CIENCIA XV UANL
ANIVERSARIO

Programación pieza-molde-máquina en planeación de la producción

O.L. Chacón Mondragón*, O.J. Ibarra Rojas*, Y.A. Ríos Solís*, M.A. Saucedo Espinosa*

La planeación es, en cualquier empresa de manufactura, una tarea vital que, muchas veces, no se cumple adecuadamente. En este estudio se aborda un problema de programación de tareas en una empresa local, cuyo objetivo es maximizar la ponderación de la demanda cumplida de piezas, con diferentes ponderaciones, para tratar de cumplir con la planeación. Las ponderaciones representan costos derivados de la compra de piezas no producidas o prioridades en la producción. Para la fabricar cada pieza, es necesario un molde que se monta en una máquina de inyección de plástico. Montar y desmontar un molde implica considerar un tiempo muerto, referenciado como tiempo de preproceso, tanto en la bibliografía como en este trabajo. En la figura 1 se ilustra (de izquierda a derecha) un conjunto de piezas, un conjunto de moldes y un conjunto de máquinas. Los arcos entre éstos representan la posible asignación de equipo. Los moldes y las máquinas son compatibles sólo con ciertas piezas y moldes, respectivamente, mientras la velocidad de producción de cada pieza se determina por el molde que la fabricará.

En la figura 2 ilustramos las consideraciones del proceso de producción. Cada barra representa una máquina; la longitud de la barra representa el límite de tiempo, mientras que el par *(j, f)* representa la programación de la pieza *j* con el molde *f*.

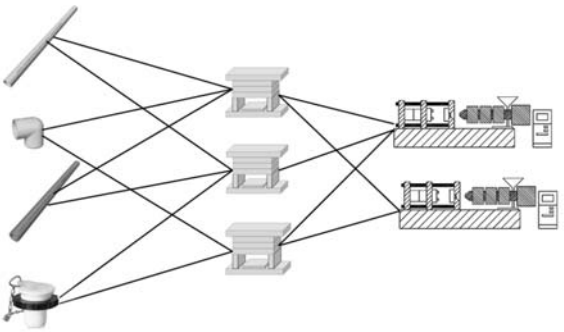

Fig. 1. Esquema de equipo de producción.

Se requiere que sólo un molde sea utilizado en cada máquina, para cualquier instante de tiempo. Se tiene, además, un tiempo disponible de uso para cada máquina, el cual no se puede sobrepasar. En la figura 2 se hace la producción (trabajos, moldes) en tres máquinas. En la M1 se considera un tiempo de preproceso al inicio, que representa la instalación del molde 3. De la misma forma, existe otro tiempo de preproceso que se genera al cambiar del molde 3 al molde 4. Con el molde 3 se hacen en las piezas 1 y 5, considerando un tiempo de preproceso al cambiar de pieza. Este tiempo es mucho más pequeño que el generado al cambiar moldes.

*Universidad Autónoma de Nuevo León, Facultad de Ingeniería Mecánica y Eléctrica.
olchacon.uanl@gmail.com

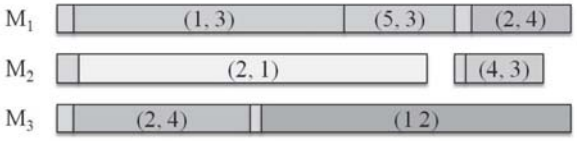

Fig. 2. Ilustración de tres máquinas de producción.

Estas características se han abordado por separado en diferentes problemas, principalmente de secuenciación. Primero, podemos encontrar diversos trabajos como los de Shim y Kim[1] y Weng *et al.*,[2] en los que se implementan tiempos de preproceso dependientes de la secuencia en máquinas paralelas, mas no los requerimientos de equipo auxiliar, como los moldes. Adicionalmente, en estudios, como el de Chen y Wu,[3] se considera la implementación de un equipo auxiliar para la producción de piezas en máquinas paralelas con la implementación de tiempos de preproceso. Sin embargo, en este tipo de problemas, la velocidad de producción depende de la máquina que procesa el trabajo. Asimismo, no es posible fabricar una pieza con diferentes tipos de molde, por lo que las decisiones de asignación no se toman en cuenta. Este último caso, es decir, la falta de asignación pieza-molde, se presenta también en problemas complejos de ensamble como los de Jacobson *et al.*,[4] que consideran las demás características, como los tiempos de preproceso dependientes de la secuencia, máquinas paralelas y diferentes ponderaciones para cada pieza.

