Jueves, 07 de octubre de 1999

Modelos de maquinas únicas (Determinísticos)

Scheduling Theory Algorithms, and System
Michael Pinedo.
RESUMEN PRESENTADO POR: MARCO ANTONIO CRUZ CHAVEZ.

Los modelos de maquinas únicas son importantes ya que aportan bases para las heurísticas de los modelos más complejos (maquinas en paralelo o en serie). Los problemas de Scheduling de ambientes mas complicados son descompuestos frecuentemente en subproblemas que tratan con maquinas simples.

Tiempo de terminación ponderado total.

1 | | Σ wj Cj

En este modelo para encontrar una asignación optima se utiliza una de las reglas mas conocidas en la teoría de Scheduling, la regla de WSTP (Primer tiempo de procesamiento mas corto ponderado). De acuerdo a esta regla los trabajos se ordenan en orden decreciente de wj/pj.

Ejemplo:

En una asignación optima S se asume que un par de tareas adyacentes (j, k) tal que la tarea j precede a la tarea k y se cumple que

$$\frac{wj}{pj} > \frac{wk}{pk}$$

Si en la asignación se intercambian estas tareas se tiene otra asignación S' (figura 1). El **TTPT** (Tiempo de terminación ponderado total) de las tareas antes de j y k no se afecta por el intercambio.

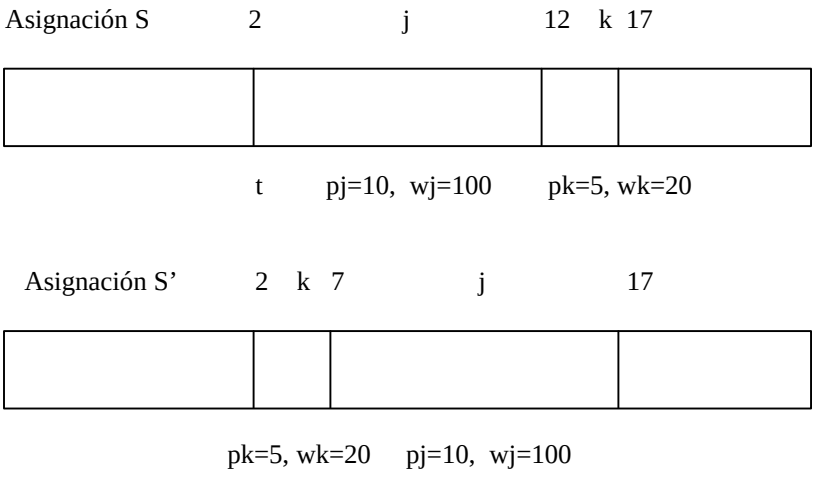


FIGURA 1.

En S Ck = t + pj + pk y Cj = t + pj

En S’ Ck = t + pk y Cj = t + pk + pj

Para S el **TTPT** de las tareas j y k es Cj wj + Ck wk

(t + pj) wj + (t + pj + pk) wk

(2 + 10)100 + (2 + 10 + 5)20 = 1200 + 340 = 1540

mientras que para S’ es

(t + pk) wk + (t + pk + pj) wj

(2 + 5)20 + (2 + 5+ 10)100 = 140 + 1700 = 1840

1 | prec | Σ wj Cj

Aplicando las restricciones de precedencia más simples como lo son en forma de cadenas. Considerar dos cadenas donde tienen las siguientes restricciones de precedencia:

1 -> 2 -> … -> k cadena 1

y

k +1 -> k + 2 -> … -> n. Cadena 2

En este scheduler se asume que si se comienza a procesar los trabajos en una de las cadenas, esta tiene que ser completada antes de que se permita comenzar con la otra. ¿Cuál cadena se procesara primero si se desea minimizar el **TTPT**?.

Si

$$\frac{\sum_{j=1}^{k} wj}{\sum_{j=1}^{k} pj} > \frac{\sum_{j=k+1}^{n} wj}{\sum_{j=k+1}^{n} pj}$$

La cadena de trabajos 1,…k precede a la cadena de trabajos k+1,….n.

