Minimización Heurística del Número de Tareas Tardías al Secuenciar Líneas de Flujo

M. Angélica Salazar Aguilar
División de Posgrado en Ing. de Sistemas
FIME, UANL
e-mail: angy@yalma.fime.uanl.mx

Roger Z. Ríos Mercado
División de Posgrado en Ing. de Sistemas
FIME, UANL
e-mail: roger@uanl.mx

Resumen: En éste artículo se presenta una heurística (método de aproximación) que permite encontrar una secuencia de n tareas en un ambiente de líneas de flujo de m máquinas para el problema de minimizar el número de tareas que se entregan tarde (después de su tiempo de entrega). Este procedimiento, basado en el algoritmo de Moore para problemas de una máquina, se evalúa y se compara computacionalmente contra un método que genera secuencias sin tomar en cuenta la estructura del problema en una variedad de problemas bajo dos escenarios distintos: fechas de entrega estrictas y no estrictas. Se observa que el procedimiento propuesto brinda mejores resultados en cuanto a la calidad de la solución encontrada.

Palabras clave: Investigación de operaciones, problema de secuenciamiento, línea de flujo, tiempos de entrega, tareas tardías, heurística.

Abstract: In this paper a heuristic for minimizing the number of late jobs in a flow shop environment is presented. The proposed procedure, based on Moore's algorithm for single-machine problems, is implemented and empirically evaluated on a variety of problem instances under two different scenarios: tight and loose due dates. The heuristic was observed to deliver good-quality sequences.

Key-words: Operations research, scheduling problem, flow shop, due dates, late jobs, heuristic.

INTRODUCCIÓN

La ciencia de la toma de decisiones o investigación de operaciones (IO) está presente en todos los niveles y en todo tipo de industrias. La IO permite llevar a cabo la toma de decisiones, a través del análisis de la operación de cualquier sistema, la formulación y uso de modelos que permiten lograr los objetivos y metas propuestos, con un adecuado aprovechamiento de los recursos disponibles.

Su ámbito de aplicación es muy amplio, aplicándose a problemas de fabricación, transporte, construcción, telecomunicaciones, planificación y gestión financiera, ciencias de la salud, servicio, etc. En general, puede aplicarse en todos los problemas relacionados a la gestión, planificación y diseño [1].

Un problema muy frecuente en sistemas de fabricación es el de secuenciamiento de tareas en líneas de flujo. Lo que se busca es

secuenciar las tareas de tal forma que se aprovechen los recursos lo mejor posible y se cumpla con los objetivos propios del sistema.

Existen diversos objetivos que pueden ser optimizados utilizando una secuencia óptima, como por ejemplo:

- Cumplir con las fechas de entrega.
- Minimizar la tardanza de los trabajos.
- Minimizar el tiempo de respuesta.
- Minimizar el tiempo en el sistema.
- Minimizar horas extra.
- Maximizar la utilización de máquinas y mano de obra.
- Minimizar tiempo ocioso.
- Minimizar inventario de trabajo en proceso.

La investigación en el campo de secuenciamiento y programación de tareas (*sequencing and scheduling*) ha sido muy amplia. Un panorama excelente en el tema, incluyendo resultados de complejidad computacional, esquemas de optimización exacta y algoritmos de aproximación puede encontrarse en el trabajo de Lawler et al [2].

Una adecuada programación de tareas permite el buen funcionamiento de cualquier sistema manteniendo la eficiencia y el control de las operaciones [3].

En este trabajo se plantea el problema de encontrar una secuencia de n tareas en un ambiente de línea de flujo (*flow shop*) de m máquinas tal que minimice el número de tareas que se entregan tarde, denotado por $\sum U_j$ Matemáticamente, el problema consiste en encontrar una permutación de las tareas que permita entregar el menor número de tareas tarde, el cual es un problema típico de secuenciamiento de tareas. Es importante que las tareas cumplan con los tiempos de entrega especificados, porque en la mayoría de los casos, la violación de estos tiempos implica una penalización (U_j).

