ESTUDIO DE ALGORITMOS DINÁMICOS PARA EL PROBLEMA DE SECUENCIACIÓN DE TRABAJOS EN UNA MÁQUINA SIMPLE[*]

Jairo Rafael Montoya Torres[**]

Gloria Rodríguez Verján[***]

Liliana Merchán Alba[****]

Resumen: la teoría clásica de la programación (secuenciación) de tareas se ha dedicado a estudiar y evaluar la pertinencia de reglas o algoritmos cuando toda la información del grupo de tareas por ejecutar se conoce de manera anticipada. Estos escenarios se llaman de tipo estático (*off-line)*. Recientemente se ha dedicado gran interés al estudio de algoritmos dinámicos (*on-line*), los cuales deben tomar decisiones de ejecución de las tareas en tiempo real conociendo únicamente la información disponible al instante de toma de la decisión. En este artículo, se estudia el problema de secuenciación *on-line* de tareas en un recurso único y se presenta el estudio de las reglas SPT (*Shortest Processing Time*) y FIFO (*First In, First Out*). Estas reglas son inicialmente analizadas con respecto a su competitividad para el peor de los casos y, posteriormente, se desarrolla una serie de experimentos de

[*] *Fecha de recepción: 13 de julio de 2006. Fecha de aceptación para publicación: 6 de septiembre de 2006.*
[**] *Ingeniero Industrial, Universidad del Norte, Barranquilla. M.Sc., Institut National Polytechnique de Grennoble, Francia. Doctor en Ingeniería Industrial, École des Mines de Saint-Étienne y Université Jean Monnet, Saint-Étienne, Francia. Profesor Asistente, Departamento de Procesos Productivos, Pontificia Universidad Javeriana, Bogotá. Correo electrónico: jairo.montoya@javeriana.edu.co*
[***] *Estudiante de décimo semestre de Ingeniería Industrial, Pontificia Universidad Javeriana, Bogotá. Correo electrónico: g.rodríguez@javeriana.edu.co*
[****] *Estudiante de noveno semestre de Ingeniería Industrial, Pontificia Universidad Javeriana, Bogotá. Correo electrónico: merchana@javeriana.edu.co*

simulación para verificar dichos postulados y comparar las reglas aplicándolas a diferentes instancias de trabajo.

Palabras clave: secuenciación dinámica de tareas, algoritmos dinámicos, competitividad de reglas de secuenciación.

Abstract: classical scheduling theory has traditionally considered the study and evaluation of scheduling algorithms based on the hypothesis of perfect advanced knowledge of the information needed to make the sequencing decisions. These algorithms are called to be used in an off-line context. Recently, a great interest has been dedicated to the study of on-line scheduling algorithms, which make sequencing decision on real-time, knowing only the information about jobs already arrived at the decision time. This paper considers the problem of scheduling on-line jobs on a single machine environment, and studies the well-known SPT (Shortest Processing Time) and FIFO (First In, First Out) scheduling rules in an on-line context. The study of these two rules is first presented from the theoretic stand point by analyzing their worst-case competitiveness. Afterwards, the algorithms are compared from the practical point of view by a complete set of simulation experiments.

Keywords: *on line* scheduling, dynamic algorithms, competitiveness of scheduling rules.

1. Introducción

La programación (secuenciación) de tareas (*scheduling*) es una parte de la investigación de operaciones que se interesa por estudiar la asignación de recursos limitados a un conjunto de trabajos por realizar en un horizonte de tiempo, con el fin de optimizar uno o más objetivos. Es un proceso de toma de decisiones que constituye uno de los problemas más importantes en gestión de la producción, tanto desde el punto de vista teórico como práctico [Pinedo, 1995]. En los modelos clásicos se considera que toda la información necesaria para establecer la secuencia de ejecución de un conjunto de tareas es conocida desde el instante inicial del horizonte de programación. Se habla entonces de modelos estáticos o modelos *off-line*. Sin embargo, las condiciones actuales de competencia de mercados y algunas aplicaciones específicas en sistemas de producción han obligado a los investigadores a considerar nuevos modelos y métodos de secuenciación de trabajos, esta vez reactivos, adaptativos y, sobre todo, evolutivos en tiempo real, con el fin de satisfacer las nuevas necesidades industriales. De esta manera, se habla de modelos dinámicos o modelos *on-line* [Sgall, 1999].

Durante los últimos años, las investigaciones en el área de modelos *on-line* de secuenciación de tareas han tenido un gran desarrollo. Los modelos y algoritmos propuestos están basados tanto en reglas clásicas utilizadas desde hace mucho tiempo por los industriales, como en estrategias más sofisticadas [Sgall, 1999]. Puesto que estos algoritmos sólo tienen en cuenta la información disponible en un instante de tiempo dado de toma de decisión, los criterios matemáticos y computacionales para la evaluación de su eficiencia sólo consideran la complejidad y aproximabilidad (competitividad) para el peor

de los casos posibles (*worst-case analysis*) [Garey y Johnson, 1979], [Motwani *et al.*, 1993] y dejan de lado los experimentos computacionales prácticos.

En este contexto, el objetivo de este artículo es doble. Por un lado, se presenta la aplicación en contexto *on-line* de las reglas FIFO (*First In, First Out*) y SPT (*Shortest Processing Time*), las cuales son muy conocidas tanto en la academia como en la práctica para el problema de secuenciación de tareas en una máquina simple. Estas reglas se analizan desde el punto de vista teórico. De otro lado, se pretende presentar un estudio experimental extenso, basado en datos (instancias) generados computacionalmente, que permita analizar el comportamiento de estas reglas (algoritmos) de secuenciación en un ambiente *on-line*.

Este artículo está organizado de la siguiente manera. La segunda sección presenta la descripción formal del problema basada en la notación clásica utilizada en la literatura. Posteriormente, las nociones fundamentales sobre el cálculo de la competitividad teórica de los algoritmos *on-line* se presenta en la tercera sección. La cuarta sección está dedicada al análisis de la competitividad teórica para las reglas de secuenciación SPT y FIFO. El estudio experimental comparativo del comportamiento de las reglas de secuenciación se presenta en la quinta sección. Finalmente, las conclusiones de este trabajo se presentan en la sexta sección.

2. Planteamiento y formalización del problema

Desde el punto de vista formal, el problema aquí considerado consiste en considerar un conjunto de n tareas (o trabajos) diferentes para las cuales se debe encontrar una secuencia admisible de ejecución en una máquina simple. Cada tarea, denotada J_j, tiene un tiempo de procesamiento p_j y está disponible en el sistema a partir del instante r_j, con j=1, 2, ..., n. No se autoriza la interrupción de la tarea actualmente en ejecución en la máquina (*preemption*), y esta última siempre está disponible a lo largo del horizonte de trabajo (no existen fallas ni actividades de mantenimiento preventivo o reactivo). Puesto que sólo se considera una máquina en el taller, ésta puede representar el recurso cuello de botella o una agregación de todos los recursos en el sistema. El objetivo es optimizar el nivel de inventario de producto en proceso, puesto que esto permite disminuir los costos asociados a la manutención de capital inmovilizado, así como los resultados globales de la cadena logística [Lee et al., 1997]. Este objetivo está representado por la minimización del tiempo total de terminación de las tareas (es decir, suma de los instantes en los cuales las tareas son terminadas), $\sum C_j$, donde C_j representa el instante de terminación de la tarea J_j. Esta función objetivo también es equivalente a la minimización tanto del tiempo total (o promedio) de flujo de tareas, $\sum F_j$, como del tiempo total (o promedio) de espera de las tareas [Pinedo, 1995].

En la literatura este problema es denotado como $1 \mid r_j \mid \sum C_j$. Las equivalencias en las funciones objetivo aquí presentadas tienen su fundamento matemático en la relación que existe entre la fecha de llegada (r_j), la fecha de inicio de la ejecución del trabajo en la máquina (s_j), y los cálculos de los tiempos de terminación (C_j) y de flujo (F_j) para la tarea J_j. Esto es, para la tarea J_j, el tiempo de flujo se calcula como la diferencia entre el instante

de terminación y la fecha de llegada, $F_j = C_j - r_j$; el tiempo promedio de flujo viene entonces dado por $(1/n)\sum F_j = (1/n)\sum (C_j - r_j)$. Se puede también calcular el tiempo promedio de respuesta, el cual está dado por $(1/n)\sum (s_j - r_j)$. Puesto que $s_j = C_j - r_j$, se obtiene que el tiempo promedio de respuesta está dado por $(1/n)\left(\sum C_j - \sum (p_j + r_j)\right)$. Dado que $(1/n)\sum (p_j + r_j)$ y $(1/n)\sum r_j$ son valores constantes para una instancia dada, se deduce que minimizar $\sum F_j$ ó $(1/n)\sum F_j$ es equivalente a minimizar $\sum C_j$ ó $(1/n)\sum C_j$.