Ejemplo: de la tabla 1 que tiene los pesos y tiempos de procesamiento para ciertos trabajos tenemos que

$$w_1 p_1 + \ldots + w_k \sum_{j=1}^{k} p_j + w_{k+1} \sum_{j=1}^{k+1} p_j + \ldots + w_n \sum_{j=1}^{n} p_j$$

Σ wj Cj = 6(3) + 18(9) +12(15) + 8(20) + 8(24) + 17(32) + 18(42) = 2012

Mientras que si la cadena de trabajos k+1,….n precede a la cadena de trabajos 1,…k

$$w_{k+1}p_{k+1}+\ldots+w_n\sum_{j=k+1}^{n}p_j+w_1\left(\sum_{j=k+1}^{1}p_j+p_1\right)+\ldots+w_k\sum_{j=1}^{n}p_j$$

Σ wj Cj = 8(4) + 47(12) +18(22) + 6(25) + 18(31) + 12(37) + 8(42) = 2120

Paso de una cadena a otra sin terminar esta.

Si asumimos que se tiene la libertad para procesar cualquier numero de trabajos en una cadena sin tener que terminar todos los trabajos antes de cambiarse a la siguiente cadena. Después se retorna a la primer cadena. En el caso de minimizar el **TTPT**.

Para una cadena de 1 -> 2 -> … -> k

Se tiene a l* que satisface

$$r=\frac{\sum_{j=1}^{l^*}w_j}{\sum_{j=1}^{l^*}p_j}=\max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l}w_j}{\sum_{j=1}^{l}p_j}\right)$$

Donde l* es el trabajo que determina al factor ρ de la cadena, así, se tiene una secuencia optima que procesa a los trabajos 1,…,l* uno a continuación del otro sin interrupciones debido a los trabajos de otras cadenas. Esto también implica que los radios del peso total dividido por el tiempo de procesamiento total de los trabajos en la cadena 1,…,l* deben de estar incrementándose.

Algoritmo: Cuando la maquina esta libre y se selecciona entre las cadenas faltantes una con el valor mas alto de ρ, se procesa esta cadena sin interrupciones, incluyendo el trabajo que determina al factor ρ.

Ejemplo: Considerar las siguientes dos cadenas :

1 -> 2 -> 3 -> 4

y

5 -> 6 -> 7.

Con sus respectivos pesos y tiempos de procesamiento mostrados en la tabla 1. Encontrar la asignación optima **TTPT**.

Trabajos	1	2	3	4	5	6	7
wj	6	18	12	8	8	17	18
pj	3	6	6	5	4	8	10

Tabla 1. Trabajos a ser asignados para un óptimo **TTPT**

Trabajos	1	2	3	4	5	6	7
Σ wj /pj	2	2.6	2.4	2.2	2	2.08	1.9
Trabajos	1	2	3	4	5	6	7
Σ wj /pj	----	----	2	1.8	2	2.08	1.9
Trabajos	1	2	3	4	5	6	7
Σ wj /pj	---	---	2	1.8	---	---	1.8
Trabajos	1	2	3	4	5	6	7
Σ wj /pj	---	---	---	1.6	---	----	1.8

La asignación optima para el **TTPT** es: 1 2 5 6 3 7 4.

FUNCIONES OBJETIVOS RELACIONADAS A LA FECHA COMPROMETIDA (DUE DATE)

La **durabilidad** del trabajo j es: $L_j=C_j-d_j$

La **demora** del trabajo j es: $T_j = \max(C_j - d_j, 0) = \max(L_j,0)$

La **unidad de penalización** del trabajo j es: $U_j = \begin{cases} 1 & si\ \ C_j > d_j \\ 0 & de\ otro \bmod o \end{cases}$

$1 \mid prec \mid h_{max}$

Este modelo considerado es muy general porque las funciones h_j abarcan lo relacionado a la fecha comprometida ya que estas funciones pueden tomar las formas descritas anteriormente (**durabilidad, demora, unidad de penalización**).