Para dar solución a este problema se realizó una modificación al algoritmo de Moore, el cual genera secuencias óptimas para problemas de n tareas en una máquina. El método fue implementado y evaluado computacionalmente en un amplio conjunto de instancias del problema. La experimentación computacional arrojó resultados que demuestran que este método es bueno, dado que los valores que adquiere la función objetivo (en la mayoría de los casos) son de mejor calidad que cuando se utiliza un método que ignora completamente la estructura del problema.

DESCRIPCIÓN DEL PROBLEMA

En un ambiente de línea de flujo, se tiene un conjunto de n tareas que deben ser procesadas en un conjunto de m máquinas cada una. Cada tarea tiene el mismo orden de ruteo tecnológico a través de las máquinas, es decir, cada una de las tareas debe ser procesada primero en la máquina 1, luego en la máquina 2, y así sucesivamente hasta llegar a la máquina m. Se supone que cada tarea está disponible inicialmente y tiene asignada una fecha de entrega, además, la secuencia de tareas en cada máquina es la misma y cada máquina puede procesar una sola tarea a la vez [4]. Las secuencias se denominan técnicamente *secuencias de permutación*.

El tiempo de procesamiento de la tarea j en la máquina i se denota por p_{ij}. El tiempo de entrega de la tarea j se denota por d_j. El tiempo en que una tarea existe en el sistema (es decir, el tiempo de terminación de la tareas en la última máquina en la cual requiere ser procesada) es una variable que se denota por C_j. La unidad de penalización (variable) de la tarea j se define como:

$$U_j = \begin{cases} 1 & si \quad C_j > d_j \\ 0 & si \quad C_j \le d_j \end{cases}$$

Las restricciones tecnológicas del sistema consisten en que una máquina no puede

procesar más de una tarea a la vez. El objetivo es encontrar una secuencia de *n* tareas que permita minimizar el número de tareas que se entregan tarde. La función que se desea minimizar se conoce como *función objetivo.* Entonces, la función objetivo se representa como

$$\min \sum_{j=1}^{n} U_j$$

Una forma de representar los problemas de secuenciamiento es un diagrama de Gantt, el cual muestra las actividades planeadas y completadas contra la escala de tiempo.

Ejemplo: Un sistema de 3x4, 3 máquinas y 4 trabajos. Los tiempos de procesamiento (p_{ij}) son los siguientes:

p_{ij}	j = 1	2	3	4
i = 1	4	3	5	7
2	7	7	2	9
3	3	3	4	5

Los tiempos de entrega son : d = (25,40,20,21). Considere la secuencia: S = (T_4,T_3,T_1,T_2). La Figura 1, muestra la representación del problema utilizando un diagrama de Gantt. Como puede observarse, el procesamiento de las tareas respeta la secuencia dada con sus respectivos tiempos de procesamiento.

Lo que se busca es encontrar una secuencia que permita entregar a tiempo el mayor número de tareas. Es decir, que C_j no exceda a d_j, para que la tarea sea entregada a tiempo. Si excede, entonces se considera la penalidad (U_j). En este ejemplo, los valores que toma la variable C_j son: C =(28,35,25,21).

Comparando C_j con d_j la función objetivo obtiene el siguiente valor:

$$\sum_{j=1}^{4} U_j = 2$$

El valor de la función objetivo indica que se entregan dos tareas tarde, en este caso las tareas 2 y 3. Para minimizar este número, habría que encontrar una secuencia que permita optimizar esta función objetivo (es decir, entregar el menor número de tareas tarde).

MÉTODO DE SOLUCIÓN

Para dar solución al problema planteado se realizó una modificación al algoritmo de Moore [5], el cual genera secuencias óptimas pero sólo se aplica a problemas de *n* tareas en una máquina. La modificación que se realizó permite encontrar una secuencia de *n* tareas que deben ser procesadas en *m* máquinas tal que reduce el número de tareas tardías.

