CAPÍTULO IV

4. FORMULACION DEL MODELO MATEMATICO

Este capítulo incluye La formulación y planteamiento del modelo matemático para obtener la función de costo y solución inicial, para lo cual debemos tener muy clara la teoría de Scheduling.

4.1. SCHEDULING (PLANEACION - ASIGNACION)

4.1.1 Introducción

Scheduling es un problema de complejidad NP-completo o NP-Duro y es tradicionalmente resuelto mediante la combinación de técnicas heurísticas locales y globales, que intentan reducir el espacio de búsqueda.

4.1.2 Resumen

Scheduling es un problema de optimización que se presenta en sistemas de planificación en la producción industrial, y planificación de transporte, etc. Es un proceso de toma de decisiones que tiene como meta la optimización de uno o más objetivos. Consiste en asignar recursos limitados a tareas donde hay restricciones de tiempo. El resultado será la obtención de una solución que minimice el tiempo necesario para completar la ejecución de todas las tareas del problema.

4.2 NUESTRO PROBLEMA

En la Industria Cartonera el problema que buscamos resolver es la Planificación de la Producción que consiste en la distribución y asignación de unidades de recursos o maquinaria (máquinas corrugadoras) a una combinación de pedidos (trimaje). Si se analiza los componentes de éste problema, se tiene que en primer lugar las unidades para producir son limitadas, la forma en la que se podría efectuar la planificación de la producción de los pedidos en el menor tiempo posible, seria asignar una unidad a cada uno de los pedidos. Pero, lo más realista sería pensar que no se tienen

tantas unidades como pedidos, y el problema consiste en determinar la ruta (planificación) que debe seguirse para producir en el menor tiempo (con el menor costo) todos los pedidos.

Pero aquí hay dos problemas en los que se debe pensar: Primero, el tiempo mínimo (si es que se puede determinar, puede ser demasiado largo). Segundo, las unidades tienen una cierta capacidad de almacenamiento o producción.

Este problema contiene dentro de sí muchos más: Determinar cuál es el tamaño óptimo de la maquinaria, determinar cuáles son los pedidos que deben ser asignados a cada unidad para ser producidos y determinar cuál es la ruta (planificación) que debe seguir cada una para terminar la producción en el menor tiempo posible.

4.3 MODELO MATEMATICO

4.3.1 Función de Costo

Para conseguir una función de tiempo, debemos determinar el tiempo que se demora cada unidad en producir la combinación de ordenes asignadas,. Para lo cual debemos obtener:

a) La cantidad dada en metros que se va ha necesitar de la plancha (caja abierta) para producir cada orden, calculando:

Q_{ij}: Cantidad en mts. de p_{ij} ; ***p_{ij}:*** pedido *i* en la corrugadora *j*

Q_{ij}: $q_{ij} * L_{ij}$; ***i = (1,... m pedidos), j= (1,2 corrugadoras)***

q_{ij}: cantidad en unidades de ítem o productos de p_{ij}

L_{ij}: largo de plancha en mts. de p_{ij} , y se obtiene con el siguiente calculo: $2* l_{ij} + 2* a_{ij} + tol_{ij}$

l_{ij}: largo del item o producto de p_{ij}

a_{ij}: ancho del item o producto de p_{ij}

$to.l_{ij}$: tolerancia del largo de la plancha, que depende del test de p_{ij}.

b) La relación de la cantidad de producir la orden con respecto a la capacidad de producción de la unidad , determina el tiempo de producir dicha orden.

Tp $_{ij}$: Tiempo de Producción de p_{ij}

Tp $_{ij}$: Q_{ij} / W_j ; ***i = (1,... m pedidos), j= (1,2 corrugadoras)***

W_j: capacidad de producción de la corrugadora *j* dada en metros por hora.

c) El tiempo que demora el cambio en la programación de una orden a la siguiente, es decir cambio de medidas (m^2) y características (test y flauta) en la máquina corrugadora.