El siguiente algoritmo produce una asignación optima para el modelo $1 \mid prec \mid h_{max.}$

ALGORITMO

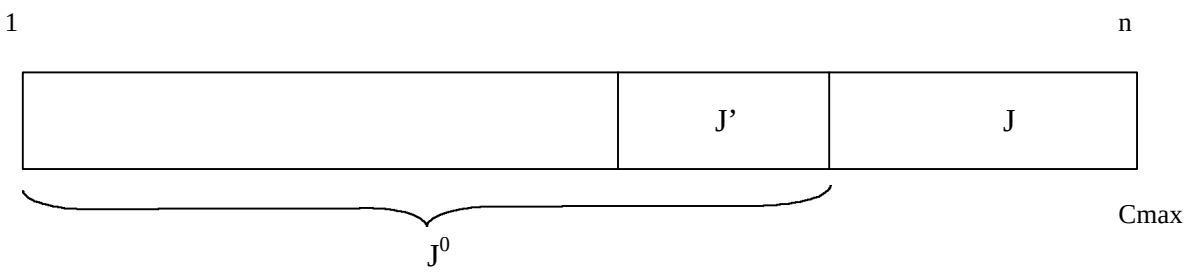
La asignación en una maquina simple empieza en forma de retroceso, conociendo el **makespan C_{max}** que es independiente de la asignación.

J = Conjunto de trabajos ya asignados los cuales se procesan en el intervalo de tiempo

$$\left| C_{max} - \sum_{j \in J} p_j, \quad C_{max} \right|$$

J^0= Complemento del conjunto J, y es el conjunto de trabajos a ser asignados.
J' = Subconjunto de J^0, conjunto de trabajos que pueden ser asignado inmediatamente antes del conjunto J.

Una asignación ya iniciada puede verse de esta manera.



Paso 1:

J = 0, $J^0 = \{1,...,n\}$

J' = conjunto de trabajos sin sucesores.

Paso 2:

Se escoge a un trabajo j^* tal que

$$h_{j^*}\left(\sum_{j\in J^0} p_j\right) = \min_{j\in J'}\left(h_j\left(\sum_{j\in J^0} p_j\right)\right)$$

sumar j* a J. Borrar j* de J^0. Modificar J' para representar al nuevo conjunto de trabajos asignables.

Paso 3: Si $J^0 = 0$, TERMINAR, de otra manera ir al paso 2.

Esto se puede comprender mejor en la figura 2.

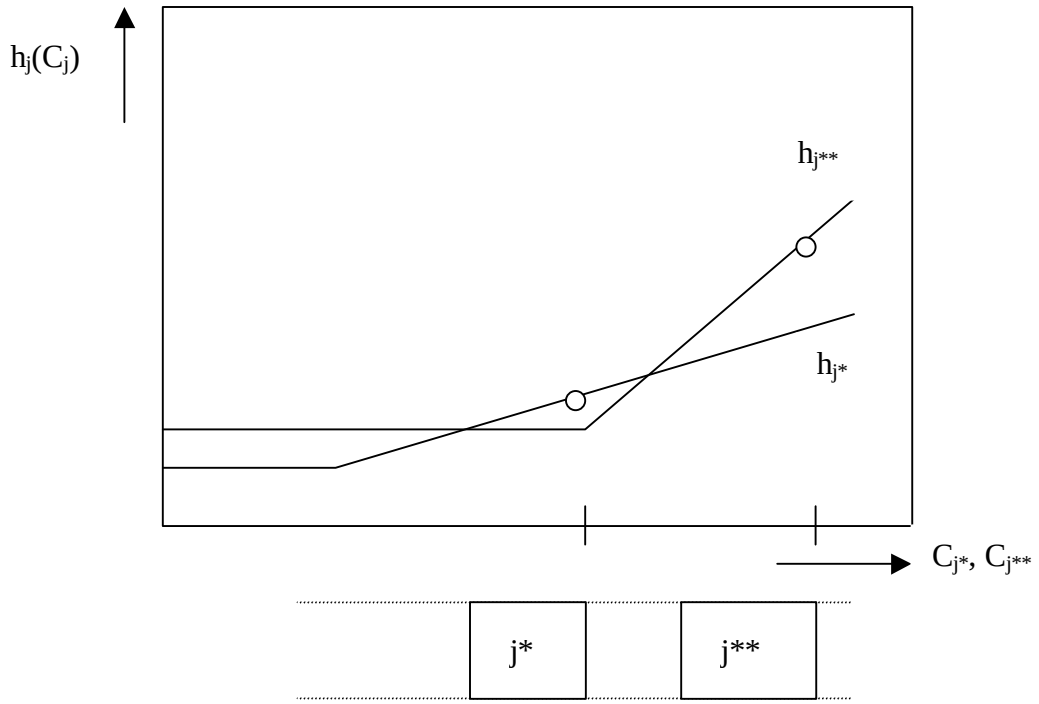


FIGURA 2(A)