Con el objetivo de simplificar la modelación, consideramos fabricar las piezas asignadas a un mismo molde, una tras otra. De esta forma, eliminamos la dependencia secuencial en el tiempo de preproceso, pudiéndose considerar como el tiempo fijo que toma colocar y retirar cada molde. Esta idea de agrupación de tareas obtiene buenos resultados al aplicarse en algoritmos de solución para problemas de secuenciación, tal como se ve en Chen y Wu.[3]

Considerando la hipótesis de tareas agrupadas, hacemos la modelación del sistema mediante un programa entero lineal, el cual es NP-duro, siendo una de nuestras principales contribuciones. Solucionar las instancias diseñadas del problema por un método exacto, como ramificación y acotación, implica un gran tiempo computacional. Por esta razón, desarrollamos una metodología de solución basada en el algoritmo de Búsqueda local iterativa, el cual genera buenos resultados para ciertos tipos de instancias, implementando poco tiempo computacional.

PROGRAMA ENTERO LINEAL

Para la formulación como problema de optimización, definimos a J, F y M como el conjunto de piezas, moldes y máquinas, respectivamente. Representamos, por medio del conjunto $J(f) \subseteq J$, las piezas que pueden fabricarse con cada molde f; así, $F(j) \subseteq F$ representa los moldes que pueden fabricar la pieza tipo j y, análogamente, $M(f)$ y $Fm(k)$. Consideramos, asimismo, los siguientes parámetros: la demanda d_j para cada tipo de pieza j, la velocidad de producción v_{jf} de la pieza tipo j con el molde f, una ponderación w_j para cada pieza j, el tiempo de preproceso s_{jf} entre trabajos, el tiempo de preproceso s_f de cada molde f, y el tiempo t_m disponible para cada máquina m.

Las variables de decisión necesarias para la modelación están dadas por: z_{jf}, que toma el valor de 1, si hay producción de la pieza j con el molde f, y_{fm}, que equivale a 1 si se asigna el molde f a la máquina m y, por último, una variable x_{jfm} (entera) que controla la cantidad de producción que se hará de la pieza tipo js, usando el molde f en la máquina m.

Como se mencionó antes, se desea maximizar la ponderación de la demanda cumplida, la cual se representa por la siguiente ecuación.

$$F(X) = \sum_{j \in J} w_j \sum_{f \in F(j)} \sum_{k \in K(f)} x_{jfk}$$

Así, el problema de programación de piezas con tiempos de preproceso y requerimiento de moldes (denotado por PMM) se define como:

min *F(X),* sujeto a:

$$\sum_{f\in F(j)}\sum_{m\in M(f)} x_{jfm} \le d_j \qquad \forall j \in J \quad (1)$$

$$\sum_{m\in M(f)} y_{fm} \le 1 \qquad \forall f \in F \quad (2)$$

$$\sum_{m\in M(f)} x_{jfm} \le z_{jf} d_j \qquad \forall j \in J,\ f \in F(j) \quad (3)$$

$$\sum_{j\in J(f)} x_{jfm} \le y_{fm} \sum_{j\in J(f)} d_j \qquad \forall f \in F, m \in M(f) \quad (4)$$

$$\sum_{f\in Fm(m)} \left(\sum_{j\in J(f)} \left(\frac{x_{jfm}}{v_{jf}} + z_{jf} s_{jf} \right) + s_f y_{fm} \right) \le t_m \quad \forall m \in M \quad (5)$$

$$x_{jfm} \in Z^+ \cup 0 \qquad \forall j \in J, f \in F(j), m \in M(f) \quad (6)$$

$$z_{jf} \in \{0,1\} \qquad \forall j \in J, f \in F(j) \quad (7)$$

$$y_{fm} \in \{0,1\} \qquad \forall f \in F, m \in M(f) \quad (8)$$

La ecuación (1) evita exceder la demanda de cada tipo de pieza *j*. Por otro lado, (2) representa la asignación de cada molde a sólo una máquina. La variable de asignación pieza-molde se controla por (3), mientras que (4) se encarga de controlar la variable de asignación molde-máquina. El no exceder el tiempo de máquina se controla mediante (5). Finalmente, el domino de las variables de decisión se cubre por (6)-(8).