En un contexto *off-line*, el problema $1 \mid r_j \mid \sum C_j$ es un problema combinatorio de tipo fuertemente *NP*-completo [Lenstra *et al*., 1977]. Esto quiere decir que el tiempo de resolución para encontrar la solución óptima a una instancia dada es una función exponencial del tamaño de dicha instancia. Para algunos casos particulares, se puede encontrar la solución óptima de cualquier instancia de manera eficiente, es decir, en un tiempo de cálculo razonable. En particular, cuando las fechas de llegada de las tareas, r_j, son todas idénticas, el problema se denota $1 \mid \mid \sum C_j$ y se resuelve de forma óptima utilizando la regla SPT [Smith, 1956], según la cual la máquina ejecuta las tareas por orden creciente de sus tiempos de procesamiento. Cuando se tienen fechas de llegada diferentes y se acepta la interrupción del trabajo en curso de ejecución (*preemption*), el problema se denota $1 \mid r_j, \text{prmp} \mid \sum C_j$ y la solución óptima se obtiene ejecutando las tareas por orden creciente de sus tiempos operatorios restantes, es decir, mediante la regla SRPT (*Shortest*

Remaining Processing Time First) [Baker, 1974], [Schrage, 1968]. Como se puede observar, la regla SRPT es de hecho una modificación de la regla SPT.

3. Nociones sobre competitividad algorítmica en un contexto dinámico

El análisis de algoritmos *on-line* es usualmente presentado a través de la evaluación del resultado obtenido por el algoritmo (o regla) con respecto a la solución óptima (*off-line*) del problema para el peor de los casos posibles (*worst-case analysis*). Esta es la noción de adversario, en la cual el algoritmo *on-line* se enfrenta a un adversario *off-line* que es capaz de conocer la instancia bajo estudio de forma anticipada y, por lo tanto, puede arrojar la mejor solución para dicha instancia.

De manera formal: para una instancia *J* compuesta de *n* tareas, sea $Z^A(J)$ el valor solución de la función objetivo dado por el algoritmo *on-line A*. El valor de la solución óptima dado por un algoritmo *off-line* OPT siempre busca minimizar (o maximizar) $Z(J)$ para cada instancia *J*. De esta forma, la relación de competitividad $R_A(n)$ para un algoritmo *on-line A* se mide con respecto a su adversario *off-line*:

$$R_A(n) = \max_{J:|J|=n} \frac{Z^A(J)}{Z^{OPT}(J)} \tag{1}$$

Por lo tanto, se dice que un algoritmo *on-line* es $f(n)-$competitivo, si $R_A(n) \leq f(n)$. Ahora bien, el algoritmo es competitivo si existe una constante *k* para la cual él es *k*–competitivo [Motwani *et al*., 1993]. Esto quiere decir que, en el peor de los casos, la solución obtenida, aplicando el método *A* es como máximo *k* veces la solución óptima obtenida por el adversario *off-line*. Esta última se logra si se conocen todos los parámetros de la instancia desde el momento inicial.

Por otro lado, no sólo se busca una buena aproximación de la solución, sino también encontrar una solución a la instancia en un tiempo de cálculo aceptable. Puesto que la mayoría de los problemas de secuenciación de tareas son problemas difíciles (*NP*-completos) en optimización combinatoria [Garey y Johnson, 1979], la medida de la complejidad de un algoritmo permite establecer una referencia, desde el punto de vista computacional, del tiempo de cálculo necesario para que un computador resuelva una instancia del problema. Se dice entonces que un algoritmo es eficaz si utiliza una cantidad razonable de tiempo y de memoria para el cálculo. El tiempo de ejecución depende principalmente del tipo de computador utilizado y de los datos a los cuales es aplicado el algoritmo. Puesto que el objetivo debe ser proponer una regla (algoritmo) que sea eficiente en cualquier tipo de computador, es entonces posible hacer abstracción y tomar únicamente en cuenta el tamaño de los datos para el peor de los casos posibles. Se considera así que la complejidad en tiempo arroja una estimación del número de operaciones elementales necesarias a la ejecución del algoritmo [Wolper, 1991]. La complejidad en tiempo es denotada $O(f)$, donde *f* puede ser una función, polinomial o no, de los parámetros o del

tamaño de la instancia. La idea es entonces proponer un algoritmo *on-line* que sea polinomial[1] en tiempo y analizar su competitividad.

Para muchos problemas de optimización, los algoritmos *on-line* han sido analizados utilizando este método. Por ejemplo, Manasse *et al.* [1990] muestran la existencia de un algoritmo *k*–competitivo para el problema del *k*–servidor. En el campo de la secuenciación de tareas, Graham [1969] fue el precursor del estudio de algoritmos *on-line* al estudiar el problema en *m* máquinas paralelas, para el cual diseñó un algoritmo de lista (*LS*). Graham demostró que, para cualquier instancia, su algoritmo de lista aplicado a la minimización del *makespan* (o tiempo máximo de terminación de todas las tareas), denotado C_{max}, es: $C_{\max}^{LS} \leq (2 - 1/m) \times C_{\max}^{*}$, donde $C_{\max}^{*}$ es el *makespan* óptimo.

En este artículo se estudia el problema de secuenciación en contexto dinámico (*on-line*) en una máquina simple. De esta manera, las tareas no están disponibles desde el instante inicial, sino que son liberadas al sistema durante el horizonte de trabajo según unas fechas de llegada. Se definen dos tipos de algoritmos (o reglas) *on-line* y se estudia su factor de competitividad. Vale la pena resaltar que, dado que las tareas tienen asociadas unas fechas de llegada el sistema, se define un algoritmo *on-line* como aquel que no tiene conocimiento de dichas fechas ni de los tiempos operatorios hasta la llegada de la tarea en cuestión.

[1] Un algoritmo de secuenciación es polinomial si el número de operaciones necesarias para el cálculo está acotado por una función polinomial del número *n* de tareas a secuenciar. Vale la pena aclarar que un algoritmo es pseudo-polinomial, si el número de operaciones necesarias para el cálculo está acotado por una función polinomial del número *n* de tareas a secuenciar y de los parámetros de la instancia (por ejemplo, tiempos operatorios p_j, fechas de llegada, r_j, etc.).

Únicamente se conocen su fecha de llegada y su tiempo de procesamiento en el momento en que la tarea ingresa al sistema.

El estudio del factor de competitividad no es nuevo en la comunidad académica; no obstante, como se mencionó anteriormente, los trabajos tradicionales han estudiado problemas estáticos en los cuales se tiene perfecto conocimiento de la información de las tareas (por ejemplo, fechas de llegada y tiempos operatorios) desde el inicio del horizonte de secuenciación. En estos casos, se habla fundamentalmente de factor de aproximabilidad. En términos matemáticos, el cálculo es el mismo que se definió arriba. Sin embargo, no se le llama factor de competitividad, puesto que éste se reserva únicamente para el estudio de algoritmos en contexto *on-line*.

4. Análisis de los algoritmos FIFO y SPT en contextos dinámicos

En esta sección se presentan dos algoritmos o reglas de secuenciación de tareas muy conocidos. Estas reglas son: FIFO (*First In, First Out*) y SPT (*Shortest Processing Time*). Aplicadas al problema *on-line* de secuenciación en una máquina, en ambas reglas se mantiene una fila en la que se contienen las tareas que van llegando al sistema pero que aún no se han ejecutado. Bajo la regla FIFO, los trabajos no ejecutados son ordenados en el mismo orden de llegada al sistema, es decir por r_j creciente (en caso de empate, va primero el trabajo con menor p_j); de otro lado, con la regla SPT, las tareas en espera de ejecución se organizan por orden creciente de sus tiempos operatorios (en caso de empate, va primero el trabajo con menor r_j). Cuando la máquina queda libre luego de terminar la ejecución de una

tarea, el primer trabajo en la lista es asignado para ejecución en la máquina. Cuando llega una nueva tarea, ésta es ubicada en la lista en la posición que le corresponde, según su fecha de llegada o su tiempo operatorio.

Tanto FIFO como SPT son algoritmos *on-line* puesto que toman las decisiones de ejecución basados en la información disponible al instante en que la máquina se libera, sin considerar las llegadas futuras. Adicionalmente, ambos algoritmos son conservadores, lo cual significa que la máquina nunca permanece inactiva siempre que haya tareas en la lista a la espera de ser ejecutadas. Vale la pena anotar que, a pesar de que estas dos reglas han sido ampliamente utilizadas tanto en la literatura como en la práctica, existen pocos resultados teóricos y experimentales acerca de su desempeño.

