Ingeniería, investigación y tecnología
ISSN: 1405-7743
Universidad Nacional Autónoma de México, Facultad de Ingeniería

Salazar-Hornig, Eduardo; Riquelme-Garrido, Mónica
Determinación de lotes y programación de múltiples productos en una máquina con tiempos de preparación
Ingeniería, investigación y tecnología, vol. XXII, núm. 3, 2021, Julio-Septiembre, pp. 22-29
Universidad Nacional Autónoma de México, Facultad de Ingeniería

DOI: https://doi.org/10.14482/INDES.30.1.303.661

Disponible en: https://www.redalyc.org/articulo.oa?id=40471798002

*Ingeniería Investigación y Tecnología*
volumen XXII (número 3), julio-septiembre 2021 1-8
ISSN 2594-0732 FI-UNAM artículo arbitrado
Información del artículo: Recibido: 12 de octubre de 2020, aceptado: 21 de abril de 2021

https://doi.org/10.22201/fi.25940732e.2021.22.3.018

Ingeniería
Investigación y
Tecnología

# Determinación de lotes y programación de múltiples productos en una máquina con tiempos de preparación

# Lot sizing and scheduling of multi-products on a single machine with sequence-dependent setup times

Salazar-Hornig Eduardo
Universidad de Concepción, Chile
Facultad de Ingeniería
Correo: esalazar@udec.cl
https://orcid.org/0000–0003–4387–0877

Riquelme-Garrido Mónica
Universidad de Concepción, Chile
Facultad de Ingeniería
Correo: moriquelme@udec.cl
https://orcid.org/0000-0003-4857-0611

## Resumen

Se desarrolla una metaheurística para resolver el problema de la determinación de lotes y programación de producción en un sistema de una máquina con múltiples productos y tiempos de preparación dependientes de secuencia. La metaheurística combina muestreo aleatorio utilizando el método Montecarlo y algoritmos genéticos para resolver de manera simultánea la configuración de lotes, así como la programación en la producción de la máquina, minimizando la cantidad de producto entregado de forma atrasada en los diferentes periodos del horizonte de programación. El proceso de optimización de la metaheurística se ilustra sobre la base de un problema generado aleatoriamente según la literatura. La metaheurística se compara con un método de búsqueda en vecindad utilizado previamente en una empresa manufacturera de bolas de molienda para la minería, evaluando un conjunto de instancias del problema de determinación de lotes y programación de producción de la práctica. En todas las instancias de prueba considerada, la metaheurística desarrollada en este trabajo mejora su solución con bajos tiempos computacionales. De acuerdo con el planteamiento del problema se establece una generalización de determinación y programación de lotes, cuya estructura y método de solución puede ser extendida a diferentes entornos productivos. La definición de la medida de desempeño, que en la aplicación tratada en este trabajo se definió como la cantidad de producto entregada atrasada (a minimizar), puede adoptar una definición diferente según la realidad en otro sistema productivo.
**Descriptores:** Determinación de lotes, algoritmos genéticos, metaheurística, programación de una máquina, muestreo aleatorio Montecarlo, tiempos de preparación dependientes de secuencia.

## Abstract

A metaheuristic approach, which combines the Monte Carlo random sampling method with genetic algorithms, to solve simultaneously the lot sizing and scheduling problem of multi-product on a single machine with sequence-dependent setup times is developed. The objective is the minimization of total backlog of the different periods in a scheduling horizon. The optimization process of the metaheuristic is illustrated on the base of a randomly generated problem in accordance to the literature. The metaheuristics is compared with a neighborhood search method previously used in a mining grinding ball manufacturing company by evaluating a set of instances of lot sizing and production scheduling problem of the practice. In almost all of the test instances considered, the metaheuristic developed in this work improves its solution with low computational times. According to the problem statement, a generalization of the lot sizing and scheduling problem is stated, which the structure and solution method can be extended to different production environments. The definition of the performance measure, which in the application discussed in this work was defined as the amount of product delivered late (to be minimized), may adopt a different definition depending on the reality in another production system.
**Keywords:** Lot sizing, genetic algorithms, metaheuristic, single machine scheduling, Monte Carlo random sampling, sequence dependent setup times.

https://doi.org/10.22201/fi.25940732e.2021.22.3.018


## Introducción

Uno de los aspectos fundamentales en la gestión de una empresa manufacturera enfocada en la mejora continua, es la gestión de sus procesos logísticos y de inventarios, lo que permite dar cumplimiento a los compromisos adquiridos con los clientes y alcanzar altos niveles de competitividad. La gestión eficiente de inventarios se apoya en el proceso de planificación y programación de producción, donde la determinación de los lotes de producción es fundamental.

