Flow Shop Sequencing

Tarea 1 - Parte 2 Optimización en Ingeniería

Gustavo Aguilar[1], Juan Barrerra[2], Juan Trujillo[3]

Resumen— **El presente informe tiene como objetivo entregar una descripción general del problema Flow Shop Scheduling. Su definición, representación, funcion objetivo, restricciones y ejemplos desarrollados. De los resultados se extraen conjeturas relativas al comportamiento de las diferentes soluciones frente a los sets de datos utilizados. Finalmente, Se propone un algoritmo de búsqueda de soluciones vecinas para el problema.**

I. Descripción del Problema

El Flow Shop Scheduling es un problema de gestion de la producción. Se describe de la siguiente manera: existe un conjunto de n trabajos y un conjunto de m máquinas. Cada trabajo considera un conjunto dee m operaciones que deben ser realizadas en diferentes máquinas. Todos los trabajos tienen el mismo orden de procesamiento a través de las máquinas. No hay restricciones de precedencia entre las operaciones de diferentes trabajos. Las operaciones no pueden ser interrumpidas y cada máquina puede procesar solo una operación a la vez. El objetivo es encontrar la secuencias de trabajos en las máquinas que minimice el tiempo máximo de completitud de todas las operaciones.

A. Definiciones

De acuerdo a la descripción del problema, tenemos las siguientes definiciones en la tabla I.

Definición	Descripción
J	Conjunto de trabajos 1,...,n.
M	Conjunto de máquinas 1,...,m.
O_p	Conjunto de operaciones 1,...,m.
j_i	El trabajo $i-ésimo$ en la permutación del conjunto de trabajos.
t_{ik}	Tiempo de procesamiento del trabajo $j_i \in J$ en la máquina k.
v_{ik}	Tiempo de espera de la máquina k antes de que comience el trabajo j_i $(\forall$ i $\in J)(\forall\ k \in M)$.
w_{ik}	Tiempo de espera del trabajo j_i después de finalizar la operación de la máquina k, mientras espera por la máquina $k+1$ a que sea liberada.

Tabla I: Definiciones del problema

[1]Dpto Ingeniería Informática, Universidad Santiago de Chile, e-mail: gustavo.aguilar@usach.cl

[2]Dpto Ingeniería Informática, Universidad Santiago de Chile, e-mail: juan.barrerab@usach.cl

[3]Dpto Ingeniería Informática, Universidad Santiago de Chile, e-mail: juan.trujillo@usach.cl

B. Variables de decisión

Dado que cada trabajo J_i debe tener asignado una máquina mientras realiza una operación, entonces se define las permutaciones de un modo binario, tal que:

$$x_{ij} = \begin{cases} 1 & \text{si } j \text{ es asignado a la } i-ésima \\ & \text{permutación, esto es cuando } J_i = j \\ 0 & \text{cualquier otro caso} \end{cases} \quad (1)$$

En otras palabras, (1) indica que si la variable tiene valor 1, entonces una máquina está ejecutando la operación de algún trabajo j.

II. Representación de la Solución

A continuación se presenta el modo de generar una solución, sobre la cual se trabajará en las posteriores secciones.
Para comprender de manera preliminar, primero considerar un set de 5 trabajos, por lo que el orden de la solución se representa de forma vectorial de largo 5, tal que distribuya los trabajos de forma secuencial:

j_1
j_2
j_3
j_4
j_5

Esto es que el trabajo j_1 es la primera operación de todas las ejecuciones, finalizando con el trabajo j_5.
Si se quiere realizar una comparación entre una solución u otra, bastaría por ejemplo, mediante *swaping*, intercambiar el orden de los trabajos, tal que la nueva distribución quede como:

j_5
j_2
j_3
j_4
j_1

Ahora bien, para considerar la solución completa debemos introducir al problema los tiempos de las operaciones dentro de las máquinas, para lo que también se introducen dos conceptos relevantes, tiempo de entrada (in) y tiempo de salida (out) de cada una de estas. Donde in corresponde al tiempo en la cual la máquina $k \in M$ comienza a operar, y por tanto el out será el tiempo en completar dicha operación.