4.1. Competitividad teórica para el peor de los casos

A continuación se analizan las reglas SPT y FIFO desde el punto de vista teórico. Phipps [1956] logró mostrar que si las fechas de llegada de los trabajos se distribuyen exponencialmente, entonces una secuencia dada por la regla SPT siempre domina una secuencia dada por la regla FIFO para la misma instancia, considerando varias medidas de desempeño. Schrage [1968] demostró que, con la regla SPT, el número de tareas en la fila en cualquier instante de tiempo siempre es inferior o igual al número de tareas en la fila al aplicar otra regla heurística de secuenciación actuando simultáneamente en la misma instancia. Para distribuciones arbitrarias de las llegadas de las tareas, Mao *et al*. [1995] demostraron que la regla SPT siempre domina a la regla FIFO cuando la función objetivo

consiste en la minimización del tiempo total o promedio de terminación. A continuación se presentan algunos de estos resultados para distribuciones arbitrarias de los tiempos entre llegadas (r_j) y de los tiempos operatorios (p_j). Las funciones objetivo consideradas son la minimización del tiempo total de terminación de todas las tareas o *makespan* (C_{max}) y la minimización del tiempo de flujo promedio de las tareas ($(1/n)\sum F_j$), o de forma equivalente, la minimización de $\sum C_j$. Los siguientes teoremas presentan los resultados obtenidos.

Teorema 1: Sea $C_{\max}^{A}(I)$ el tiempo máximo de terminación de todas las tareas o *makespan* de cualquier instancia *I* aplicando el algoritmo *A*. Entonces, si *A* es un algoritmo conservador, la secuencia dada por *A* es la secuencia de mínimo *makespan* para la instancia *I*.

Demostración: considérese cualquier algoritmo conservador *A* de secuenciación de tareas. Para cualquier instancia *I*, el *makespan* $C_{\max}^{A}(I)$ es el resultado de la suma de los tiempos de procesamiento de las tareas en la instancia, $\sum p_j$, más un tiempo muerto de la máquina, denotado *T*. Esto es:

$$C_{\max}^{A}(I) = \sum p_j + T$$

Puesto que al aplicar el algoritmo *A* la máquina únicamente permanece inactiva cuando no hay más trabajos disponibles en la fila, *A* minimiza el tiempo muerto de la máquina. Por lo tanto, el algoritmo conservador *A* construye una secuencia de ejecución de mínimo *makespan*.

Lema 1: para cualquier instancia *I*, sean $C_{\max}^{SPT}(I)$ y $C_{\max}^{FIFO}(I)$ los valores para el *makespan* utilizando las reglas SPT y FIFO, respectivamente. Entonces $C_{\max}^{SPT}(I) = C_{\max}^{FIFO}(I)$.

Demostración: como se dijo anteriormente, tanto SPT como FIFO son reglas de secuenciación conservadoras, puesto que la máquina nunca está inactiva siempre que existan tareas en la lista de espera de ejecución. Por lo tanto, el valor de *T* (el tiempo ocioso de la máquina) se debe únicamente a la inactividad ocasionada por no tener más tareas en la lista de espera para ejecución en la máquina. Así pues, debido a que este tiempo muerto es minimizado, para toda instancia *I*, tanto SPT como FIFO arrojan la solución óptima para la minimización del *makespan*. Se tiene entonces que $C_{\max}^{SPT}(I) = C_{\max}^{FIFO}(I) = C_{\max}^{*}(I)$.

El resultado del Lema 1 será utilizado a continuación para demostrar que la regla de secuenciación SPT siempre entrega un menor tiempo total de terminación de las tareas (y por consiguiente, un menor tiempo de flujo) que la regla FIFO, para cualquier instancia en una máquina simple. Este postulado fue demostrado por Mao *et al.* [1995]. Este es un resultado muy importante para el desarrollo de este trabajo; por tanto, se presenta en el siguiente teorema.

Teorema 2 [Mao *et al.*, 1995]: para el problema de minimización del tiempo total de terminación de tareas en una máquina simple, $\sum C_j^{SPT} \leq \sum C_j^{FIFO}$.

Demostración: la demostración se realiza por inducción matemática sobre el número *n* de tareas. Cuando $n = 1$, la instancia bajo estudio se denota como I_1. SPT y FIFO se comportan de la misma manera y se tiene que $\sum C_j^{SPT}(I_1) = \sum C_j^{FIFO}(I_1)$.

Por principio de inducción matemática, se asume que para una instancia I_{n-1} con $n-1$ tareas, $\sum C_j^{SPT}(I_{n-1}) \leq \sum C_j^{FIFO}(I_{n-1})$. Ahora, si se considera una instancia I_n compuesta de *n* tareas, sea *J**, con tiempo de proceso p_j^* y fecha de llegada r_j^*, la tarea que es ejecutada de última en la lista establecida por FIFO para la instancia I_n. Se puede observar que el tiempo ocioso de la máquina es $T = \max\{0, r^* - C_{\max}^{FIFO}(I_{n-1})\}$. Sea I_{n-1} la misma instancia I_n descrita pero con la tarea *J** removida. Se tiene entonces que

$$\sum C_j^{FIFO}(I_n) = \sum C_j^{FIFO}(I_{n-1}) + \left(C_{\max}^{FIFO}(I_{n-1}) + T + p^*\right) \qquad (2)$$

Ahora, si se considera la secuencia producida por SPT para la instancia I_n, sin pérdida de generalidad, se puede asumir que si hay otras tareas con el mismo tiempo de proceso y fecha de llegada que *J**, entonces *J** puede ejecutarse de última entre ellas en la secuencia

dada por la regla SPT para la instancia I_n. Sea entonces T_{J*} el tiempo ocioso de la máquina justo antes de la ejecución de *J** en la secuencia dada por la regla SPT para la instancia I_n.

Si se asume que hay $k \geq 0$ tareas siguiendo *J** en la secuencia dada por la regla SPT para la instancia I_n, todas no más pequeñas que *J** y llegando más temprano que *J**. Si $T_{J*} > 0$, *J** debe ser la última tarea en la secuencia de SPT y entonces $k = 0$; y si $T_{J*} = 0$, *J** puede estar seguida por algunas tareas más largas en la secuencia dada por SPT. Por consiguiente,

$$k \times T_{J*} = 0 \tag{3}$$

Entonces, cuando $T_{J*} > 0$, todas las tareas excepto *J** están ubicadas antes de *J** en la secuencia dada por SPT para la instancia I_n. Así, $T_{J*} = \max\{0, r* - C_{\max}^{SPT}(I_{n-1})\}$. Puesto que $C_{\max}^{SPT}(I) = C_{\max}^{FIFO}(I)$ por el Lema 1, entonces,

$$T = T_{J*} \tag{4}$$

Sea C_h el instante de terminación de la tarea J_h la cual ha sido ubicada justo a la derecha de *J** en la secuencia dada por SPT para la instancia I_n. Nótese que J_h y *J** pueden estar separadas por T_{J*}. Entonces,

$$C_h + k \times p^* \leq C_{\max}^{SPT}(I_{n-1}) \tag{5}$$

Entonces, se tiene que:

$$\begin{aligned}
\sum C_j^{SPT}(I_n) &= \sum C_j^{SPT}(I_{n-1}) + (C_h + T_{J^*} + p^* + k \times (T_{J^*} + p^*)) \\
&\leq \sum C_j^{SPT}(I_{n-1}) + C_{\max}^{SPT}(I_{n-1}) + T_{J^*} + p^* \text{ (por (2) y (4))} \\
&\leq \sum C_j^{FIFO}(I_{n-1}) + C_{\max}^{SPT}(I_{n-1}) + T_{J^*} + p^* \text{ (por hipótesis de inducción)} \\
&= \sum C_j^{FIFO}(I_{n-1}) + C_{\max}^{FIFO}(I_{n-1}) + T + p^* \text{ (por Lema 1 y (4))} \\
&= \sum C_j^{FIFO} \text{ (por (1))}
\end{aligned}$$

A continuación en los teoremas 3 y 4 se analiza la competitividad para el peor de los casos de las reglas FIFO y SPT. La demostración del Teorema 3 está inspirada de la idea de bloques presentada en el trabajo realizado por Vestjens [1997] y retomada por Montoya-Torres [2003] para la demostración de la competitividad teórica de un algoritmo *semi-online* de secuenciación.

Teorema 3 [Mao *et al*., 1995]: para el problema de minimización del tiempo total de terminación de tareas en una máquina, $\sum C_j^{FIFO}(I) \leq n \sum C_j^*(I)$ para cualquier instancia *I* de *n* tareas, donde $\sum C_j^*(I)$ es la solución óptima del tiempo total de terminación.

Demostración: la secuencia de ejecución dada por el algoritmo FIFO para una instancia *I* sigue una estructura de bloques. Dichos bloques serán denotados como B_1, B_2,..., B_x. Dentro de cada bloque no hay tiempo ocioso de la máquina y entre dos bloques consecutivos existe un periodo de inactividad de la máquina. Sea $s(B_i)$ el instante de inicio de ejecución del bloque B_i. Es claro que $r_j \geq s(B_i)$, para cualquier $J_j \in B_i$.