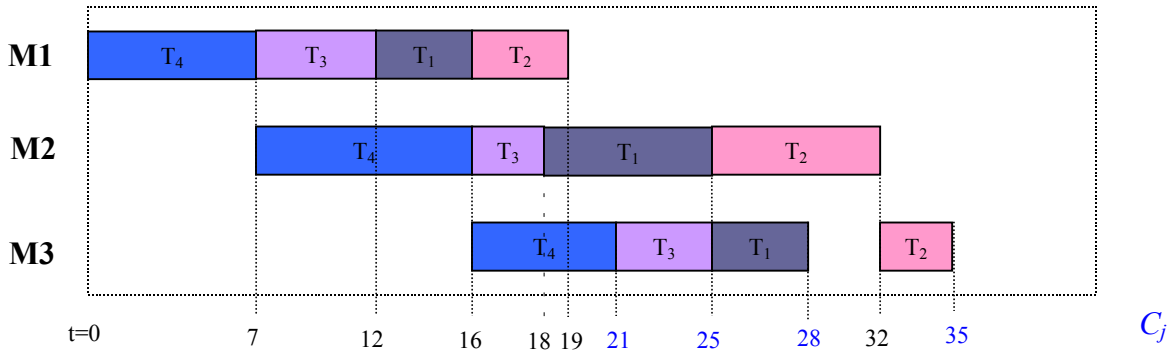

Figura 1. Diagrama de Gantt

Sea J el conjunto de tareas que cumplen con los tiempos de entrega. Sea J_d el conjunto de tareas que no cumplen con las fechas de entrega. Sea J_c el conjunto de las tareas que no se han secuenciado en la programación de tareas.

Algoritmo de Moore Modificado (AMM)

Paso 1:

$J=\varnothing$, $J_d=\varnothing$, $J_c=\{1, 2, 3, \ldots, n\}$

Paso 2:

j^* es el índice de la tarea $j \in J_c$ que satisface

$$d_{j^*} = \min\{d_j : j \in J_c\}$$

Agregar j^* a J
Eliminar j^* de J_c

Paso 3:

Si $C_{j^*} \leq d_{j^*}$
entonces, Ir al Paso 4.
Si no
Eliminar j^* de J
Agregar j^* a J_d

Paso 4:

Si $J_d=\varnothing$
entonces terminar.
Si no
Ir al Paso 2.

Al final la secuencia obtenida está formada por J y J_d. Las primeras en ser procesadas son las de J , después las de J_d . En la ejecución del algoritmo, nótese lo siguiente. Al seleccionar la tarea j^* en el paso 2, el cálculo de C_{j^*} se efectúa de forma recursiva calculando parcialmente C_{ij} (tiempo de terminación de la tarea j^* en la máquina i) para i=1, 2, ..., m. Este último valor C_{mj^*} es precisamente C_{j^*}, el cual representa tentativamente la terminación de la la tarea j^*. En el paso 3 se verifica la condición de permanencia ($C_{j^*} \leq d_{j^*}$), si la cumple, la tarea se queda, en caso contrario, se "desprograma" y la secuencia parcial queda tal y como estaba hasta antes de probar esta tarea j^*.

Este método es una buena alternativa para encontrar secuencias aceptables, sobre todo en problemas de dimensiones relativamente pequeñas. Recordemos que el algoritmo que Moore planteó genera secuencias óptimas para problemas de n tareas en una máquina. Naturalmente, este método no garantiza soluciones óptimas al problema de líneas de flujo en m máquinas; sin embargo intenta aprovechar la idea de Moore para generar secuencias de buena calidad.

EXPERIMENTACIÓN COMPUTACIONAL

El propósito del experimento es el de evaluar la efectividad del AMM. Los resultados obtenidos se comparan con los que se obtienen al considerar una secuencia generada por un método aleatorio (MA). El AMM se programó en lenguaje C usando el compilador gcc bajo el sistema operativo Solaris 7 en una estación de trabajo Sun Ultra 10.

Para llevar a cabo el experimento computacional, se consideraron dos escenarios del problema: un escenario estricto (con tiempos de entrega menores) y uno menos estricto (con tiempos de entrega mayores), con el fin de tener diversos puntos de comparación. Además se generaron 10 diferentes instancias para cada uno de éstos.

