UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
INSTITUTO DE INGENIERÍA

PLANIFICACIÓN DE LA PRODUCCIÓN DE DISPOSITIVOS ELECTRÓNICOS BASADA EN LA OPTIMIZACIÓN DEL USO DE MATERIALES

T E S I S
que para obtener el grado de DOCTOR EN CIENCIAS
Presenta
Rainier Romero Parra

DIRECTOR: Dra. Larysa Burtseva

MEXICALI, B. C. **FEBRERO DE 2013**

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN de Tesis de Rainier Romero Parra, presentada como requisito parcial para obtener el grado de DOCTOR EN CIENCIAS, Mexicali, Baja California, Febrero de 2013.

PLANIFICACIÓN DE LA PRODUCCIÓN DE DISPOSITIVOS ELECTRÓNICICOS BASADA EN LA OPTIMIZACIÓN DEL USO DE MATERIALES

Resumen aprobado por:

_____

Dra. Larysa Burtseva
Director de Tesis

Se investiga un problema real de planificación de recursos en una planta dedicada a la fabricación de interruptores eléctricos. El proceso productivo se lleva a cabo utilizando lotes de material. Existen varios tipos de lotes y cada tipo contiene material con las mismas características. El tamaño del lote es fijo. La investigación se enfoca en una operación donde las piezas de cada lote clasifican entre repositorios de acuerdo a una distribución empírica conocida. Las piezas de los lotes se clasifican y distribuyen de acuerdo a una distribución empírica conocida entre repositorios. El uso de diferentes tipos de material provoca una variabilidad en el valor de operación de las piezas y por tanto en la tendencia de distribución en los repositorios. Cada repositorio se asocia con un valor unitario de operación. Un pedido de los clientes utiliza piezas con un rango de valor de operación para su fabricación y los rangos de operación de diferentes pedidos se sobre ponen ocurriendo una colisión en el requerimiento de piezas. El problema es encontrar el orden de procesamiento de los pedidos y un modo de fragmentarlos sujeto al tipo de material que minimicen la cantidad de lotes de material requerido para cumplir con todos los pedidos. Este problema se interpreta como un problema de empaquetamiento fraccionable con repositorios de contenedores de tamaño variables y restricciones de asignación (*Fragmentable Objet Bin Packing with Variable Sized Container Repositories and Assignation Constrains,* FOBP-VRAC). Se presenta un modelo de programación dinámica entera. Se introducen tres algoritmos que trabajan de manera conjunta para resolver el problema: 1) el algoritmo MinLot-GA que busca la mejor permutación de pedidos; 2) el algoritmo MaxD que selecciona el mejor tipo de lote para satisfacer los pedidos; 3) el algoritmo NWP que asigna los pedidos en los lotes y resuelve el problema de colisión. Se realiza un experimento computacional para evaluar el desempeño del algoritmo genético y compararlo contra diferentes heurísticas.

Adicionalmente se plantea la resolución del problema a través de la búsqueda de la solución entre las prioridades de los pedidos como una forma alternativa. Final el algoritmo MinLot-GA mostro mejores resultados y una eficiencia aceptable para la implementación practica. Adicionalmente se propone una generalización original del problema de Johnson para dos máquinas considerando las operaciones de clasificado y líneas de forma. Este problema se analiza y se propone un algoritmo exacto para el caso particular.

Palabras clave: Planificación Scheduling, Packing, Clasificación, Procesamiento por lotes.

ABSTRACT of the thesis of Rainier Romero Parra, as partial requirement for obtaining the degree of DOCTOR IN SCIENCE, Mexicali, Baja California, Mexico February de 2013.

PRODUCTION PLANNING OF ELECTRONIC DEVICES BASED ON THE USED MATERIALS OPTIMIZATION

Summary approved by:

Dra. Larysa Burtseva
Director de Tesis

A real problem of resource planning and scheduling in manufacturing of electrical switches is investigated. The production process is carried out using lots of material. Exist several lot types and each one contains material with the same characteristics. The lot size is fixed. The investigation is focused on an operation where the pieces of every lot are classified between repositories according to a known empirical distribution. The use of different types of material causes a variability in the operation value on the pieces and therefore on the distribution tendency. Each repository is associated with a fixed range of the switch operation value. A customer order (a demand) requires pieces of a specific operation value range. They can be recollected from some successive repositories, and operating ranges of different orders are overlapping occurring collisions, which complicates the calculation of required material. The problem is to find the sequence of demands processing and an allocation way of demands on the lots of different types of material, through fragmentation of the demands, to minimize the number of lots required for fulfillment of all orders. This problem is interpreted as The Fragmentable Object Bin Packing problem with Variable Sized Container Repositories and Assignation Constrains (FOBP-VRAC). One lot is considered as a container divided in a number of repositories, a number of container types is used, the repository volumes are different, meanwhile the volume of anyone container is constant; a customer demand represents a packing object, and there exist several restrictions in the allocation of these objects on containers and repositories through object fragmentation. A dynamic integer programming model is presented. Three algorithms that work together are presented to solve the problem: 1) MinLot-GA genetic algorithm that seeks the best permutation of demands, 2) MaxD algorithm that selects the best type of lots, 3) the algorithm NWP assigning orders in the lots and solves the problem of collision.

Computational experiment is realized to evaluate the performance of the genetic algorithm. The results were compared against different heuristics. An alternative method is presented, which find the solution using priority rules. Final MinLot-GA algorithm showed better results and acceptable efficiency for practical implementation. Additionally, an original generalization of Johnson problem for two machines is proposed considering the classification and line form operations. This problem is analyzed and an exact heuristic algorithm is proposed for a particular case.

Key words: Planning Scheduling, Packing, Classification, Lot processing

1. INTRODUCCIÓN

1.1 OPERACIÓN DE CLASIFICADO EN MANUFACTURA SERIAL

En muchos sistemas de manufactura en serie los componentes de los productos se procesan por lotes (paletas, contenedores, boxes), y cada lote se compone de muchas piezas idénticas. El tamaño del lote frecuentemente es constante y los pedidos de los clientes se conocen para un horizonte de planeación.

En ambientes reales de producción, las piezas que pertenecen al mismo lote tienen alguna variabilidad en sus parámetros, por ejemplo, en nivel de calidad, características físicas o químicas, etc. Supongamos que estas piezas se utilizan para manufacturar diferentes productos, y que cada producto requiere un rango definido de parámetros, por lo tanto las piezas deben ser clasificadas de acuerdo a estos rangos.

Después de la clasificación el lote se divide en sublotes de acuerdo a los requerimientos de los productos. Cuando el tamaño de los pedidos es mayor que el tamaño del sublote, los pedidos se dividen y se asignan en varios lotes. Consecuentemente, un lote puede conceptualizarse como un *batch* (un grupo) compuesto por fragmentos de uno o más pedidos. El número de lotes necesarios para cumplir con todos los pedidos debe ser el mínimo posible.

El problema se vuelve más difícil cuando se procesan varios tipos de lotes, por ejemplo, cuando se utiliza la materia prima con diferente calidad. Esto tiene efecto directo sobre los resultados de distribución de los lotes, por lo tanto en la cantidad de las piezas obtenidas en un rango de operación deseado. La selección del tipo de lote representa un problema combinatorio complejo de planificación y programación de la producción. Esta situación se presenta en la fabricación de semiconductores y dispositivos digitales, además de la industria de cerámicas y agrícola, cuando un lote de producción es clasificado de acuerdo a la calidad, color, tamaño, etc., de las piezas que se utilizarán para diferentes propósitos.

Se examina un problema real de planificación que ocurre en la producción de interruptores eléctricos en una planta. Se utiliza materia prima de diferentes tipos en la planta, y este hecho tiene un efecto directo en los resultados de clasificado. El objetivo es minimizar la cantidad de lotes necesarios para cumplir todos los pedidos.

1.2 ORGANIZACIÓN DEL PROCESO PRODUCTIVO DE INTERRUPTORES ELÉCTRICOS

Los interruptores (*reed switches*) se utilizan en sensores eléctricos y relevadores. Los componentes de un interruptor son dos navajas de contacto ferromagnéticas (*blades* o *reeds*) de una cierta forma posicionadas en una capsula de vidrio sellada herméticamente, con una separación (*gap*) entre sí (Figura 1.1). La fabricación de los interruptores se realiza de acuerdo a los pedidos de los consumidores.

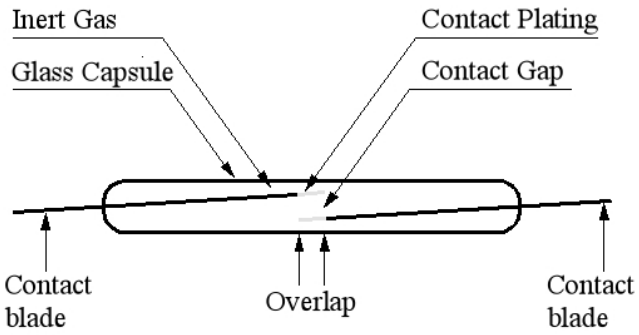

Figura 1.1 Esquema de un interruptor (reed switch)

Un pedido incluye:

- un número de parte (código del producto),
- una cantidad,
- una fecha de entrega.

El campo magnético expresado en NI o ampere-vuelta (*ampere-turns*, AT) que al aplicarse al interruptor lo acciona, se conoce como valor de operación (*operate value*). A un número de parte (*part number*) le corresponde un rango del valor de operación aceptable y la forma de las navajas. Se distinguen 4 formas de navajas. En la planta se fabrican aproximadamente 50 números de parte.

Se utilizan tubos de vidrio seleccionados entre cuatro diferentes diámetros de núcleo (tipo de vidrio) cuando se requiere obtener la mayoría de piezas en distintos rangos del valor de operación.

La ruta tecnológica por la que pasa todo interruptor se compone de 8 operaciones (etapas) (Figura 1.2):

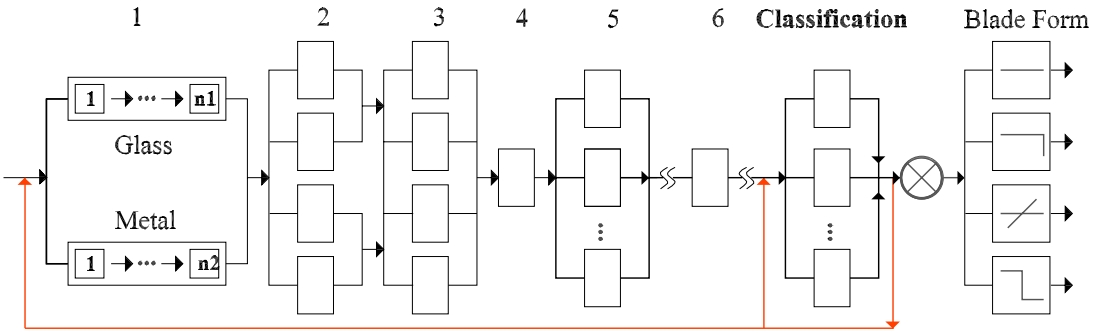

Figura 1.2 Esquema de proceso de producción

1. *Preparación de componentes, vidrio y metal*: Los tubos de vidrio y las navajas ferromagnéticas se procesan por separado. Los tubos de vidrio son lavados para eliminar cualquier impureza. Las navajas ferromagnéticas se someten a un tratamiento de recocido para endurecerlas.
2. *Ensamble de componentes*: Las navajas ferromagnéticas se introducen al tubo de vidrio para formar el interruptor. El montaje se realiza en charolas especiales que evitan que se desarme el interruptor.
3. *Sellado*: Los extremos de los tubos de vidrio se sellan con láser atrapando las navajas en el tubo.
4. *Prueba de fugas*: Los interruptores se sumergen en una sustancia reactiva bajo luz negra. Si los interruptores no están bien sellados se iluminarán al pasarlos por luz negra.
5. *Envejecimiento*: Los interruptores se introducen en unas cámaras de vibración magnética. Su función es activar y desactivar los interruptores en una frecuencia muy alta para estabilizar la operación.
6. *Estañado*: Aplicación de una capa de platinado para alterar las características físico-químicas de las navajas expuestas del interruptor.
7. *Clasificado*: En cada interruptor se mide el valor de operación para clasificar entre 25 rangos de operación.

8. *Forma de navajas y empaquetamiento*: De acuerdo al número de parte se toman los interruptores de cierto rango de operación y se aplica una de cuatro formas a las navajas. Posteriormente se empacan para su envío al cliente.

La operación de estañado se realiza de manera externa con duración de un día, dividiendo el proceso en dos partes relativamente aisladas. La investigación se enfoca en las operaciones que siguen después del estañado: el clasificado y la forma de navajas. Además de estar separadas, estas operaciones tienen varias complicaciones, que no permiten describir directamente el modelo productivo como un taller de flujo para secuenciar y procesar los pedidos, así como lo requiere la planificación de la producción.

En la operación de clasificado se identifica el valor de operación de cada interruptor, y se asigna a un repositorio (*bin*). A continuación, el rango de operación que corresponde a un repositorio se identifica por el número del último.

En la entrada de la operación, los interruptores llegan agrupados en lotes de 5500 piezas ensambladas con el mismo tipo de vidrio. Una vez que un lote comienza a procesarse, la clasificación no se detiene hasta terminar. El procedimiento se realiza por lotes en 8 máquinas idénticas (Figura 1.3).

Las operaciones anteriores a la etapa de clasificado no garantizan una magnitud exacta del valor de la operación de un interruptor. Las fuentes de las desviaciones del rango de operación son:

1. La manipulación de los materiales por los operadores entre las etapas 1 a 5.
2. Incertidumbre en la calibración de las máquinas de sellado que provoca variaciones que afectan el valor de operación.
3. Incertidumbre en la calibración de las máquinas de estañado que afecta a las propiedades físicas del material.

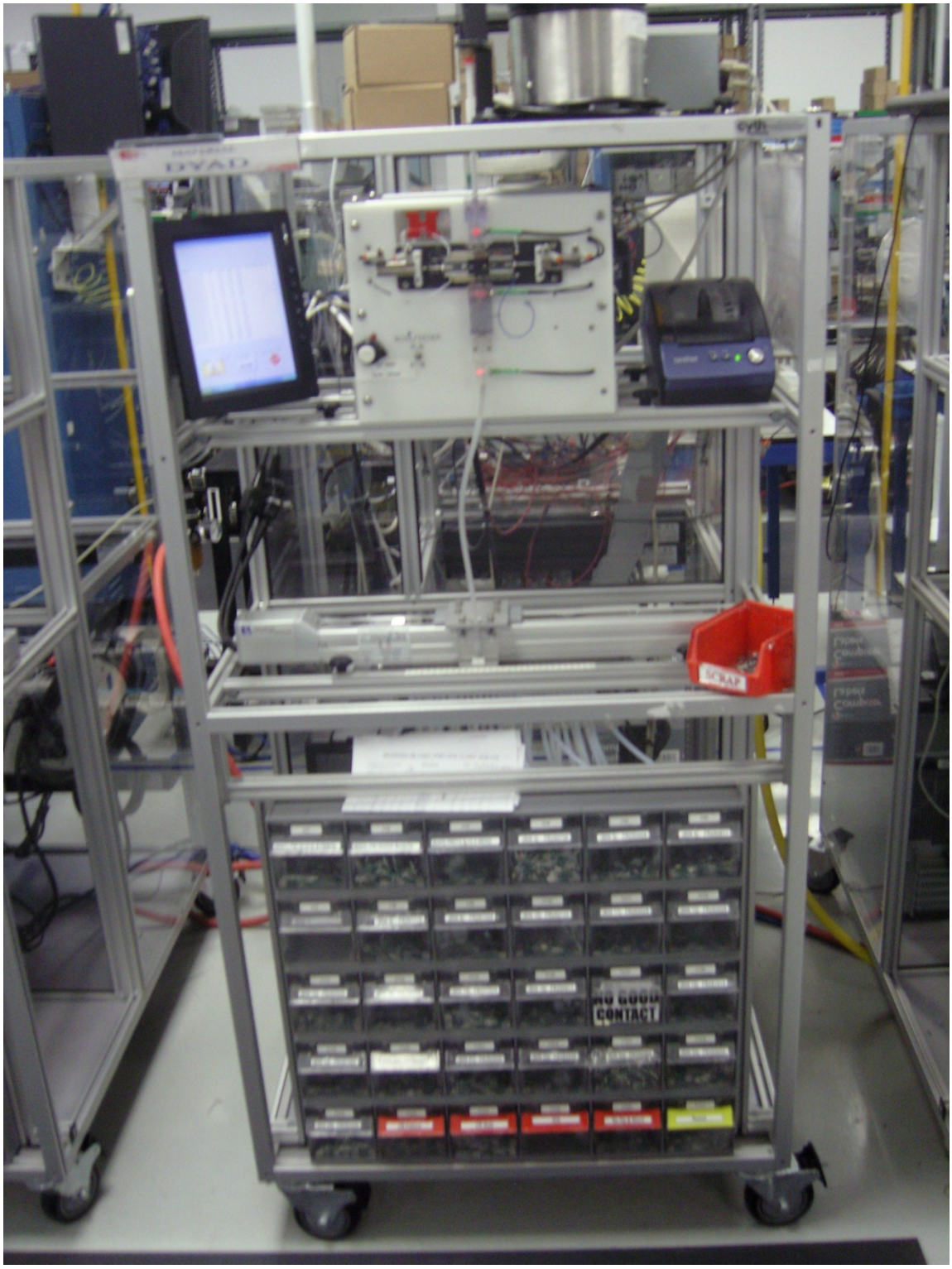

Figura 1.3 Máquina de clasificado con 25 repositorios.