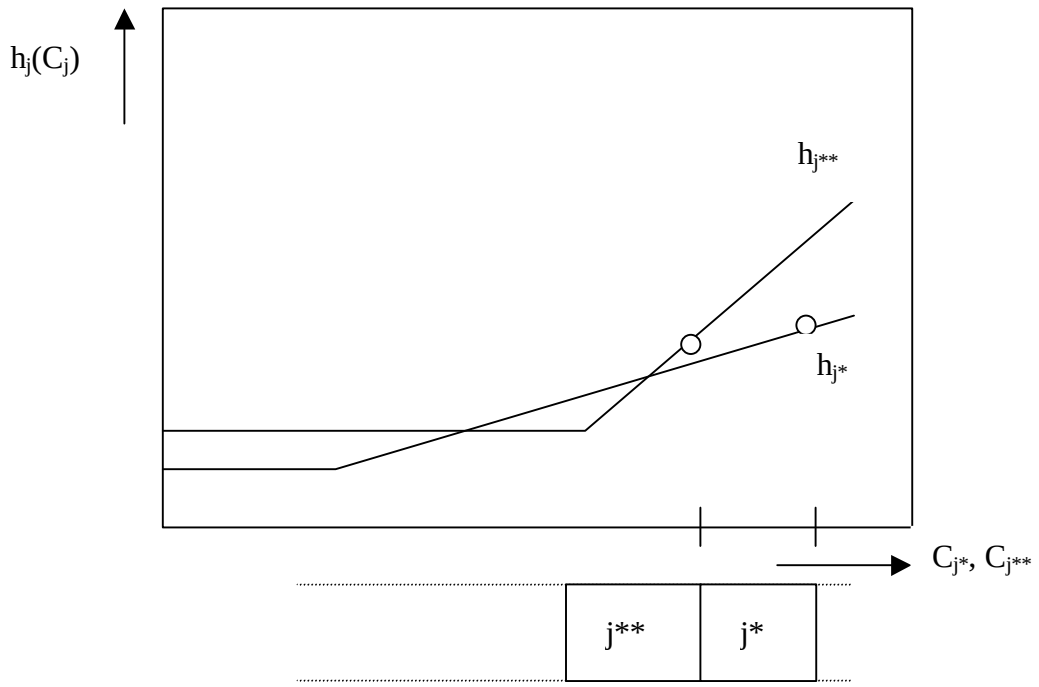


FIGURA 2(B)

Para ejemplificar este algoritmo veamos en una asignación de trabajos el siguiente modelo:

$$\mathbf{1} \mid \ \mid \ \sum h_j(C_j)$$

C_{max}	TRABAJOS	1	2	3	4	5	6
	pj	2	4	1	7	6	3
	hj(Cj)	$1+C_1$	$1.2C_2$	C_3	$4+C_4$	$2.2C_5$	$C^2{}_6$
23	h(23)	24	27.6	**23**	27	50.6	529
22	h(22)	**23**	26.4		26	48.4	484
20	h(20)		24		**24**	44	400
13	h(13)		**15.6**			28.6	169
09	h(9)					**19.8**	81

El Makespan es Cmax = 23 y la asignación optima basada en el algoritmo anterior es 652413 cuyo costo de la función objetivo es:

$$\sum h_j(C_j) = (3)^2 + 2.2(3+6) + 1.2(3+6+4) + [4+(3+6+4+7)] + [1+(3+6+4+7+2)] + (3+6+4+7+2+1) = 106.3$$

que seria el costo mínimo encontrado al asignar las tareas en el orden antes mencionado.

Si asignáramos los trabajos en el orden 123456 entonces.

$$\sum h_j(C_j) = 608.2$$

ALGORITMO PARA EL MAKESPAN CON SUCESIONES SUJETAS A DISPOSICIONES DE TIEMPO.

El Makespan que incluye sucesiones sujetas a disposiciones de tiempo depende de cómo se realiza la asignación. El modelo que representa a este problema es:

$$\mathbf{1 \mid s_{jk} \mid C_{max}}$$

El trabajo j se le asocian dos parámetros a_j y b_j y $s_{jk} = \mid a_k - b_j \mid$. Esta estructura indica que después de la terminación de j la maquina es puesta en el estado b_j y para comenzar el trabajo k la maquina tiene que ser conducida dentro del estado a_k.

