Rev.Acad.Canar.Cienc., I, 217-226 (1990)

SECUENCIACION DE TRABAJOS SOBRE UNA MAQUINA CON RESTRICCIONES DE PRECEDENCIA

D. Alcaide[2] y J. Sicilia[1]

[1]Departamento de Estadística e I.O.
Universidad de La Laguna
38204-La Laguna, España

[2]Departamento de Informática y Sistemas
Universidad de Las Palmas de Gran Canaria
38200-La Laguna, España

ABSTRACT

Suppose n jobs are to be processed on a single machine. These jobs can not be performed in any order, because a precedence relation between the jobs is specified. This paper develops an heuristic method to find a schedule which minimizes the total tardiness.

Keywords: Scheduling, Tardiness, Precedence Constraints.

INTRODUCCION

Al amparo de la Investigación Operativa, y en los últimos años, se han desarrollado diversas disciplinas dirigidas a optimizar la utilización de los escasos recursos disponibles, para lograr la máxima eficacia en la aplicación de los mismos.

Una de esas disciplinas, conocida como Planificación y Secuenciación, estudia la asignación entre actividades o tareas a realizar y máquinas disponibles para realizarlas, buscando la ordenación de trabajos que optimice alguna medida o criterio considerado.

Entre dichas medidas podemos citar el tiempo total de completación de los trabajos, el número de trabajos que exceden de unas fechas límite, la tardanza total de todos los trabajos, etc.; siendo ésta última la considerada en el presente artículo.

Si los trabajos pueden secuenciarse en cualquier orden, se dice que los mismos son independientes entre sí. No obstante, en la práctica, éste no es el caso mas común, debido tal vez a consideraciones de tipo tecnológico o humano que obligan a realizar un trabajo después de haber finalizado otros que le preceden. Tales restricciones de precedencia imponen un orden parcial sobre los trabajos e incrementan la complejidad computacional del

correspondiente problema de secuenciación.

La teoría de la Complejidad Computacional ofrece un soporte teórico para distinguir formalmente entre problemas bien resueltos, aquellos para los cuales se ha encontrado un algoritmo cuyo tiempo de computación está acotado por una función polinomial del tamaño del problema, y los llamados NP-duros, para los cuales no se encuentran algoritmos en tiempo polinomial y, al mismo tiempo, la búsqueda de dicho algoritmo es altamente dificultosa.

Dentro de este último grupo, se encuentra el problema que nos ocupa. Tanto Baker (1) como French (3) comentan en sus respectivos libros la necesidad de abordar el problema de la tardanza total mediante métodos heurísticos, debido a que la complejidad del problema requiere un tiempo exponencial. Más concretamente, Lenstra y Rinnooy Kan (7) demuestran que el problema de minimizar la tardanza total en una máquina con restricciones de precedencia es NP-duro.

En tales circunstancias, es ampliamente justificada la consideración de métodos heurísticos que ofrezcan soluciones aproximadas al problema planteado. En ese contexto se enmarca el presente trabajo.

PLANTEAMIENTO DEL PROBLEMA

Sean n trabajos que se desean procesar sobre una máquina, los cuales cumplen las siguientes características: todos están disponibles para ser procesados en cualquier momento y una vez iniciada la realización de un trabajo éste no puede interrumpirse para comenzar ningún otro. La secuenciación de los trabajos debe adecuarse a unas relaciones de precedencia entre trabajos fijada a priori. Dicha relación es recogida en un grafo acíclico G cuyos nodos sean los trabajos numerados de tal forma que si (i,j) es una arista del grafo, se entenderá que i debe ir siempre antes que j.

Denotaremos por $\Gamma(j)$ al conjunto de trabajos que deben realizarse posteriormente a la completación del trabajo j, esto es, el conjunto de vértices siguientes a j en el grafo G. Dar una estructura de precedencia R sobre los trabajos, equivaldrá a especificar los correspondientes conjuntos $\Gamma(j)$, $\forall j$.

Una secuencia S se dice que es compatible con la estructura de precedencia R, si no vulnera ninguna de las restricciones de precedencia fijadas en R, es decir, no podran existir dos trabajos con la ordenación "i anterior a j" en la secuencia S, si $i \in \Gamma(j)$.

Cada trabajo j lleva asignado un tiempo de procesamiento p_j conocido y

una fecha límite dj dada. Si no se ha completado el procesamiento de un trabajo antes de su fecha límite, se incurre en una tardanza la cual es medida por el número de unidades de tiempo en que se ha superado dicha fecha.