El problema de dimensionamiento de lotes y programación de producción consiste en obtener un programa de producción que logre satisfacer de forma eficiente la demanda de los clientes, cumpliendo restricciones técnicas y operativas. Las principales decisiones de las que se ocupa son: La determinación de las órdenes de producción generadas como resultado de la determinación de los lotes de producción y las fechas de inicio y término de su procesamiento.

Existe una gran variedad de problemas de loteo dependiendo de las características que se tomen en consideración, como el CLSP (*Capacitated Lot Sizing Problem*), el cual posee una dificultad *NP-hard*, además, al incorporar tiempos de preparación o *setup* tiene una dificultad *NP-completa*. Debido a ello, este problema ha sido resuelto principalmente por heurísticas o metaheurísticas.

Gómez *et al.* (2013) presentan un modelo de programación lineal entera, donde encuentran una solución con brecha de 10 % respecto de la solución óptima. Ik *et al.* (2011) proponen una heurística de dos etapas, mejorando el desempeño de una heurística existente. En la primera etapa se obtiene una solución inicial y se determinan los tamaños de lotes, en la siguiente etapa se mejora la solución encontrada en la etapa anterior. Gupta & Magnusson (2005) formulan una heurística (que los autores llaman ISI), la cual consiste en un primer paso de inicialización donde se obtienen los tamaños de lote, posteriormente se continúa con la etapa de secuenciamiento utilizando una heurística *greedy*. Ocampo y Salazar (2014) proponen un enfoque de solución mixto, donde se utiliza una heurística para la generación de secuencias de lotes que son evaluadas mediante un modelo de optimización continua no lineal. Almada *et al.* (2007) presentan una heurística de cinco pasos, la que secuencia lotes de productos sin considerar restricciones de capacidad, luego se aplica un procedimiento que corrige no factibilidades de capacidad. Gonçalves y Sousa (2011) desarrollan una solución híbrida combinando algoritmos genéticos y una formulación de programación lineal. Mirabi (2011) presenta una metaheurística que consta de una primera fase de inicialización para lograr obtener una solución inicial y una segunda fase de mejora de la solución utilizando *simulated annealing*.

En este trabajo se generaliza un problema de dimensionamiento y programación de lotes de producción en una máquina, considerando múltiples productos y tiempos de preparación, dependientes de la secuencia basada en un problema de fabricación de bolas de molienda para la minería en una empresa chilena (Iribarren, 2011).

## Planteamiento del problema

En un sistema constituido por una máquina única que procesa múltiples productos en un horizonte de planificación finito (por ejemplo, un mes) dividido en períodos de tiempo (por ejemplo, semanas). Al realizar un cambio de tipo de producto a procesar se lleva a cabo una preparación o *setup* de la máquina, cuyo tiempo de preparación depende del tipo de producto que sale y el tipo de producto que entra. Además, cada tipo de producto posee una tasa conocida de proceso en la máquina y la demanda de cada tipo de producto se conoce en cada periodo del horizonte de programación.

Se busca elevar el nivel de servicio al cliente, es decir, maximizar el volumen de productos entregados a tiempo (asimismo minimizar el volumen de productos entregados de manera atrasada). De esta forma, el objetivo del problema es establecer los tamaños de lotes de cada tipo de producto dando origen a las órdenes de producción que deben secuenciarse en la máquina. Debido a factores de eficiencia y restricciones técnicas, existen lotes de producción mínimos por tipo de producto.

A continuación se describen los parámetros y variables de decisión del problema, donde los parámetros representan las características particulares de toda instancia del problema de determinación y programación de lotes, pero a su vez, con diferentes valores de instancia a instancia:

$T$ : Número de períodos del horizonte de programación
$N$ : Número de tipos de productos
$loteMin_i$ : Lote mínimo del producto tipo $i$; $i = 1, \ldots, N$
$tasa_i$ : Tasa de producción del producto tipo $i$; $i = 1, \ldots, N$
$s_{ij}$ : Tiempo de *setup* dependiente de la secuencia (paso de producto tipo $i$ al $j$)
$d_{it}$ : Demanda de producto $i$ en período $t$; $i = 1, \ldots, N$; $t = 1, \ldots, T$
$nMax_i$: Máximo de lotes del producto tipo $i$; $i = 1, \ldots, N$

$M_i$ : Número de lotes del producto tipo $i$ ($1 \leq M_i \leq nMax_i$); $i = 1, \ldots, N$
$M$ : Número total de de órdenes de producción (lotes) a secuenciar ($M = \sum_{i=1}^{N} M_i$)
*tipo* ($k$): Tipo de producto de orden $k$; $k = 1, \ldots, M$
*Ok* : Tamaño de orden $k$; $k = 1, \ldots, M$
*pk* : Tiempo de producción de orden $k$ ($p_k = tasa_{tipo(k)} \cdot o_k$); $k = 1, \ldots, M$

