 3

Resultados del algoritmo QL con $\gamma = 0.8$, $\alpha = 0.1$ y $\epsilon = 0.2$ para los escenarios 1, 2 y 3.

Escenario 1				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1031	979	485	494	94,95635
1108	1007	509	498	90,884476
1120	1025	505	520	91,51786
1142	1073	496	577	93,95797
1130	1045	539	506	92,477875
1104	1003	529	474	90,85145
1026	996	506	490	97,07603
1164	1068	535	533	91,75258
1142	1057	536	521	92,556915
1159	1028	504	524	88,69715
1069	1012	497	515	94,667915
1154	1036	516	520	89,7747
1107	990	478	512	89,43089
1136	1034	494	540	91,021126
1176	1034	525	509	87,92517

Escenario 2				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1026	1000	494	506	97,46589
1092	1072	540	532	98,168495
1137	1111	547	564	97,71328
1088	1060	536	524	97,42647
1098	1078	537	541	98,178505
1136	1116	575	541	98,23943
1107	1093	547	546	98,73532
1073	1052	547	505	98,04287
1136	1123	567	556	98,85564
1143	1127	582	545	98,600174
1109	1093	546	547	98,55726
1160	1129	601	528	97,32758
1056	1035	527	508	98,01136
1128	1110	527	583	98,40426
1087	1071	525	546	98,52806

Escenario 3				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1040	1034	515	519	99,42308
1126	1122	558	564	99,64476
1121	1111	534	577	99,10794
1111	1105	558	547	99,459946
1099	1094	551	543	99,545044
1179	1170	591	579	99,23664
1120	1116	565	551	99,64286
1128	1117	573	544	99,024826
1141	1129	573	556	98,94829
1114	1107	535	572	99,371635
1132	1122	535	587	99,11661
1030	1024	469	555	99,41747
1106	1101	549	552	99,54792
1150	1139	562	577	99,04348
1136	1128	550	578	99,29578

Anexo 4

Resultados del algoritmo QL con $\gamma = 0.9$, $\alpha = 0.1$ y $\epsilon = 0.2$ para los escenarios 1, 2 y 3.

Escenario 1				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1109	998	488	510	89,99098
1070	994	476	518	92,897194
1117	1028	535	493	92,03223
1058	994	520	474	93,95085
1131	1009	491	518	89,21309
1131	1074	548	526	94,96021
1152	1052	513	539	91,31944
1139	1064	531	533	93,415276
1122	1060	520	540	94,47415
1092	1056	566	490	96,7033
1134	1029	554	475	90,74074
1051	1026	520	506	97,621315
1059	981	495	486	92,63456
1079	1000	496	504	92,678406
1119	1015	532	483	90,705986

Escenario 2				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1094	1077	515	562	98,44607
1104	1089	543	546	98,641304
1107	1093	530	563	98,73532
1169	1145	586	559	97,94696
1132	1125	600	525	99,38162
1080	1057	559	498	97,87037
1068	1058	542	516	99,06367
1071	1055	542	513	98,506065
1129	1104	585	519	97,78565
1074	1063	525	538	98,97579
1089	1079	531	548	99,081726
1153	1128	545	583	97,83174
1118	1096	548	548	98,0322
1072	1066	543	523	99,4403
1061	1021	517	504	96,22997

Escenario 3				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1105	1099	539	560	99,457016
1078	1068	516	552	99,07236
1102	1095	536	559	99,36479
1098	1086	536	550	98,907104
1083	1080	576	504	99,72299
1128	1117	571	546	99,024826
1087	1084	534	550	99,724014
1117	1108	583	525	99,19427
1162	1147	566	581	98,70912
1093	1083	543	540	99,08509
1104	1094	546	548	99,0942
1135	1123	545	578	98,94273
1122	1115	553	562	99,376114
1057	1048	522	526	99,14854
1119	1109	548	561	99,106346

Anexo 5

Resultados del algoritmo QL con $\gamma = 0.9$, $\alpha = 0.1$ y $\epsilon = 0.1$ para los escenarios 1, 2 y 3.