De esta forma una solución de $5x2$ (Trabajos x Máquinas), se representa de forma matricial (S), de tamaño $5x4$ como en la tabla II, cuyo tiempo de interés para este orden es el elemento $S_{52} = t^5_{22}$.

...→	Máquina k		...→
	in	out	
	t_{ik-1} (o t_{i-1k})	t_{ik-1} (o t_{i-1k}) $+t_{ik}$	

$S =$

	Máquinas			
	M_1		M_2	
J	in	out	in	out
j_1	t^1_{11}	t^1_{12}	t^1_{21}	t^1_{22}
j_2	t^2_{11}	t^2_{12}	t^2_{21}	t^2_{22}
j_3	t^3_{11}	t^3_{12}	t^3_{21}	t^3_{22}
j_4	t^4_{11}	t^4_{12}	t^4_{21}	t^4_{22}
j_5	t^5_{11}	t^5_{12}	t^5_{21}	t^5_{22}

Tabla II: Solución matricial, con tiempos de entrada y salida de cada máquina para cada trabajo de alguna secuencia en particular.

Notar que los tiempos t^k_{i1} y t^k_{i2} se comprende que es de forma continua, cuyos valores están asociados a las restricciones que más adelante se detallan, por lo que no puede ocurrir, por ejemplo que $t^k_{i1} > t^k_{i2}$ o que $t^{k-1}_{i1} > t^k_{i1}$, es decir esta representación, debe satisfacer las restricciones de tal forma que los tiempos sean consistentes.

III. Función Objetivo

Teniendo que m como la última máquina del conjunto $M = \{1, 2, ..., m\}$, nos interesa optimizar el tiempo de ejecución de todos los trabajos de esta, en conjunto con el tiempo de espera, es decir:

$$C_{max} = \sum_{i}^{n}(v_{im} + \sum_{j=1}^{n} p_{jm}x_{ij}) \tag{2}$$

IV. Restricciones

$$\forall i \in J : \sum_{j=1}^{n} x_{ij} = 1 \tag{3}$$

$$\forall j \in J : \sum_{i=1}^{n} x_{ij} = 1 \tag{4}$$

Las restricciones (3) y (4), limitan a que las máquinas puedan ejecutar sólo un trabajo a la vez, y que un trabajo sólo pueda ser ejecutado por una máquina a la vez.

$$\forall k \in M - \{m\} : w_{1k} = 0 \tag{5}$$

De la restricción (5), indica que para el trabajo J_1, todos los tiempos de espera a que se desocupe alguna máquina $k+1$, luego de terminar una máquina k, son 0. Esto es evidente ya que no hay ningún trabajo que le antecede y que haya ocupado alguna máquina $k+1$.

$$\forall k \in M - \{1\} : v_{1k} = \sum_{r=1}^{k-1}\sum_{i=1}^{n} p_{ir}x_{1i} \tag{6}$$

Cuando se observa una máquina k del primer trabajo (i.e $i = 1$), su tiempo de espera (v_{1k}) va estar dado por los tiempos de procesamiento de las máquinas que le anteceden, exceptuando la máquina 1, que en cuyo caso es 0 por definición, por lo que la restricción (6) garantiza que la máquina se ejecute en el orden correcto para el trabajo J_1.

$(\forall i \in J - \{n\})(\forall k \in M - \{m\}) :$

$$v_{i+1,k} + \sum_{j=1}^{n} p_{jk}x_{i+1,j} + w_{i+1,k} = w_{ik} + \sum_{j=1}^{n} p_{j,k+1}x_{ij} + v_{i+1,k+1} \tag{7}$$

La restricción (7) nos proporciona la consistencia y viabilidad entre los tiempos de inactividad de la máquina y el trabajo [1].

V. Ejemplos

A. Ejemplo set 3x2

Sea un conjunto de 3 trabajos y un conjunto de 2 máquinas, tal que sus tiempos de procesamiento se definen como en la tabla III

	m_1	m_2
j_1	3	2
j_2	1	4
j_3	5	6

Tabla III: Tabla de tiempos de procesamiento para un set de 3x2.