Cada máquina en la etapa de clasificado tiene 25 repositorios (Figura 1.4a) enumerados de manera natural de 1 a 25, uno para cada unidad del rango de operación,

y repositorios para piezas defectuosas[1] (*scrap*). Con cada repositorio está asociado un intervalo determinístico del valor de operación, así que estos intervalos no se traslapan. De tal manera, en el primer repositorio se depositan interruptores con el valor de operación en rango [7,8) AM, en el segundo [8,9) AM, en el tercero [9,10) AM, etc. Como resultado, un lote se distribuye entre 25 repositorios de la máquina, es decir, un lote se divide (*splits*) entre 25 sublotes (Figura 1.4b).

Al terminar el lote, los repositorios se vacían directamente en recipientes de la línea de forma, donde se realiza la última etapa, para cumplir con los pedidos (Figura 1.4c). La siguiente etapa se compone de cuatro líneas de producción, una para cada forma de navajas. Un pedido se asigna a su línea de manera unívoca, de acuerdo al número de parte, ejecutándose por completo o por fracciones, a medida que estén listas a procesarse.

Las piezas sobrantes cuyos rangos de operación no fueron requeridos por los pedidos generan el llamado ”trabajo en proceso” (w*ork in process, WIP*), o “productos finales” (*finished goods*), los cuales se acumulan en el búfer de espera ilimitado de acuerdo a su rango de operación, y deben tomarse en cuenta para los siguientes pedidos (Figura 1.4*c*, 1.5). Después del clasificado el concepto de lote desaparece y el producto se maneja en piezas, puesto que el contenido de un lote se distribuye entre los repositorios.

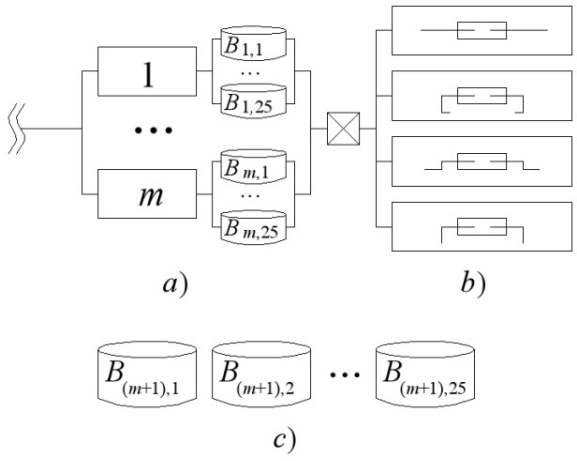

Figura 1.4 La operación de clasificación: a) Las *m* máquinas de clasificado idénticas con 25 repositorios cada una; b) Líneas de forma de navajas; c) Búferes ilimitados (WIP).

[1] Las piezas defectuosas no se consideran en la investigación.

Figura 1.5 Productos finales (finished goods)

1.3 PLANTEAMIENTO DEL PROBLEMA

La planificación de la producción permite mejorar el proceso de manufactura y elevar la productividad de la empresa optimizando el uso de material. Para aplicar sus métodos es necesario analizar los datos obtenidos sobre el procesamiento de los pedidos en las máquinas, con el propósito de obtener el modelo formal del funcionamiento de la planta.

Como el tipo de vidrio del lote influye directamente al modo de la distribución de piezas entre los repositorios, éste debe tomarse en cuenta para el cálculo de la cantidad de lotes necesaria para cumplir con un conjunto de los pedidos. La complejidad del cálculo de cantidad de lotes proviene de las consideraciones siguientes:

1. Para que la planta funcione de manera eficiente, la cantidad de lotes para cumplir con los pedidos programados a ejecutar debe ser la mínima posible.

2. La cantidad de piezas que se deposita a un repositorio a partir de un lote depende del tipo de vidrio utilizado en la fabricación de piezas. Es apropiado utilizar esta propiedad para maximizar la cantidad de piezas en el rango de operación deseable seleccionando el tipo de vidrio más apropiado para cada pedido, y así minimizar la cantidad total de lotes.
3. Además de piezas con el rango de operación necesario para cumplir con un pedido, en el resultado del clasificado de un lote se obtienen piezas con otros rangos de operación, los cuales de alguna manera deben tomarse en cuenta para otros pedidos.
4. En el caso general, a un número de parte le corresponden varios repositorios de una máquina. Por lo tanto, hay que tomar en cuenta no solo el contenido de cada repositorio, sino su capacidad sumaria, lo que complica los cálculos.
5. Los mismos repositorios se utilizan para cumplir con distintos pedidos.

Analizando la parte final de la ruta tecnológica en una planta que elabora interruptores se concluye que la complejidad de la información acerca de los tiempos de procesamiento de los pedidos en las máquinas no permite secuenciar y planificar el proceso de manufactura directamente en términos de talleres de flujo puesto que involucra:

- Distintas medidas del producto: lote, pieza, pedido, los cuales se transforman uno al otro en el proceso de la producción dificultando la aplicación del concepto del trabajo, apropiado en la planificación de la producción;

- Varias características del material utilizado y el producto, relacionadas entre sí, como el número de parte, tipo de vidrio, rango de operación, rango de repositorio;

- Procesamiento de un pedido por partes de tal manera que al mismo tiempo éstos se encuentran en varios repositorios de una o más máquinas, en búfer ilimitado como WIP y terminados.

De una manera empírica se observa que la cantidad de piezas depositadas a un repositorio a partir de un lote, depende del tipo de vidrio utilizado, de tal manera que lote con un tipo de vidrio produce la mayoría de las piezas en un cierto rango de

operación y el resto se distribuye entre otros rangos. Por lo tanto es imposible manejar la distribución como uniforme.

El incumplimiento de pedidos en los tiempos de entrega produce pérdidas económicas y desorganización del proceso productivo.

Debido a la complejidad de la información, la programación de la producción en la empresa se realiza de manera empírica, basándose en la experiencia anterior, y por tanto sufre de resultados inesperados al finalizar la producción.

El problema principal de la empresa es cumplir con los pedidos de clientes de acuerdo con sus fechas de entrega. La selección de los pedidos con la misma fecha para la programación de su procesamiento permite cumplir esta condición concentrándose en la minimización del tiempo de ejecución.

1.4 OBJETIVOS

Objetivo general:

Optimizar el uso de material en la producción de dispositivos electrónicos, a través del análisis del proceso productivo y el desarrollo de algoritmos basados en programación matemática y técnicas metaheurísticas.

Objetivos específicos:

1. Minimizar el número de lotes de material necesarios para cumplir con los pedidos.
2. Definir el orden de procesamiento de los lotes tomando en cuenta diferencias en las características del material.
3. Desarrollar un método que permita dividir los pedidos y asignar los fragmentos en los repositorios y lotes.

1.5 METODOLOGIA

1. Analizar el proceso productivo, recoger datos reales y desarrollar un modelo de la producción.
2. Realizar un análisis estadístico enfocándose en la distribución de lotes entre los repositorios en la etapa de clasificación.
3. Realizar estado del arte sobre los problemas computacionales de planificación que involucran el procesamiento por lotes.
4. Formalizar el problema y desarrollar un modelo matemático.
5. Formular el problema como un problema de ciencias computacionales y evaluar su originalidad.
6. Evaluar la complejidad computacional y desarrollar algoritmos que solucionen el problema.
7. Verificar el funcionamiento de los algoritmos desarrollados a través de un experimento computacional y un análisis estadístico de resultados.

1.6 ESQUEMA GENERAL DE LA TESIS

El resto del documento se organiza como sigue:

En el capítulo 2 se exponen temas de la teoría de planificación y programación de la producción relacionados con el problema y algunos conceptos necesarios de procesamiento por lotes.

En el capítulo 3 se describe el problema de cumplimiento de pedidos en la producción de interruptores eléctricos. Se detalla y analiza el proceso de clasificado para establecer patrones de distribución de los lotes de material entre los repositorios de las máquinas. Se describe la complejidad de minimizar el uso de material (lotes) debido a que diferentes pedidos comparten la utilización de ciertos repositorios, además de que existen varios tipos de material que afectan la forma en que las piezas se distribuyen entre los repositorios. Finalmente se analiza el problema para una partición de trabajos establecida y se presentan algunos lemas que reducen al problema clásico de Johnson para dos máquinas. El teorema que concluye el capítulo permite encontrar la solución óptima de la generalización obtenida del problema de Johnson para el caso considerado.

En el capítulo 4, el problema de minimización de lotes se interpreta en términos del problema de empaquetamiento (*bin packing*) realizando un análisis en la literatura relacionada con la presente investigación. Se considera por primera vez un grupo de características que nunca antes han sido abordadas juntas. Además se presentan las notaciones, suposiciones y modelo matemático que describen el problema. Finalmente se proponen tres algoritmos originales que de una manera integral solucionan el problema: dos algoritmos heurísticos y un genético. Estos algoritmos proporcionan solución a los problemas: la asignación de pedidos a los lotes y la división de pedidos, la selección del tipo de lote y la minimización del número de lotes. Adicionalmente se propone una algoritmo alternativo que busca la solución de los mismos problemas considerando varias prioridades de los trabajos a procesar.

En el capítulo 5 de describe el experimento computacional donde se calibra el algoritmo genético con el fin de establecer los parámetros más adecuados para su ejecución. Posteriormente se diseña un segundo experimento para comparar el desempeño del algoritmo genético con cuatro algoritmos heurísticos y se presentan los resultados.

Finalmente en el capítulo 6 se presentan las conclusiones de la investigación, las contribuciones y el trabajo futuro.

2. PARTICULARIDADES DE LA PLANIFICACIÓN EN SISTEMAS DE MANUFACTURA MODERNOS CON PROCESAMIENTO POR LOTES

La producción por lotes es típica en los sistemas de manufactura modernos, especialmente en la manufactura de semiconductores, debido al nivel alto de estandarización en el diseño de productos y volúmenes grandes de los pedidos. A continuación se consideran algunos modelos, métodos y conceptos relacionados con el tema de investigación.

2.1 PLANIFICACIÓN DE LA PRODUCCIÓN EN TALLERES DE FLUJO HÍBRIDO

2.1.1 Modelos de talleres de flujo híbrido

En la planificación de la producción, existen procesos en los que los productos deben pasar por un número de etapas en el mismo orden, a este tipo de proceso se le conoce como taller de flujo (*flow shop*). En un taller de flujo simple (*simple flow shop*), cada etapa cuenta con una sola máquina, la cual procesa una operación a la vez. Para evitar el llamado "efecto de cuello de botella" y elevar la productividad, frecuentemente se aplican los modelos de los talleres en los cuales cada etapa está conformada por un número de máquinas que operan en paralelo. A este modelo se le conoce como taller de flujo híbrido (*Hybrid Flow Shop*, HFS por sus siglas en inglés) (Ruben Ruiz y Vázquez-Rodríguez, 2010). Es posible que en alguna de las etapas exista sólo una máquina pero un taller se clasifica como HFS cuando en al menos una de las etapas se cuenta con por lo menos 2 máquinas que trabajan en paralelo. Los trabajos que se procesan en este modelo se ejecutan en una de las máquinas en cada etapa y el flujo ocurre en una dirección.

Los modelos de HFS son comunes en las industrias que tienen la misma ruta tecnológica para todos los productos como una secuencia de etapas, donde algunas etapas tienen un grupo de máquinas capaces de realizar la misma operación. Varios procesos industriales, tales como metalúrgicos, químicos, farmacéuticos, petroleros, alimenticios, manufacturas de semiconductores, tableros de circuito impreso (*printed*

circuit board, PCB), automóviles, entre otros, se modelan como un HFS. En tales industrias, las instalaciones de equipo son paralelizadas para incrementar la productividad total, o para balancear las capacidades de las etapas, eliminar o reducir el impacto de etapas con cuello de botella relacionados con las capacidades de taller (R. Ruiz et al., 2008).

Un típico HFS tiene las siguientes características:

- Existen k etapas de procesamiento que ocurren en un orden lineal: 1, 2, …, K, para procesar n trabajos. Cada etapa k tiene un número predeterminado m_k de máquinas en paralelo, y el número de máquinas varía de etapa a etapa, m_1,…, m_K.
- Cada trabajo visita las etapas en orden secuencial, aun cuando no requieren ser procesados por todas las etapas. Un trabajo puede saltar alguna etapa particular, pero el flujo de trabajos por etapas es el mismo.
- El tiempo de procesamiento de un trabajo se conoce de antemano al procesarse en cualquier máquina y es constante para la máquina dada.
- Un trabajo no se divide entre las máquinas, y una máquina procesa no más que un trabajo en el mismo tiempo. Cuando se inicia una operación en una máquina, no se interrumpe hasta terminar.
- Típicamente, el orden de procesamiento de trabajos no tiene una relación de precedencia.
- Entre las etapas se disponen buffers ilimitados para guardar productos intermedios.

2.1.2 Descripción formal

Para describir todos los detalles de un HFS, se utiliza la notación $\alpha|\beta|\gamma$, llamada *tripleta de Graham* (Pinedo, 2008). El campo α denota la configuración del taller, incluyendo el tipo de flujo y el ambiente de las máquinas por etapas. El campo α se descompone en cuatro parámetros, es decir α_1, α_2, α_3, y α_4, que aparecen como $\alpha_1\alpha_2$ ($\alpha_3\alpha_4^{(1)}$, $\alpha_3\alpha_4^{(2)}$, …, $\alpha_3\alpha_4^{(\alpha 2)}$). Aquí el parámetro α_1 indica el tipo de taller considerado, el parámetro α_2 muestra el número de etapas. Para la notación de HFS, en la posición α_1 se denota HF, y el valor del parámetro α_2 debe ser mayor a uno. En cada etapa, los parámetros α_3 y α_4

indican el ambiente del conjunto de máquinas. Más específico, α_3 denota información acerca de tipo de máquinas, mientras α_4 indica el número de máquinas por etapas.

Los posibles desarrollos del conjunto de máquinas en la etapa *k* de un HFS son los siguientes:

- *Una máquina* (1): algunas etapas (no todas) en un HFS tienen una sola máquina;
- *Máquinas idénticas en paralelo* (Pm_k): el trabajo *j* puede ser procesado en cualquiera de m_k máquinas; el tiempo de procesamiento es equivalente;
- *Máquinas uniformes en paralelo* (Qm_k): las m_k máquinas de la etapa tienen diferentes velocidades; cualquier máquina de la etapa es capaz de procesar un trabajo *j*, mientras el tiempo de procesamiento es proporcional a la velocidad de la máquina;
- *Máquinas no relacionadas en paralelo* (Rm_k): existe un conjunto de m_k máquinas diferentes en paralelo. El tiempo que un trabajo se queda en la máquina depende tanto del trabajo como de la máquina;
- *Máquinas dedicadas en paralelo* (Dm_k): para cada máquina se conoce de antemano un subconjunto de trabajos que procesa.