Las variables de decisión se refieren al tiempo de finalización $\boldsymbol{C_k}$ de cada orden de producción $\boldsymbol{k}$ (a través del tiempo de finalización es posible, restando su tiempo de proceso $\boldsymbol{p_k}$ y derivando el tiempo de inicio del proceso de toda orden de producción, por lo tanto, de la secuencia de trabajos en la máquina), la producción $\boldsymbol{x_{it}}$ de cada tipo de producto $\boldsymbol{i}$ que se obtiene en cada periodo $\boldsymbol{t}$, y el balance de inventario $\boldsymbol{I_{it}}$ para cada producto $\boldsymbol{i}$ al final del periodo $\boldsymbol{t}$:

$C_k$ : Tiempo de finalización de orden de producción $k$ ($k = 1. \ldots, M$)
$x_{it}$ : Produción del producto tipo $i$ en el período $t$ ($i = 1, \ldots, N$; $t = 1, \ldots, T$)
$I_{it}$ : Inventario del producto tipo $i$ al final del período $t$ ($i = 1, \ldots, N$; $t = 1, \ldots, T$)

La evaluación del programa de producción se realiza mediante la medida de desempeño (función objetivo) definida como *Déficit*, que consiste en el total de la producción entregada de forma atrasada en los diferentes periodos, y corresponde a la suma del déficit observado para cada tipo de producto $\boldsymbol{i}$ en cada periodo $\boldsymbol{t}$: $\boldsymbol{Def_{it}}$. Para su cálculo se realiza el balance de inventario para cada tipo de producto $\boldsymbol{i}$ al final de cada periodo $\boldsymbol{t}$:

$$I_{it} = I_{i,t-1} + x_{it} - d_{it}$$

Si $I_{it} < 0$ existe un déficit igual a $-I_{it}$ para el producto $\boldsymbol{i}$ en el periodo $\boldsymbol{t}$, esto es:

$$Def_{it} = max\ (0, -I_{it})$$

El déficit total de un programa será la suma del déficit resultante para cada tipo de producto en todos los periodos del horizonte de programación:

$$Déficit = \sum_{t=1}^{T} \sum_{i=1}^{N} Def_{it}$$

Para dar contexto al problema se asume, sin pérdida de generalidad, la producción en un horizonte de un mes subdividido en cuatro periodos semanales ($T = 4$), sin inventario al inicio del horizonte de programación (en caso de que este exista se corrige la demanda restándole este inventario).

## Metaheurística MCGA

Para resolver el problema de conformación y programación de lotes se propone una metaheurística híbrida, llamada (*Monte Carlo Genetic Algorithm*), la que integra muestreo aleatorio basado en el método Montecarlo con algoritmos genéticos.

Conociendo la demanda total del mes y el tamaño mínimo de lote para cada producto $\boldsymbol{i}$, se obtiene el número máximo de lotes como la división entera entre la demanda total y el tamaño mínimo de lote:

$$nMax_i = \left\lfloor \frac{\sum_{t=1}^{T} d_{it}}{loteMin_i} \right\rfloor$$

Para la conformación de lotes se genera, utilizando el método de Montecarlo, el número $M_i$ de lotes para cada tipo de producto $i$, el que se genera de manera (pseudo) aleatoria en el intervalo discreto [1, $nMax_i$] asumiendo una distribución uniforme discreta (todos los valores tienen igual probabilidad de ser generados). Si $M_i = 1$, el único lote conformado es por el total de la demanda del producto $i$; si $M_i > 1$ los $M_i$ lotes serán todos del mismo tamaño. Cada lote se transforma en una orden de producción $k = 1, \ldots M$ con $M = \sum_{i=1}^{N} M_i$. El tiempo de proceso de la orden $k$ se obtiene multiplicando la tasa de producción correspondiente al tipo de producto de la orden $k$ por la cantidad de producto asociada a la mencionada orden: $p_k = tasa_{tipo(k)} \cdot o_k$ con $o_k = \sum_{t=1}^{T} d_{tipo(k),t} / M_{tipo(k)}$.

Una vez definidos todos los lotes las órdenes de producción resultantes se secuencian mediante un algoritmo genético cuyos individuos (cromosomas) corresponden a una secuencia de las $\boldsymbol{M}$ órdenes de producción.

El algoritmo utiliza el operador genético de cruzamiento *OX* (*Order Cossover*), el cual selecciona aleatoriamente dos puntos de corte en la cadena del cromosoma en ambos padres. Los trabajos comprendidos dentro de los dos puntos de corte se transfieren manteniendo las posiciones a la decendencia. Luego, los trabajos restantes de un padre se transfieren a partir de la parte final del cromosoma del respectivo padre manteniendo el orden de los trabajos y excluyendo aquellos trabajos que ya fueron transferidos desde el otro padre a la parte central del hijo (Figura 1).