Escenario 1				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1130	1000	492	508	88,495575
1062	989	506	483	93,126175
1126	1027	510	517	91,20782
1094	1007	523	484	92,04753
1080	989	493	496	91,57407
1136	1031	540	491	90,75704
1143	1054	520	534	92,21347
1086	992	504	488	91,34438
1144	1035	534	501	90,47203
1124	1051	497	554	93,50534
1099	1048	543	505	95,35942
1087	987	517	470	90,80037
1097	947	453	494	86,32635
1119	1060	547	513	94,72743
1122	1006	516	490	89,661316

Escenario 2				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1097	1078	549	529	98,268005
1089	1075	535	540	98,71442
1122	1099	557	542	97,95009
1139	1119	555	564	98,24407
1107	1096	554	542	99,006325
1105	1089	541	548	98,55203
1120	1089	569	520	97,23214
1133	1117	567	550	98,58782
1133	1111	564	547	98,05825
1059	1047	528	519	98,86685
1094	1076	535	541	98,35466
1148	1127	546	581	98,17073
1146	1138	563	575	99,30192
1027	1006	497	509	97,95521
1076	1066	518	548	99,07063

Escenario 3				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1125	1115	574	541	99,111115
1104	1096	540	556	99,27536
1156	1150	557	593	99,48097
1089	1081	541	540	99,26538
1124	1116	590	526	99,28825
1075	1066	542	524	99,16279
1116	1108	554	554	99,28316
1114	1107	576	531	99,371635
1156	1143	566	577	98,875435
1111	1102	572	530	99,18992
1072	1062	520	542	99,06716
1131	1123	554	569	99,29266
1112	1106	528	578	99,460434
1110	1102	549	553	99,27928
1089	1083	557	526	99,449036

Anexo 6

Resultados del algoritmo QL con $\gamma = 0.8$, $\alpha = 0.1$ y $\epsilon = 0.1$ para los escenarios 1, 2 y 3.

Escenario 1				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1046	981	524	457	93,78585
1064	1041	506	535	97,83835
1117	985	493	492	88,18263
1134	1025	493	532	90,38801
1095	1010	490	520	92,23744
1093	1017	534	483	93,04666
1062	974	480	494	91,713745
1166	1036	532	504	88,85077
1156	1073	522	551	92,82007
1112	1037	514	523	93,255394
1098	1027	510	517	93,5337
1103	1003	499	504	90,933815
1133	1069	546	523	94,35128
1104	1048	529	519	94,927536
1104	982	497	485	88,94927

Escenario 2				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1068	1044	559	485	97,75281
1134	1118	556	562	98,589066
1202	1185	608	577	98,58569
1169	1150	567	583	98,37468
1083	1064	535	529	98,24561
1128	1111	559	552	98,492905
1149	1132	597	535	98,520454
1069	1029	515	514	96,25819
1137	1116	571	545	98,15304
1151	1118	579	539	97,13293
1102	1090	541	549	98,91107
1136	1121	560	561	98,67958
1116	1110	581	529	99,462364
1117	1107	563	544	99,104744
1065	1051	554	497	98,68545

Escenario 3				
Trabajos generados	Trabajos procesados	Trabajos procesados de tipo 1	Trabajos procesados de tipo 2	% de trabajos procesados
1085	1079	556	523	99,44701
1101	1091	564	527	99,091736
1147	1133	580	553	98,77943
1119	1111	557	554	99,28507
1103	1094	560	534	99,184044
1119	1113	562	551	99,463806
1098	1089	578	511	99,18033
1089	1082	536	546	99,35721
1113	1102	566	536	99,01168
1159	1147	573	574	98,96462
1102	1091	532	559	99,001816
1140	1132	593	539	99,29825
1099	1086	522	564	98,81711
1130	1122	566	556	99,29204
1138	1132	557	575	99,472755