Cuando existen varias etapas consecutivas con el mismo ambiente de máquinas en diferentes etapas, los parámetros α_3 y α_4 se agrupan como $((\alpha_3\alpha_4)^k)_{k=s}^{i}$, donde *s* e *i* son los índices de la primera y la última etapas consecutivas, correspondientemente. Por ejemplo, la notación $FH4, (1, P2^{(i)})_{i=2}^{3}, Q3^{(4)})$ refiere a una configuración de HFS con cuatro etapas, donde una sola máquina se encuentra en la primera etapa, dos máquinas paralelas idénticas en la segunda y tercera etapas, y tres máquinas uniformes en la cuarta etapa.

El campo β incluye las propiedades del taller, los detalles y condiciones del procesamiento, también puede ser vacío si estos no existen. A continuación se describen algunas propiedades asociadas frecuentemente con un problema de planificación en un HFS con el procesamiento por lotes (el campo β):

batch(*b*) Procesamiento de trabajos en batches. Una máquina es capaz de procesar hasta *b* trabajos simultáneamente, es decir, en el mismo tiempo, y sin ningún ajuste.

brkdown	Interrupción (*breakdown*) en una máquina. Implica que una máquina puede no estar disponible continuamente.
M_{jk}	Restricciones de elegibilidad de la máquina. El procesamiento del trabajo j es restringido con un conjunto M_j de las máquinas en la etapa k.
s_{in}	En taller existen tiempos de ajuste de las máquinas independientes de la secuencia de trabajos.
s_{sd}	Los tiempos de ajuste de las máquinas son dependientes de la secuencia de trabajos.
w_j	Factor de prioridad. Denota el peso, o importancia, de un trabajo j con relación a otros trabajos en el sistema.

El campo γ establece el objetivo a minimizar. Las funciones objetivo más comunes en un problema de planificación para HFS son:

C_{max} – el tiempo máximo de cumplimiento (*makespan*);

F_{max} - el tiempo máximo de flujo de un trabajo (*flow time*);

L_{max} - el retraso (*lateness*) máximo;

T_{max} – la tardanza (*tardiness*) máxima;

E_{max} – el anticipado (*earliness*) máximo;

El criterio más utilizado para un problema de planificación en un HFS, es el tiempo de cumplimiento, el cual indica el momento cuando el último trabajo abandona el sistema, generalmente se refiere a éste como *makespan*, o C_{max}.

Un problema estándar de planificación en un HFS con K etapas y un número $M^{(k)}$ de máquinas paralelas idénticas en la etapa k, $k = 1, \ldots, K$, se denota como FHK, $((PM^{(k)})_{k=1}{}^{K})||C_{max}$. En este caso, el campo β es vacío, y el objetivo es el minimizar makespan.

La Figura 2.1 muestra la relación física entre máquinas y etapas, que corresponde a la notación $FH3$, $(1, P3^{(2)}, R2^{(3)})$ $|M_{j3}, s_{sd}|C_{max}$, refiriendo a un HFS de tres etapas. La etapa uno tiene una sola máquina, la etapa 2 tiene tres máquinas idénticas en paralelo, y

la etapa 3 tiene dos máquinas no relacionadas; M_{j3} y s_{sd} indican que en la etapa 3 existen las restricciones de elegibilidad de máquinas y los tiempos de ajuste son dependientes de la secuencia de trabajos. El objetivo es minimizar makespan. La Figura 2.1 muestra además que existen buffers ilimitados entre las etapas para el almacenamiento del WIP.

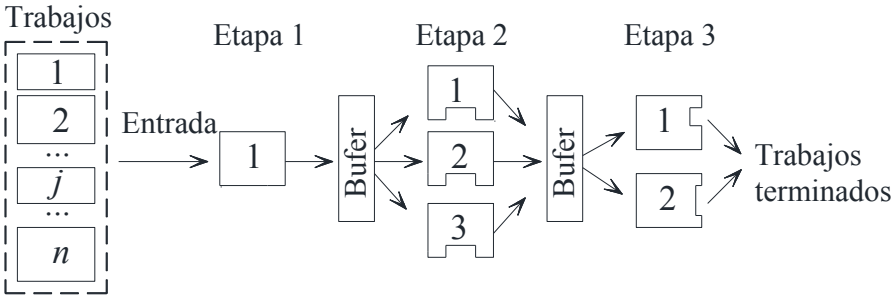

Figura 2.1 Un ejemplo que ilustra un modelo de un TFH con tres etapas.

2.1.3 Flexibilidad de un taller de flujo hibrido

Un sistema de producción debe ser *flexible* para clasificarse como un HFS. Es importante conocer la diferencia entre un sistema flexible de producción y uno tradicional, es decir, definir exactamente el concepto de flexibilidad y justificar el uso de modelos especiales para sistemas flexibles de la producción. Los sistemas automatizados de manufactura muestran la flexibilidad en modos múltiples y entrecruzados, pertinentes al equipo, procesos, productos, volúmenes de producción, etc. Entre los más importantes conceptos se encuentran los siguientes (Crama, 1997), (Vairaktarakis, 2004):

- *flexibilidad de máquinas* (*machine flexibility*), es la habilidad de las máquinas para realizar varios tipos de operaciones sin requerir un ajuste costoso cuando se cambia de una operación a otra;
- *flexibilidad en manejo de material* (*material flexibility*), es la habilidad del sistema de manejar los materiales moviendo eficientemente diferentes partes de material para la colocación propia y el procesamiento a través de facilidades de manufactura;

- *flexibilidad de operaciones* (*operation flexibility*), es la habilidad de realizar las mismas operaciones en diferentes modos;
- *flexibilidad del procesamiento* (*processing flexibility*), se trata de la capacidad de identificar trabajos que saltan etapas, o existe un conjunto de tipos de partes que se realizan con la misma configuración de máquinas;
- *flexibilidad en asignación de ruta* (*routing flexibility*), es la habilidad de los sistemas de manufactura para procesar una parte de productos utilizando rutas alternativas.

Un modelo de planificación de la producción con conjuntos de máquinas en paralelo debe cumplir con uno de estos conceptos para ser clasificado como un taller de flujo flexible, tomando en cuenta que el flujo de productos en la planta es unidireccional. La *hibridación* ocurre cuando algunos productos requieren condiciones especiales para su fabricación, por ejemplo, diferentes calidades o capacidades de máquinas en la misma etapa, asignación de algunos trabajos a ciertas máquinas, entre otras.

2.1.4 Otros modelos de HFS

Mientras el HFS ha sido bien investigado desde los años 70, los investigadores prestan mucha atención a este modelo, y han sido descubiertos varios nuevos diseños en años recientes. Este hecho, probablemente, implica confusiones en la terminología y notaciones. Actualmente, no existe en la literatura una clasificación convencional de esta clase de talleres de flujo. Se recomienda interpretar como un HFS una variedad de modelos conocidos o sus casos especiales. Estos son:

Taller de flujo flexible (*Flexible flow shop*, FFS por sus siglas en ingles); es un HFS con máquinas paralelas en el que las máquinas en cada conjunto son idénticas (flexibilidad de procesamiento en cada etapa de producción, la cual esta derivada desde la habilidad de procesar un trabajo en cualquier máquina de la etapa). Algunos autores reconocidos, como, por ejemplo, Pinedo (2008), Jungwattanakit et al. (2009), no utilizan la notación de HFS, y describen las más complejas configuraciones como un FFS con máquinas no idénticas al menos en una etapa. Más aun, una variedad de autores no difieren entre términos de FFS y HFS, refiriéndose a este modelo como un

Taller de flujo flexible (*hibrido*) (Allahverdi et al., 2008) o utilizan el término de HFS en un desarrollo con máquinas paralelas idénticas (Naderi et al., 2009).

Línea de flujo flexible, o *Taller de flujo con múltiples procesadores* (*Flexible flow line,* FFL; *Flow shop with multiple processors***,** FSMP, MPFS) son equivalentes a un FFS (Lin y Liao, 2003). Zandieh et al. (2006) consideran que un modelo de HFS se conoce como una línea de flujo flexible, debido a que el flujo de trabajos en este sistema es unidireccional.

Taller de flujo hibrido flexible o *Línea de flujo hibrido flexible* (*Hybrid flexible flow shop or Flexible hybrid flow line* (HFFL); este modelo es equivalente a un HFS donde los trabajos pueden saltar etapas (*flexibilidad de procesamiento* atravesando etapas) (Ruben Ruiz y Vázquez-Rodríguez, 2010), (Allahverdi et al., 2008).

Los sistemas de HFS paralelos (*Parallel HFS system*, PHFS) representa un HFS descompuesto en un número de menores HFS, los cuales son sub-diseños operados en paralelo. Más específico, un PHFS es compuesto de un número de sub-diseños independientes, cada uno de los cuales es un HFS con una ruta unidireccional (*flexibilidad en asignación de ruta*) (Vairaktarakis, 2004).

Un problema de HFS, restringido con dos etapas de procesamiento, hasta el caso más simple, cuando una etapa contiene dos máquinas paralelas idénticas y una sola máquina en la segunda etapa, ya es NP-difícil (NP-*hard*) de acuerdo a (Gupta, 1988). Más aún, un caso particular cuando existe una máquina en la etapa, conocido como un taller de flujo (*simple flow shop*), y el caso más simple, cuando existe una sola etapa con varias máquinas, conocido como un desarrollo con máquinas paralelas idénticas, también son NP-difícil (Glover y Laguna, 1997). La complejidad de problemas de la planificación en un HFS es esencialmente más alta cuando se trata de procesos estocásticos.

2.2 PROCESAMIENTO DE TRABAJOS EN GRUPOS

2.2.1 Tecnología de grupos

Los productos que se manufacturan en una planta con frecuencia tienen algunas similitudes técnicas; por lo tanto, es posible clasificar estos productos en grupos, de acuerdo a su diseño o atributos manufactureros, tales como la forma de una detalle, el tamaño, la textura de superficie, el tipo de material, la materia prima, etc. Las similitudes técnicas de los productos dentro de un grupo permiten reducir el número de ajustes en una máquina. Consecutivamente, el tiempo de procesamiento decrece y el uso de máquina mejora.

Esta idea ha sido adoptada en la Tecnología de Grupos (*Group Technology*, GT). La GT es un enfoque solicitado en la administración de manufactura e ingeniería para lograr la eficiencia en la producción de volúmenes grandes explotando las similitudes de diferentes productos, así como las actividades en su producción/ejecución (Chen et al., 2007). El concepto de GT está basado en la simplificación y la estandarización de procesos, y de acuerdo a Burbidge (1975), apareció en el principio del siglo XX. Numerosas compañías manufactureras tomaron beneficio del GT para elevar la productividad y competitividad; para referencias ver, por ejemplo, Wemmerlöv y Hyer (1989), Tatikonda y Wemmerlöv (1992), Hadjinicola y Kumar (1993), Gunasekaran et al. (2001). La primera publicación sobre la planificación en los ambientes con el uso de la GT se debe a Petrov (1966).

La GT originalmente surge como un concepto relacionado con una sola máquina que ha sido creado para reducir los tiempos de ajuste (Mitrofanov, 1966). Después se extendió hasta el problema del HFS con tiempos de ajuste dependientes de la secuencia de trabajos (Li, 1997). Andrés et al. (2005) introdujo el concepto del *coeficiente de la similitud* entre cada uno de los productos; originalmente tuvo el rol de un parámetro, permitiendo agruparse a los productos utilizando un método heurístico, en contra del concepto básico de *explotación de similitudes* (tomado de la filosofía de la GT). El primer concepto permite a los ingenieros diseñadores salvar los diseños existentes para el soporte de estandarización en el diseño de nuevas partes, así como hacer una exacta estimación de los costes. El segundo produce mejoras en el proceso de control, así como la reducción de tiempos de ajuste y planes de procesos estandarizados (Kusiak, 1987).

Del paradigma GT surgen dos conceptos importantes: familia de productos y batch. Los trabajos (*jobs*) son supuestos de estar divididos en *F familias*, $F \geq 1$. Un *batch* es un conjunto de trabajos de la misma familia que se procesan conjuntamente (Brucker, 2004). *Batching* ocurre sólo si costes de ajuste, o tiempos, son no insignificantes; además, varios trabajos de la misma familia tienen que producirse. Un batch se llama *factible* (*feasible*), si se procesa sin un cambio de herramientas. El tiempo del procesamiento depende solamente de la familia del batch. Cuando el procesamiento se realiza por batches de unidades idénticas (*lotes*), las operaciones se ejecutan simultáneamente. Así el tiempo de completar todos los trabajos en un batch es el tiempo de finalización del último trabajo. Una vez el procesamiento de un batch se inició, no se interrumpe; tampoco se permite agregar un trabajo en el batch. La motivación para el batching de trabajos es de ganar en la eficiencia: el procesamiento de trabajos en un batch es más barato o más rápido en comparación con el procesamiento individual (Chirs N. Potts y Kovalyov, 2000).

El término de familia denota una partición inicial de trabajos, mientras el término batch denota la parte de la solución. La tarea de calcular el tamaño de un batch es decidir, cuántas unidades deben ser procesadas consecutivamente. En el trabajo de C.-Y. Liu y Chang (2000) está indicado que los tamaños de batches deben ser optimizados, puesto que el procesamiento de trabajos en los batches largos incrementa la utilización de máquina y se reduce el tiempo total de ajuste; sin embargo, crece el tiempo de flujo. Por lo tanto, tiene sentido una discusión sobre un equilibrio entre el tiempo de flujo y la utilización de máquina a través de la selección del tamaño del batch. De acuerdo al GT, una familia no debe ser dividida, sino un único batch se forma para cada familia.

Muchas publicaciones utilizan el término batch para denotar una partición inicial de trabajos, y emplean diferentes términos como subbatch, lote, sublote, etc., para denotar un conjunto de trabajos de la misma familia que se procesa consecutivamente en la misma máquina. En el lenguaje español, no existe diferencia entre los términos “lote” y “batch”, mientras en la literatura especializada estos son dos conceptos diferentes.

2.2.2 Modelos de batching

Los modelos del ajuste por batches se dividen en los modelos de la *disponibilidad por batches* (*batch availability*) y los de la *disponibilidad por trabajos* (*job availability*) (Chirs N. Potts y Kovalyov, 2000). De acuerdo al modelo de disponibilidad por batches, todos los trabajos en el mismo batch juntos reciben la disponibilidad para el procesamiento, así como abandonan la máquina. Por ejemplo, tal situación ocurre si los trabajos en un batch son colocados en una paleta, y la paleta se remueve de la máquina sólo cuando todos estos trabajos ya son procesados. Una suposición alternativa es la disponibilidad de trabajos, generalmente conocida en la literatura como la *disponibilidad por unidades* (*item availability*), en la cual un trabajo recibe disponibilidad inmediatamente después de completar el procesamiento, y los tiempos de terminación son independientes de otros trabajos en el batch.

De acuerdo a Lushchakova y Strusevich (2010), el tiempo de procesamiento de un batch se calcula como sigue:

- En el *batching serial* (*serial batching*), también conocido como *s-batch* ó *sum-batch*, es equivalente al total de los tiempos de procesamiento de sus trabajos.
- En el *batching paralelo* (*parallel batching*), también conocido como *p-batch* ó *max-batch*, es equivalente al tiempo más largo de procesamiento de sus trabajos.

Cuando los tamaños de los trabajos son considerados, tal caso usualmente recibe el nombre de un problema con *batches restringidos* (*bounded batch*) si los tamaños totales de los trabajos, que contiene un batch, no exceden a la capacidad del mismo, por ejemplo, $b > n$. Como cada trabajo puede tener un tamaño diferente, el número de trabajos en cada batch también puede ser diferente. Por otro lado, si se permite incluir en un batch cualquier número de trabajos, tal problema se llama *batch no-restringido* (*unbounded batch*), respectivamente, $b \leq n$. En este caso, los batches no son restringidos en procesamiento con cualquier número de trabajos (Yazdani y Jolai, 2010).

Cuando un batch se completa, se debe ajustar el recurso para el siguiente batch. El tiempo necesario para las actividades de ajuste dependen de las familias de ambos batches adyacentes.