Defínase ahora otra instancia *I'* que contenga las tareas $J'_1, J'_2, ..., J'_n$, donde el tiempo de procesamiento de J'_j es el mismo que J_j, y la fecha de llegada al sistema de J'_j es $s(B_i)$ si $J_j \in B_i$ en la secuencia dada por FIFO para la instancia *I*. La secuencia óptima para *I'* tiene entonces la misma estructura de bloques definida por FIFO para *I*. Además, se debe notar que cada bloque tiene las mismas tareas que su correspondiente bloque en la secuencia dada por FIFO para *I*. También, como lo establece Smith [1956], los trabajos en cada bloque en la secuencia óptima para *I'* están ordenados por valor de tiempo operatorio p_j creciente (es decir, según la regla SPT).

Se tiene entonces que $\sum C_j^*(I) \geq \sum C_j^*(I')$, puesto que las instancias *I* e *I'* tienen tareas con los mismos tiempos operatorios pero las fechas de llegada en *I'* son todas al menos tan pronto como en *I*. Asúmase que el bloque B_i tiene los trabajos J_{i1}, ..., J_{ik}, con $p_{i1} \leq ... \leq p_{ik}$. Sean $\sum C_j^*(I', B_i)$ y $\sum C_j^{FIFO}(I, B_i)$ las sumas de los tiempos de terminación de las tareas en B_i, respectivamente, en la secuencia óptima para la instancia *I'* y en la secuencia dada por FIFO para la instancia *I*. Entonces,

$$\sum C_j^*(I', B_i) = k_i s(B_i) + k_i p_{i1} + (k_i - 1) p_{i2} + ... + p_{ik_i}$$

y

$$\sum C_j^{FIFO}(I, B_i) \leq k_i s(B_i) + p_{i1} + 2p_{i2} + ... + k_i p_{ik_i} \leq k_i \sum C_j^*(I', B_i).$$

Por consiguiente,

$$\begin{aligned}
\sum C_j^{FIFO}(I) &= \sum C_j^{FIFO}(I, B_1) + ... + \sum C_j^{FIFO}(I, B_x) \\
&\leq k_1 \sum C_j^*(I', B_1) + ... + k_x \sum C_j^*(I', B_x) \\
&\leq n\left(\sum C_j^*(I', B_1) + ... + \sum C_j^*(I', B_x)\right) \\
&= n \sum C_j^*(I') \\
&\leq n \sum C_j^*(I) \ \blacksquare
\end{aligned}$$

Teorema 4: para el problema de minimización del tiempo total de terminación de tareas en una máquina, $\sum C_j^{SPT}(I) \leq n \sum C_j^*(I)$ para cualquier instancia I de n tareas, donde $\sum C_j^*(I)$ es el tiempo total óptimo de terminación.

Demostración: la demostración está basada en los teoremas 2 y 3.

A continuación se va mostrar que los factores de competitividad obtenidos en los teoremas 3 y 4 se pueden obtener, puesto que son realizables para al menos una instancia. La instancia presentada a continuación fue tomada de Montoya-Torres [2002]. Considérese

una instancia *I* compuesta de *n* tareas. Sean $p_1 = M$ y $r_1 = 0$ el tiempo operatorio y la fecha de llegada de la tarea J_1 en dicha instancia, con *M* un número positivo arbitrariamente muy grande. Sean $p_j = 1$ y $r_j = \varepsilon$, para $j = 2,3,...,n$ y ε un número positivo arbitrariamente muy pequeño, los tiempos operatorios y las fechas de llegada de las $n-1$ restantes tareas de la instancia *I*.

En la secuencia óptima, la máquina permanece intencionalmente inactiva durante las primeras ε unidades de tiempo en espera de la llegada de las tareas J_2, ..., J_n las cuales tienen tiempo de proceso unitario. Entonces, en la secuencia óptima primero se ejecutan estas tareas a partir de su instante de llegada y posteriormente se ejecuta la tarea más larga J_1. De esta forma, se tiene:

$$\sum C_j^*(I) = (1+\varepsilon) + (2+\varepsilon) + ... + (n-1+\varepsilon) + (M+n-1+\varepsilon) = M + \frac{1}{2}n(n+1) - 1 + n\varepsilon .$$

En las secuencias de trabajo construidas por SPT y FIFO, la máquina primero ejecuta la tarea J_1, puesto que no conoce la información sobre las llegadas de las demás tareas y, al terminar su ejecución, se programan las tareas J_2, ..., J_n. Por lo tanto, se tiene que:

$$\sum C_j^{SPT}(I) = \sum C_j^{FIFO}(I) = M + (M+1) + (M+2) + ... + (M+n-1) = nM + \frac{1}{2}n(n-1) .$$

De esta forma, el factor de competitividad para la instancia *I* viene calculado como:

$$R_{SPT}(I) = \frac{\sum C_j^{SPT}(I)}{\sum C_j^*(I)} = R_{FIFO}(I) = \frac{\sum C_j^{FIFO}(I)}{\sum C_j^*(I)} = \frac{nM + \frac{1}{2}n(n-1)}{M + \frac{1}{2}n(n+1) - 1 + n\varepsilon}.$$

En el límite, cuando el número de tareas tiende a infinito y el valor de ε es muy pequeño, se tiene que:

$$\lim_{\substack{n\to\infty \\ \varepsilon\to 0}} \frac{nM + \frac{1}{2}n(n-1)}{M + \frac{1}{2}n(n+1) - 1 + n\varepsilon} \to n.$$

La complejidad informática, es decir, el número de pasos necesarios para ejecutar una instancia, es de $O(n)$ tanto para SPT como para FIFO.

4.2. Una cota inferior generalizada

A partir de la discusión anterior, se observa que la competitividad teórica definida para el peor de los casos tanto para la regla SPT como para la regla FIFO arroja un resultado muy malo, puesto que tiene un valor de *n*, el número de tareas a programar, y *n* puede ser extremadamente grande dependiendo de la instancia. En esta sección, el interés está en saber si existe algún algoritmo *on-line* para el problema $1|r_j|\sum C_j$ que tenga una competitividad acotada por una constante, en lugar de estar acotada por un parámetro de la instancia. Mao *et al.* [1995] demostraron que la competitividad de cualquier algoritmo *on-*

line está acotada por una constante *c*. Posteriormente, Hoogeveen y Vestjens [1996] encontraron el valor de dicha constante y demostraron que no existe ningún algoritmo determinístico *on-line* que tenga una competitividad inferior a dos para el problema de minimización del tiempo total de terminación de tareas en una máquina simple. El resultado se presenta en el siguiente teorema.

Teorema 5 [Hoogeveen y Vestjens, 1996]: para el problema de minimización del tiempo total de terminación de tareas en una máquina simple, no existe ningún algoritmo *on-line* determinístico A y un valor $\varepsilon>0$, de tal forma que A sea $(2-\varepsilon)$-competitivo.

Demostración: para demostrar el teorema se procede por contradicción. Supóngase que para el problema en cuestión exista un algoritmo A que sea $(2-\varepsilon)$-competitivo, siendo ε algún número pequeño estrictamente mayor que cero. Defínase la siguiente secuencia de tareas. La tarea J_1 con tiempo de proceso $p_1=1$ llega al instante $r_1=0$. Si el tiempo de inicio de la ejecución de dicha tarea $S_1>1-\varepsilon$, entonces ningún otro trabajo llega después de éste y la competitividad del algoritmo es entonces $(S_1+1)/1>2-\varepsilon$, lo cual es una contradicción. Así entonces, se puede asumir que $S_1\leq 1-\varepsilon$. Entonces, $n-1$ tareas adicionales llegan al instante $1/2+S_1/2$ para ser trabajadas en la máquina con tiempo de proceso igual a cero. El tiempo total de terminación de la secuencia de trabajo dada por el algoritmo A para dichas tareas es al menos $n(S_1+1)$, mientras que la secuencia óptima entrega un tiempo total de terminación de $n(1/2+S_1/2)+1$. Se puede así calcular la competitividad de A para este caso, la cual es de

2 si el número *n* de tareas a trabajar tiende a infinito. Esto también contradice la hipótesis de que *A* es (2–ε)-competitivo.

Si se desea tener en cuenta el tiempo de flujo total, $\sum F_j$ (o promedio $(1/n)\sum F_j$), los factores de competitividad son los mismos puesto que $\sum F_j = \sum (C_j - r_j)$, y $\sum r_j$ es una constante para una instancia dada, como se indicó anteriormente. Es, sin embargo, necesario recordar que la complejidad computacional de la implementación (número de pasos en la ejecución) de un algoritmo que minimice $\sum F_j$ puede ser diferente a la de un algoritmo que minimice $\sum C_j$.