Las instancias en el escenario estricto se generaron aleatoriamente de acuerdo a lo siguiente: $p_{ij} \in [1,30]$, por lo que $p_{\max}=30$, $d_j \in [1, 1.2d]$, donde $d=0.5p_{\max}(m+n-1)$. Las instancias en el escenario menos estricto se generaron aleatoriamente considerando lo siguiente: $p_{ij} \in [1,30]$, por lo que $p_{\max}=30$, $d_j \in [1, 1.7d]$, donde $d=0.5p_{\max}(m+n-1)$.

Las Tablas 1 y 2 muestran los resultados que se obtuvieron al probar computacionalmente el AMM con 10 instancias del problema con

Tamaño	# de veces que minimiza $\sum U_j$			# veces que minimiza C_{max}	
m x n	**MA**	**Empate**	**AMM**	**MA**	**AMM**
2 x 50		1	9	4	6
2 x 100		2	8	8	2
2 x 500	1	1	8	5	5
10 x 50	1		9	2	8
10 x 100	3	1	6	5	5
10 x 500	6		4	8	2
20 x 50	2	2	6	6	4
20 x 100	2	1	7	8	2
20 x 500	5		5	9	1
Total	**20**	**8**	**62**	**55**	**35**

Tabla 1. Resultados obtenidos al probar el AMM bajo las condiciones de un escenario estricto.

Tamaño	# de veces que minimiza $\sum U_j$			# veces que minimiza C_{max}	
m x n	**MA**	**Empate**	**AMM**	**MA**	**AMM**
2 x 50		6	4	5	5
2 x 100		8	2	7	3
2 x 500		4	6	5	5
10 x 50	1	2	7	2	8
10 x 100	1	3	6	9	1
10 x 500	2		8	6	4
20 x 50		5	5	6	4
20 x 100	3	3	4	2	8
20 x 500	3	2	5	7	3
Total	**10**	**33**	**47**	**49**	**41**

Tabla 2. Resultados obtenidos al probar el AMM bajo las condiciones de un escenario menos estricto.

cada una de las dimensiones que se indican en la primera columna. La segunda columna (MA) indica cuántas veces de 10 la secuencia aleatoria arrojó mejores resultados que la secuencia generada por el AMM. La tercer columna (Empate) indica el número de veces que el MA y el AMM arrojaron los mismos resultados, es decir, la función objetivo tomó el mismo valor sin importar el método utilizado. La cuarta columna (AMM), indica cuántas veces de 10 la secuencia generada por el AMM arrojó mejores resultados que la secuencia generada por el MA. La quinta y sexta columna indican el número de veces que cada método encontró una mejor solución usando como función objetivo el tiempo de terminación de la última tarea procesada en el sistema C_{max} (*makespan*). Esto es desde luego un objetivo diferente y se llevó a cabo para ilustrar que el método de solución debe tomar en cuenta el objetivo del problema.

Las gráficas, con los resultados de cada experimento de forma detallada se encuentran disponibles por parte de los autores. Como puede observarse, en la mayoría de los resultados obtenidos, el AMM arroja mejores valores para la función objetivo que los que se producen con la secuencia generada de forma aleatoria, el cual asemeja una práctica común en la industria, ya que se secuencian tareas sin tener un conocimiento de la estructura del problema. En el escenario menos estricto, la diferencia es más notoria. Esto se debe a que los tiempos de entrega asociados a las tareas son mayores, por lo que la

probabilidad de que las tareas cumplan con sus tiempos de entrega es mayor. En el escenario estricto se puede apreciar que la mejora se presenta en los problemas que utilizan un número menor de máquinas, esto se debe a que el AMM se basa en un método que se aplica sólo a problemas de una máquina.