2.3 DIVISION DE TRABAJOS

En la mayoría de estudios de la planificación, asociados con procesamiento de productos, un lote de producción se trata como una entidad llamada *trabajo* (*job*) que consiste de una sola parte. Usualmente se supone que la interrupción de trabajos o su transferencia parcial en subsiguientes máquinas es imposible (Glass et al., 1994). Como resultado, un plan no se puede mejorar mientras se tienen los tiempos muertos en las máquinas; es decir, la transferencia parcial de los trabajos procesados a la siguiente máquina se considera imposible. Como los lotes de producción con frecuencia son largos, las piezas ya procesadas deben esperar un tiempo largo en los búferes de salida de la máquina mientras la máquina siguiente puede ser desocupada. Tal situación lleva al WIP excesivo en inventario entre las máquinas y alarga el tiempo de cumplimiento.

Cuando las suposiciones mencionadas son relajadas en el problema de planificación correspondiente, por ejemplo, cuando se supone que un trabajo es divisible, es posible mejorar la calidad del plan resultante. *División de trabajos* (*job splitting*) refiere ruptura de un trabajo dado (lote, pedido) en sublotes más pequeños durante la producción de la manera que al ser dividido, un trabajo se fragmenta en sublotes continuos y se procesa simultáneamente en diferentes máquinas (K. R. Baker y Trietsh, 2009), (Tahar et al., 2006).

Como apuntan C. N. Potts y Van Wassenhove (1992), existen dos ventajas principales de división de trabajos en sublotes:

1. Se mejora el servicio de consumidores; cada sublote puede ser entregado al consumidor inmediatamente después de completarse, sin detenerse, sin esperar los sublotes restantes del mismo trabajo.
2. En un sistema de producción multi-etapa se reduce el makespan, el cual es tiempo cuando todo procesamiento es terminado en el taller.

Más aún, el tiempo del flujo, el inventario WIP, el almacenamiento interno con requerimientos de espacio, y el sistema de mantenimiento de materiales se reducen (Truscott, 1986). Las soluciones se implementan con un menor tiempo y costos más bajos en comparación con la reorganización.

De tal manera, si un trabajo consiste en el procesamiento de piezas idénticas, este puede ser dividido en las cuatro situaciones siguientes:

1) Preferencias (*preemptions*) son permitidas;

2) Procesamiento simultáneo (independiente) de sublotes del mismo trabajo es permitido en máquinas paralelas;

3) Técnicas de *lote continuo* (*lot streaming*) se aplican en un sistema de producción multi-etapa;

4) De una manera natural, en resultado del procesamiento de un lote, por ejemplo, después de la clasificación de piezas.

Este agrupamiento se basa en los trabajos de K. R. Baker y Trietsh (2009), K. R. Baker y Pyke (1990), y Xing y Zhang (2000). El cuarto caso de la división de trabajos es considerado por la primera vez en esta investigación.

La *preferencia* significa el admitir la interrupción del procesamiento de un trabajo para otro más urgente (K. R. Baker y Pyke, 1990), y de tal manera, un trabajo es dividido. En realidad, los conceptos de preferencia y división de trabajos son diferentes, puesto que las secciones del trabajo no pueden ser procesados simultáneamente si sólo la preferencia es permitida (Tahar et al., 2006).

Cuando los requerimientos de procesamiento de un trabajo son considerados como un pedido total de un producto en la planificación de la producción, los trabajos pueden ser divididos arbitrariamente en continuos sublotes y procesados independientemente en *m* máquinas para terminar el procesamiento de todas las demandas lo más rápido posible (Xing y Zhang, 2000).

2.4 LOTE CONTINUO

En muchas situaciones prácticas, la división de un lote de producción es posible y deseable. Cuando un lote se divide en un número de sublotes, cada sublote puede ser procesado por una máquina incluso si otros sublotes todavía no son procesados por las máquinas anteriores. Diferentes sublotes del mismo trabajo se encuentran procesando simultáneamente en diferentes etapas. En resultado del traslape de operaciones, la producción se acelera notablemente y los tiempos muertos en las máquinas se reducen.

Existen varias definiciones de *lote continuo* (*lot streaming*) en la literatura (Çetinkaya y Duman, 2010), (Pott y Baker, 1989), (K. R. Baker y Jia, 1993), (Glass et al., 1994), (Zhang et al., 2005). Sumando las características mencionadas anteriormente,

definimos lote continuo como un concepto y una práctica que se refiere a la división de un lote entero de la producción (*process batch*) en sublotes (*transfer batches*) y la planificación de estos sublotes de una manera traslapada, para acelerar el progreso en la producción. Un trabajo es definido aquí como una orden de producción (lote) compuesto de muchas piezas idénticas (Pott y Baker, 1989), (K. R. Baker y Jia, 1993), (Çetinkaya y Duman, 2010).

Este concepto es distinto del concepto de preferencias, puesto que involucra cada trabajo en particular: más que dar la prioridad al otro trabajo, el objetivo es expedir el trabajo (K. R. Baker y Trietsh, 2009). El lote continuo es aplicable solo si el número de etapas en el taller es no menor a dos y se considera como un tipo especial de división de trabajos (K. R. Baker y Trietsh, 2009).

Generalmente, el *makespan* se minimizará incluyendo si un lote contiene una sola pieza (Vickson y Alfredson, 1992). Sin embargo, existen consideraciones prácticas cuando no es deseable tener un número grande de sublotes con tamaño de una unidad. En el problema de lote continuo se busca un compromiso entre tamaño de batches de proceso y sublotes cuando los ajustes (*setups*) son largos y complejos.

El *problema de lote continuo* combina decisiones de tamaño de lote (*lot sizing*), así como la planificación, los cuales tradicionalmente se tratan por separado (Zhang et al., 2005), (K. R. Baker y Jia, 1993). Es una integración de las siguientes decisiones:

- el número óptimo de sublotes para cada trabajo;
- el tamaño óptimo de cada sublote;
- la secuencia óptima para el procesamiento de sublotes, tal que se minimice el tiempo de producción.

Las ideas de Zhang et al. (2005) sobre el problema de lote continuo pueden ser extendidas hasta el problema de división de trabajos.

En la Figura 2.2 se muestra un ejemplo del beneficio de lote continuo para un taller de flujo de tres máquinas, el único trabajo con los tiempos de procesamiento de 6, 3 y 6 (Pan et al., 2011). Si el trabajo no se divide en sublotes, el tiempo de cumplimiento del trabajo es de 15 unidades (Figura 2.2a). Cuando el trabajo es dividido en tres sublotes, y no se permiten tiempos muertos entre cualesquiera dos etapas adyacentes, el tiempo

de cumplimiento se reduce a 11 unidades (Figura 2.2b), mientras en caso de permitir los tiempos muertos, el tiempo de cumplimiento de trabajo se reduce más hasta 9 unidades (Figura 2.2c). Es claro que el tiempo de cumplimiento del trabajo en caso de permitir los tiempos muertos es más corto que en el caso de no permitirlo para el mismo tamaño de lote. Sin embargo, existen muchas aplicaciones prácticas para el lote continuo en el taller de flujo, donde los tiempos muertos no se permiten.

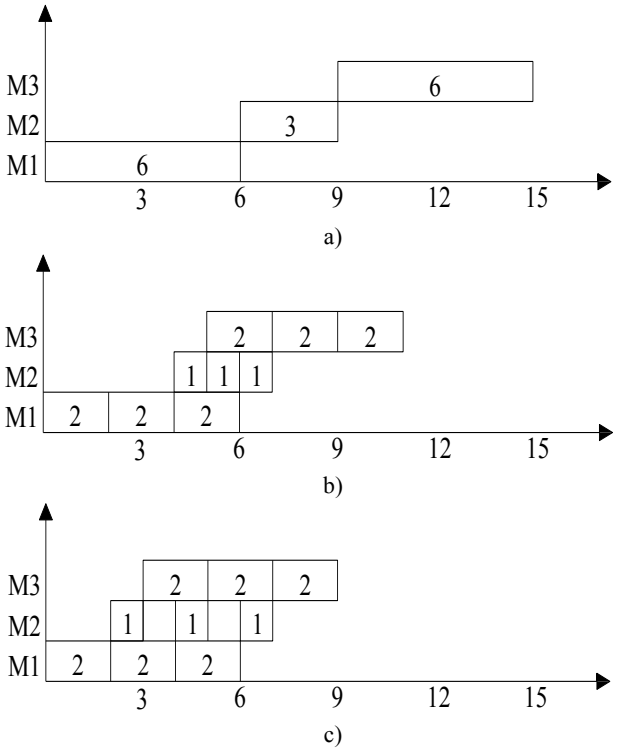

Figura 2.2. Un ejemplo de la planificación el taller de flujo: a) planificación sin sublotes; b) planificación con sublotes en caso si los tiempos muertos no se permiten.; c) planificación con los tiempos muertos.

El concepto y la práctica de lote continuo, donde se permite el traslape de operaciones, no son nuevos. El término de lote continuo (*lot streaming*) fue introducido por Reiter (1966). Graves y Kostreva (1986) definieron la idea de operaciones traslapadas (*overlapping operations*) en sistemas de la planificación con requerimientos de materiales (*material requirements planning systems*, MRP).

El uso de batches de transferencia (*transfer batches*), o sublotes, es un elemento clave en la manufactura sincronizada (Umble y Srikanth, 1990). En las dos últimas décadas, con el incremento del interés en *just-in-time* (JIT) y filosofías de tecnologías de producción optimizadas (*optimized production technology*, OPT) en manufactura, la

aplicación de idea de lote continuo en problemas de planificación recibió una atención considerable. El enfoque JIT trata cada pieza en el lote como una unidad, o trabajo, lo que resulta en el mínimo makespan. A través del uso extensivo de sistemas JIT en la manufactura, las medidas de rendimiento relacionadas con las penalidades: terminación anticipada (*earliness*) y tardanza *(tardiness)* recibieron atención significativa en la literatura sobre lote continuo.

2.5 CONCLUSIONES DEL CAPITULO

Es esta sección se presentó una descripción de los modelos de taller de flujo considerando principalmente los talleres de flujo hibrido por ser uno de los más comunes en la industria. Se presenta la notación $\alpha|\beta|\gamma$, mejor conocida como *tripleta de Graham,* con la que se describen las características, restricciones y objetivos de cualquier problema de la planificación en talleres de flujo.

Además se presentan algunas formas en las que se procesan los trabajos dentro de los talleres de flujo. El procesamiento de trabajos en grupos describe que con frecuencia existen algunos trabajos que tienen algún tipo de similitud por lo que pueden ser clasificados y organizados en grupos de acuerdo con sus requerimientos de manufactura.

Cuando se agrupan los trabajos por similitud, se reducen el número de ajustes en las máquinas que deben procesarlos, así como el tiempo de ejecución total. Se presenta un análisis de la literatura relacionada con investigaciones de tecnología de grupos de donde se desprende el concepto de batching.

Otra característica importante en esta investigación es la de división de trabajos la cual trata de atributos que permiten que un trabajo se divida o interrumpa para procesar otro trabajo o para enviar la parte restante a otra máquina. De lo anterior se desprende el concepto de lote continuo, en donde se permite que la parte ya procesada de un batch pase a la siguiente etapa aun cuando no se ha terminado de procesar la totalidad del batch.

Se comparan los conceptos del lote y trabajo. Se describen y analizan los cuatro casos cuando un trabajo puede ser dividido. El cuarto caso (la división de trabajos de

manera natural) y la combinación de dos casos de división de trabajos se consideran por primera vez en esta investigación, así como el caso cuando los pedidos de los consumidores y los lotes de material están presentes simultáneamente en un problema.

3. PROBLEMA DE CUMPLIMIENTO DE PEDIDOS EN LA PRODUCCIÓN DE INTERRUPTORES ELÉCTRICOS

El cumplimiento de pedidos de acuerdo a las fechas de entrega depende de la planificación correcta, la cual implica la definición: cuántos lotes se necesitan para la ejecución de los pedidos y qué tipos de lotes deben ser utilizados para obtener un máximo de piezas en los rangos requeridos del valor de operación (*planning problem*); cuál orden de procesamiento de lotes es preferible (*scheduling problem*). Actualmente, la compañía no tiene un procedimiento que le ayude solucionar estos problemas de una forma eficiente.

El problema de cumplimiento de pedidos se descompone en dos partes:

- Minimizar la cantidad de lotes en la etapa de clasificación y definir la cantidad de piezas por rangos de operación en cada repositorio necesarios para cumplir con todo pedido.
- Secuenciar los lotes en las dos etapas del HFS con varias máquinas paralelas idénticas en la primera etapa y cuatro máquinas dedicadas de líneas de forma en la segunda etapa, de tal manera, que se minimice el *makespan*.

3.1 ANÁLISIS DE LA OPERACIÓN DE CLASIFICADO

La clasificación es la operación clave en la planta. La eficiencia de la producción depende del tiempo consumido en esta etapa. Por lo tanto, el objetivo general es disminuir el tiempo de clasificación minimizando la cantidad de lotes, necesarios para cumplir con un conjunto de demandas especificado con una orden de producción, lo que implica la reducción de material procesado.

A pesar de que la producción tiene ocho etapas, sólo después de la clasificación de piezas según el rango de operación resulta claro, si será posible cumplir con los pedidos de acuerdo a sus fechas de entrega. Es necesario garantizar que se produce el material requerido en cantidades suficientes. Por lo tanto, la operación del clasificado tiene un papel clave en la planificación de la producción de la planta. El análisis de esta etapa permite la toma de decisiones no sólo acerca del orden de procesamiento de los pedidos

en las líneas de forma, sino alimenta las operaciones anteriores con la información actual.

De la experiencia se observó, que el tipo de vidrio de un lote influye de alguna manera al rango de operación de los interruptores, y consecuentemente a la distribución de piezas entre los repositorios de la máquina. El conocimiento acerca del tipo de la distribución tiene mucha importancia, puesto que permitiría con un nivel de confianza predecir las cantidades de piezas que caen en cada uno de los 25 repositorios de una máquina a partir de un lote (Glenberg y Andrzejewki, 2008).

Como primer acercamiento se analizaron las distribuciones de cada lote, tomando en cuenta el tipo de vidrio, para entender si tienen algunas características comunes. En la Figura 3.1 se aprecia la diferencia en la frecuencia relativa entre tres lotes de tipo g2. La grafica muestra además que la moda varía entre 7 y 9, disminuyendo hacia los repositorios extremos.

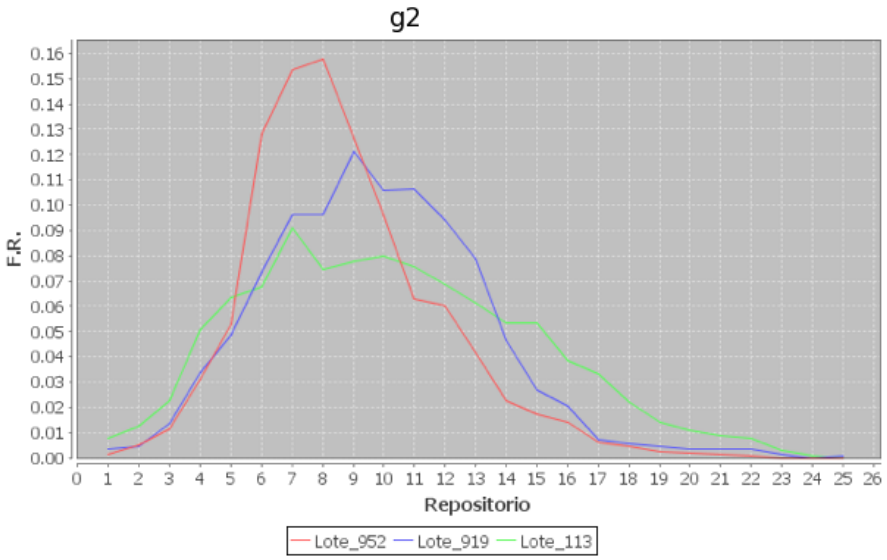

Figura 3.1. Frecuencia relativa de los datos para el vidrio tipo 2 (g_2).