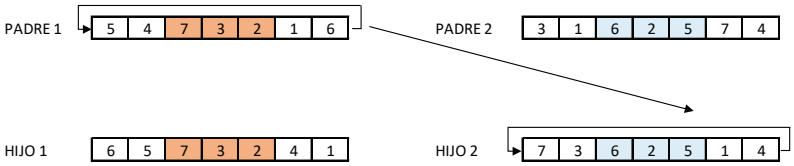


Figura 1. Operador de cruzamiento *OX*

Como operador genético de mutación se utiliza el operador de intercambio (*swap*), el cual selecciona aleatoriamente dos posiciones (*genes*) de un cromosoma, intercambiando los trabajos contenidos en dichas posiciones (Figura 2).

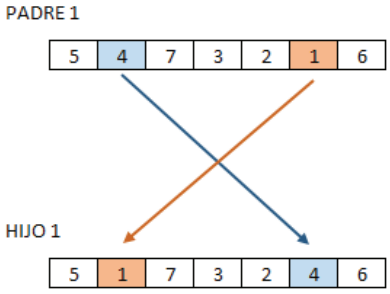


Figura 2. Operador de mutación *swap*

La probabilidad de selección de un individuo se basa en el criterio de aptitud *Déficit*, a menor valor de *Déficit* los individuos tendrán mayor probabilidad de ser seleccionados.

El método Montecarlo para generar el número de lotes por tipo de producto se combina con el algoritmo genético, dando paso al algoritmo ***MCGA***. La Figura 3 muestra la estructura (pseudocódigo) de la metaheurística ***MCGA*** diseñada para resolver este problema, ya que refleja la integración entre el muestreo aleatorio y el algoritmo genético. La variable $s_b$ almacena la mejor solución conocida, inicialmente almacena una solución generada por el procedimiento ***solución_inicial***(), lo que permite posteriormente inicializar la comparación de las soluciones que se van obteniendo en el proceso iterativo. Se realiza un número total igual a ***muestra*** de problemas de programación de una máquina, en cada uno de ellos se genera de forma aleatoria aplicando el método Montecarlo con el número $M_i$ de lotes para cada tipo de producto ***i***, definiendo a continuación las ***M*** órdenes de producción del problema a través del procedimiento ***DefinirOP***(). Las ***M*** órdenes de producción se secuencian en la máquina mediante el algoritmo genético del procedimiento ***SecuenciarOP***(). La solución ***s*** entregada por este procedimiento se compara con $s_b$, la mejor solución conocida hasta el momento. Si ***s*** mejora a $s_b$, esta reemplaza a $s_b$. Al finalizar la mejor solución encontrada queda almacenada en la variable $s_b$.

```
metaheurística MCGA( )
{
 sb ← solución_inicial( )
 m ← 1
 while (m ≤ muestra)
 {
  for (i=1, … , N) {generar Mi}
  DefinirOP()
  s ← SecuenciarOP()
  if ( f(s) < f(sb) ) {sb ← s }
  m ← m+1
 }
 Salida: sb
}
```

Figura 3. Metaheurística MCGA (pseudocódigo)

Los parámetros utilizados en la experimentación son los siguientes (para el algoritmo genético se consideraron valores recomendados en problemas de programación de trabajos de la literatura y para el valor de *muestra* se utiliza un valor observado empíricamente en pruebas preliminares):

- Tamaño de muestra Montecarlo *muestra* = 100
- Tamaño población algoritmo genético = 50
- Generación aleatoria de la población inicial del algoritmo genético
- Número de generaciones del algoritmo genético = 100
- Probabilidad de cruzamiento = 0.8
- Probabilidad de mutación = 0.1
- Número de réplicas del algoritmo genético en cada muestra = 10

Se ilustra el proceso de optimización de la metaheurística ***MCGA*** a través de un problema de $N = 8$ tipos de productos, generado aleatoriamente según lo propuesto por Almada *et al.* (2007) asumiendo que las unidades de producto y tiempo son genéricas y compatibles, la Tabla 1 resume la información general del problema de ilustración:

- Demanda: $d_{it} \sim$ *Uniforme* (40, 60)
- Tiempos de *setup*: $sij \sim$ *Uniforme* (5, 10)

- Tasa de produción: $tasa_i = 1$
- Tamaño mínimo de lote: $M_i = 30$

El número máximo de lotes por producto se calcula como: $nMax_i = [dT_i/loteMin_i]$ donde $dT_i = \sum_{t=1}^{T} d_{it}$ es la demanda total del producto $i$ (por ejemplo, para el producto tipo 1 se tiene $nMax_1 = \lfloor 199/30 \rfloor = \lfloor 6.63 \rfloor = 6$, por lo que en cada muestra $M_i$ se genera aplicando el método de Montecarlo a la distribución uniforme discreta en el intervalo [1, $nMax_i$] (por ejemplo, en el intervalo [1, 6] para el producto tipo 1).