El instante en el cual el trabajo j se termina de procesar se denomina el tiempo de completación del trabajo j y se denota por Cj. Lógicamente este tiempo varía en relación con la posición que ocupa el trabajo j en la secuencia correspondiente de trabajos. Así, si dicho trabajo ocupase el lugar k, entonces $C_j = \sum_{i=1}^{k} p_{l(i)}$, siendo l(i) el trabajo que estaría en la posición i.

Se define la Tardanza de un trabajo j por la expresión

$$T_j = \max\{ 0, \ L_j = C_j - d_j \}$$

El problema de la Tardanza Total consiste en encontrar una ordenación o secuencia $[l(1),\ldots,l(n)]$ de trabajos que minimice la tardanza total

$$\sum_{j=1}^{n} T_{l(j)}.$$

Métodos exactos que permiten encontrar una solución óptima para el problema pueden verse en Baker (1), Baker y Martin (2), Lawler (4) y en Potts y Van Wassenhove (8). Estos últimos autores hacen un estudio exhaustivo de varios de estos métodos. El problema es que tales procedimientos requieren un gran esfuerzo computacional cuando el número de trabajos es de varias decenas. En tales circunstancias los métodos heurísticos pueden ofrecer soluciones aceptables, ya que la pérdida de bondad de la solución obtenida queda contrarrestada por la notable mejora del tiempo invertido en su búsqueda.

En anteriores trabajos, referencias (9) y (10), abordamos el estudio de la tardanza total sin considerar restricciones de precedencia; con el presente, vemos como incide sobre el problema considerado, la incorporación de dichas restricciones.

Describimos a continuación un método heurístico que guíe la búsqueda de una solución aproximada para el problema comentado.

ALGORITMO

Antes de describir paso a paso el algoritmo propuesto, comentemos brevemente las ideas generales del mismo.

La estructura de precedencias entre los trabajos, permite clasificar los mismos en k niveles de forma que, la ejecución de cualquier trabajo del nivel i-ésimo, requiere haber realizado previamente al menos un trabajo del nivel i-1.

Dentro de cada nivel se considera la correspondiente secuencia EDD (ordenación de menor a mayor de los diversos trabajos de acuerdo con las fechas límites) y posteriormente se agrupan las diferentes ordenaciones en un solo vector, poniendo en primer lugar las de menor nivel. Dicho vector será nuestra secuencia semilla.

Si en dicha secuencia hubiesen dos o más trabajos de un mismo nivel con fechas límites iguales, se ordenan estos trabajos de acuerdo con sus tiempos de procesamiento de menor a mayor.

El organigrama del algoritmo puede verse en la figura 1. Tras determinar la secuencia inicial, se calculan las tardanzas de los trabajos y se selecciona el trabajo candidato a cambiar de lugar (paso 3). Posteriormente se decide cuál debe ser el cambio más favorable, es decir, si el trabajo elegido debe adelantarse ó retrasarse, teniendo siempre presente que el cambio seleccionado debe ser compatible con la estructura de precedencia establecida inicialmente.

La posible ganancia en la reducción de la tardanza, si adelantáramos el trabajo, viene dada en los pasos 5 y 6; mientras que si se retrasara, la bondad de tal medida es cuantificada por los pasos 8 y 9 del algoritmo.

La decisión de efectuar el cambio (siempre al lugar que ofrece una mayor reducción) se aborda en el paso 10. Si desistimos de hacerlo, será debido a que no decrece la tardanza total y, en consecuencia, elegiremos un nuevo trabajo candidato a cambiar de lugar. El proceso finaliza cuando ningún trabajo proporcione un cambio que permita reducir la tardanza total.