5. Estudio experimental

El objetivo de esta sección es examinar el desempeño de las reglas de secuenciación STP y FIFO en contexto *on-line* utilizando instancias generadas aleatoriamente. El desarrollo de estos experimentos pretende ir más allá de los resultados teóricos tradicionalmente presentados en la literatura, a través del análisis de las reglas de secuenciación sobre una amplia variedad de instancias. Para este efecto, las reglas se programaron y simularon utilizando el software Arena® versión 9.0. Los factores y niveles (tratamientos) del diseño experimental se muestran en la Tabla 1. Los tiempos operatorios de las entidades se determinaron de manera aleatoria según una distribución uniforme entre 1 y 12. Cuando el tiempo entre llegadas de tareas a la máquina sigue la distribución normal, el valor de la desviación estándar se tomó como el 5% del valor de la media. Se diseñó entonces un

experimento con factores múltiples. Se efectuaron tres repeticiones para cada combinación, lo que arroja un total de 1.296 instancias (3×3×3×16×3).

Tabla 1. Factores y niveles de los escenarios de simulación.

Factor	**Significado práctico**	**Niveles**	**Número de niveles**
Regla de secuenciación	Orden de ejecución de los trabajos en la máquina	FIFO, SPT, LPT	3
Tipo de distribución para el lanzamiento de trabajos a la máquina	Variabilidad asociada a la llegada de tareas al sistema	Exponencial (mucha variabilidad), Normal (variabilidad media), Constante (poca variabilidad)	3
Parámetros de las distribuciones de llegada	Medida del tiempo promedio entre las llegadas de las tareas	3, 5 y 10	3
Tamaño de la instancia (valor de *n*)	Número de trabajos a ejecutar	20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, 350, 400 y 500	16

Fuente: presentación propia de los autores.

Vale la pena anotar que en este estudio experimental también está siendo considerada la regla de secuenciación LPT (Longest Processing Time). En las secciones precedentes esta regla no se analizó desde el punto de vista teórico (es decir, no se analizó su factor de competitividad), debido a que bajo la regla LPT las tareas siempre se ejecutan en la máquina según el tiempo de proceso más largo. Se debe entonces comentar que, puesto que LPT es también un algoritmo conservativo, dado que cada vez que la máquina se libera siempre se escoge una tarea para ejecución, el valor del C_{max} va a ser igual al valor obtenido utilizando SPT o FIFO. Sin embargo, el valor de $\sum C_j$ (y, por consiguiente, tanto el valor de $\sum F_j$ como el del nivel de producto en proceso) se va a ver afectado de manera considerable. Es de esperarse, entonces, que en el estudio experimental, se observe un deterioro de este indicador de desempeño. Se decidió incluir la regla LPT con el fin de tener un punto de referencia en el desarrollo de los experimentos.

Las Tablas 2, 3 y 4 presentan los resultados obtenidos para el *makespan* aplicando las reglas FIFO, SPT y LPT, respectivamente. Las Tablas 5, 6 y 7 presentan a su vez los resultados para el tiempo promedio de flujo aplicando, respectivamente, las reglas FIFO, SPT y LPT. Finalmente, los resultados para el nivel de utilización de la máquina aplicando cada una de las reglas se presentan en las Tablas 8, 9 y 10. Los resultados presentados en las tablas corresponden a los promedios obtenidos de las tres réplicas para cada escenario.

Tabla 2. Resultados para el *makespan* mediante la regla FIFO.

	FIFO, makespan								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	2,467	2,467	2,467	2,467	2,467	2,467	2,467	2,467	2,467
30	3,700	3,700	3,700	3,700	3,700	3,700	3,700	3,700	3,700
40	4,800	4,800	4,800	4,800	4,800	4,800	4,800	4,800	4,800
50	6,033	6,033	6,033	6,033	6,033	6,033	6,033	6,033	6,033
60	7,317	7,317	7,317	7,317	7,317	7,317	7,317	7,317	7,317
70	8,867	8,867	8,867	8,867	8,867	8,867	8,867	8,867	8,867
80	9,867	9,867	9,867	9,867	9,867	9,867	9,867	9,867	9,867
90	10,850	10,850	10,850	10,850	10,850	10,850	10,850	10,850	10,850
100	12,483	12,483	12,483	12,483	12,483	12,483	12,483	12,483	12,483
150	17,883	17,883	17,883	17,883	17,883	17,883	17,883	17,883	17,883
200	24,333	24,333	24,333	24,333	24,333	24,333	24,333	24,333	24,333
250	30,033	30,033	30,033	30,033	30,033	30,033	30,033	30,033	30,033
300	35,367	35,367	35,367	35,367	35,367	35,367	35,367	35,367	35,367
350	41,267	41,267	41,267	41,267	41,267	41,267	41,267	41,267	41,267
400	47,617	47,617	47,617	47,617	47,617	47,617	47,617	47,617	47,617
500	58,467	58,467	58,467	58,467	58,467	58,467	58,467	58,467	58,467

Fuente: presentación propia de los autores.

Tabla 3. Resultados para el *makespan* mediante la regla SPT.

	SPT, makespan								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	2,467	2,467	2,467	2,467	2,467	2,467	2,467	2,467	2,467
30	3,700	3,700	3,700	3,700	3,700	3,700	3,700	3,700	3,700
40	4,800	4,800	4,800	4,800	4,800	4,800	4,800	4,800	4,800
50	6,033	6,033	6,033	6,033	6,033	6,033	6,033	6,033	6,033
60	7,317	7,317	7,317	7,317	7,317	7,317	7,317	7,317	7,317
70	8,867	8,867	8,867	8,867	8,867	8,867	8,867	8,867	8,867
80	9,867	9,867	9,867	9,867	9,867	9,867	9,867	9,867	9,867
90	10,850	10,850	10,850	10,850	10,850	10,850	10,850	10,850	10,850
100	12,483	12,483	12,483	12,483	12,483	12,483	12,483	12,483	12,483
150	17,883	17,883	17,883	17,883	17,883	17,883	17,883	17,883	17,883
200	24,333	24,333	24,333	24,333	24,333	24,333	24,333	24,333	24,333
250	30,033	30,033	30,033	30,033	30,033	30,033	30,033	30,033	30,033
300	35,367	35,367	35,367	35,367	35,367	35,367	35,367	35,367	35,367
350	41,267	41,267	41,267	41,267	41,267	41,267	41,267	41,267	41,267
400	47,617	47,617	47,617	47,617	47,617	47,617	47,617	47,617	47,617
500	58,467	58,467	58,467	58,467	58,467	58,467	58,467	58,467	58,467

Fuente: presentación propia de los autores.

Tabla 4. Resultados de las simulaciones para el *makespan* mediante la regla LPT.

	LPT, makespan								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	2,467	2,467	2,467	2,467	2,467	2,467	2,467	2,467	2,467
30	3,700	3,700	3,700	3,700	3,700	3,700	3,700	3,700	3,700
40	4,800	4,800	4,800	4,800	4,800	4,800	4,800	4,800	4,800
50	6,033	6,033	6,033	6,033	6,033	6,033	6,033	6,033	6,033
60	7,317	7,317	7,317	7,317	7,317	7,317	7,317	7,317	7,317
70	8,867	8,867	8,867	8,867	8,867	8,867	8,867	8,867	8,867
80	9,867	9,867	9,867	9,867	9,867	9,867	9,867	9,867	9,867
90	10,850	10,850	10,850	10,850	10,850	10,850	10,850	10,850	10,850
100	12,483	12,483	12,483	12,483	12,483	12,483	12,483	12,483	12,483
150	17,883	17,883	17,883	17,883	17,883	17,883	17,883	17,883	17,883
200	24,333	24,333	24,333	24,333	24,333	24,333	24,333	24,333	24,333
250	30,033	30,033	30,033	30,033	30,033	30,033	30,033	30,033	30,033
300	35,367	35,367	35,367	35,367	35,367	35,367	35,367	35,367	35,367
350	41,267	41,267	41,267	41,267	41,267	41,267	41,267	41,267	41,267
400	47,617	47,617	47,617	47,617	47,617	47,617	47,617	47,617	47,617
500	58,467	58,467	58,467	58,467	58,467	58,467	58,467	58,467	58,467

Fuente: presentación propia de los autores.

Tabla 5. Resultados para el tiempo promedio de flujo mediante la regla FIFO.

	FIFO, tiempo de flujo promedio								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	0,865	0,609	0,253	0,853	0,546	0,123	0,851	0,544	0,122
30	1,116	0,684	0,224	1,154	0,670	0,124	1,154	0,671	0,124
40	1,480	0,830	0,248	1,541	0,899	0,123	1,542	0,903	0,122
50	1,795	0,996	0,259	1,864	1,046	0,124	1,865	1,049	0,124
60	2,176	1,238	0,239	2,251	1,271	0,125	2,254	1,276	0,124
70	2,558	1,392	0,260	2,635	1,490	0,127	2,638	1,498	0,127
80	2,928	1,572	0,261	3,003	1,694	0,127	3,005	1,699	0,126
90	3,212	1,806	0,258	3,299	1,813	0,124	3,302	1,818	0,124
100	3,391	1,693	0,237	3,492	1,843	0,122	3,496	1,851	0,122
150	5,167	2,644	0,247	5,309	1,822	0,122	5,313	2,830	0,122
200	6,484	3,146	0,269	6,664	3,352	0,122	6,671	3,364	0,121
250	8,167	3,994	0,256	8,362	4,217	0,122	8,369	4,234	0,122
300	10,140	5,045	0,255	10,358	5,375	0,122	10,366	5,388	0,122
350	11,604	5,802	0,256	11,825	6,026	0,122	12,832	6,045	0,121
400	13,038	6,295	0,264	13,252	6,606	0,122	13,260	6,619	0,121
500	16,767	8,478	0,263	16,953	8,644	0,122	16,958	8,653	0,122

Fuente: presentación propia de los autores.