En este modelo de optimización se consideran las decisiones de asignación pieza-molde y molde-máquina, además del flujo de producción, respetando la limitante de tiempo de máquina, lo cual es vital para la programación de la producción. Las decisiones del modelo propuesto no se encontraron en ningún modelo integral de problemas de secuenciación, de asignación o de problemas complejos de ensamble, según una búsqueda hecha en la bibliografía.

Ibarra Rojas *et al*[5] demostraron que el PMM pertenece a la categoría de los problemas NP-duros, e hicieron un estudio del programa usando la implementación de un algoritmo de ramificación y acotación (B&B, por sus siglas en inglés) usando GAMS y Cplex9.0. Notamos que resolver el PMM con los tamaños de instancia observados en problemas reales (120 piezas, 80 moldes y 20 máquinas) ocupa una gran cantidad de tiempo de cómputo. En particular, la relajación lineal del problema presenta también una lenta convergencia del B&B, a excepción de la relajación lineal de la variable y_{fk}, lo cual lleva a una rápida convergencia (476 segundos en promedio). Sin embargo, se pierde la estructura del problema. Aun relajando algunos componentes del programa entero, se presentan dificultades en la convergencia de un algoritmo exacto como el B&B. Por esta razón, se propone otro tipo de método de optimización.

METODOLOGÍA DE SOLUCIÓN

Dada la complejidad del problema y la necesidad de los sistemas reales de obtener soluciones de manera rápida y sin costear programas especializados de optimización, proponemos el uso de la heurística Búsqueda local iterativa (BLI) para solucionar el problema PMM. El algoritmo BLI se ha usado para la programación de tareas en máquinas y no sólo es aplicable como algoritmo de solución, sino como componente de procedimientos más sofisticados, como en Álvarez y Tamarit.[6]

Como apuntan Lourenço *et al*.,[7] BLI es una herramienta simple y rápida que puede aplicarse y mejorarse, aumentar la calidad de cada uno de los módulos. Cuando se hacen dichas mejoras, la interacción entre intensificación y diversificación de la metodología es vital y es, además, un problema desafiante. A continuación presentamos las componentes que usaremos en nuestra propuesta de solución con base en una BLI.

CONSTRUCCIÓN INICIAL

Implementamos un algoritmo miope con base en una lista de prioridades. La idea es programar la combinación de pieza-molde que genere un mayor ahorro en la máquina con mayor tiempo disponible. Se toma en cuenta un orden $(\rangle, J \times F)$ para cada combinación pieza-molde. Se dice que $(j_1, f_1) \rangle (j_2, f_2)$ cuando $w_{j1}v_{j1f1} > w_{j2}v_{j2f2}$. Análogamente, se establece un orden entre las máquinas $(\rangle, M)$. Se dice que una máquina $k_1 \rangle k_2$ cuando $t_{m1} > t_{m2}$.

Diseñamos el algoritmo *Construcción( )*, que construye una lista *lista(J, F)* de combinaciones pieza-molde y una lista *lista(M)* de máquinas ordenadas decrecientemente. Para *lista(J,F)* y *lista(M)* se consideran sólo piezas j con $d_j>0$ y máquinas m con $t_m>0$, respectivamente. Mientras haya elementos en las listas o hasta que se cumpla la demanda, se toma la primera combinación de la *lista(J, F)* para asignarse a la primera máquina m de la lista compatible con el molde f. Se determina el flujo de producción por $x_{jfm}=\max\{d_j, v_{jf}t_m^*\}$, donde t_m^* representa el tiempo libre de la máquina k, considerando los tiempos de preproceso. Luego se hace la actualización de t_k, *lista (J, F),* y d_j. Finalmente, se reordena *lista(M)*.