A.1 Representación

Luego la representación de una solución con orden $j_1 \to j_2 \to j_3$, queda como en la tabla IV. Notar que los valores de los in y out van intrínsecamente las restricciones acorde a lo explicado anteriormente.

$S_1 =$

	Máquinas			
	M_1		M_2	
J	in	out	in	out
j_1	0	3	3	5
j_2	3	4	5	9
j_3	4	9	9	15

Tabla IV: Solución matricial, con tiempos de entrada y salida de cada máquina para cada trabajo de alguna secuencia en particular.

A.2 Cálculo función objetivo

De la ecuación (2) reemplazamos $m = 2$, como la última máquina de nuestro set.

$$C_{max} = \sum_{i=1}^{3}(v_{i2} + \sum_{j=1}^{3} p_{j2}x_{ij}) \tag{8}$$

$$C_{max} = v_{12} + p_{12}x_{11} + p_{22}x_{12} + p_{32}x_{13} + v_{22} + p_{12}x_{21} + p_{22}x_{22} + p_{32}x_{23} + v_{32} + p_{12}x_{31} + p_{22}x_{32} + p_{32}x_{33} \tag{9}$$

De (8) se tiene que los terminos x_{ij} para $i \neq j$ son 0 y en caso contrario los valores son 1, según (1). Reemplazando los valores de la tabla III en (V-A.2):

$$C_{max} = 3 + 2 * 1 + 0 + 4 * 1 + 0 + 6 * 1 = 15 \tag{10}$$

Por lo que coincide con el último elemento de la tabla IV correspondiente a t^3_{22}.

A.3 ¿Qué ocurre al usar otra permutación?

Ahora el orden de los trabajos de acuerdo a la III, se utilizará $j_3 \rightarrow j_1 \rightarrow j_2$.

	m_1	m_2
j_3	5	6
j_1	3	2
j_2	1	4

Tabla V: Tabla de tiempos de procesamiento para un set de 3x2 permutada.

A.4 Representación

$S_1 =$

	Máquinas			
	M_1		M_2	
J	in	out	in	out
j_3	0	5	6	11
j_1	5	8	11	13
j_2	8	9	13	17

Tabla VI: Solución matricial, con tiempos de entrada y salida de cada máquina para cada trabajo de una secuencia permutada de la configuración III.

A.5 Cálculo función objetivo

$$C_{max} = 5 + 6 * 1 + 0 + 2 * 1 + 0 + 4 * 1 = 17 \quad (11)$$

Notamos que de acuerdo a la ecuación (11), la permutación $[j_3, j_1, j_2]$ nos arroja un tiempo de 17 unidades, lo cual es menor que la permutación presentada anteriormente.

B. Ejemplo set 5x2

Sea un conjunto de 5 trabajos y un conjunto de 2 máquinas, tal que sus tiempos de procesamiento se definen como en la tabla III

	m_1	m_2
j_1	5	3
j_2	2	7
j_3	9	1
j_4	0	6
j_5	8	4

Tabla VII: Tabla de tiempos de procesamiento para un set de 5x2.

B.1 Representación

Luego la representación de una solución con orden $j_4 \rightarrow j_2 \rightarrow j_5 \rightarrow j_1 \rightarrow j_3$, queda como en la tabla VIII.

$S_1 =$

	Máquinas			
	M_1		M_2	
J	in	out	in	out
j_4	0	0	0	6
j_2	0	2	6	13
j_5	2	10	13	17
j_1	10	15	17	20
j_3	15	24	24	25

Tabla VIII: Solución matricial, con tiempos de entrada y salida de cada máquina para cada trabajo de alguna secuencia en particular.