Tabla 6. Resultados para el tiempo promedio de flujo mediante la regla SPT.

	SPT, tiempo de flujo promedio								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	0,659	0,483	0,233	0,651	0,423	0,123	0,648	0,418	0,122
30	0,857	0,536	0,211	0,878	0,512	0,124	0,880	0,509	0,124
40	1,060	0,614	0,225	1,096	0,637	0,123	1,098	0,635	0,122
50	1,286	0,717	0,233	1,335	0,737	0,124	1,340	0,740	0,124
60	1,532	0,872	0,220	1,575	0,877	0,125	1,578	0,880	0,124
70	1,844	0,999	0,232	1,893	1,043	0,127	1,895	1,047	0,127
80	2,054	1,081	0,232	2,108	1,146	0,127	2,111	1,148	0,126
90	2,211	1,205	0,234	2,260	1,199	0,124	2,263	1,199	0,124
100	2,355	1,149	0,214	2,431	1,233	0,122	2,433	1,236	0,122
150	3,488	1,702	0,227	3,578	1,804	0,122	3,582	1,804	0,122
200	4,487	2,065	0,236	4,622	2,196	0,122	4,627	2,201	0,121
250	5,650	2,623	0,230	5,791	2,753	0,122	5,796	2,763	0,122
300	6,851	3,219	0,225	6,998	3,403	0,122	7,005	3,411	0,122
350	7,851	3,683	0,226	8,004	3,805	0,122	8,009	3,617	0,121
400	13,038	4,058	0,232	9,120	4,240	0,122	13,260	4,247	0,121

500	16,767	5,362	0,233	11,429	5,440	0,122	16,958	5,442	0,122

Fuente: presentación propia de los autores.

Tabla 7. Resultados para el tiempo promedio de flujo mediante la regla LPT.

	LPT, tiempo de flujo promedio								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	1,004	0,703	0,272	0,994	0,675	0,123	0,992	0,673	0,122
30	1,412	0,906	0,239	1,463	0,900	0,124	1,467	0,901	0,124
40	1,858	1,105	0,266	1,926	1,225	0,123	1,928	1,229	0,122
50	2,367	1,421	0,286	2,457	1,523	0,124	2,455	1,526	0,124
60	2,800	1,738	0,264	2,905	1,808	0,125	2,912	1,833	0,124
70	3,406	2,048	0,293	3,469	2,216	0,127	3,472	2,225	0,127
80	3,869	2,304	0,291	3,953	2,520	0,127	3,955	2,530	0,126
90	4,185	2,563	0,297	4,323	2,649	0,124	4,332	2,678	0,124
100	4,614	2,612	0,266	4,726	2,841	0,122	4,731	2,851	0,122
150	6,954	3,992	0,292	7,105	4,325	0,122	7,110	4,330	0,122
200	9,157	5,205	0,320	9,375	5,562	0,122	9,362	5,592	0,121
250	8,167	6,548	0,302	11,729	6,893	0,122	11,736	6,917	0,122
300	10,140	5,045	0,225	14,142	8,583	0,122	14,152	8,596	0,122
350	11,604	5,802	0,295	16,349	6,026	0,122	16,356	9,837	0,121
400	13,038	6,295	0,309	18,702	6,606	0,122	18,710	11,165	0,121
500	16,767	8,478	0,306	23,408	8,644	0,122	23,415	14,169	0,122

Fuente: presentación propia de los autores.

Tabla 8. Resultados para el nivel de utilización de la máquina mediante la regla FIFO.

	FIFO, utilización del recurso								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	100%	97%	75%	100%	100%	74%	100%	100%	74%
30	100%	97%	65%	100%	100%	73%	100%	100%	73%
40	100%	99%	67%	100%	100%	62%	100%	100%	72%
50	100%	99%	68%	100%	100%	72%	100%	100%	72%
60	100%	99%	71%	100%	100%	72%	100%	100%	72%
70	100%	100%	72%	100%	100%	73%	100%	100%	73%
80	100%	100%	72%	100%	100%	73%	100%	100%	73%
90	100%	99%	67%	100%	100%	71%	100%	100%	72%
100	100%	100%	67%	100%	100%	70%	100%	100%	70%
150	100%	100%	68%	100%	100%	70%	100%	100%	70%
200	100%	100%	68%	100%	100%	70%	100%	100%	70%
250	100%	100%	69%	100%	100%	70%	100%	100%	70%
300	100%	100%	69%	100%	100%	70%	100%	100%	70%
350	100%	100%	69%	100%	100%	70%	100%	100%	70%
400	100%	100%	70%	100%	100%	70%	100%	100%	70%
500	100%	100%	70%	100%	100%	70%	100%	100%	70%

Fuente: presentación propia de los autores.

Tabla 9. Resultados para el nivel de utilización de la máquina mediante la regla SPT.

	SPT, utilización del recurso								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	100%	97%	75%	100%	100%	74%	100%	100%	74%
30	100%	97%	65%	100%	100%	73%	100%	100%	73%
40	100%	99%	67%	100%	100%	62%	100%	100%	72%
50	100%	99%	68%	100%	100%	72%	100%	100%	72%
60	100%	99%	71%	100%	100%	72%	100%	100%	72%
70	100%	100%	72%	100%	100%	73%	100%	100%	73%
80	100%	100%	72%	100%	100%	73%	100%	100%	73%
90	100%	99%	67%	100%	100%	71%	100%	100%	72%
100	100%	100%	67%	100%	100%	70%	100%	100%	70%
150	100%	100%	68%	100%	100%	70%	100%	100%	70%
200	100%	100%	68%	100%	100%	70%	100%	100%	70%
250	100%	100%	69%	100%	100%	70%	100%	100%	70%
300	100%	100%	69%	100%	100%	70%	100%	100%	70%
350	100%	100%	69%	100%	100%	70%	100%	100%	70%
400	100%	100%	70%	100%	100%	70%	100%	100%	70%
500	100%	100%	70%	100%	100%	70%	100%	100%	70%

Fuente: presentación propia de los autores.

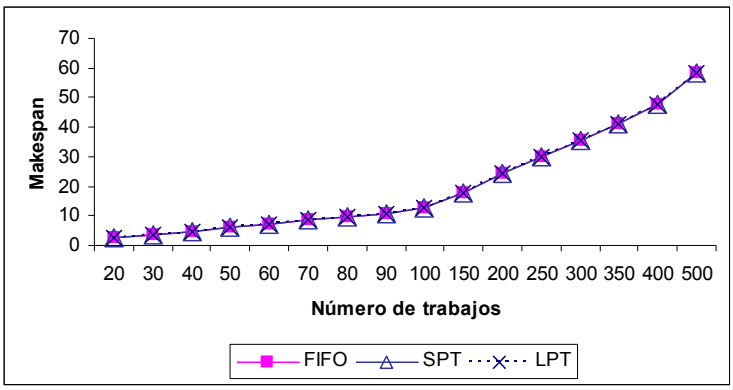
Tabla 10. Resultados para el nivel de utilización de la máquina mediante la regla LPT.

	LPT, utilización del recurso								
	Tiempo entre llegadas								
	Exponencial (λ)			**Normal(μ; 0,05μ)**			**Constante**		
n	**3**	**5**	**10**	**3**	**5**	**10**	**3**	**5**	**10**
20	100%	97%	75%	100%	100%	74%	100%	100%	74%
30	100%	97%	65%	100%	100%	73%	100%	100%	73%
40	100%	99%	67%	100%	100%	62%	100%	100%	72%
50	100%	99%	68%	100%	100%	72%	100%	100%	72%
60	100%	99%	71%	100%	100%	72%	100%	100%	72%
70	100%	100%	72%	100%	100%	73%	100%	100%	73%
80	100%	100%	72%	100%	100%	73%	100%	100%	73%
90	100%	99%	67%	100%	100%	71%	100%	100%	72%
100	100%	100%	67%	100%	100%	70%	100%	100%	70%
150	100%	100%	68%	100%	100%	70%	100%	100%	70%
200	100%	100%	68%	100%	100%	70%	100%	100%	70%
250	100%	100%	69%	100%	100%	70%	100%	100%	70%
300	100%	100%	69%	100%	100%	70%	100%	100%	70%
350	100%	100%	69%	100%	100%	70%	100%	100%	70%
400	100%	100%	70%	100%	100%	70%	100%	100%	70%
500	100%	100%	70%	100%	100%	70%	100%	100%	70%

Fuente: presentación propia de los autores.