Las limitaciones que presenta el AMM se deben a que este algoritmo realiza cambios en la programación de tareas considerando sólo la ultima tarea seleccionada, es decir, no considera las tareas que fueron tomadas en cuenta con anterioridad en la programación de tareas, algo que el algoritmo de Moore si hace en el caso de una máquina. Es por eso que la secuencia generada no es una secuencia óptima. Si para generar la secuencia se tomara en cuenta las tareas ya consideradas en la programación de tareas, el tiempo de utilización de CPU sería mayor, ya que se requeriría de un número mayor de operaciones para reprogramar las tareas. Sin embargo, puede valer la pena si la mejora en el objetivo resultara significativa.

Nótese que, si bien es cierto que el AMM genera secuencias que reducen el número de tareas tardías, esto no implica que la secuencia minimice el tiempo de terminación de la última tarea (C_{max}). Lo anterior demuestra que el objetivo que se desea optimizar en cualquier sistema debe ser planteado de forma precisa, para que los resultados obtenidos impacten en el rendimiento del mismo, mejorando la medida de desempeño que se desee optimizar.

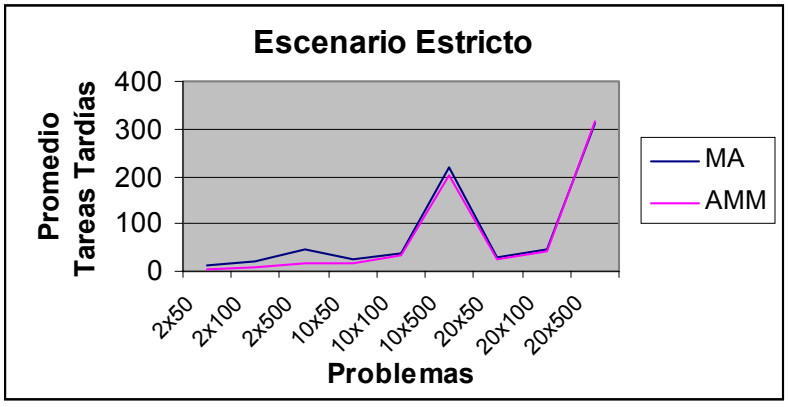

Figura 2. Número promedio de tareas tardías, en un escenario estricto.

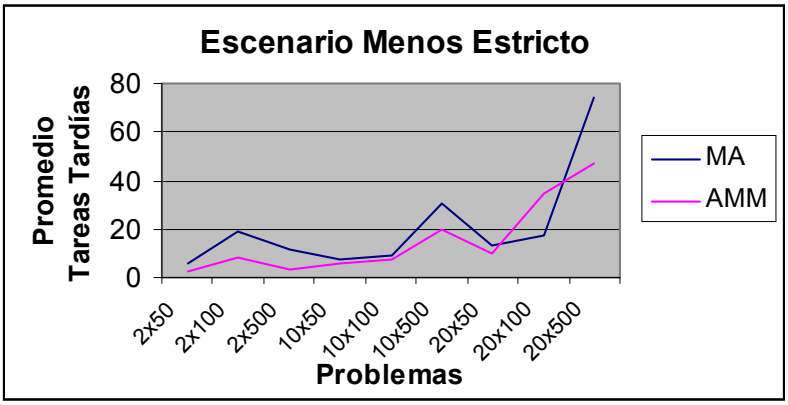

Figura 3. Número promedio de tareas tardías, en un escenario menos estricto.

Las Figuras 2 y 3 muestran el número promedio de tareas tardías que se generó al experimentar computacionalmente con el AMM y el MA. El eje x corresponde a las dimensiones del problema. Recuérdese que se analizaron 10 para cada una de estas. Los resultados de la Figura 1 corresponden a las instancias que se generaron bajo las condiciones de un escenario estricto. Los resultados de la Figura 2 corresponden a las instancias que se generaron bajo las condiciones de un escenario menos estricto.

CONCLUSIONES

En este trabajo se ha presentado el AMM para el problema de minimización del número de tareas tardías en un problema de secuenciamiento de líneas de flujo. Este método pretende encontrar secuencias aceptables que permitan encontrar una solución de mejor calidad que la que se encontraría utilizando una secuencia aleatoria.