El análisis estadístico se enfocó en establecer si realmente existe dependencia entre el tipo de vidrio y el rango de operación de un interruptor, de ser así, predecir las cantidades de piezas que caen en los repositorios a partir de un lote de un tipo en particular.

Se recolectaron 10 lotes para cada tipo de vidrio (g1, g2, g3, y g4) como ejemplos reales del proceso de clasificación. La información usada para el análisis varía en tamaño de lote por la presencia de desechos (*scrap*). Para homogenizar el tamaño, se extrajeron 1000 piezas de cada lote, utilizando generador discreto uniforme de MATLAB para los números aleatorios. Se evaluaron la media, la moda y desviación estándar (DE) para la distribución de piezas entre los 25 repositorios de los cuatro tipos de vidrio (g1, g2, g3 y g4). Las muestras revelan una tendencia central de las modas para cada tipo de vidrio.

El proceso de clasificado muestra fluctuaciones inesperadas debido a que en ocasiones los lotes llegan incompletos o incluso se mezclan distintos tipos de vidrio por fallas en el seguimiento de la información. La Tabla 3.1 contiene los resultados del análisis estadístico para los cuatro tipos de vidrio.

Tabla 3.1. Ejemplo estadístico

Ejemplo	TIPO DE VIDRIO											
	g1			g2			g3			g4		
	Moda	Media	DE	Moda	Media	DE	Moda	Media	DE	Moda	Media	DE
1	6	6.31	3.04	8	8.83	3.13	10	10.6	4.56	12	13.75	4.28
2	5	6.58	3.54	7	8.88	4.03	7	10.79	4.53	15	13.94	4.33
3	5	6.62	3.7	7	8.91	3.75	9	10.8	4.48	12	13.96	4.27
4	5	6.63	3.49	7	8.94	3.83	11	10.86	4.44	14	14.01	3.99
5	5	6.67	4.42	8	8.96	3.65	9	10.87	4.74	14	14.01	4.19
6	5	6.78	3.82	9	9.18	3.37	11	11.01	4.45	14	14.05	4.03
7	5	6.82	4.19	9	9.27	3.55	9	11.02	4.45	15	14.12	3.95
8	5	6.92	4.04	7	9.28	3.51	10	11.25	4.4	14	14.14	4.1
9	5	6.96	3.92	7	9.34	3.84	9	11.28	4.24	14	14.23	4.01
10	5	7.18	4	9	9.59	3.52	11	11.46	4.14	16	14.59	3.74
Media	5	6.75	3.81	7	9.12	3.62	9	10.99	4.44	15	14.08	4.09

Como muestra la Tabla 3.1, las medias revelan una influencia del tipo de vidrio a los resultados de distribución, de tal manera que el más grande valor de la media corresponde al tipo más alto de vidrio. Las medias de cada tipo fluctúan dentro de un rango, y estos rangos no se intersectan entre sí.

El valor de la moda para los dos primeros tipos de vidrio muestra una desviación hacia la izquierda con respecto a la media. Las modas para los dos tipos restantes, cuyas medias están bien ajustadas al centro de distribución (repositorios 12-13), varían a los dos lados de la media. La desviación estándar para todos los tipos varía dentro del rango [3.04, 4.75].

Las muestras se sumaron por separado para cada tipo de vidrio para revelar las tendencias de distribución para cada uno. La Figura 3.2 muestra el histograma para los tipos de vidrio g1, g2, g3, y g4; Figura 3.3 ilustra las tendencias de distribución para cada tipo de vidrio.

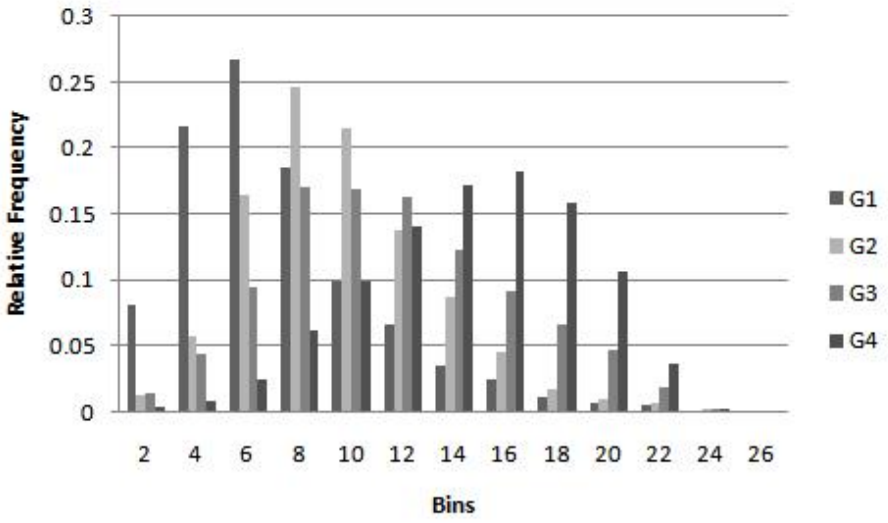

Figura 3.2 Histograma de distribución

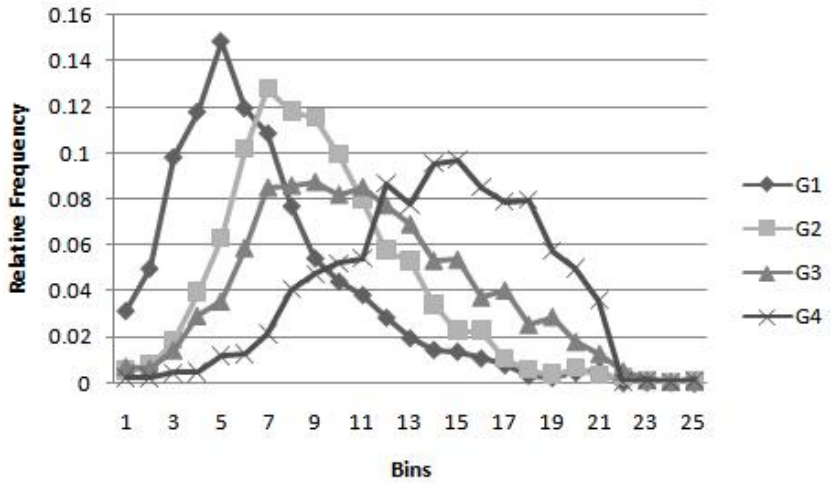

Figura 3.3. Tendencias de distribución

A partir del análisis de las distribuciones de los lotes con piezas elaboradas de distintos tipos de vidrio se establece que cada tipo de vidrio tiene la tendencia central de las estadísticas principales: la moda, la media y la desviación estándar, con respecto al rango de valor de operación por repositorio, de tal manera, que la mayoría de las piezas se localizan alrededor del centro de una distribución, disminuyendo en los extremos.

Cuando las distribuciones empíricas se suman para cada tipo de vidrio, las distribuciones resultantes tienden a la normal, cada una con su propia media y distribución estándar, esto es porque los valores de operación representan una medición de un parámetro y debido al Teorema de Límite Central (Montgomery y Runger, 2011) (Figura 3.4a).

En este estudio se utiliza un enfoque determinístico, es decir, las distribuciones se suponen conocidas y fijas para cada tipo de vidrio, lo cual a continuación referimos como el *tipo de lote*. Por lo tanto, el contenido de cada lote puede ser asignado a los pedidos, de una manera anticipada, para cumplir con estos parcialmente o por completo.

3.2 MINIMIZACIÓN DE LA CANTIDAD DE LOTES

El número de parte no indica un tipo de vidrio, solo especifica el rango del valor de operación que permite el cumplimiento de las demandas por partes, utilizando varios lotes, sin requerir un tipo de vidrio en específico. De otro lado, la selección del tipo de vidrio para procesar un lote implica la selección de los repositorios, en los cuales se deposita la mayoría de piezas. Como resultado, la misma cantidad de piezas en un rango de operación requerido se obtiene de diferente número de lotes. Por lo tanto, el problema es satisfacer los pedidos utilizando el número mínimo de lotes tomando ventajas de diferencias en distribución que genera cada tipo de vidrio.

La complejidad del problema está condicionada con las dos consideraciones siguientes: Cómo un lote tiene su estructura propia, definida por la distribución de piezas entre los repositorios, la cantidad requerida de piezas se obtiene a partir de diferente cantidad de lotes a través de la selección de tipo para cada siguiente lote (Figura 3.4a). Del otro lado, la división de pedidos, para ser asignados a los repositorios permitidos, no es evidente cuando ocurren *colisiones*, es decir, diferentes números de

parte tienen un traslape parcial (*overlapping*) en rangos admisibles de valores de operación; por ejemplo, un pedido necesita repositorios 1, 2, el otro 2, 3, el tercero 2, 3, 4, etc. Consecutivamente, un pedido debe ser balanceado entre repositorios y lotes. La Figura 3.4b ilustra el problema de selección de repositorios para diferentes tipos de lotes y varios pedidos.

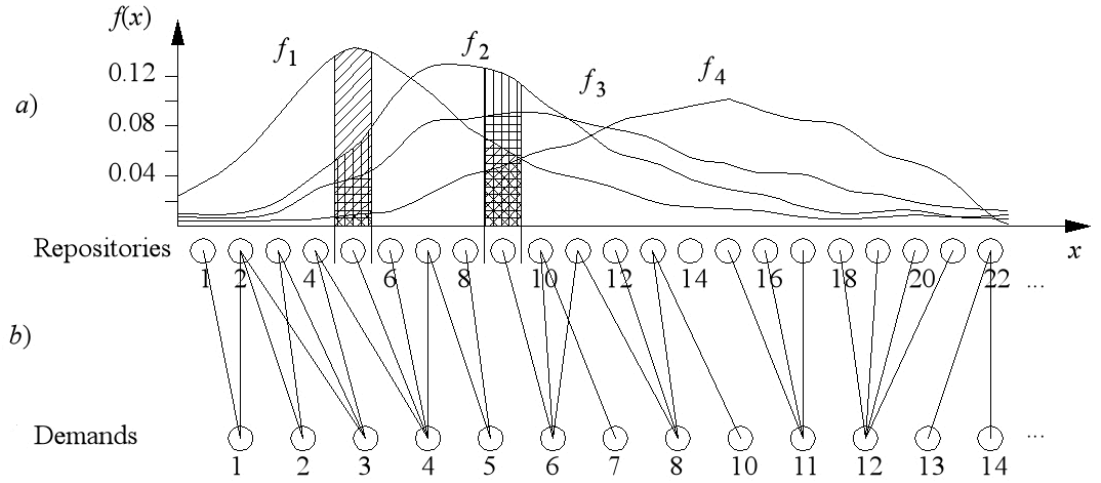

Figura 3.4. Una ilustración de elección entre cuatro tipos de lotes: a) Distribuciones de piezas entre los repositorios con las funciones de densidad de probabilidad f(x); b) Un ejemplo del grafo bipartido de la relación entre repositorios y números de parte; las colisiones ocurren en repositorios 2,3,4,7,10,11,13.

Se deben tomar tres decisiones mutualmente dependientes en el problema estudiado para encontrar la cantidad mínima de lotes:

1) El orden de asignación de los pedidos a los lotes (*demand scheduling*);

2) El orden de procesamiento de los lotes, tomando en cuenta el tipo de lote (*lot scheduling*);

3) El modo de división (*splitting*) de los pedidos entre los repositorios los lotes.

Cuando el objetivo es encontrar la cantidad mínima de lotes de un tamaño equivalente (tiempo de procesamiento equivalente), se obtiene simultáneamente el *makespan* mínimo, el cual significa el momento cuando el último trabajo abandona el taller se anota en la literatura de planificación (*scheduling*) como C_{max}. Utilizando la tripleta de Graham, y sus extensiones propuestos recientemente por (Ruben Ruiz y Vázquez-Rodríguez, 2010), el problema considerado se denota como:

$$PM\,|p_{lot}=p,\ lot\ split, job\ split, job\ constraints\ |C_{\max}, \quad (3.1)$$

lo que indica el ambiente de máquinas paralelas idénticas, donde el tiempo de procesamiento de cualquier lote es constante, los dos el lote y el trabajo (pedido) se permite dividir, se consideran restricciones de asignación trabajo-a-sublote, y el objetivo es minimizar el *makespan*.

3.3 ANÁLISIS DEL PROBLEMA PARA UNA PARTICIÓN DE TRABAJOS

El problema del uso óptimo de material incluye la optimización de los tres elementos siguientes: el número de lotes, la selección de tipo para cada lote, la división de pedidos para la colocación a los lotes tomando en cuenta los parámetros de su distribución entre los repositorios. Supongamos que la división de los pedidos ya está realizada. A continuación se discute la agrupación (*batching*) de los fragmentos que pertenecen al mismo pedido (la misma forma de navajas) después de la etapa de clasificación para procesarse en la segunda etapa, y la planificación (*scheduling*) del procesamiento de los pedidos en las dos etapas del HFS con un número de máquinas paralelas idénticas en la primera etapa y las cuatro máquinas dedicadas de la segunda. Primeramente, el batching de sublotes en la primara y la segunda etapas se analiza bajo la suposición que el número de lotes y la partición del trabajo en sublotes son conocidos; después se calcula la ocupación de las máquinas.

3.3.1 Batching de sublotes

Sea L el número de lotes necesarios para la colocación de N trabajos, y los trabajos enteros se dividen entre sublotes d_{lrj} del tamaño s_{lrj}. Sea Z_l el conjunto de sublotes d_{lrj} asignados al lote l, $l = 1,\ldots, L$, $r = 1, 2,\ldots, R$, $j = 1,\ldots, N$, de tal manera que el conjunto de lotes

$$A=\bigcup_{l=1}^{L} Z_l\ ,\ Z_l \neq \varnothing, Z_\alpha \cap Z_\beta = \varnothing,\ \alpha \neq \beta,$$

y el tiempo total de su procesamiento es

$$\sum_{l=1}^{L}\sum_{r=1}^{R} s_{lrj} = S_j, j = 1,\ldots, N.$$

Consecuentemente, el lote l puede ser considerado como un batch Z_l de sublotes más WIP.

Los m_1 máquinas de la primera etapa procesan m_1 lotes simultáneamente, iniciando y terminando juntos. Los lotes son asignados en las máquinas de la etapa 1 utilizando el método de "rotación": lote 1 se asigna a la máquina 1, lote 2 a la máquina 2, ..., lote m_1 a la máquina m_1, lote m_1 + 1 a la máquina, y así consecutivamente (J. Liu, 2008). El procesamiento de m_1 lotes en las máquinas paralelas es denominado como el ciclo h. El número total de ciclos es $h = \lceil L/m_1 \rceil$, donde $\lceil L/m_1 \rceil$ es el límite superior entero del valor L/m1; y los lotes procesados en la máquina i_1 forman el conjunto H_{i_1} de ($\lceil L/m_1 \rceil$ [-1]) lotes, $i_1 = 1,\ldots, m_1$.

Después de la primera etapa, hay un tiempo de ajuste, el cual es no-anticipado, es decir, adicionado al tiempo de procesamiento, e independiente de la secuencia de los trabajos (*non-anticipatory sequence independent setup time*). Este tiempo se utiliza para la separación de sublotes, el reagrupamiento y la preparación de los alimentadores (*feeders*) en máquina de la segunda etapa. Este tiempo se aplica una vez para todos d_{lrj} procesados en el ciclo h. Sin perder la generalidad, se establece $s_{in} = const$ y se incluye en el tiempo de procesamiento del lote:

$$p^{(1)} = p^{(1)} + s_{in}. \tag{3.2}$$

Entonces, el tiempo de ocupación de la máquina i_1 es:

$$P_{i_1}^{(1)} = \left(\lceil L/m_1 \rceil [-1]\right) p^{(1)},\ i = 1,\ldots,m_1. \tag{3.3}$$

Ahora, sea J el conjunto de los trabajos, $J = \{j_1,\ldots, j_N\}$, y J_{i_2} un subconjunto de trabajos cuya segunda operación está asignada a la máquina i_2 de la segunda etapa, $i_2 = 1,\ldots, m_2$. Entonces, $J = \bigcup_{i_2=1}^{m_2} J_{i2}$ forma la partición del J en m_2 disjuntos, así que $J_{i_2} \neq \varnothing$, $J_\gamma \cap J_\eta = \varnothing$ si $\gamma \neq \eta$. El tiempo puro de la ocupación de la máquina i_2 es $\sum_{j \in J_{i_2}} S_j p_{i_2}^{(2)}$.