Como se puede observar en las Tablas 2 a la 4, los valores obtenidos para el *makespan* no varían cuando se cambia la regla de secuenciación o la distribución para los tiempos entre llegadas, para un tamaño de instancia dado (ver Figura 1). Esto se explica por el hecho de que los algoritmos implementados (SPT, FIFO y LPT) son todos conservadores, es decir, $C_{\max}^{FIFO} = C_{\max}^{SPT} = C_{\max}^{LPT} = \sum p_j + T$, donde T es el tiempo ocioso de la máquina, el cual es minimizado en la secuencia de ejecución al aplicar estas reglas. Se debe anotar que debido a que las mismas instancias son empleadas para analizar cada regla de secuenciación, los valores de $\sum p_j$ y de $\sum r_j$ permanecen constantes para una instancia dada. Por lo tanto, el tiempo ocioso de la máquina se debe fundamentalmente a la variabilidad en las llegadas de las tareas al sistema. Desde el punto de vista numérico este tiempo ocioso se verá reflejado en el nivel de utilización de la máquina, ilustrado en las Tablas 8, 9 y 10.

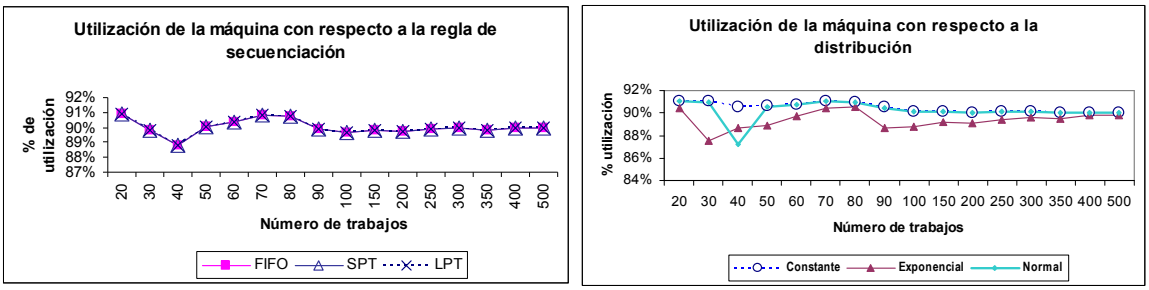
Figura 1. *Makespan* en función del tamaño de la instancia.



Fuente: presentación propia de los autores.

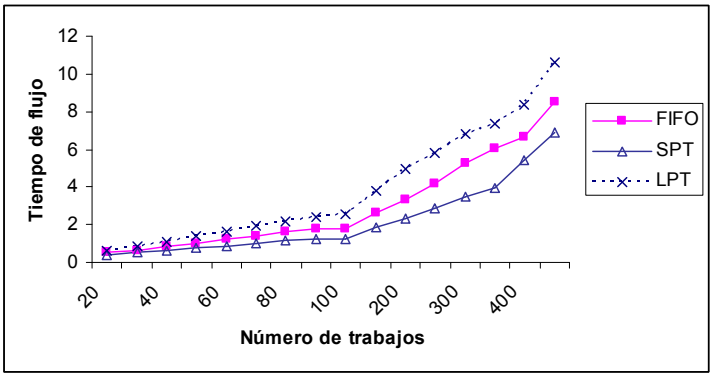
Con el fin de ver más claramente el comportamiento del sistema cuando se varía el tamaño de la instancia, las Figuras 2 y 3 muestran, respectivamente, el comportamiento del nivel de utilización de la máquina y la evolución del tiempo promedio de flujo con respecto al número de trabajo a ejecutar. En la Figura 2 se puede observar el comportamiento del porcentaje de utilización del recurso a medida que se incrementa el número de tareas en el sistema, apreciándose un decrecimiento en el porcentaje de utilización de la máquina a medida que aumenta el número de tareas. También, se analiza cómo las llegadas con distribuciones constante y normal tienen más utilización del recurso que la distribución exponencial. Es importante resaltar la semejanza entre las distribuciones constante y normal en casi la totalidad de los puntos de estas gráficas. Esto último puede deberse a la desviación estándar de la distribución normal, la cual fue calculada como el 5% del valor de la media. Esto indicaría una variabilidad muy pequeña y relativamente poco significativa con respecto a una llegada regular de la tareas al sistema.

Figura 2. Nivel de utilización de la máquina en función del tamaño de la instancia.



Fuente: presentación propia de los autores.

Figura 3. Tiempo de flujo en función del tamaño de la instancia.



Fuente: presentación propia de los autores.

Al analizar la diferencia de los promedios dados en las tablas, se observa que con la estrategia LPT el flujo de entidades es mayor en 1,7415 unidades de tiempo que con la estrategia SPT. Con la utilización de la estrategia FIFO, el flujo de las entidades es 0,8167 unidades de tiempo mayor que con la estrategia SPT. Se utilizó entonces el software SPSS versión 12.0 para realizar un análisis de varianza (ANOVA) de un solo factor (ver Figura 4). Se realizó una prueba de comparación múltiple para determinar la diferencia entre las estrategias, y con base en esta prueba se concluye que para el indicador del tiempo promedio de flujo sí existe diferencia significativa entre las estrategias SPT y LPT. Sin embargo, no se encuentra diferencia significativa entre las estrategias FIFO y SPT y entre FIFO y LPT.

Figura 4. Resultados del análisis de varianza.

Multiple Comparisons

Dependent Variable: Flujo

	(I) Estrategia	(J) Estrategia	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Scheffe	SPT	LPT	-1,7415882*	,4765283	,001	-2,912093	-,571083
		FIFO	-,8167028	,4765283	,231	-1,987208	,353802
	LPT	SPT	1,7415882*	,4765283	,001	,571083	2,912093
		FIFO	,9248854	,4765283	,153	-,245620	2,095390
	FIFO	SPT	,8167028	,4765283	,231	-,353802	1,987208
		LPT	-,9248854	,4765283	,153	-2,095390	,245620
Tamhane	SPT	LPT	-1,7415882*	,4871177	,001	-2,912915	-,570261
		FIFO	-,8167028	,4113016	,137	-1,804854	,171449
	LPT	SPT	1,7415882*	,4871177	,001	,570261	2,912915
		FIFO	,9248854	,5241993	,218	-,334594	2,184365
	FIFO	SPT	,8167028	,4113016	,137	-,171449	1,804854
		LPT	-,9248854	,5241993	,218	-2,184365	,334594

*. The mean difference is significant at the .05 level.

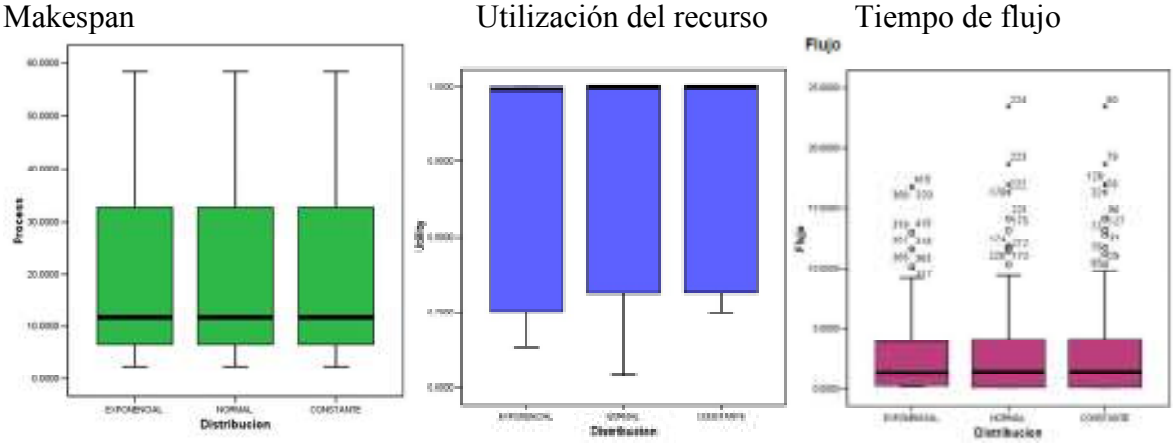
Fuente: presentación propia de los autores.

Vale la pena aclarar que, si bien no se encuentran diferencias estadísticamente significativas entre las reglas SPT y FIFO para el valor del tiempo de flujo, según el Teorema 2 presentado anteriormente el valor obtenido aplicando la regla SPT siempre será inferior o igual al valor obtenido aplicando FIFO, para cualquier instancia (es decir, SPT siempre domina a FIFO). Este resultado teórico se verifica experimentalmente puesto que SPT tiene un mejor desempeño que FIFO en el 75% de los casos, mientras que en el restante 25% SPT y FIFO arrojan valores solución iguales para el tiempo de flujo promedio (y, por consiguiente, para el tiempo total de terminación).