Tcp $_{ij}$: Tiempo de Cambio en Programación de p_{ij}

$$Tcp_{ij}: \begin{cases} 15 \text{ min. ; si trimaje}_{ij} \neq \text{ trimaje}_{i+1j} \\ 0 \text{ min. ; si trimaje}_{ij} = \text{trimaje}_{i+1j} \end{cases}$$

Trimaje$_{ij}$: *flauta, test, m^2 de p_{ij}*

d) La suma del tiempo de producción con el tiempo de cambio en programación, determina el tiempo total que demora la producción de dicha orden.

T_{ij} **:** Función de Tiempo de p_{ij}

$T_{ij} : Tp_{ij} + Tcp_{ij}$; ***i = (1,... m pedidos), j= (1,2 corrugadoras)***

e) La suma de la formulación anterior, determina el tiempo total de producir todas las ordenes asignadas

C: Función de Tiempo para los pedidos asignados

$$C: \sum_{J=0}^{2} \sum_{i=0}^{m} T_{ij;}$$ ***i = (1,... m pedidos), j= (1,2 corrugadoras)***

f) Para obtener un tiempo total óptimo de producir todas las ordenes debemos minimizar la función de tiempo.

Min. *C*

4.3.2 Solución Inicial

Para obtener una solución inicial, se debe considerar:

a) Numero de Máquinas disponibles para la producción

b) Capacidad de cada máquina

c) Características de cada máquina

d) Determinar el conjunto de pedidos que pueden asignarse a cada máquina.

Para determinar cuales son los pedidos que deben ser asignados a cada unidad, debemos considerar ciertas características de dichas unidades, como se mencionó en el capítulo 1, se tienen diferentes tipos de corrugados, los de Pared Simple (*flauta B y C*) y los de Pared Doble (*flauta BC*), y se tienen dos máquinas Corrugadoras, la ***S & S*** y la ***Langston***, la primera es usada solo para corrugados de flauta ***C***, y la segunda para los corrugados de flauta ***B, C*** y ***BC***.

De manera que lo primero que se debe hacer es particionar el conjunto

$\boldsymbol{P = \{ p_1, p_2, p_3, \ldots, p_n \}}$ de pedidos en 2 subconjuntos,

$\boldsymbol{P = \{ LSS, L \}}$;

$\boldsymbol{LSS = \{ p_1, p_2, p_3, \ldots, p_m \}}$ y $\boldsymbol{L = \{ p_1, p_2, p_3, \ldots, p_r \}}$; $n = m+r$

$Si\ p_i \in LSS \Rightarrow (flauta.p_i = B \vee flauta.p_i = C \vee flauta\ p_i = BC)$

$Si\ p_i \in L \Rightarrow flauta.p_i = C$

Pero no es suficiente seleccionar que pedidos deben ser asignadas a cada unidad, lo mas importante es proponer en que orden pueden ser producidos, y esto se puede solucionar sabiendo cuales son los que tienen prioridad, es decir los mas urgentes de cumplir, lo que se indica en la fecha de despacho del pedido. Por lo que ordenaremos cada partición o subconjunto ***LSS*** y ***L***, es decir se particionará en grupos por fecha de despacho y estos grupos deben ordenarse de menor a mayor.

$\boldsymbol{LSS = \{}\ LSS_1, LSS_2, \ldots, LSS_p\ \boldsymbol{\}}$ y $\boldsymbol{L = \{}\ L_1, L_2, \ldots, L_q\ \boldsymbol{\}}$;

$\boldsymbol{LSS_k = \{}\ p_{k1}, p_{k2, \ldots,} p_{kn}\ \boldsymbol{\}}$

$Si\ p_i\ , p_j \in LSS_k \Rightarrow FechaDespacho.\ p_i = FechaDespacho.\ p_j$;

$k = (1,2, \ldots,p)$

$\boldsymbol{L_k = \{}\ p_{k1}, p_{k2, \ldots,} p_{km}\ \boldsymbol{\}}$

Si p_i , $p_j \in L_k \Rightarrow$ *FechaDespacho.* p_i = *FechaDespacho.* p_j *; k = (1,2, ...,q)*