BÚSQUEDA LOCAL

Utilizamos el intercambio de asignaciones pieza-molde y molde-máquina como métodos de búsqueda local. El movimiento *Búsqueda_moldes(s)* se refiere al intercambio de moldes en diferentes máquinas, mientras que *Búsqueda_piezas(s)* se refiere al intercambio de piezas en diferentes moldes. Definimos las búsquedas locales como algoritmos miopes que realizan el primer intercambio que produzca una mejora, dada una solución semilla s. El algoritmo termina después de un número de iteraciones sin producir cambios.

PERTURBACIÓN Y CRITERIO DE ACEPTACIÓN

La perturbación para la BLI debe ser un movimiento irreversible por la búsqueda local. Esto se controla por el tipo y la "fuerza" del movimiento. Un movimiento "débil" podría explorar pocas soluciones, mientras que un movimiento "fuerte" haría una exploración casi aleatoria. En nuestro caso, implementamos el movimiento de 3-intercambio (ilustrado en la figura 3), que se encarga de cambiar la asignación dada entre tres parejas diferentes. Diseñamos dos tipos de perturbación a una solución s dada: *Perturbación_moldes(s,k)* y *Perturbación_piezas(s,k)*, que realizan un número k de 3-intercambio entre asignación molde-máquina y pieza-molde, respectivamente. En ambos casos, k es el parámetro que representa la "fuerza" de la perturbación.

El criterio de aceptación es fundamental en la búsqueda realizada por la BLI, ya que un algoritmo voraz podría arrojar poca diversidad en las soluciones. Sin embargo, en nuestro caso obtienen buenos resultados. El criterio de aceptación Aceptación consiste en aceptar el movimiento sólo si hay mejora en la función objetivo F(X).

ALGORITMOS CON BASE EN BLI

Las soluciones generadas con el algoritmo *Construcción( )* hacían uso de una gran cantidad de moldes, comparado con soluciones óptimas de instancias pequeñas. Teniendo esto en mente, proponemos eli-

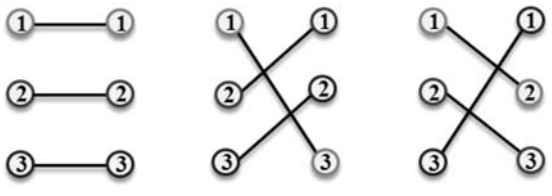

Fig.3. Visualización de los dos posibles 3-intercambios.

minar un porcentaje *per%* de los moldes que generen menor ahorro en la solución *s*, situación que se implementa en la función *Eliminar _moldes(s,per)*. Esto genera mayores ganancias al momento de aplicar los algoritmos de búsqueda local. A continuación se ilustra el algoritmo base de BLI.

Algoritmo BLI
Entrada: Instancia de PMM, salida: óptimo local.

```
1:  s ? Construccion()
2:  s ? Eliminar_moldes(s, per)
3:  s* ? Busqueda_moldes(s)
4:  s* ? Busqueda_piezas(s*)
5:  while (iter1 < limite1 ) do
6:    s´? Perturbacion_moldes(s*)
7:    s'* ? Busqueda_piezas(s')
8:    s'* ? Busqueda_moldes(s'*)
9:    s* ? Aceptacion(s*, s'*)
10: end While
```

RESULTADOS COMPUTACIONALES

Para la experimentación utilizamos *per*=20% y *k*=3, según un ajuste de parámetros. También generamos 10 repeticiones para dos tamaños diferentes de instancias, denotados por (|J|, |F|, |M|). Además, variamos el porcentaje de compatibilidad entre piezas y moldes, denotado por *cjf*, el cual representa un porcentaje de las $|J||F|$ posibles asignaciones. Es decir, *cjf* toma el valor de 100%, si todos los moldes son compatibles con todas las piezas. Análogamente, definimos a *cfm* para la compatibilidad de moldes y máquinas.