Después de finalizar el ciclo c, los sublotes que pertenecen al trabajo procesado j se juntan y después se agrupan sin mezclarse en un batch F_{h,i_2} con otros trabajos cuya

segunda etapa está asignada a la máquina i_2; $\bigcup_{h=1}^{\lceil L/m_1 \rceil} F_{h,i_2} = J_{i_2}$. El tiempo puro del procesamiento del batch F_{h,i_2} es equivalente a

$$P_{h,i_2}^{(2)} = p_{i_2}^{(2)} . f_{h,i_2}, \tag{3.4}$$

donde $f_{h,i2} = \sum_{l \in H_h, r \in \{1,\ldots,R\}, j \in J_{i_2}} s_{lrj}$, $i_2 = 1,\ldots, m_2$.

Consecuentemente, el tiempo de la ocupación de la máquina i_2 es:

$$\sum_{h=1}^{\lceil L/m_1 \rceil} P_{h,i_2}^{(2)} = \sum_{h=1,\ldots,\lceil L/m_1 \rceil, f_{h,i_2} \neq 0} p_{i_2}^{(2)} \cdot f_{h,i_2} = \sum_{j \in J_{i_2}} S_j p_{i_2}^{(2)}. \tag{3.5}$$

3.3.2 Generalización del problema de Johnson para secuenciación de lotes

Es evidente, que el número mínimo de lotes lleva el mínimo al total del tiempo del procesamiento de los lotes en la primera etapa, y en su turno, la cantidad de lotes depende de sus tipos respectivos y del modo de la partición de los trabajos. La finalización de la segunda etapa se sujeta a los resultados de la primera etapa debido a la pre-asignación de trabajos a las máquinas dedicadas. Consecuentemente, el resultado de descomposición de los trabajos en los sublotes afecta directamente el valor de $C_{\max}$. Entonces, la secuenciación de los trabajos es sujeta a la asignación de los trabajos en los lotes.

Supongamos que la asignación de trabajos es realizada, es decir, los tamaños de los sublotes s_{lrj} y las cantidades de lotes de cada tipo son conocidos. Entonces, el problema (3.1) se transforma a:

$$FH2, PM^{(1)}, DM^{(2)} \mid p_l^{(1)} = p^{(1)}, p_h^2 = P_{h,i_2}^{(2)} \mid C_{\max} \tag{3.6}$$

A continuación se consideran los casos de una y varias máquinas de clasificado.

Una máquina de clasificado

Cuando se tiene una sola máquina en la primera etapa, el problema (3.6) se transforma de la manera siguiente:

$$FH2,1^{(1)}, DM^{(2)} \mid p_l^{(1)} = p^{(1)}, p_l^2 = P_{l,i_2}^{(2)} \mid C_{\max} \tag{3.7}$$

Lema 1: El problema (3.7) representa una generalización del problema de Johnson $F2||\ C_{\max}$.

Prueba: La única máquina en la primera etapa procesa sucesivamente L lotes. El tiempo de procesamiento de un lote es $p^{(1)}$ para cualquier lote l. Ahora, las m_2 máquinas de la segunda etapa se inician simultáneamente. Una máquina i_2, $i_2 = 1,\ldots, m_2$, inicia independientemente después de finalizar el lote l, o cuando se libera, para procesar el batch F_{h,i_2}, y trabaja continuamente $P_{l,i_2}^{(2)}$ unidades de tiempo. Entonces, el problema (3.6) se descompone en m_2 problemas de Johnson, donde $p^{(1)}$ y $P_{l,i_2}^{(2)}$ son tiempos de procesamiento del lote l en las máquinas de la primera y la segunda etapa, respectivamente, $i_2 = 1,\ldots, m_2$.

La solución óptima del i_2-problema $F2||\ C_{\max}$ es la permutación $\pi_{i_2}^*$ creada de acuerdo a la *regla de Johnson*, donde el lote l_1 precede al lote l_2 (Błażewich et al., 2007):

$$\min\{p^{(1)}, P_{l_2,i_2}^{(2)}\} \le \min\{P_{l_1,i_2}^{(2)}, p^{(1)}\},\ 1 \le l_1, l_2 \le L. \tag{3.8}$$

Eso implica que si el tiempo de procesamiento en la primera etapa es constante, los trabajos en una permutación óptima $\pi_{i_2}^*$ son arreglados en el orden *Larger Processing Time First* (LPT) (Pinedo, 2008) de la duración de la segunda etapa.

Conclusión 1: El límite inferior LB del problema (3.7) es:

$$LB = \max_{1 \le i_2 \le m_2} C_{\max}(\pi_{i_2}^*). \tag{3.9}$$

Las permutaciones en (3.9) son generalmente diferentes. Una solución óptima $C(\prec^*)$ del (3.7) alcanza el LB si el orden general de lotes es $l_1 \prec l_2 \prec \cdots l_{i_2} \prec \cdots \prec l_L$, para todas las permutaciones $\pi_{i_2}^*$.

Para el problema real considerado, es característico que el tiempo de procesamiento del lote es esencialmente más largo que la duración de la segunda operación, incluyendo si se tienen varias máquinas en la primera etapa (Figura 3.5).

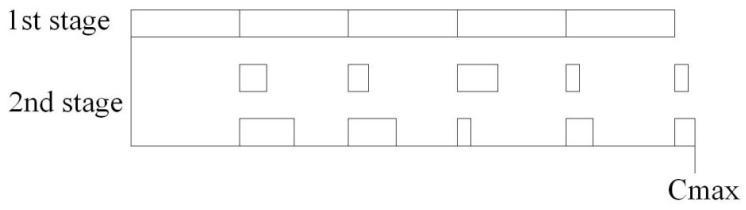

Figura 3.5. Un ejemplo de la solución del problema Johnson generalizado con una máquina en la primera etapa (m_1 =1) *y dos máquinas en la segunda etapa* (m_2 =2).

Lema 2: La solución óptima del problema (3.7), cuando $P^{(2)}_{l,i_2} << p^{(1)}$, $\forall l$, i_2, es una permutación π*, donde los lotes l son arreglados en el orden de decrecimiento del valor $\max_{i2}\ P^{(2)}_{l,i_2}$, $l = 1,\ldots, L$.

En tal caso, (3.7) representa el problema de Johnson de dos máquinas con tiempos $p^{(1)}$ y $\max_{i_2}\ P_{l,i_2}{}^{(2)}$ en la primara y la segunda máquinas respectivamente; y el *makespan* es:

$$C_{\max} = \max_{1\le l\le L}(lp^{(1)} + \max_{1\le i_2\le m_2}\sum_{k=l}^{L} P^{(2)}_{k,i_2}). \qquad (3.10)$$

Teorema 1: Existe una planificación óptima del problema (3.7), donde el número de lotes es mínimo.

Prueba. Sea L* el número mínimo de lotes necesarios para asignación de N trabajos, sea $L^1 = L^*+1$, y respectivamente, los tiempos mínimos del cumplimiento son C* y C^1. Los tiempos del procesamiento en la primera etapa son: $L^*p^{(1)} < (L^*+1)\,p^{(1)} = L^1p^{(1)}$. El tiempo de ocupación de las máquinas en la segunda etapa son fijos e iguales a $\sum_{j\in J_{m_2}} D_j p^{(2)}_{i_2}$, para cualquier cantidad de lotes. El tiempo muerto (*idle time*) de la máquina m_2 solo crece cuando crece la cantidad de lotes. Consecutivamente, C^1 solo puede ser mayor que C*.

Conclusión 2: La planificación óptima del problema (3.7) se alcanza con la cantidad mínima de lotes.

Múltiples máquinas de clasificado

El caso de una máquina de clasificado se extiende a múltiples máquinas con una sola diferencia en el cálculo del tiempo de procesamiento en la segunda etapa: los batches F_{h,i_2} son formados de los sublotes obtenidos de los m_1 máquinas y sus tiempos de procesamiento son calculados de acuerdo a la formula (3.11).

El *makespan* se define así:

$$C_{\max} = \max_{1 \le h \le \lceil L/m_1 \rceil} (hp^{(1)} + \max_{1 \le i_2 \le m_2} \sum_{l \in \{H_h\}} P^{(2)}_{l,i_2}). \qquad (3.11)$$

La conclusión 2 permite obtener la solución del problema (3.11) minimizando la cantidad de lotes necesarios para la colocación de los *N* trabajos, tomando ventajas de diferencias entre las distribuciones que genera cada tipo de lote.

3.4. CONCLUSIONES DEL CAPITULO

En esta sección se presenta la descripción del problema de cumplimiento de pedidos dependientes de la disponibilidad de los recursos en la producción de interruptores eléctricos que ocurre en una empresa de manufactura real. Se presenta el análisis del proceso productivo, se ubica la etapa de clasificado como la etapa clave, de la cual depende la generación de material para cumplir con los pedidos; se observó que esta etapa recibe el material a procesar en lotes de tamaño 5500 piezas.

Adicionalmente se tienen cuatro tipos de lote diferentes entre sí por el tipo de material utilizado en las piezas. En la etapa de clasificado se toma cada pieza y se deposita en uno de 25 repositorios de acuerdo a su valor de operación. Inicialmente se conoce que el tipo de lote afecta a la forma en que las piezas se distribuyen entre los repositorios. Se presenta un análisis estadístico que permite conocer las tendencias de distribución para cada tipo de lote.

Los pedidos de los clientes están formados por un número de parte y cantidad. El número de parte describe el rango del valor de operación y la forma de la pieza. Algunos números de parte comparten el uso de material de los repositorios por lo que existe una complejidad dada por la necesidad de decidir a que trabajo se le proporcionará el material de un repositorio. Generalmente se requiere más de un lote de

material para satisfacer un pedido, por lo que se deben dividir los trabajos entre varios lotes. Más aun, es posible aprovechar tendencias de distribución de cada tipo de lote a través de su combinación para satisfacer los pedidos. Para resolver el problema es necesario encontrar el orden en el cual se deben asignar los pedidos a los lotes, el modo en el que se dividirán los trabajos entre los lotes, el orden en el que se deben procesar los lotes.

Se describe el problema como una generalización del problema de Johnson (Vignier et al., 2010)a través de la suposición de que se conoce la partición de los trabajos a procesar y encontrando una solución al problema de minimización de lotes para colocar *N* trabajos aprovechando las diferencias en las distribuciones de cada tipo de lote.

4. GENERALIZACIÓN DEL PROBLEMA BIN PACKING

4.1 INTERPRETACIÓN DEL PROBLEMA EN TÉRMINOS DE BIN PACKING

El problema de minimizar la cantidad de lotes permite interpretarse como una generalización del problema de empaquetamiento (*bin packing*). En la variante clásica (Figura 4.1) NP-difícil se requiere empaquetar una lista dada de objetos, cuyos tamaños pertenecen al intervalo (0, 1], en el menor número posible de contenedores con tamaño igual a la unidad (Coffman et al., 2007).

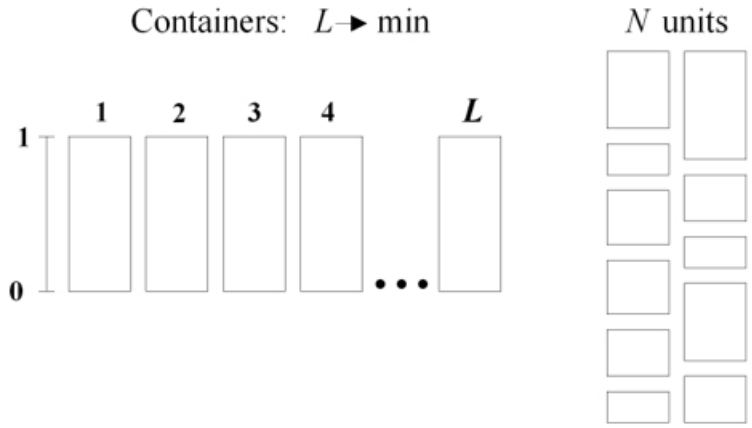

Figura 4.1 Empaquetamiento clásico.

Sea R el número de repositorios de una máquina cuya capacidad completa se representa por k_{gr}, $r = 1,\ldots, R$, cuando se distribuye un lote de tipo g, $g = 1,\ldots, G$ (Figura 4.2). Luego los R repositorios forman un contenedor (*bin*) con capacidad $\sum_{r=1}^{R} k_{gr} = U$, $\forall g = 1,\ldots, G$. Se tiene una colección de G tipos de contenedores, los cuales difieren entre sí por las capacidades k_{gr}. Las N demandas, que consumen estas capacidades, pueden ser consideradas como objetos a empacar, o asignar. Existen varias restricciones para la asignación de un objeto en un repositorio y contenedor.

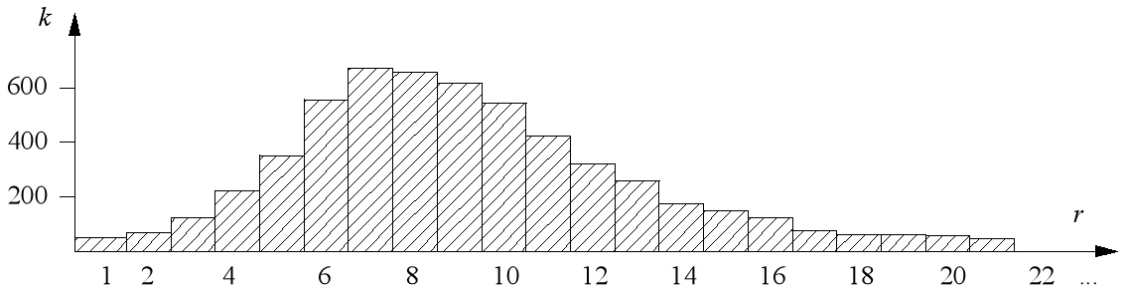

Figura 4.2. Un contenedor formado por R repositorios, $r = 1,...,R$ *, de capacidad* $k_{2,r}$ *asociados con el lote de tipo g = 2 en el problema real considerado.*

Generalmente, se supone que un objeto a empacar es indivisible y por lo tanto no puede ser depositado en dos contenedores. Cuando un objeto, por ejemplo un pedido, consiste en un número de piezas idénticas, este puede ser dividido (*split*) arbitrariamente en fragmentos y empaquetado de manera separada. En este documento se considera que los objetos pueden ser divididos arbitrariamente y al mismo tiempo, demasiado grandes para ser asignados en un simple repositorio.

La Figura 4.3 muestra algunas características de la generalización de problema de empaquetamiento utilizado en esta investigación. Se tiene un conjunto de objetos a empaquetar en contenedores divididos en *R* repositorios; existen restricciones de asignación de los objetos a los repositorios. Un objeto representa un número de piezas. Es posible fragmentar los objetos para asignarlos a los repositorios permitidos. Existe variabilidad de los tamaños de los repositorios para los diferentes repositorios. Un objeto puede requerir más de un contenedor para ser asignado por completo.

Si no se permite la fragmentación, solo se utiliza un tipo de contenedor, el cual consiste en un repositorio de un tamaño dado, y no existe restricción para colocación de objetos, entonces se recibe el problema clásico de empaquetamiento. El problema presentado es por lo tanto, su generalización.