Conociendo el número de lotes por producto se determina el tamaño de cada orden de producción (lote) $O_k$, dividiendo el total de la demanda por el número de lotes del respectivo tipo de producto: $o_k = dT_{tipo(k)}/M_{tipo(k)}$ (por ejemplo, si $M_6 = 4$ el tamaño de lote para las cuatro órdenes de producción de producto tipo 6 serán de 192/4 = 48.

Para efectos de la ilustración se consideran tres muestras (*muestra* = 3) y dos réplicas del algoritmo genético por cada muestra. En la Tabla 2 se presentan las tres muestras del número de lotes por tipo de producto. Se generan tres problemas de programación de producción de 24, 20 y 33 órdenes de producción que se resuelven mediante el algoritmo genético. La solución entregada por la metaheurística es aquella configuración de lotes y secuencia de producción de menor valor de *Déficit* entre todas las soluciones generadas durante el proceso de búsqueda.

En la Figura 4 se aprecia la evolución de la mejor solución encontrada en el proceso de optimización realizado por la metaheurística ***MCGA*** a través del muestreo y réplicas del algoritmo genético. El problema de 24 trabajos generado en la muestra 1 se resuelve mediante el algoritmo genético que compone las soluciones con ***Déficit* = 2229** (réplica 1) y ***Déficit* = 2192** (réplica 2). Al cabo del proceso de las tres muestras la mejor solución que se obtiene es la de la réplica 2, de la muestra 1 con ***Déficit* = 2192**.

## Experimentos y resultados

Para mostrar la efectividad del método desarrollado se consideran tres problemas tratados por Iribarren (2011) en una empresa productora de bolas de molienda para la minería, enfocando su trabajo en el proceso de una de las máquinas del sistema productivo en la que se procesan diversos tipos de productos (bolas de molienda de diferentes diámetros). Los resultados obtenidos con la metaheurística *MCGA* se comparan con los resultados obtenidos con el método de búsqueda en vecindad *IP* utilizado en el mencionado trabajo. En la Tabla 3 se muestran los parámetros generales para el problema considerando seis tipos de productos.

El trabajo de la referencia la determinación de los tamaños de lotes se realiza de forma manual (no sistemática y externa al método *IP*). Siguiendo criterios técnicos, la demanda total de cada producto se divide en partes iguales, formando lotes de igual tamaño, tenien-

Tabla 1. Parámetros problema de ilustración ($N$ =8)

| Tipo | $s_{ij}$ (*hora*) | | | | | | | | $d_{it}$ | | | | $dT_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 8 | 7 | 8 | 5 | 10 | 6 | 7 | 41 | 56 | 53 | 49 | 199 |
| 2 | 9 | 0 | 8 | 8 | 5 | 5 | 9 | 5 | 58 | 52 | 59 | 40 | 209 |
| 3 | 6 | 6 | 0 | 8 | 6 | 10 | 6 | 7 | 53 | 50 | 53 | 47 | 203 |
| 4 | 8 | 7 | 8 | 0 | 8 | 8 | 10 | 5 | 45 | 47 | 49 | 59 | 200 |
| 5 | 9 | 8 | 10 | 9 | 0 | 7 | 10 | 10 | 55 | 47 | 52 | 48 | 202 |
| 6 | 10 | 6 | 6 | 6 | 8 | 0 | 8 | 10 | 45 | 51 | 52 | 44 | 192 |
| 7 | 10 | 7 | 8 | 9 | 7 | 6 | 0 | 8 | 45 | 41 | 41 | 49 | 176 |
| 8 | 5 | 5 | 7 | 9 | 10 | 8 | 6 | 0 | 48 | 51 | 48 | 54 | 201 |

Tabla 2. Lotes ($M_i$)

| Tipo | $nMax_i$ | Muestra 1 | Muestra 2 | Muestra 3 |
|---|---|---|---|---|
| 1 | 6 | 1 | 3 | 6 |
| 2 | 6 | 2 | 1 | 1 |
| 3 | 6 | 3 | 1 | 3 |
| 4 | 6 | 3 | 3 | 6 |
| 5 | 6 | 6 | 2 | 6 |
| 6 | 6 | 2 | 4 | 4 |
| 7 | 5 | 4 | 2 | 5 |
| 8 | 6 | 3 | 4 | 2 |
| $M$ | | 24 | 20 | 33 |