Y para minimizar el tiempo, se debe minimizar el número de paradas entre ordenes, por cambio de medidas (mts.^{2} = largo de la plancha * ancho de la plancha) y características del pedidos (*test y flauta*), que como se menciono en el capitulo 1, es una de las principales causas por las que se para la máquina, lo que ocasiona una demora de 15 a 20 min. mas. Por lo que después de particionar ordenadamente por la fecha de despacho serán agrupados o combinados por test, luego por flauta, y por último por medidas, (a esta clase de combinación de pedidos se le denomina trimaje en el ámbito de la Producción Cartonera). Es decir que cada partición o subconjunto de ***LSS*** y *de* ***L*** será a su vez particionado en los grupos de combinaciones de pedidos:

$\boldsymbol{LSS_k = \{ O_{k1}, O_{k2}, \ldots , O_{kn} \}}$;

$\boldsymbol{O_{k1} = \{ p_1, p_{2, \ldots, p_a} \}}$ $\boldsymbol{O_{k2} = \{ p_1, p_{2, \ldots, p_b} \}}$,..., $\boldsymbol{O_{kn} = \{ p_1, p_{2, \ldots, p_c} \}}$;

$p = a+b+\ldots+c.$

$Si\ p_i\ ,\ p_j \in O_k \Rightarrow (test.p_i = test.p_j \wedge flauta.p_i = flauta.p_j \wedge m^2.p_i = m^2.p_j\ )$;

$k = (1,2, ...,p)$

$\boldsymbol{L_i = \{\ O_{i1}, O_{i2}, ..., O_{im}\}}$;

$\boldsymbol{O_{k1} = \{\ p_1, p_{2, ...,} p_d\}\ O_{k2} = \{\ p_1, p_{2, ...,} p_e\}}$,..., $\boldsymbol{O_{kn} = \{\ p_1, p_{2, ...,} p_f\}}$;

$q = d+e+...+f.$

$Si\ p_i\ ,\ p_j \in O_k \Rightarrow (Test.p_i = Test.p_j \wedge Flauta.p_i = Flauta.p_j \wedge m^2.p_i = m^2.p_j\ )$;

$k = (1,2, ...,q)$

Bien, una vez obtenido las particiones o grupos de ordenes, sabemos bajo que criterios podemos asignar los pedidos a cada unidad y en que orden, ya que hemos considerado ciertas restricciones: orden por fecha de despacho, agrupación por características de flauta y test, y medidas de mts^2 por lo que a este resultado lo denominamos ***Conjunto Solución Inicial***

4.3.3 Costo de la Solución

Para determinar el Costo de la Solución, procedemos a obtener los siguientes parámetros:

feci$_{ij}$: fecha inicio, fecha que empieza la producción el p_{ij}

fecf$_{ij}$: fecha fin, fecha que termina la producción el p_{ij} , está dada por: $fi_{ij} + Tp_{ij}$

t$_{ij}$: test de p_{ij}

f$_{ij}$: flauta de p_{ij}

k$_j$: numero de pedidos asignados a la corrugadora j

m$^2_{ij}$: metros cuadrados de la plancha de p_{ij} , está dado por:

$L_{ij} * A_{ij}$

A$_{ij}$: ancho de plancha del pedido i en la corrugadora *j* , y se obtiene con el siguiente calculo: $a_{ij} + h_{ij} + to.a_{ij}$

a$_{ij}$: ancho del item o producto de p_{ij}

h$_{ij}$: alto del item o producto de p_{ij}

to.a$_{ij}$: tolerancia del ancho de la plancha, que depende del test de $p_{ij.}$

Algoritmo 4.1: Algoritmo de la Función para obtener el Costo de la Solución

---

$\rightarrow$ fecha

$j \leftarrow 1$ to 2

begin

$\rightarrow k_j$

$fecf_{0j} \leftarrow$ fecha

$C \leftarrow 0$

$f_{0j} \leftarrow f_{1j}$

$t_{0j} \leftarrow t_{1j}$

$m^2_{0j} \leftarrow m^2_{1j}$

$j \leftarrow 1$ to k_j

begin

$\rightarrow q_{ij}$

$\rightarrow L_{ij}$

$\rightarrow W_j$

Si $f_{i-1j} <> f_{ij}$ or $t_{i-1j} <> t_{ij}$ or $m^2_{i-1j} <> m^2_{ij}$ entonces