Los parámetros se generan con una distribución uniforme entre los intervalos observados en un sistema real: $d_j \in [1000, 18000]$, $v_{jf} \in [120, 1000]$, $s_{jf} \in [0, 0]$, $s_f \in [.75, 1.15]$ y, finalmente, t_m=24. Sin embargo, no se cuenta con suficiente información para estimar las ponderaciones de las piezas, por lo que se genera $w_j \in [1, 10]$. Implementamos los algoritmos con base en BLI en un equipo distribuido por HP, con un procesador Intel Core 2 Duo T5300, 1.83GHz, y 1GB de memoria RAM, mientras que para el método B&B utilizamos GAMS y Cplex9.0, en una terminal con procesador Sun Fire V440, conectado a cuatro procesadores de 1602 Hhz, Ultra SPARC III con 1 MB de cache y memoria de 8 GB. En ambos métodos, medimos la desviación promedio (GAP) de la mejor solución encontrada, con respecto a la mejor cota obtenida por Cplex9.0 (durante una hora de ejecución). Se reporta, además, el tiempo de ejecución de BLI.

La tabla I describe el GAP promedio y la necesidad del sistema real, es decir, el porcentaje cumplido de la demanda. Observamos que aunque hay una diferencia significativa en el GAP promedio, la diferencia en el porcentaje de la demanda cumplida es muy pequeña. Otra cuestión a notar es que la diferencia en el GAP de ambos métodos es menor para los grupos de instancias de mayor tamaño que para los tamaños pequeños y medianos. Se observa, además, que el aumento de asignaciones potenciales pieza-molde repercute enormemente en la calidad de ambos algoritmos de solución.

Como mencionamos anteriormente, la rapidez con que se actualizan las cotas en el método B&B se ve comprometida también por *cjf*. En la figura 4

Tabla I. Resultados de algoritmos de solución B&B y BLI.

(120,80,20), *cjf=5% y cfm=60%*	B&B	BLI
GAP promedio	2.83%	10.48%
Cumplimiento demanda	74.39%	74.17%
Tiempo (seg)	3600	56.33
(120,80,20), *cjf=15% y cfm=60%*	B&B	BLI
GAP promedio	29.72%	40.55%
Cumplimiento demanda	80.49%	78.47%
Tiempo (seg)	3600	146.07
(200,120,25), *cjf=5% y cfm=60%*	B&B	BLI
GAP promedio	7.07%	10.32%
Cumplimiento demanda	59.67%	58.99%
Tiempo (seg)	3600	182.47
(200,120,25), *cjf=15% y cfm=60%*	B&B	BLI
GAP promedio	25.90%	30.62%
Cumplimiento demanda	70.42%	68.82%
Tiempo (seg)	3600	682.22

observamos los valores objetivo alcanzados con B&B y BLI, así como las cotas encontradas para las instancias de tamaño (120, 80, 20). La mayor diferencia entre las cotas y los valores de la función objetivo se presentan en las últimas diez instancias, las cuales tienen un mayor porcentaje *cjf*.

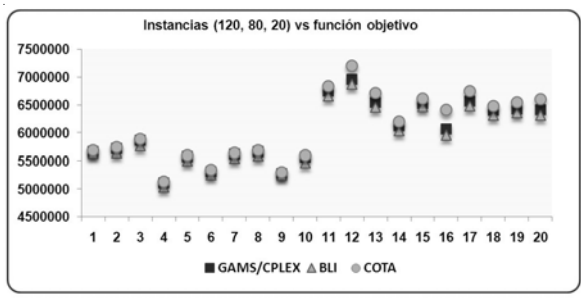

Fig. 4. Comparación de las soluciones obtenidas por B&B y BLI contra las cotas encontradas por Cplex9.0.

En la figura 5 ilustramos los valores de la demanda cumplida para las mismas instancias, donde es posible apreciar que los niveles de ambos métodos son muy parecidos, aun y cuando existe una diferencia significativa en el GAP promedio.

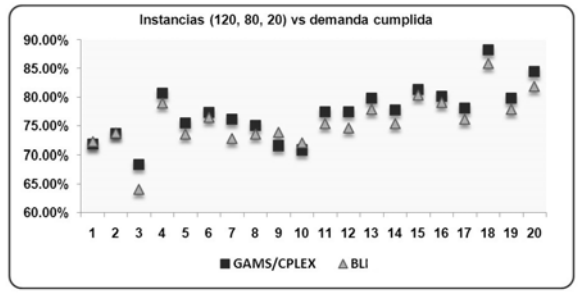

Fig. 5. Comparación de la demanda cumplida en soluciones obtenidas por B&B y BLI.