El estudio computacional reveló que:

- La calidad de los resultados obtenidos con el AMM es bastante buena, lo cual se puede ver reflejado en los valores obtenidos en la función objetivo.
- Los resultados obtenidos dependen de la medida de desempeño que se desee optimizar. Se pudo observar que al minimizar el número de tareas tardías no necesariamente se obtiene un valor mínimo en el tiempo de terminación de la última tarea.
- Este procedimiento arroja buenos resultados, aunque requiere mayor tiempo de utilización de CPU que cuando se genera una secuencia aleatoria.

Una posible mejora sería ordenar las tareas que se entregan tarde en orden ascendente, para que los tiempos de terminación no difieran tanto de los tiempos de entrega establecidos, esto no implica que el valor de la función objetivo se minimice, simplemente se reduce el tiempo de retraso.

Las limitaciones que presenta el AMM se deben a que este algoritmo realiza cambios en la programación de tareas considerando sólo la ultima tarea seleccionada, es decir, no considera las tareas que fueron tomadas en cuenta con anterioridad en la programación de tareas, algo que el algoritmo de Moore si hace para el caso de una máquina. Por lo anterior, la secuencia generada no es una secuencia óptima. Si para generar la secuencia se tomara en cuenta las tareas ya consideradas en la programación de tareas, el tiempo de utilización de CPU sería mayor, ya que se requeriría de un número mayor de operaciones para reprogramar las tareas. Sin embargo, puede valer la pena si la mejora en el objetivo resultara significativa.

Agradecimientos: El trabajo de María Angélica Salazar Aguilar fue apoyado por una beca otorgada por la Academia Mexicana de Ciencias, dentro del XIII Verano de Investigación Científica.

REFERENCIAS

[1] F. Hillier y G. J. Lieberman. *Introducción a la Investigación de Operaciones*. 3ra. Edición, Mc Graw Hill, México, 1982.

[2] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan y D. Shmoys. Sequencing and scheduling: Algorithms and complexity. En S. S. Graves, A. H. G. Rinnooy Kan y P. Zipkin, editores, *Handbook in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory*, 445-522. North-Holland, New York, EUA, 1993.

[3] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Prentice-Hall, Englewood Cliffs, New Jersey, EUA, 1995.

[4] R. Z. Ríos y J. F. Bard. Heurísticas para secuenciamiento de tareas en líneas de flujo. *Ciencia UANL,* 4(1):48-54, 2001.

[5] J. M. Moore. An *n* job, one machine sequencing algorithm for minimizing

the number of late jobs. *Management Science*, 5:102-109, 1968.

Fichas Biográficas

María Angélica Salazar Aguilar

María Angélica Salazar Aguilar cursa actualmente el primer semestre de la Maestría en Ciencias en Ingeniería de Sistemas, en la División de Posgrado de Ingeniería de Sistemas de la FIME, UANL. Es egresada de la carrera de Ing. en Sistemas Computacionales con especialidad en Redes y Sistemas Distribuidos del el Instituto Tecnológico de Querétaro. Obtuvo una beca por parte de la Academia Mexicana de Ciencias para participar en el XIII Verano de Investigación Científica 2003, el cual llevó a cabo en la División de Posgrado de Ingeniería de Sistemas de la FIME, UANL, bajo la tutela del Dr. Roger Z. Ríos.

Roger Z. Ríos Mercado

El Dr. Roger Z. Ríos Mercado labora actualmente como Profesor de Tiempo Completo y Exclusivo en el Programa de Posgrado en Ingeniería de Sistemas de la FIME, UANL. Recibió sus títulos de Doctor y Maestro en Ciencias en Investigación de Operaciones e Ingeniería Industrial de la Universidad de Texas en Austin, y su título de Lic. en Matemáticas de la UANL. Sus áreas de interés son investigación de operaciones, desarrollo de heurísticas y optimización estocástica, con aplicación a problemas de toma de decisiones provenientes de la industria del gas y procesos de manufactura. Más sobre su trabajo puede encontrarse en: *http://yalma.fime.uanl.mx/~roger/*