begin

$Tcp \leftarrow 15$

else

$Tcp_{ij} \leftarrow 0$

end

$Tp_{ij} \leftarrow (q_{ij} * L_{ij}) / W_j$

$C \leftarrow C + Tp_{ij} + Tcp_{ij}$

$feci_{ij} \leftarrow fecf_{i-1j} + Tcp_{ij}$

$fecf_{ij} \leftarrow feci_{ij} + Tp_{ij}$

end

end

Mediante la implementación del algoritmo anterior a la partición o grupo de ordenes correspondiente a cada unidad hemos asignado:

a) Fecha en que inicia la producción

b) Tiempo que demora la producción

c) Fecha en que termina la producción

Además hemos obtenido el costo de producir los resultados del algoritmo considerados como una solución

Entonces decimos

$$C : \sum_{J=0}^{2} \sum_{i=0}^{m} T_{ij}$$: costo o tiempo de producir las ordenes asignadas

i* = *(1,... m pedidos), j= (1,2 corrugadoras)

Nuestro trabajo consiste en saber cuando rechazar o aprobar una solución lo que podemos determinar mediante el costo de la solución

4.3.4 Aprobar o Rechazar una Solución

Para rechazar o aprobar un conjunto solución, debemos obtener su costo el que debe ser evaluado por medio de métodos de búsqueda por entornos, que están dentro de las técnicas Meta-Heurísticas, que como hemos dicho anteriormente en el capítulo 2 son Heurísticas de éxito usadas para problemas intratables o NP Completos como es nuestro caso. Se ha escogido entre dichas técnicas la del Recocido Simulado.

Esta evaluación permite rechazar o aprobar el conjunto solución

Algoritmo 4.2: Algoritmo de la Función del Recocido Simulado

$\rightarrow k_j$

$T \leftarrow 0.005 * C_j(Sol_{inicial})$

$N \leftarrow 5*(k_j)$ número de iteraciones estimado

$e \leftarrow 10 \wedge -5$ epsilon o error estimado

$T_f = e /(k_j / 2)$

While $T >= T_f$

begin

$i \leftarrow 1$ to N

begin

$Sol_{cand} \leftarrow$ Permutar (Sol_{act})

$delta \leftarrow C_j(Sol_{cand}) - C_j (Sol_{act})$

$boltzmann \leftarrow Exp(-(delta / T_0))$

Si (Rand < boltzman) Or (delta < 0) entonces

begin

$Sol_{act} \leftarrow Sol_{cand}$

end

$T \leftarrow Rnd * (0.99 - 0.88) + 0.88 * T$

end

end

Algoritmo 4.3: Algoritmo de la Función Permutación de n elementos

---

```
→ n
i ← 1  to  n
  begin
        → V (i)
  end
Randomize
x ← Rand * (n - 1) + 1
y ← Rand * (n - 1) + 1
Si  x < y entonces
   Begin
        i ← 1  to  x-1
        begin
              U (i) ← V(i)
        end
        k←0
        i ← x  to  y
        begin
              U (i) ← V(y-k)
              k←k+1
        end
```

```
        i ← y+1  to  n
        begin
                U (i) ← V(i)
        End
Else
        k←0
        i ← 1  to  x
        begin
                U (i) ← V(x-k)
                k←k+1
        end
        k←0
        i ← x+1  to  y-1
        begin
                U (i) ← V(i)
        End

        i ← y  to  n
        begin
                U (i) ← V(n-k)
                k←k+1
        end
```

```
 end
i ← 1  to  n
   begin
        U (i) → ' mostrar la Permutacion
   End
```

Algoritmo 4.4: Algoritmo de la Función Factorial de n

```
→ n
Si n > 1 entonces
 begin
    fact ← 1
    i ← 0 to n – 1
      begin
         fact ← fact * (n - i)
      end
      Factorial ← fact
 else
     Si n = 0 Or n = 1 entonces
       Factorial ← 1
 end
```