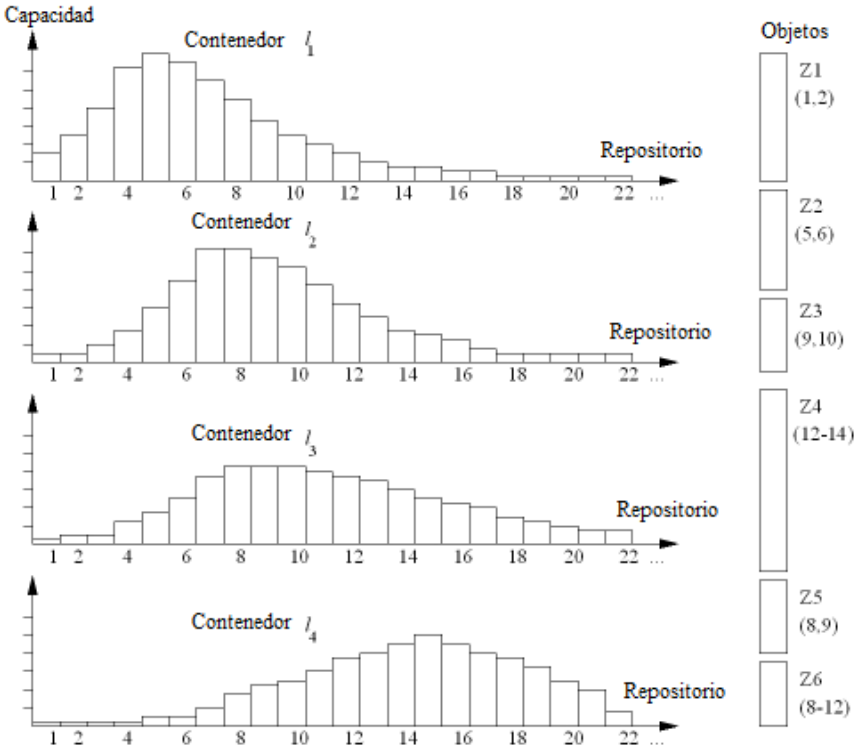

Figura 4.3 Generalización del problema de empaquetamiento

4.2 REVISIÓN DE LITERATURA

El problema de empaquetamiento es uno de los problemas más tratados en la literatura computacional y tiene muchas variantes. En el caso clásico unidimensional, el cual se conoce como NP-difícil, se requiere empaquetar una lista dada de objetos, cuyos tamaños son menores o iguales a la unidad, en el menor número posible de repositorios de tamaño uno. La aplicación del empaquetamiento aparece en numerosas áreas, como planificación, programación televisiva, trasporte, almacenamiento computacional, asignación de ancho de banda, etc. Los estudios más recientes han sido desarrollados por Coffman y Csirik (2007).

El problema de empaquetamiento se ha estudiado en escenarios online y offline. En un escenario online los objetos deben ser procesados en el mismo orden en el que llegan, sin conocimiento alguno sobre la secuencia completa de los objetos. En un escenario offline, se dispone de la lista completa de objetos a empaquetar. El escenario offline corresponde con el ambiente investigado donde los datos de entrada se conocen de antemano.

Se presentan algunas variantes del problema de empaquetamiento en la literatura, relacionadas con la presente investigación.

Tipos diferentes de repositorios, repositorios de capacidad variable

La variabilidad en el tamaño de contenedores fue inicialmente estudiado en el artículo de Langston (1984), donde se investiga el problema de maximizar el número de objetos empaquetados dentro de *m* repositorios disponibles, y el tamaño de los repositorios es diferente. Más tarde, (Friesen y Lagnston, 1986) introdujeron el problema de empaquetamiento con repositorios de tamaño variable (*Variable Sized Bin Packing*, VSBP), el cual es como sigue: Se tiene una lista dada de objetos $L = \{a_1, a_2,\ldots, a_n\}$ y una colección finita l de tamaños (costos) de repositorios con una inagotable reserva de repositorios de cada tamaño. El objetivo es empaquetar los objetos en los repositorios de tal manera que la suma de los tamaños sea mínima. El problema clásico de empaquetamiento es un caso particular de VSBP donde l =1. Dado que VSBP es un problema NP-difícil, comúnmente se utilizan algoritmos aproximados para resolverlo. Los autores sugieren esquemas eficientes de aproximación que obtienen los peores casos con límites asintóticos, ver, por ejemplo, Friesen y Lagnston (1986), Murgolo (1987), Dawande et al. (2001). En estos artículos se supone el costo es fijo y se define igual al volumen del repositorio correspondiente o directamente proporcional a él. Friesen y Lagnston (1986) presentan tres algoritmos aproximados basados en *Next Fit* (NF) y *First Fit Decreacing* (FFD):

- *Next Fit using Largest bins only* (NFL) que solo utiliza repositorios de tamaño grande;
- *First Fit Decreacing using Largest bins, at end Repack to smallest posible bins* (FFDLR) utilizando repositorios grandes con reempaque en repositorios pequeños;
- *First Fit Decreacing using Largest bins, but Shifting as necesary* (FFDLS) utilizando repositorios grandes pero con intercambio si es necesario.

Murgolo (1987) desarrolló un esquema de aproximación en tiempo polinomial.

En el artículo (Kang y Park, 2003) se considera un problema VSBP, donde el objetivo es minimizar el costo total de utilizar repositorios cuando el costo por unidad

de tamaño de cada repositorio no se incrementa cuando se incrementa el tamaño del repositorio. Se describen dos algoritmos voraces basados en los algoritmos FFD y *Best Fit Decreacing* (BFD), que además se analizan en tres casos especiales: a) el tamaño de los objetos y los repositorios es divisible, respectivamente, b) solo el tamaño del repositorio es divisible, y c) el tamaño del repositorio no es divisible.

Xing (2002) considera un VSBP con objetos cuyo tamaño excede el tamaño del contenedor más grande (*oversized ítems,* BPOS). Los contenedores no pueden ser sobre empacados; se permite dividir los objetos de tamaño excesivo hasta que las partes no sean mayores que el tamaño del contenedor más grande. Se propone un procedimiento de dos etapas: primero empacar los objetos de tamaño excesivo en los contendores más grandes hasta llenarlos y luego empacar el remanente.

En Dawande et al. (2001) se considera un VSBP con restricción de color. Los objetos vienen con dos atributos: color y tamaño, Se requiere que cada repositorio contenga objetos con a lo más p colores distintos. Zhou et al. (2009) consideran un problema similar, introduciendo una restricción adicional al VSBP: algunos de los objetos se etiquetan como diferentes tipos y deben empaquetarse separados en diferentes repositorios. Los autores modelan el problema como un tipo restringido y un problema de empaquetamiento con repositorios variables (*Type-Constrained and Variable Sized Bin Packing Problem*, TVSBPP) y lo resuelven con un método de ramificación y acotamiento (*Branch and Bound*).

Correia et al. (2008) proponen una formulación de programación entera (*Less Straightforward Integer Programming Formulation*) de el VSBP obtenido por una técnica de reformulación de modelo; se plantean nuevas inecuaciones válidas con las variables del modelo discretizado para fortalecer los limites lineales originales de la relajación.

Epstein y Levin (2008) proponen el llamado *Generalized Cost Variable Sized Bin Packing.* Se tiene un suministro infinito de repositorios de r tipos, cuyos tamaños se denotan por $b_r < \ldots < b_1 = 1$. Los objetos de tamaños en (0,1] se separan en subconjuntos. Un repositorio de tipo i se asocia con un costo fijo c_i, el cual puede ser pequeño o grande en comparación con el tamaño del repositorio. Los costos de repositorios de diferentes tamaños no están relacionados. El objetivo es encontrar una

solución factible cuyo costo total sea mínimo. Se propone un esquema polinomial asintótica de aproximación completa (*Asymptotic Fully Polynomial Time Approximation Scheme*, AFPTAS) para el problema generalizado.

En el artículo de (Haouari y Serairi, 2009) se presentan y comparan varios algoritmos para solucionar el problema VSBP. Estos algoritmos incluyen cuatro heurísticas constructivas, donde las columnas se generan eficientemente utilizando una heurística de empaquetamiento aleatorizada y un algoritmo genético eficiente con tres operadores de cruzamiento: *Two-point*, *Three-point*, y *Similar? Job One-Point*.

Baldi et al. (2012) resumen las investigaciones anteriores que consideran varias reglas y restricciones en el acomodo de los objetos en los repositorios, así como diferentes atributos de objetos y repositorios e introducen *Generalized Bin Packing Problem* (GBPP), donde se tienen un conjunto dado de objetos caracterizado por el volumen y beneficio y un conjunto de repositorios con un volumen dado y costo. El objetivo es seleccionar un subconjunto de objetos beneficiosos los repositorios apropiados para optimizar una función objetivo combinando el costo de utilizar un repositorio y el beneficio dado cargar los objetos seleccionados. El GBPP generaliza muchos problemas de empaquetamiento presentados en la literatura: *Bin Packing* y VSBP, así como *Knapsack*, *Multiple Homogeneous* y *Heterogeneous Knapsack*, u otros. Los autores presentan dos formulaciones de programación entera mezclada del GBPP. La primera se basa en la decisión de asignación de objetos a repositorios y requiere un número polinomial de variables y restricciones. Este modelo es útil en discusiones sobre como el GBPP generaliza el problema de empaquetamiento mencionado anteriormente. El segundo modelo se basa en patrones viables de carga y en la idea de cobertura de conjunto, lo que facilita el desarrollo de métodos de solución eficiente. Se proponen un método eficiente de límite inferior basado en generación de columnas (*column-generation based lover bound method*), así como algoritmos FF, BF y procedimientos de límite superior basado en generación de columnas (*column-generation based upper bound procedures*). En el artículo se introduce un nuevo conjunto de instancias y se presentan resultados de extensivos experimentos computacionales, los cuales muestran que los procedimientos propuestos son eficientes y los límites son muy estrechos ajustados? (*tight*).

Una generalización del problema titulado *Variable Cost and Size Bin Packing Problem* (VCSBPP), la cual se caracteriza por sus atributos físicos y económicos, volumen de repositorios (capacidad) y selección de los costos fijos, se estudia en Crainic et al. (2011). El objetivo del problema es seleccionar los repositorios para empaquetar todos los objetos con un mínimo costo total de selección de repositorio. Se proponen límite inferior eficiente y una heurística.

Bang-Jensen y Larsen (2012) presentan una heurística de búsqueda local (*local search*) basada en una tabla de programación dinámica. Se consideran varios casos particulares, e introduce un algoritmo exacto para pequeñas instancias de problemas reales de VSBP. Por otra parte, los autores proporcionan un breve análisis de aproximaciones de VSBP.

En el trabajo de Hemmelmayr et al. (2012) se propone una variante de metaheurísticas conocida como *neighbourhood search*, para atacar el problema VSBP. El objetivo es encontrar el costo mínimo de empaquetar un conjunto dado de objetos ponderados en un conjunto de repositorios de tamaños y costos variables. El algoritmo propuesto puede verse como una metaheurística híbrida, esto debido a que utiliza una técnica de límite inferior y programación dinámica en varios componentes del algoritmo. Se muestra una alta competitividad del algoritmo con otras soluciones de problemas cercanos en extensivos experimentos.

Fragmentación de repositorios

El problema de empaquetamiento con fragmentación de repositorios se considera primeramente en el trabajo de Shachnai y Tamir (2004). Se considera un *Class-Constrained Bin Packing problem* (CCBP), donde los objetos pueden ser de diferentes clases (colores). Se tiene un conjunto de repositorios, cada uno de los cuales tienen capacidad v y c compartimientos, así como n objetos de M diferentes clases y el mismo tamaño (unidad). Cada repositorio tiene una capacidad y un número límite de compartimientos. Los objetos de diferentes colores no pueden colocarse en el mismo compartimiento. El empaquetamiento es factible si se satisface las restricciones tradicionales de capacidad, así como la restricción de clase. Se requiere llenar los repositorios con objetos, de acuerdo a las restricciones de capacidad, tal que los objetos

de diferentes clases se coloquen en compartimientos separados; así, cada repositorio puede contener objetos de al menos *c* clases distintas. El objetivo es empaquetar todos los objetos en el mínimo número de repositorios. Se conoce que el problema CCBP es NP-difícil en sentido estricto. Se estudian las variantes online y temporaria. Un CCBP temporario indica que los objetos deben ser empaquetados durante un intervalo restringido de tiempo, el cual es desconocido de antemano. Se derivan estrechos límites para ambas variantes.

Fragmentación de objetos

Mandal et al. (1998) introdujo el problema *Fragmentable Object Bin Packing* (FOBP), donde se permite fragmentar los objetos mientras se empaqueten en un repositorio de capacidad fija. La capacidad utilizada para empaquetar cualquier objeto (ya sea entero o después de fragmentar) de tamaño *k* es *k*+1. Se muestra que el problema de decisión para *N* objetos fragmentables es NP-difícil cuando $N \geq 2$.

Namman y Rom (2001) investigaron el mismo problema, llamado *Bin Packing problem with Item Fragmentation* (BP-IF). Utilizaron una versión del bien conocido algoritmo NF, con la capacidad de fragmentar objetos. Se investigó el rendimiento. Se presentan el peor caso y el promedio, se comparan con el caso donde no se permite la fragmentación. Este problema se deriva de un problema de programación (scheduling) que ocurre con los datos que se trasmiten en redes de tipo *Community Antenna Television* (CATV).

Menakerman y Rom (2001) investigaron dos variantes del FOBP donde se tienen dos posibles funciones de costo. El fragmentar un objeto se asocia con un costo, el cual hace el problema NP-difícil. En la primera variante, llamada *Bin Backing with Size-Increasing Fragmentation* (BP-SIF), se permite fragmentar cualquier objeto, pero se agrega una unidad adicional a cada fragmento. En la segunda variante cada objeto tiene un tamaño y un costo y el fragmentar un objeto aumenta el costo del objeto pero no cambia su tamaño. Esta variante se llama *Bin Packing with Size Preserving Fragmentation* (BP-SPF). Se investigan varios algoritmos basados en algoritmos bien conocidos como FF y FFD, además se desarrollan otros algoritmos originales. El

modelo de empaquetamiento también desprende de un problema de programación de los datos que se trasmiten en redes de tipo CATV.

Shachnai y Tamir (2004) consideran el mismo problema descrito en Menakerman y Rom (2001) y desarrollan un AFPTAS para cada una de las dos variantes del problema.

En el artículo (Epstein et al., 2012), los autores estudian el problema de empaquetamiento con objetos divisibles y con restricciones de carnalidad. En este problema, se tiene un conjunto de objetos a empaquetar en la menor posible cantidad de repositorios. Se permite dividir los objetos, pero cada repositorio contiene a lo máximo k (partes de) objetos, donde k es una constante. Los objetos pueden ser tan grandes como la unidad, la cual es el tamaño del repositorio. Los autores terminan este problema proporcionando un esquema asintótico de aproximación de tiempo polinomial (*Asymptotic Polynomial Time Approximation Scheme,* APTAS) y un límite superior.

Chung et al. (2006) introducen un problema de empaquetamiento motivado por la asignación de dos puertos de memoria a el bus de un procesador. Se tiene un número ilimitado de repositorios, cada uno con capacidad de 1 y una lista de pesos, $W = (w_1, w_2, w_3, \ldots w_n)$, donde w_i es un número no negativo y puede ser mayor que 1, en general. Los pesos W se empaquetan en b repositorios si existe una forma de dividir el objeto I_j de tipo j con peso w_j, para $1 \leq j \leq n$, tal que todas las partes quepan en b repositorios. En otras palabras, para cada k, las partes que se agrupan dentro del k-esimo repositorio tienen en total un peso no mayor que 1. El problema es: Para una lista dada W, encontrar la forma de empaquetar W dentro del mínimo número de repositorios tal que cada repositorio tenga partes de por lo menos dos tipos. Se muestra que el problema de asignación es NP-difícil en general, pero tiene un algoritmo rápido de aproximación asintótica con un factor de 3/2. El algoritmo se ilustra con la gráfica asociada. También se proporciona una mejora al algoritmo.

Epstein y van Stee (2011) proporcionan un algoritmo simple de aproximación asintótica 3/2 para el mismo problema de Chung et al. (2006), el cual es en realidad un algoritmo online. Adicionalmente, este algoritmo obtiene buen rendimiento para los casos más generales donde cada repositorio contiene a lo máximo k partes de objetos. Los autores muestran que esta generalización es estrictamente NP-difícil para cualquier

$k \geq 3$, y designan un algoritmo aproximado eficiente, para el cual la relación de aproximación puede hacerse arbitrariamente cercana a 7/5.