https://doi.org/10.22201/fi.25940732e.2021.22.3.018


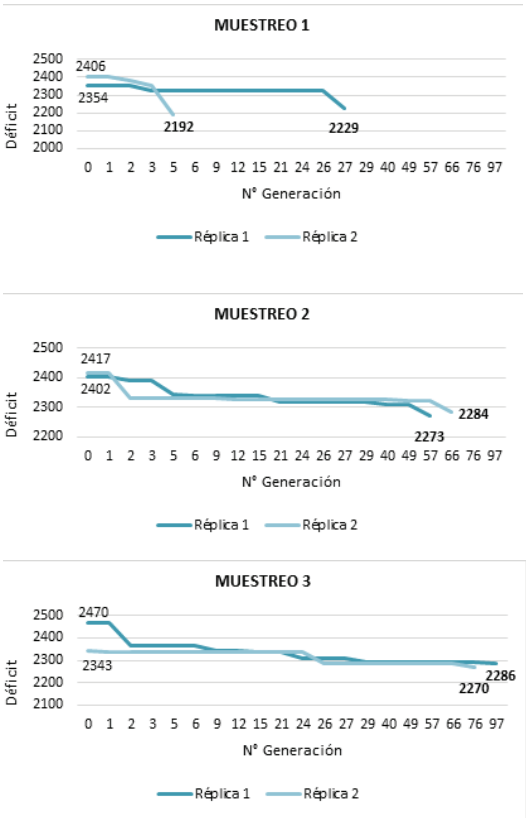


Figura 4. Evolución de valor *Déficit* para el problema ilustrativo

Tabla 3. Parámetros generales

| Tipo | $tasa_i$ (*ton / hora*) | $loteMin_i$ (*ton*) | $s_{ij}$ (*hora*) |
|---|---|---|---|
| 1 | 4.1 | 500 | 0 4 5 6 7 8 |
| 2 | 5.8 | 500 | 4 0 4 5 6 7 |
| 3 | 6.6 | 500 | 5 4 0 4 5 6 |
| 4 | 9.2 | 500 | 6 5 4 0 4 5 |
| 5 | 9.2 | 500 | 7 6 5 4 0 4 |
| 6 | 9.2 | 500 | 8 7 6 5 4 0 |

do en consideración la restricción de tamaño mínimo de lote. Una vez determinados los lotes se procede a secuenciar las órdenes de producción mediante un método de búsqueda en vecindad *IP* utilizando como solución inicial aquella que prioriza los trabajos según la demanda de los productos a través del tiempo (periodos). Se seleccionan tres problemas (los problemas Serie 950, Serie 1040 y Serie 1160) tratados en Iribarren (2011), los que se diferencian en el número de productos considerados y sus demandas.

La demanda mensual por tipo de producto expresada en toneladas/semana se muestra en la Tabla 4 (notar que no en todos los problemas se considera la producción de todos los productos).

La determinación de lotes se realiza de forma manual de acuerdo con los siguientes criterios: Sin subdivisión, máximo una subdivisión y máximo tres subdivisiones. La solución inicial se determina acorde con el criterio de priorizar aquellas órdenes de productos con mayores demandas en las primeras semanas del mes, pero al mismo tiempo evitando producir escasez en las restantes.

Los experimentos se llevan a cabo utilizando los problemas Serie 950, Serie 1040 y Serie 1160. Los resultados producidos por la metaheurística *MCGA* se comparan con la búsqueda en vecindad *IP*. Las instancias se resuelven utilizando el programa *SPS_MCGA_Scheduler* adaptando rutinas en lenguaje C/C++ de *SPS_Optimizer* (Salazar, 2010). Todos los experimentos se realizaron en un computador equipado con un procesador Intel Core i5-8350U, con 1,70 GHz y 4 GB de RAM.

La Tabla 5 muestra el detalle de las instancias mejor evaluadas del método de búsqueda en vecindad, por ejemplo, la instancia mejor evaluada del problema Serie 950 considera un máximo de una subdivisión, estable-

https://doi.org/10.22201/fi.25940732e.2021.22.3.018


Tabla 4. Demandas (ton/semana) $d_{it}$ y total $dT_i$

| Serie 950 | | | | | Serie 1040 | | | | | Serie 1160 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 140 | 140 | **280** | 0 | 0 | 168 | 84 | **252** | 112 | 112 | 224 | 28 | **476** |
| 0 | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | **0** | 0 | 28 | 0 | 28 | **56** |
| 84 | 84 | 196 | 168 | **532** | 560 | 140 | 476 | 224 | **1400** | 0 | 56 | 168 | 84 | **308** |
| 224 | 28 | 28 | 0 | **280** | 0 | 0 | 0 | 0 | **0** | 0 | 0 | 56 | 0 | **56** |
| 128 | 0 | 156 | 28 | **312** | 84 | 56 | 84 | 56 | **280** | 56 | 56 | 56 | 56 | **1024** |
| 812 | 644 | 952 | 1176 | **3584** | 840 | 476 | 350 | 420 | **2086** | 784 | 756 | 84 | 84 | **1708** |