B.2 Cálculo función objetivo

De la ecuación (2) reemplazamos $m = 2$, como la última máquina de nuestro set.

$$C_{max} = \sum_{i=1}^{5}(v_{i2} + \sum_{j=1}^{5} p_{j2}x_{ij}) \quad (12)$$

$C_{max} = v_{12} + p_{12}x_{11} + p_{22}x_{12} + p_{32}x_{13} + p_{42}x_{14} + p_{52}x_{15} + v_{22} + p_{12}x_{21} + p_{22}x_{22} + p_{32}x_{23} + p_{42}x_{24} + p_{52}x_{25} + v_{32} + p_{12}x_{31} + p_{22}x_{32} + p_{32}x_{33} + p_{42}x_{34} + p_{52}x_{35} + v_{42} + p_{12}x_{41} + p_{22}x_{42} + p_{32}x_{43} + p_{42}x_{44} + p_{52}x_{45} + v_{52} + p_{12}x_{51} + p_{22}x_{52} + p_{32}x_{53} + p_{42}x_{54} + p_{52}x_{55}$(13)

De (12) se tiene que los terminos x_{ij} para $i \neq j$ son 0 y en caso contrario los valores son 1, según (1). Reemplazando los valores de la tabla VII en la ecuación:

$$C_{max} = 0+3*1+0+7*1+4+1*1+0+6*1+0+4*1 = 25 \quad (14)$$

Por lo que coincide con el último elemento de la tabla VIII correspondiente a t_{22}^5.

VI. Diseño Algoritmo

A continuación, se presenta una propuesta en pseudocódigo de un algoritmo de generación de soluciones vecinas para el problema del Flow shop sequencing, cuya implementación está basada en el paper [2]:

Listing 1: Algoritmo de generación de soluciones vecinas

```
S = [j1,j2,j3,...,jn]
Repetir
    ji = Seleccionar trabajo pivote i
    Repetir z = 1 hasta z = p // p = parámetro
         de máxima vecindad
        T(S) // tiempo de operación
        S' = seleccionar par de trabajos [i,i+z]
        y hacer swap de largo = z
        T(S') // tiempo de operación de solución
             S'
        Si T(S') <= T(S)
            S=S'
        SINO
            z++
    Fin repetir
Fin hasta satisfacer alguna condición
Retornar la mejor permutación con un tiempo T
// de la coleccion de soluciones se escoge
// la de mejor tiempo
```

Haciendo uso de este algoritmo, se podría generar un vecindario de soluciones para el cual se podría aplicar una solución metaheurística para el problema de planificación de trabajos.

A. Ejemplo aplicación

A continuación, de acuerdo a lo propuesto anteriormente, se utilizará el set (III), para aplicar la función de vecindad.

$$S = [j_1, j_2, j_3] \quad (15)$$

De lo anterior, se toma como pivote para j_1, por lo tanto el primer swap corresponde al par $[j_1, j_2]$, así S_1' queda como:

$$S_1' = [j_2, j_1, j_3] \quad (16)$$

Luego $T(S'_1) = 15$, es una solución factible y continuamos con el siguiente vecino, donde el pivote corresponde a j_2, y repitiendo el proceso, se obtiene:

$$\begin{aligned} S'_2 &= [j_2, j_3, j_1] \quad \text{con,} \\ T(S'_2) &= 13 \end{aligned} \tag{17}$$

Notamos que el algortimo se detiene a la 3ra permutación, encontrando 2 soluciones óptimas, con tiempos 15 y 13 unidades, para los conjuntos S'_1 y S'_2 respectivamente. Por otra parte, notamos que por enumeración exhaustiva, todas las soluciones se encuentran entre $[13, 17]$ unidades de tiempo, por lo que mediante este algortimo propuesto, se logra encontrar una solución óptima, reduciendo el número de permutaciones, en comparación a una solución más clásica, que implicaría probar 6 permutaciones.
Finalmente el algoritmo presentado tiene complejidad algorítmica:

$$O\binom{n}{2}$$

Referencias

[1] Willem J. Selen and David D. Hott, *A Mixed-Integer Goal-Programming Formulation of the Standard Flow-Shop Scheduling Problem*, Palgrave Macmillan Journals on behalf of the Operational Research Society, 1986.

[2] Marcus Ritt Alexander J. Benavides, *Iterated Local Search Heuristics for Minimizing Total Completion Time inPermutation and Non-Permutation Flow Shops*, 2015.