Con respecto a la comparación entre distribuciones para las llegadas de las tareas, se realizó la comparación a través de diagramas de caja y bigotes, como se muestra en la Figura 5. Se

observa que los resultados de la media y la desviación estándar para el *makespan* se comportan igual en los tres tipos de distribuciones y a su vez arroja como resultado unos valores de error estándar, valores mínimos y máximos iguales. Para el caso de los indicadores de nivel de utilización del recurso y tiempo de flujo promedio, se observa que existen diferencias entre los grupos. A continuación se realizó la prueba de análisis de varianza (ANOVA) de un factor, donde las variables dependientes son los indicadores de nivel de utilización y el tiempo de flujo promedio, y las variables independientes son los tipos de distribución para los tiempos entre llegadas. La tabla ANOVA, presentada en la Figura 6, muestra que la variabilidad entre las llegadas de las tareas al sistema no afecta de manera estadísticamente significativa el porcentaje de utilización de la máquina ni el tiempo promedio de flujo de las tareas.

Figura 5. Diagramas de caja y bigotes para la comparación de la variabilidad entre llegadas de tareas.



Fuente: presentación propia de los autores.

Figura 6. Resultados del análisis de varianza para la comparación de la variabilidad entre llegadas de tareas.

Comparación para la utilización de la máquina

Scheffe

Dependent Variable	(I) Distribucion	(J) Distribucion	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Utility	EXPONENCIAL	NORMAL	-,0082590	,0164903	,882	-,048764	,032246
		CONSTANTE	-,0105590	,0164903	,815	-,051064	,029946
	NORMAL	EXPONENCIAL	,0082590	,0164903	,882	-,032246	,048764
		CONSTANTE	-,0023000	,0164903	,990	-,042805	,038205
	CONSTANTE	EXPONENCIAL	,0105590	,0164903	,815	-,029946	,051064
		NORMAL	,0023000	,0164903	,990	-,038205	,042805

Comparación para el tiempo de flujo promedio

Scheffe

Dependent Variable	(I) Distribucion	(J) Distribucion	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Flujo	EXPONENCIAL	NORMAL	-,1493389	,4836381	,983	-1,337308	1,038630
		CONSTANTE	-,3287937	,4836381	,794	-1,516763	,859175
	NORMAL	EXPONENCIAL	,1493389	,4836381	,983	-1,038630	1,337308
		CONSTANTE	-,1794549	,4836381	,933	-1,367424	1,008514
	CONSTANTE	EXPONENCIAL	,3287937	,4836381	,794	-,859175	1,516763
		NORMAL	,1794549	,4836381	,933	-1,008514	1,367424

Fuente: presentación propia de los autores.

6. Conclusiones y perspectivas de trabajo futuro

En este artículo se analizó el problema de la programación (secuenciación) de tareas en un modelo de máquina simple. Las tareas estaban sujetas a fechas de llegada distintas y el ordenamiento en la ejecución en la máquina debía hacerse de manera dinámica (*on-line*)

durante el horizonte de tiempo. Primero, se presentó la formalización del problema y las nociones fundamentales sobre el cálculo de la competitividad de los algoritmos (reglas) *on-line* de secuenciación. Seguidamente, se adelantó el estudio teórico de la competitividad de las reglas SPT (*Shortest Processing Time*) y FIFO (*First In, First Out*). Estas dos reglas han sido altamente utilizadas tanto en la práctica como en la literatura para la creación de algoritmos más sofisticados tanto en problemas de tipo *off-line* (estáticos) como *on-line* (dinámicos). Finalmente, se hizo un estudio experimental completo cuyo objetivo fue el análisis del comportamiento de estas dos reglas, comparándolas entre sí y también con respecto a la regla LPT (*Longest Processing Time*).

Este estudio experimental permitió observar que no existe diferencia estadística significativa en el desempeño promedio de las reglas SPT y FIFO, con respecto a la minimización del tiempo promedio de flujo (o minimización del inventario en proceso). Sin embargo, vale la pena resaltar que estas conclusiones deben tomarse en cuenta sólo para casos generales, puesto que valores particulares (valores extremos) de los parámetros de las tareas pueden conducir a resultados indeseables de la función objetivo. Se debe resaltar que para los valores absolutos de las instancias consideradas, y teniendo en cuenta los resultados teóricos y experimentales obtenidos, se comprueba que para el problema de secuenciación *on-line* de tareas en una sola máquina, la regla SPT arroja la mejor solución para el tiempo de flujo que las reglas FIFO y LPT, mientras que los valores del *makespan* y de la utilización del recurso resultan iguales bajo las tres reglas.

En la práctica, se reconoce que la ignorancia sobre el futuro puede ser altamente perjudicial para la obtención de buenas secuencias de ejecución de las tareas. Una pregunta interesante es, entonces, ¿cómo cuantificar los beneficios que se pueden obtener si se conociera la información futura? O dicho de otra forma, ¿cuál es la información pertinente sobre el futuro que permite mejorar los indicadores de desempeño de un sistema productivo? Intuitivamente, puede decirse que el conocimiento del futuro permite lograr un mejoramiento considerable de dichos indicadores. Sin embargo, en la literatura existen pocos trabajos focalizados en responder a esta pregunta desde el punto de vista cuantitativo. Entre ellos vale la pena resaltar los desarrollados en [Montoya-Torres, 2002, 2003] y en [Sepúlveda y Frein, 2004].

Referencias

Baker, K.R. *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons, 1974.

Garey, M.R., Johnson, D.S. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co, 1979.

Graham, R.L. Bounds on Multiprocessing Timing Anomalies. En: *SIAM Journal of Applied Mathematics*, 17, 1969, 416-429.

Hoogeveen, J.A., Vestjens, A. Optimal on-line Algorithms for Single-Machine Scheduling. En: *Lecture Notes in Computer Science*, 1084, 1996, 404-414.

Lee, H.L., Padmanabhan, V., Wang, S. The Bullwhip Effect in a Supply Chain. En: *Sloan Management Review*, Spring, 1997, 93-102.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P. Complexity of Machine Scheduling Problems. En: *Annals of Discrete Mathematics*, 1, 1977, 343-362.

Manesse, M.S., McGeoch, L.A., Sleator, D.D. Competitive Algorithms for Server Problems. En: *Journal of Algorithms*, 11, 1990, 208-230.

Mao, W., Kincaid, R.K., Rifkin, A. On-line Algorithms for a Single Machine Scheduling Problem. En: S. Nash, A. Sofer (eds.) *The Impact of Emerging Technologies on Computer Science and Operations Research*. Norwell: Kluwer Academic Publishers, 1995, p. 157-173.

Merchán-Alba, A.L., Rodríguez-Verjan, G.L. *Estudio del impacto de las estrategias de cooperación entre los eslabones de una cadena logística a nivel de la programación de la producción*. Trabajo de Grado en Ingeniería Industrial. Bogotá: Pontificia Universidad Javeriana, 2006.

Montoya-Torres, J.R. *Une étude de l'influence de l'information anticipée en ordonnancement dynamique*. Master of Science Thesis. Grennoble: Institut National Polytechnique de Grenoble, 2002.

Montoya-Torres, J.R. Competitive Analysis of a Better On-line Algorithm to minimize total Completion Time on a Single-Machine. En: *Journal of Global Optimization*, 27 (1), 2003, 97-103.

Motwani, R., Phillips, S., Torng, E. Non-Clairvoyants Scheduling. *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 422-431. Austin, USA. January 25-27, 1993.

Phipps, T.E. Machine Repair as a Priority Waiting-Line Problem. En: *Operations Research*, 4, 1956, 45-61.

Pinedo, M. *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, 1995.

Schrage, L.E. A Proof of the Optimality of the Shortest Remaining Processing Time Discipline. En: *Operations Research*, 16, 1968, 678-690.

Sepúlveda, J.P., Frein, Y. *About the Effect of Coordination and Information Sharing on the Performance of a Typical Supply Chain.* Preprints of the Third International Conference on Management and Control of Production and Logistics. CD-ROM. Santiago de Chile, 2004.

Sgall, J. On-Line Scheduling – A Survey. En: A. Fiat, G.I. Woeginger (eds.). *On-line Algorithms: The State of the Art*. Vol. 1.442. Berlin: Springer-Verlag, 1999. p. 196-231.

Smith, W.E. Various Optimizers for Single-Stage Production. En: *Naval Research Logistics Quarterly*, 3, 1956, 56-66.

Vestjens, A.P. *On-line Machine Scheduling*. PhD thesis. Eindhoven University of Technology, 1997.

Wolper, P. *Introduction à la calculabilité*. Collection IIE, InterEditions, 1991.