Tabla 5. Instancias mejor evaluadas IP

| Problema | M (subdivisión) | Parámetros | Déficit |
|---|---|---|---|
| Serie 950 | 6 (1) | $o_k$: 500, 532, 500, 500, 1792, 1792<br>*tipo* ($k$): 1, 3, 4, 5, 6, 6<br>*SI:* $O_5 - O_2 - O_4 - O_1 - O_3 - O_6$<br>*IP:* $O_3 - O_5 - O_4 - O_2 - O_6 - O_1$ | <br><br>1803<br>387 |
| Serie 1040 | 7 (2) | $o_k$: 500, 700, 700, 500, 696, 695<br>*tipo* ($k$): 1, 3, 3, 5, 6, 6, 6<br>*SI:* $O_5 - O_2 - O_6 - O_3 - O_7 - O_4 - O_1$<br>*IP:* $O_5 - O_2 - O_6 - O_4 - O_7 - O_3 - O_1$ | <br><br>536<br>396 |
| Serie 1160 | 7 (1) | $o_k$: (500, 500, 500, 500, 500, 854, 854)<br>*tipo* ($k$): (1, 2, 3, 4, 5, 6, 6)<br>*SI:* $O_7 - O_1 - O_6 - O_3 - O_4 - O_5 - O_2$<br>*IP:* $O_7 - O_1 - O_6 - O_5 - O_4 - O_3 - O_2$ | <br><br>224<br>178 |

ciendo seis órdenes de producción, de las cuales, las cuatro primeras son de lote único para los productos 1, 3, 4, 5, mientras que las dos últimas consideran una subdivisión del producto tipo seis. Se tienen por lo tanto, las órdenes de producción: $O_1$, $O_2$, $O_3$, $O_4$, $O_5$ y $O_6$ de productos tipo 1, 3, 4, 5, 6 y 6, respectivamente, con tamaños de lote de 500, 532, 500, 500, 1792 y 1792 toneladas, respectivamente. La solución inicial considerada para resolver esta instancia con el método de búsqueda en vecindad es la secuencia *SI* con valor de *Déficit* = 1803 *ton*. A partir de esta solución se realiza la búsqueda en vecindad obteniendo la secuencia *IP* con valor de *Déficit* = 387 *ton*. Análogo para los otros dos problemas.

En la Tabla 6 se muestran las soluciones obtenidas con la metaheurística *MCGA* y en la Tabla 7 se comparan estas con el método *IP* agregando el tiempo CPU de la metaheurística *MCGA* (el tiempo CPU del método *IP* son fracciones de segundo). Si bien, la diferencia del tiempo computacional entre las metaheurísticas *MCGA* e *IP* es significativa, en este caso esto no se considera un aspecto negativo, dado que 35 segundos es un tiempo muy adecuado que no interfiere con la toma de decisiones en problemas de este tipo, atendiendo además que la solución se mejora en forma significativa.

Para el problema Serie 950 se observa que la metaheurística *MCGA* resuelve el problema utilizando 10 órdenes de producción, mejorando la solución en 21.19 % en comparación con la solución obtenida con el método *IP*, la que considera seis órdenes de producción. Por otro lado, el problema Serie 1040 es el que presenta la mayor reducción del mejorando la solución en un 78.79 % utilizando ocho órdenes de producción, a diferencia de la solución encontrada con la búsqueda en vecindad *IP*, la cual utiliza siete lotes para su resolución. Observando los resultados obtenidos para el problema Serie 1160 se observa que también el método *MCGA* mejora la solución entregada por el método *IP*, observándose una mejora de 5.06 %.

Tabla 6. Solución MCGA

| Problema | M | Parámetros | Déficit |
|---|---|---|---|
| Serie 950 | 10 | $o_k$: 500, 532, 500, 500, 597, 597, 597, 597, 597, 599<br>*tipo* ($k$): 1, 3, 4, 5, 6, 6, 6, 6, 6, 6<br>*MCGA:* $O_4 - O_{10} - O_3 - O_2 - O_8 - O_6 - O_7 - O_1 - O_9 - O_5$ | <br><br>305 |
| Serie 1040 | 8 | $o_k$: 500, 700, 700, 500, 521, 521, 521, 523<br>*tipo* ($k$): 1, 3, 3, 5, 6, 6, 6, 6<br>*MCGA:* $O_2 - O_5 - O_8 - O_7 - O_4 - O_3 - O_1 - O_6$ | <br><br>84 |
| Serie 1160 | 8 | $o_k$: 500, 532, 500, 500, 1792, 1792<br>*tipo* ($k$): 1, 3, 4, 5, 6, 6<br>*SI:* $O_6 - O_7 - O_1 - O_8 - O_5 - O_3 - O_4 - O_2$ | <br><br>169 |