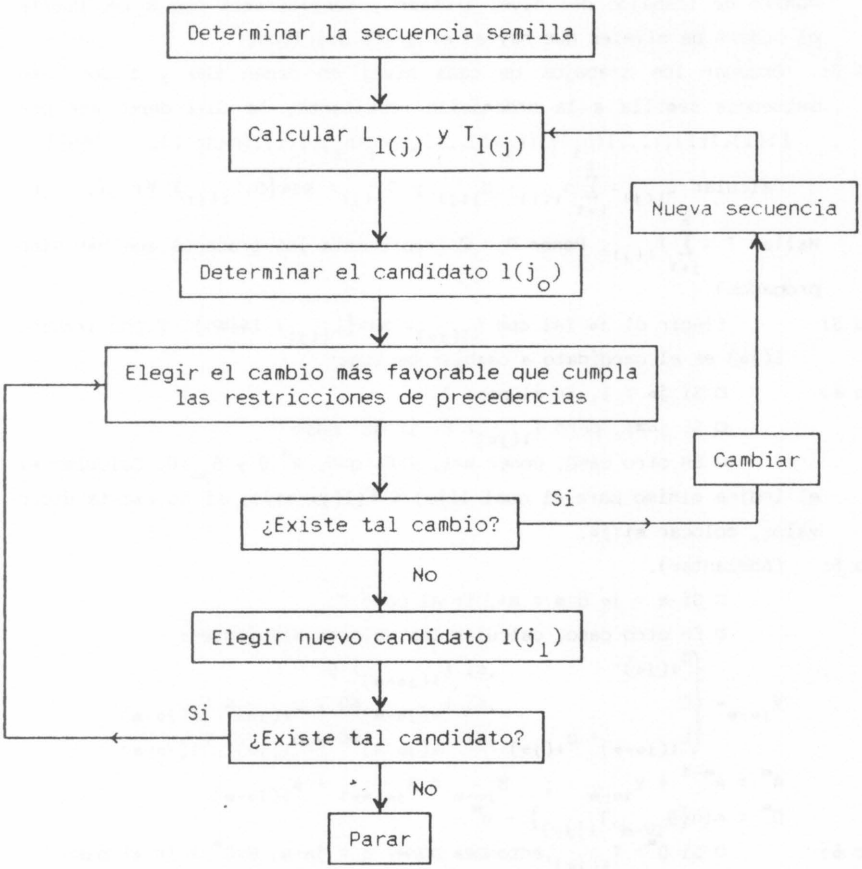


Figura 1: Organigrama del algoritmo heurístico.

Algoritmo

Paso 1: Establecer la ordenación de trabajos por niveles de acuerdo con el grafo acíclico que fija las relaciones de precedencia. Sea n_i el número de trabajos del nivel i-ésimo y consideremos que k representa el número de niveles que hay en el grafo acíclico.

Paso 2: Ordenar los trabajos de cada nivel en orden EDD y tomar como secuencia semilla a la ordenación resultante, la cual denotamos por

$$[1(1),1(2),\ldots,1(n_1);1(n_1+1),\ldots,1(n_1+n_2);\ldots;1(n-n_k+1),\ldots,1(n)]$$

Calcular $L_{1(j)}=\sum_{i=1}^{j} p_{1(i)} - d_{1(j)}$ y $T_{1(j)}=\max\{0,L_{1(j)}\}$ $\forall j = 1,\ldots,n$.

Hallar $T=\sum_{j=1}^{n} T_{1(j)}$. Poner $P=\emptyset$ (representa los trabajos que han sido probados).

Paso 3: Elegir el jo tal que $L_{1(jo)}=\max\{L_{1(j)} / 1\leq j\leq n\}$. Dicho trabajo 1(jo) es el candidato a cambiar de lugar.

Paso 4:

- Si jo = 1, ir al paso 7.
- Si jo≠1, pero $T_{1(jo)}=0$, ir al paso 7.
- En otro caso, poner m=1, G=0, q=0, $A^0=0$ y $B_{jo}=0$. Calcular m1 el índice mínimo para el cual $1(jo)\in\Gamma(1(jo-m1))$; si no existe dicho valor, colocar m1=jo.

Paso 5: (Adelantar).

- Si m = jo ó m = m1, ir al paso 7.
- En otro caso, calcular los valores siguientes:

$$V_{jo-m}=\begin{cases} p_{1(jo)} & \text{,si } L_{1(jo-m)}>0 \\ 0 & \text{,si } L_{1(jo-m)}\leq 0 \text{ y } p_{1(jo)}\leq L_{1(jo-m)} \\ L_{1(jo-m)}+p_{1(jo)} & \text{,si } L_{1(jo-m)}\leq 0 \text{ y } p_{1(jo)}> L_{1(jo-m)} \end{cases}$$

$$A^m = A^{m-1} + V_{jo-m} \quad ; \quad B_{jo-m} = B_{jo-m+1} + p_{1(jo-m)}$$

$$D^m = \min\{B_{jo-m}, T_{1(jo)}\} - A^m.$$

Paso 6:

- Si $D^m = T_{1(jo)}$, entonces poner q = jo-m, $G=D^m$ e ir al paso 7.
- Si $D^m \neq T_{1(jo)}$ y $D^m > G$, entonces poner q = jo-m y $G = D^m$. Hacer m = m+1 y volver al paso 5.
- Si $D^m \neq T_{1(jo)}$ y $D^m \leq G$, entonces hacer m=m+1 y volver al paso 5.