Como el modelo es NP-hard se tiene que hacer una suposición para que se pueda utilizar un algoritmo de tiempo polinomial: se asume que al tiempo cero el estado es b_0 y después de terminarse de hacer el ultimo trabajo la maquina se tiene que dejar en un estado a_0.

$s_{jk} = \mid a_k - b_j \mid$ es la disposición de tiempo total para llevar la maquina del estado b_j al a_k (ver figura 1).

A continuación se muestra el algoritmo a través de un ejemplo del problema del agente viajero: Encontrar la mejor ruta con el costo mínimo (menor tiempo de recorrido esto es el Makespan)

Tabla 1.

CIUDADES	0	1	2	3	4	5	6
b_i	1	15	26	40	3	19	31
a_i	7	16	22	18	4	45	34

ALGORITMO	***EJEMPLO***
PASO1: Reordenar las ciudades de tal forma que $b_j <= b_{j+1}$. Obtenga las permutaciones que minimicen los costos.	Tabla 2.

CIUDAD	0	1	2	3	4	5	6
b_j	1	3	15	19	26	31	40
a_j	7	4	16	45	22	34	18
$a_{\phi^*(j)}$	4	7	16	18	22	34	45
$\phi^*(j)$	1	0	2	6	4	5	3

	Ver Figura 1. Se toma la ciudad con el menor b_j y se une a la ciudad con el menor a_j , y así sucesivamente hasta terminar. Se busca una permutación donde las flechas no crucen, pues esto genera un costo mayor.
PASO 2: Formar un grafo no dirigido con n+1 nodos y $A_{j,\phi^*(j)}$ arcos conectando los nodos j y $\phi(j)$. Si el grafo actual tiene un solo componente (un solo recorrido) TERMINAR de lo contrario continuar en el paso 3.	
PASO 3: Calculo de los costos de intercambio $c\phi^*l(j,j+1)$ para j=0,…,n-1. $c\phi^*l(j,j+1) = 2\max(\min(b_{j+1}, a_{\phi^*(j+1)}) - \max(b_j, a_{\phi^*(j)}))$	Cambios en costo debido al intercambio. $c\phi^*l(0,1) = 0$ $c\phi^*l(0,1) = 2(15-7)=16$ $c\phi^*l(0,1) = 2(18-16)=4$ $c\phi^*l(0,1) = 2(22-19)=6$ $c\phi^*l(0,1) = 2(31-26)=10$ $c\phi^*l(0,1) = 2(40-34)=12$ Ver Tabla 2, Figura 1.

PASO 4:

Seleccionar el costo mas pequeño $c\phi^*l(j,j+1)$ tal que j este en un componente y j+1 en otro.
Insertar el arco $A_{j,j+1}$ no dirigido dentro del grafo y repetir este paso hasta que todos los componentes en el grafo no dirigido estén conectados.

Los arcos no dirigidos $A_{1,2}$, $A_{2,3}$, $A_{3,4}$, $A_{4,5}$ se insertan dentro del grafo.

PASO 5:

Se dividen los arcos seleccionados en el paso 4 en dos grupos.

Grupo 1: los arcos $A_{j,j+1}$ donde $a_{\phi^*(j)} >= b_j$.
Grupo 2: los restantes.

Los cuatro arcos se particionan en dos grupos. Para hacer esto, cada b_j se tiene que comparar con su correspondiente $a_{\phi^*(j)}$.

ARCOS	b_j	$a_{\phi^*(j)}$	GRUPO
$A_{1,2}$	$b_1=3$	$a_{\phi^*(1)}=a_0=7$	1
$A_{2,3}$	$b_2=15$	$a_{\phi^*(2)}=a_2=16$	1
$A_{3,4}$	$b_3=19$	$a_{\phi^*(2)}=a_3=18$	2
$A_{4,5}$	$b_4=26$	$a_{\phi^*(4)}=a_4=22$	2

PASO 6:

Encontrar el índice mas grande j_1 tal que el arco $A_{j,j+1}$ este en el grupo 1.
Encontrar el segundo mas grande j_2 y continuar así.
Encontrar el índice mas pequeño j_1 tal que el arco $A_{k,k+1}$ este en el grupo 1.
Encontrar el segundo mas pequeño k_2 y continuar así.

$J_1 = 2$, $j_2 = 1$, $k_1 =3$ y $k_2 = 4$.