Tabla 7. Comparación IP vs MCGA

| Problema | IP | MCGA | %Mejora | CPU (seg) |
|---|---|---|---|---|
| Serie 950 | 387 | 305 | 21,19 | 35 |
| Serie 1040 | 396 | 84 | 78,79 | 32 |
| Serie 1160 | 178 | 169 | 5,06 | 35 |

## Conclusiones

El presente trabajo se centra en el problema de dimensionamiento y programación de lotes de producción de múltiples productos en una máquina, con tiempos de preparación dependientes de la secuencia. En este trabajo se propone una mejora a un método de búsqueda en vecindad *IP* que resuelve el problema de definición de lotes y programación de producción en una empresa manufacturera que produce bolas de molienda de diferente diámetro para la minería.

Para la resolución del problema se desarrolla la metaheurística *MCGA* que integra muestreo aleatorio utilizando el método Montecarlo para generar el número de lotes (órdenes de producción) a procesar por tipo de producto. Por tipo de producto, las órdenes de producción se consideran todas de igual tamaño. Luego, las órdenes de producción se secuencian utilizando un algoritmo genético.

Como objetivo se define la minimización del *Déficit*, es decir, la cantidad total entregada de producto de forma atrasada en el horizonte de programación.

Por otro lado, la metaheurística *MCGA* desarrollada en este trabajo propone un método generalizado para resolver el problema de determinación de lotes y programación de producción de forma simultánea, susceptible de ser aplicada a diferentes entornos productivos. Además, a modo de ilustración del método, se generó un problema en forma aleatoria según la literatura.

La metaheurística *MCGA* se utiliza para resolver un conjunto de problemas de la práctica de una empresa manufacturera en la fabricación de bolas de molienda para la minería. En los problemas evaluados se puede apreciar que la metaheurística *MCGA*, además de automatizar la generación de los lotes de producción por tipo de producto, logra mejorar las soluciones en comparación con un método de búsqueda en vecindad *IP* con bajos tiempos computacionales, adecuados y suficientes para el proceso de toma de decisiones.

## Referencias

Almada, B., Klabjan, D., Carravilla, M. A. & Oliveira, J. F. (2007). Single machine multi-product capacitated lot sizing with sequence-dependent setup. *International Journal of Production Research*, 45(20), 4873-4894. https://doi.org/10.1080/00207540601094465

Gómez, J. A., Escobar, J. W. & Figueroa, Á. (2013). A multi-product lot-sizing model for a manufacturing company. *Ingeniería Investigación y Tecnología*, 14(3), 413-419.

Gonçalves, J. F. & Sousa, P. S. (2011). A genetic algorithm for lot-sizing and scheduling under capacity constraints and allowing backorders. *International Journal of Production Research*, 49(9), 2683-2703. https://doi.org/10.1080/00207543.2010.532936

Gupta, D. & Magnusson, T. (2005). The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers and Operations Research*, 32(4), 727-747. http://dx.doi.org/10.1016/j.cor.2003.08.014

Ik-Soo S., Hyeok-Chol K., Hyoung-Ho D., Dong-Ho L. (2011). A two-stage heuristic for single machine capacitated lot-sizing and scheduling with sequence-dependent setup costs. *Computer and industrial engineering*, 61(4), 920-929. http://dx.doi.org/10.1016/j.cie.2011.06.002

Iribarren, R. A. (2011). *Modelo para la programación de la producción con configuración de lotes-Caso de Moly-Cop Chile S. A.* Concepción, Chile: Universidad de Concepción.

Mirabi, M. (2011). A hybrid simulated annealing for the single-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times and costs and dynamic release of jobs. *International Journal of Advanced Manufacturing Technology*, 54(9), 1109-1119. http://dx.doi.org/10.1007/s00170-010-2988-5

Ocampo, H. & Salazar, E. (2014). Dimensionamiento de lotes y programación de una máquina para múltiples productos con setup y escasez. *Ingeniería Industrial*, 13(2), 9-62.

Salazar, E. (2010). Programación de sistemas de producción con SPS_Optimizer. *Revista ICHIO,* 1 (2), 33-46.

**Cómo citar:** Salazar-Hornig E. & Riquelme-Garrido M. (2021). Determinación de lotes y programación de múltiples productos en una máquina con tiempos de preparación. *Ingenieria Investigacion y Tecnologia*, 22 (03), 1-8. https://doi.org/10.22201/fi.25940732e.2021.22.3.018