Paso 7:

- Si jo = n, ir al paso 10.
- Si jo ≠ n, poner m = 1, Fo = 0 y $Z^0 = 0$. Sea m2 el índice mínimo para el cual $1(jo+m2)\in\Gamma(1(jo))$; si no existe tal valor, poner m2 = n-jo+1.

Paso 8: (Retrasar).

□ Si $m = n - j_o + 1$ ó $m = m2$, ir al paso 10.

□ En otro caso, calcular los valores siguientes:

$$F_m = F_{m-1} + p_{l(j_o+m)}$$

$$Y^m = \begin{cases} F_m & , \text{ si } L_{l(j_o)} > 0 \\ \max\{0,\ L_{l(j_o)} + F_m\}, & \text{si } L_{l(j_o)} \leq 0 \end{cases}$$

$$X_{j_o+m} = \begin{cases} 0 & , \text{ si } L_{l(j_o+m)} \leq 0 \\ \min\{L_{l(j_o+m)},\ p_{l(j_o)}\}, & \text{si } L_{l(j_o+m)} > 0 \end{cases}$$

$$Z^m = Z^{m-1} + X_{j_o+m} \quad ; \quad D^m = Z^m - Y^m.$$

Paso 9: □ Si $D^m > G$, poner $q = j_o + m$ y $G = D^m$. Hacer $m = m + 1$ y volver al paso 8.

□ Si $D^m \leq G$, hacer $m = m + 1$ y volver al paso 8.

Paso 10: (Cambiar)

□ Si $q = 0$, no hacer ningún cambio. Colocar $P = P \cup \{j_o\}$ e ir al paso 11 (búsqueda de un nuevo candidato).

□ Si $q>0$, cambiar el trabajo $l(j_o)$ al lugar q y rodar los trabajos que sean necesarios, esto es: Sea $MIN = \min(q,j_o)$ y $MAX=\max(q,j_o)$.

- Si $MIN = q$, entonces $s(q) = l(j_o)$, $s(q+1) = l(q),\ldots, s(q+k) = l(q+k-1),\ldots, s(j_o) = l(j_o-1)$.

- Si $MIN = j_o$, entonces $s(j_o) = l(j_o+1),\ldots, s(j_o+k) = l(j_o+k+1), \ldots, s(q-1) = l(q), s(q) = l(j_o)$.

Antes de salir de este paso, renombrar los subíndices haciendo $l(i) = s(i)$, $\forall i \in [MIN,MAX]$ y $l(i) = l(i)$, $\forall i \notin [MIN,MAX]$.

Volver a calcular $L_{l(j)}$ y $T_{l(j)}$ para $j \in [MIN,MAX]$ según las fórmulas del paso 2. Poner $T = T-G$, $P = \emptyset$ e ir al paso 3.

Paso 11: □ Si $Card(P) = n$, parar. La secuencia $[l(1),l(2),\ldots,l(n)]$ nos dá la solución, siendo T el valor de la tardanza total para dicha secuencia.

□ Si $Card(P) \neq n$, entonces calcular el j_1 tal que $L_{l(j_1)} = \max\{L_{l(k)}/k \notin P\}$. Hacer $j_o = j_1$ y volver al paso 4.

COMPLEJIDAD DEL ALGORITMO

El algoritmo tiene una complejidad $O\left(n^2 \sum_{i=1}^{n} pi\right)$, desglosada de la forma siguiente:

El paso 1 requiere $O(n^2)$ operaciones, mientras que el paso 2 consume a lo sumo $O(n \log n)$.

Los pasos del 3 al 11 requieren $O(n)$ por cada iteración. El número de iteraciones para estos pasos es a lo sumo de $\sum_{j} T_j(\text{Semilla}) - T_{max}(\text{EDD})$, siendo $T_j(\text{Semilla})$ la tardanza del trabajo j correspondiente a la secuencia inicial y $T_{max}(\text{EDD})$ la tardanza máxima para la secuencia EDD. La cuestión será determinar qué valor puede tomar dicha expresión. La variabilidad de la misma es grande pero siempre estará acotada por $n.\sum_{i=1}^{n} pi$.