Este mismo comportamiento (diferencia en el GAP promedio y semejanza en el porcentaje de demanda cumplida) se presenta también en las instancias más grandes con aún mejores resultados

CONCLUSIONES

Se hizo una modelación representativa de un sistema real de manufactura mediante un programa entero lineal. El objetivo es maximizar la ponderación de la demanda cumplida, considerando el tiempo disponible de máquina y respetando la compatibilidad de equipo. En este modelo se determinan las asignaciones pieza-molde y molde-máquina, así como las cantidades a producir. Según nuestro conocimiento, no hay un modelo en la bibliografía que integre todas estas características.

Se generaron soluciones de "buena" calidad para instancias con pocas asignaciones potenciales de pieza-molde, mediante una Búsqueda local iterativa basada en movimientos de intercambio entre asignaciones de equipo. El método de solución propuesto permite cumplir un alto porcentaje de la demanda, al compararse con las cotas encontradas. Más aún, se encontró que las asignaciones pieza-molde tienen una gran repercusión en la calidad y convergencia de los algoritmos de solución diseñados. Sin embargo, los movimientos de intercambio propuestos en este trabajo para tal asignación no son eficientes, por lo que se determina así un área de trabajo futuro.

RESUMEN

Abordamos un proceso de manufactura de piezas con requerimiento de moldes y uso de maquinaria. Representamos el sistema como un problema de optimización, mediante una formulación de programación entera lineal que determina las cantidades y el tipo de piezas a producir, así como la asignación de piezas a moldes y de moldes a máquinas. El objetivo es maximizar la ponderación de la producción cumplida. Dado que el problema abordado pertenece a la clase de problemas NP-duros, se implementa una búsqueda local iterativa. Mediante la experimenta-

ción y resultados computacionales concluimos sobre la dificultad de las decisiones presentes en el proceso.

Palabras clave: Asignación, Secuenciación, Búsqueda Local iterativa, Flujo en redes, Programa entero lineal.

ABSTRACT

We address a manufacturing process of pieces, which uses molds and parallel machines. We represent the system as an optimization problem by using a linear integer programming formulation, which determines the quantity and type of pieces to produce. Our formulation determines also the piece-mold and mold-machine assignments. The objective is to maximize the weighted production of the fulfilled demand. Since the problem belongs to the class of NP-hard problems, we design a solution methodology based on Iterative Local Search Algorithm. Through experimentation and computational results, we make conclusions about the difficulty of the decisions involved in the process.

Keywords: Assignment, Scheduling, Iterated local search, Network flow, Integer linear program.

REFERENCIAS

1. S. Shim, Y. Kim, "A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property," *Computers and Operations Research, 35*(3), 863-875, 2006.
2. M.X. Weng, J. Lu, H. Ren, "Unrelated parallel machine scheduling with preproceso consideration and a total weighted completion time objective," *International journal of production economics, 70*(3), 215-226, 2001.
3. J. Chen, T. Wu, "Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints," *Omega The International Journal of Management Science, 34*(1), 81-89, 2006.
4. S. Jacobson, A. Johnson, K. Sullivan, M. Fleischer, A. Kumar, "Metaheuristic for a Flexible Assembly System Design Problems," *Journal of Heuristic, 3*(2), 139-159, 1997.
5. O.J. Ibarra-Rojas, O.L. Chacón-Mondragón, Y.A. Ríos-Solís, "Piece–mold–machine manufacturing planning," *Intelligent Systems in Operations: Models, Methods, and Applications in the supply chain*, Capítulo 6, 105-117, 2010.
6. R. Álvarez, J. M. Tamarit, "Algoritmos heurísticos deterministas y aleatorios en secuenciación de proyectos con recursos limitados," *Qüestiió, 13*(1), 173-191, 1989.
7. H.R. Lourenço, O. Martin, T. Stützle, "A Beginner´s Introduction to Iterated Local Search," *Metaheuristics International Conference*, Porto, Portugal, pp. 1-11, 2001.

Recibido: 1 de octubre de 2011
Aceptado: 25 de enero de 2012