PASO 7: El recorrido optimo ϕ^{**} se construye aplicando la siguiente secuencia de intercambios sobre la permutación ϕ^{*}: $\phi^{**}=\phi^{*}l(j_1,j_1+1)l(j_2,j_2+1)\ldots l(j_l,j_1+1)$ $l(k_1,k_1+1)\ l(k_2,k_2+1)\ldots\ l(k_m,k_m+1)$.	El recorrido optimo se obtiene después de los siguientes intercambios. $\phi^{**}= \phi^{*}l(2,3)\ l(1,2)\ l(3,4)\ l(4,5)$. Ver figura 2 y tabla 2. De esta manera el recorrido optimo es 0 → 1 → 6 → 3 → 4 → 5 → 2 → 0 El costo de este recorrido que se calcula como se muestra en la figura 1. 3 + 15 + 5 + 3 + 8 + 15 + 8 = 57

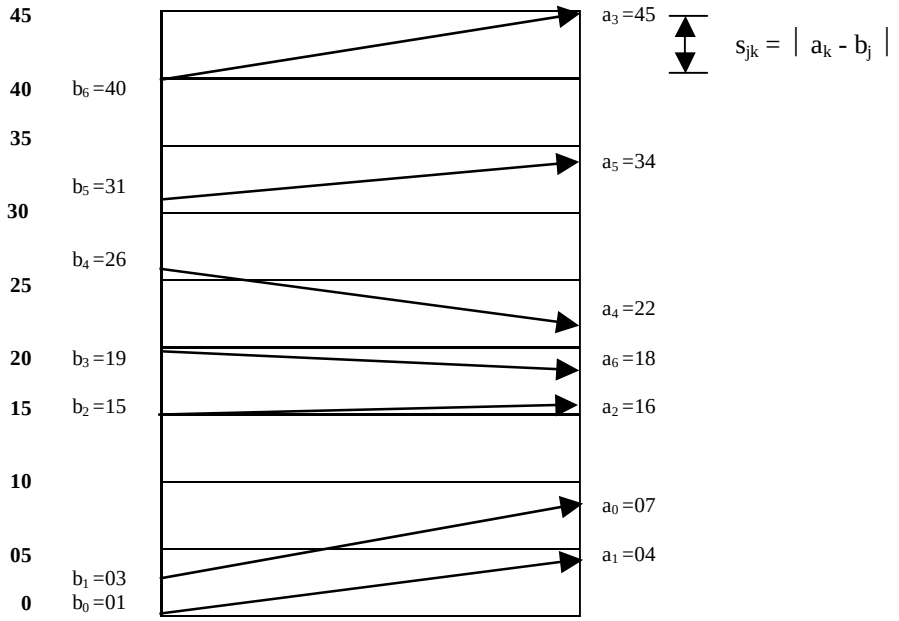


FIGURA 1. Primera permutación.

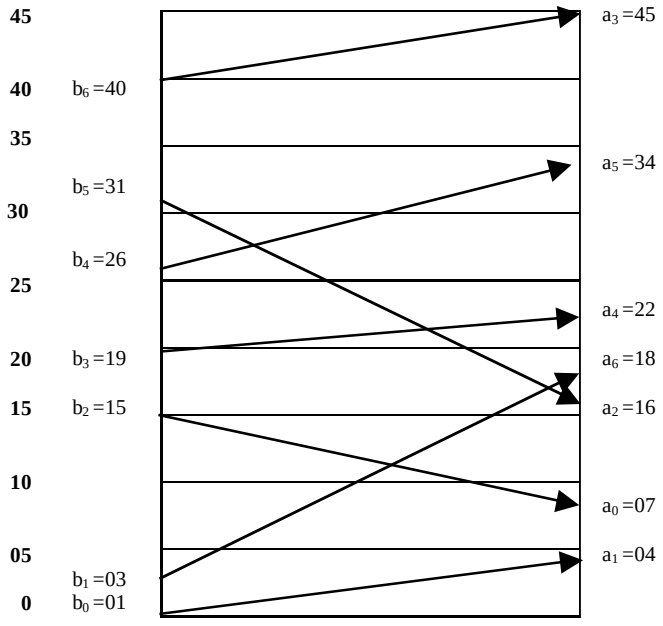


FIGURA 2. Asignación optima.