Por tanto, la recursividad de los pasos tercero al décimo primero nos lleva a una complejidad $O\left(n^2.\sum_{i=1}^{n} pi\right)$ que supera ampliamente la de los pasos 1 y 2; en consecuencia, dicha expresión será la complejidad del algoritmo.

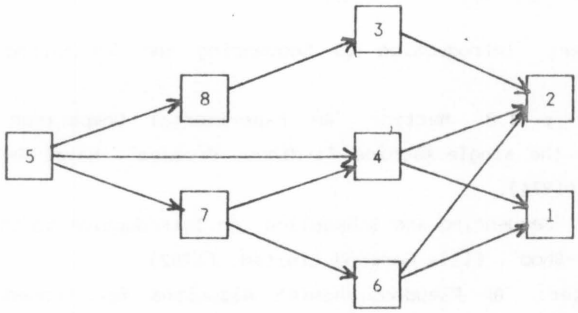
Comentemos por último que el algoritmo se ha probado en varios casos particulares. Exponemos a continuación uno de los casos estudiados.

EJEMPLO

Consideramos el problema de secuenciar ocho trabajos, cuyos tiempos de procesamiento y fechas límites se recogen en la siguiente tabla

j	1	2	3	4	5	6	7	8
p_j	121	147	102	79	130	83	96	88
d_j	260	269	400	266	337	336	683	719

Las relaciones de precedencia vienen dadas por el grafo acíclico



Siguiendo el algoritmo propuesto, la secuencia semilla será

5	7	8	4	6	3	1	2

a la que corresponden los Lj

(-207, -457, -405, 127, 140, 178, 439, 577)

y una tardanza total de 1461.

Inicialmente el candidato a cambiar es el trabajo 2 pero el mismo no ofrece cambio favorable; lo mismo ocurre para los trabajos 1 y 3. La elección del trabajo 6 nos proporciona una reducción de la tardanza en 57 unidades, si instalamos dicho trabajo en la posición 3 de la secuencia, obteniendo así la nueva secuencia

5	7	6	8	4	3	1	2

De la misma forma se descartan los candiatos 2 y 1, y se elige cono nuevo candidato el trabajo número 4. Dicho trabajo se coloca en la tercera posición por ser el lugar que otorga mayor descenso en la tardanza total, quedándo ésta en 1285. La nueva secuencia será

5	7	4	6	8	3	1	2

Repitiendo el proceso, el algoritmo finaliza con la secuencia

5	7	4	6	1	8	3	2

con tardanza 1216.

Nótese que a lo largo del desarrollo del ejemplo, las diversas soluciones parciales que se van obteniendo verifican las restricciones de precedencia establecidas en el grafo de partida.

BIBLIOGRAFIA

(1) K.R. Baker: "Introduction to Sequencing and Scheduling". J. Wiley. (1974).

(2) K.R.Baker y J.B. Martin: "An Experimental Comparison of Solution Algorithms the single-machine Tardiness Problem". Naval Res. Log. Quar. 21, nº 1. (1974).

(3) S. French: "Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop". Ellis Horwood limited. (1982).

(4) E.L. Lawler: "An Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness". Ann. Disc. Math. 1, 331 - 342. (1977).

(5) E.L. Lawler; J.K. Lenstra y A.H.G. Rinnooy Kan: "Recent Developments in deterministic sequencing and Scheduling: a survey", en Deterministic and Stochastic Scheduling editado por Dempster, et al. Reidel Publishing Company. (1982).

(6) J.K. Lenstra, A.H.G. Rinnooy Kan y P. Brucker: "Complexity of Machine Scheduling Problems". Ann. Disc. Math. 1, 343 - 362. (1977).

(7) J.K. Lenstra, A.H.G. Rinnooy Kan: "Complexity of scheduling under Precedence Constraints". Oper. Res. 26, n. 1, 22 - 35. (1978).

(8) C.N. Potts y L.N. Van Wassenhove: "Dynamic Programming and Decomposition Approaches for the Single Machine Total Tardiness Problem". European Journal of Oper. Research, 32, 405 - 414. (1987).

(9) J. Sicilia y D. Alcaide: "Algoritmo aproximado para la minimización de la tardanza total sobre una máquina". XIV Jornadas Hispano-Lusas de Matemáticas. Tenerife. (1989).

(10) J. Sicilia y D. Alcaide: "Minimización de la tardanza total considerando diferente disponibilidad de los trabajos". Actas de la XVIII Reunión Nacional de Estadística e Investigación Operativa. Santiago de Compostela. (1989).