Restricciones de asignación

Xavier y Miyazawa (2008) presentan un algoritmo aproximado para un CCBP, donde los objetos deben ser separados por estantes (*non-null shelf divisions*). Este problema se denota como *Class Constrained Shelf Bin Packing* (CCSBP). Se tienen repositorios de tamaño B, valores no negativos d y Δ, así como una lista L de objetos, cada objeto $e \in L$ con tamaño s_e y clase c_e. Un estante se define como un subconjunto de objetos empaquetados en un repositorio con un total de tamaño de objetos a lo máximo Δ, tal que todos los objetos en este estante pertenecen a la misma clase. Dos subsecuentes estantes deben ser separados por una división de estante de tamaño d. El tamaño del estante es el tamaño total de sus objetos más el tamaño de la división de estantes. El problema CCSBP es empaquetar los objetos de la lista L en el mínimo número de repositorios tal que los objetos se dividan entre los estantes y el tamaño total del estante en un repositorio sea no más que B. Se presenta un algoritmo híbrido basado en FFD y BFD y un APTAS para el problema CCSBP cuando el número de diferentes clases está limitado por una constante C.

Zhou et al. (2009) introducen una restricción adicional para el problema VSBP. Prácticamente, algunos objetos deben ser empaquetados por separado en diferentes repositorios debido a requerimientos específicos. Por lo tanto, estos objetos se etiquetan como diferentes tipos. Los repositorios se utilizan para empaquetar cualquier tipo de objetos si se encuentran vacíos desde el principio o del mismo tipo de objeto el cual ya tienen empaquetados. Los autores modelan el problema como un *Type-constrained and Variable Sized Bin Packing Problem* (TVSBPP), y lo resuelven a través de un método de ramificación y acotamiento (*Branch and Bound*). Se propone un procedimiento de búsqueda en reversa (*Backtracking*) para mejorar los resultados del algoritmo.

Epstein et al. (2010) estudian una variante de CBPP, propuesto en (Shachnai y Tamir, 2004). Se tiene un conjunto dado de objetos, donde cada uno tiene (no negativo) un tamaño y color. Adicionalmente se tiene un parámetro entero k, y el objetivo es dividir los objetos en el número mínimo de subconjuntos tal que para cada subconjunto

S en la solución, el tamaño total de objetos en *S* sea no más que 1 (tal como en el problema clásico de empaquetamiento) y el número total de colores de los objetos en *S* sea no más que *k* (el cual distingue el problema de la versión clásica). Los autores siguen un trabajo anterior sobre este problema y estudian el problema en escenarios *offline* y *online*.

Khanafer et al. (2010) estudian *Bin Packing problem with Conflicts* (BPC). Este problema consiste en la minimización del número de repositorios a utilizar para empaquetar un conjunto de objetos, donde algunos objetos no pueden empaquetarse juntos en el mismo repositorio debido a restricciones de compatibilidad, que son dadas por una gráfica indirecta de conflictos. A partir de los conceptos conocidos, como función dual factible (*dual-feasible functions*, DFF) y función dual factible dependiente de datos (*data-dependent dual-feasible function*, DDFF), los cuales se utilizan para mejorar las soluciones de algunos problemas de corte (*cutting*) y empaquetamiento, se propone un framework general para derivar nuevas FFDD, y se introduce un nuevo concepto de funciones dual factibles generalizados dependientes de datos (*generalized data-dependent dual-feasible functions*, GDDFF), una generalización conflicto de DDFF. Se muestra que estas técnicas pueden ser usadas para mejorar los límites inferiores (*lower bounds, LB*) existentes.

Shachnai y Tamir (2004) definen el problema *Class-Constrained Bin Packing* donde los repositorios tienen una capacidad *v* y *c* compartimentos. En su problema cada objeto tiene el mismo tamaño y un color. Los objetos deben ser depositados en un repositorio, sujetos restricciones de capacidad tal que los objetos de diferentes colores se colocan en diferentes compartimentos; el objetivo es minimizar el número de repositorios utilizados.

En (Chung et al., 2006; Epstein y van Stee, 2007) el problema de empaquetamiento puede verse como un caso especial del problema de corte (*cutting*), donde el número de repositorios está restringido a 1 y cada objeto está caracterizado por un volumen y un valor. El problema de maximizar el valor de los objetos que pueden introducirse al repositorio se conoce como problema de la mochila (*Knapsack problem*). A pesar del hecho de que se trata de un problema NP-difícil, pueden producirse soluciones óptimas para instancias muy grandes con algoritmos sofisticados. Además, se han desarrollado muchas heurísticas: el algoritmo FF provee de una solución rápida pero pocas veces

óptima; involucra colocar cada objeto en el primer contenedor, en el cual cabe. Requiere de un tiempo $\Theta(n \log n)$, donde n es el número de elementos a ser empaquetados. El algoritmo puede hacerse más eficiente si primero se acomodan los elementos de la lista en orden decreciente (FFD), aunque esto aún no garantiza obtener una solución óptima, y se incrementa el tiempo de ejecución para listas largas. Sin embargo se sabe que por lo menos existe un orden de los objetos que proporciona una solución óptima (Lewis, 2009).

La revisión de la literatura presentada en el texto anterior muestra que no existe ninguna publicación que considere en conjunto 1) la división de contenedores de la capacidad equivalente en volúmenes más pequeños llamados aquí repositorios, 2) los repositorios de capacidad variable, 3) la división (fragmentación) de objetos a empaquetar, 4) restricciones de asignación de objetos a depositar a ciertos repositorios, y 5) colisiones de asignación entre objetos. Así mismo, este problema contiene nuevas características e interpretaciones y puede referirse como problema de empaquetamiento con repositorios de capacidad variable del contenedor, y restricciones de asignación (*The Fragmentable Object Bin Packing Problem with Container Repositories of Variable Capacity and Allocation Constrains,* FOBP-VRAC).

4.3 FORMULACIÓN MATEMÁTICA

Se considera el problema FOBP-VRAC. Las notaciones, suposiciones del problema y modelo matemático se presentan a continuación.

4.3.1 Notaciones

Índices

j	Índice de demanda, $j = 1,\ldots, N$.
r	Índice de reposito, $r = 1,\ldots, R$.
g	Tipo de lote, $g = 1,\ldots,G$.
l	Índice de lote, $l = 1,\ldots, L$.

Entrada del modelo

S_j	Tamaño del pedido j.

U	Tamaño del lote.
$V = [v_j]$	Lista de N códigos de productos (números de parte); v_j es el código del producto asociado con el pedido j;
$O = [o_{v_j,r}]$	Utilización del repositorio r en el pedido j cuyo código de producto es v_j; representa N cadenas $[o_{v_j,1}, o_{v_j,2}, ..., o_{v_j,R}]$ de R bits.
$K = [k_{gr}]$	Capacidad del repositorio r en el lote de tipo g.

Valores de decisión, variables y permutaciones.

π	Permutación de N pedidos, $\pi = \{j_1, j_2, \ldots, j_N\}$.
L	Cantidad de lotes.
σ	Lista de tipos de lotes, $\sigma = \{g_1, g_2, \ldots, g_l, \ldots, g_L\}$, $g_l \in \{1, \ldots, G\}$.
d_{ljr}	Cantidad de piezas del pedido j asignadas en el repositorio r del lote l.

4.3.2 Suposiciones del problema

- Se dispone de un inagotable suministro de lotes de todo tipo para procesar.
- El tamaño de cualquier lote es fijo y se considera igual a una constante U en este estudio. El lote, seleccionado para procesarse, se identifica completamente por su tipo g.
- La distribución de las piezas entre los repositorios es conocida y fija para cada tipo de lote.
- Todos los pedidos tienen la misma prioridad.
- Un pedido puede asignarse a uno o más repositorios de uno más lotes. El contenido de un repositorio se asigna a uno o más pedidos.
- Un pedido puede asignarse a cualquier tipo de lote.
- Un pedido j puede ser asignado solo en los repositorios asociados con éste a través del código del producto v_j.
- Una vez que inicia, un pedido se asigna en todos los lotes subsecuentes hasta terminar.

- Las piezas sobrantes de un lote, que no fueron consumidas por los pedidos forman el WIP. El contenido del WIP se considera para cumplir con futuros pedidos.

4.3.3 Modelo

El problema de optimización se define como:

$$\min_{\pi} L(\pi) = \text{Length}[\sigma^*(\pi)], \tag{4.1}$$

Sujeto a:

$$\sigma^*(\pi) = (g_1^*, \ldots, g_l^*, \ldots, g_L^*), \qquad g_l^* \in \{1, \ldots, G\}. \tag{4.2}$$

$$\sum_{j=1}^{N}\sum_{r=1}^{R} d_{ljr} \le \sum_{r=1}^{R} k_{gr} = U, \qquad l = 1, \ldots, L\ ;\ g = 1, \ldots, G. \tag{4.3}$$

$$\sum_{l=1}^{L}\sum_{r=1}^{R} d_{ljr} = S_j, \qquad j = 1, \ldots, N. \tag{4.4}$$

$$d_{ljr} \cdot o_{jr} \le k_{gr}, \qquad \forall\ l, r;\ g \in \{1, \ldots, G\}, \tag{4.5}$$

$$o_{jr} = \begin{cases} 1, \text{if } r \in V_j \\ 0, \text{otherwise} \end{cases} \qquad \forall\ r, j, \tag{4.6}$$

$$k_{gr}, d_{ljr} \in Z^0, \qquad \forall g, r, j, l. \tag{4.7}$$

La asignación de los elementos en los lotes se realiza dinámicamente de acuerdo con las siguientes fórmulas:

$$f_0 = 0; \tag{4.8}$$

$$f_l = f_{l-1} + \max_{g}\{\max_{j,r}\{\sum_{j=1}^{N}\sum_{r=1}^{R} d_{l_g jr}\}\}, g = 1, \ldots, G, \tag{4.9}$$

donde $d_{l_g jr}$ satisface las condiciones (4.3-4.5), g^* es el índice del valor voraz $\max_{j,r}\{\sum_{j,r} d_{l_g jr}\}$ en la iteración l. Este proceso se repite hasta que todos los elementos estén asignados, esto es, hasta $\sum_{l=1}^{L} f_l = \sum_{j=1}^{N} S_j$.

El objetivo es encontrar tal permutación π de pedidos que la longitud $L(\pi)$ de una lista σ^* basada en π (4.2), es mínima. El índice L del último elemento en σ^* denota el número de lotes obtenidos cuando los pedidos se asignan de acuerdo con la permutación π. La cantidad de elementos consumidos por los pedidos a partir de un lote se restringe por el tamaño del lote U (4.3). La restricción (4.4) indica que todos los pedidos deben ser asignados por completo; y (4.5) restringe la cantidad de piezas asignadas en el repositorio r con la cantidad k_{gr}, si el tipo de lote es g. El valor binario o_{jr} en (4.6) se define como: $o_{jr} = 1$ si el repositorio r puede ser utilizado para asignar el pedido j; en otro caso $o_{jr} = 0$. Los tamaños k_{gr} y d_{ljr} son los números enteros no negativos (4.7).

La solución del problema (4.1) incluye el valor desconocido d_{ljr}, el orden de procesamiento de los lotes σ^* incluyendo el índice de su último elemento L, y la permutación π.

4.4 ALGORITMOS

Como se mostró en el análisis de la literatura, se han empleado una gran variedad de técnicas para resolver las generalizaciones relacionadas con el problema de empaquetamiento. Las más utilizadas son las heurísticas, tales como NF, FF, FFD. Se han encontrado algoritmos de aproximación basados en heurísticas en los trabajos de Friesen y Lagnston (1986), Xing (2002), Kang y Park (2003), Chung et al. (2006), Haouari y Serairi (2009), Baldi et al. (2012). Soluciones basadas en el LB y algunos esquemas de aproximación, como APTAS y AFPTAS, se introducen en los trabajos de Murgolo (1987), Epstein y Levin (2008), Xavier y Miyazawa (2008), Khanafer et al. (2010), Crainic et al. (2011). El método de ramificación y acotamiento se utiliza para resolver el problema VSBP (Correia et al., 2008) y el VSBP con restricciones de color (Zhou et al., 2009). La complejidad de generalizaciones del problema de empaquetamiento motivan a los investigadores a desarrollar algoritmos de solución utilizando técnicas metaheurísticas y evolutivas.

Recientemente Hemmelmayr et al. (2012), Bang-Jensen y Larsen (2012) propusieron metaheurísticas de búsqueda *variable neighborhood* y de búsqueda local para abordar variantes del problema VSBP. Los autores de ambos artículos aplican elementos de programación matemática en componentes algorítmicos. Haouari y Serairi (2009)

proponen también el uso de algoritmos genéticos basados en orden (*order-based*) para el problema VSPB; la solución se calcula utilizando una variante del algoritmo NF. En la literatura no se reporta la aplicación de algoritmos metaheurísticos para solucionar problemas de empaquetamiento donde se permite la división (*splitting*) de los objetos.

Para solucionar el problema (4.1) se introduce el algoritmo genético MinLot (MinLot-GA). Este integra tres procedimientos: MinLot-GA encuentra la mejor permutación π de los pedidos utilizando el algoritmo MaxD para evaluar la aptitud del π; el algoritmo MaxD crea la lista $\sigma^*(\pi)$ de lotes seleccionando el mejor tipo de lote en cada iteración hasta que se asignan todos los pedidos, y llama al algoritmo *North West Packing* (NWP) como el método de solución de colisiones y la fragmentación de los pedidos.

4.4.1 Método de la esquina noroeste

El algoritmo NWP se basa en el método (regla) de la esquina noroeste (*North West Corner Rule*) que originalmente se utiliza en la programación matemática para obtener una solución factible del problema de trasporte donde comúnmente se tienen una cantidad j de almacenes con cierta capacidad a llenar y una cantidad i de centros de suministros con una capacidad finita de envió de material. A continuación se presenta los pasos de la regla de esquina noroeste clásica (Winston y Goldberg, 2004), (Ahuja et al., 1993).

North West Corner Rule:

1 Inicializar la matriz de esquina noroeste (i, j) con $i = 1$ y $j = 1$.
2 Asignar tanto unidades de recursos desde i hasta j en las celdas a través de $x_{i,j} = \min\{a_i, b_j\}$.
3 Ajustar el suministro $a_i = a_i - x_{i,j}$ y el pedido $b_j = b_j - x_{i,j}$. Si $a_i = 0$ entonces $i = i + 1$; y Si $b_j = 0$, $j = j + 1$.
4 Si aún existe alguna demanda por satisfacer y suministros por asignar, entonces seguir al paso 2.

4.4.2 Asignación de pedidos a los lotes

NWP recibe como entrada:

1) la matriz $D[N, R+1]$ con celdas (j, r), marcadas como no disponibles para asignar el pedido j, $j = 1,\ldots, N$, en el repositorio r, $r = 1, \ldots, R$, y el tamaño de pedido S_j en la columna $R+1$;

2) el vector $T[R]$ de unidades de suministro disponibles para asignarse en cada repositorio.

El algoritmo llena la matriz siguiendo la regla de la esquina noroeste asignando las unidades de suministro a las celdas disponibles y ajustando el valor de la columna $R+1$ de la matriz D.

Algoritmo NWP

Entrada:

$T[R]$	Lista de unidades de suministro;
$D[N, R+1]$	Matriz marcada con celdas disponibles y tamaño de los pedidos $d_{j,R+1} = S_j$ en la columna $R+1$.

Output:

$D[N,R+1]$ Matriz llena con las piezas asignadas d_{jr} del pedido;

$T[R]$ lista de unidades sobrantes.

1 Inicializar NWP(j, r) con $j = 1$ and $r = 1$.

2 Revisar las celdas del renglón j desde r hasta R, buscando celdas disponibles y asignando tantas unidades como sea posible de t_r a d_{jr} estableciendo $d_{jr} = \min(t_r, d_{j,R+1})$, ajustando los suministros $t_r = t_r - d_{jr}$, y el pedido $d_{j,\,R+1} = d_{j,R+1} - d_{jr}$.

3 Si t_r no es 0, y existe un valor $d_{j+1,R+1},\ldots, d_{N,R+1}$ no 0 con una celda r disponible, $r = 1,\ldots, R$, entonces $j = j + 1$, $r = 1$; ir al paso 2.

4 Regresar $D[N, R+1]$.

A continuación se presenta un ejemplo de la asignación realizada de acuerdo con el algoritmo NWP para la lista de pedidos $S = (30, 50, 20, 40)$, $N = 4$. El vector inicial de suministros es $T = (20, 30, 40, 10)$, y el número de repositorios es $R = 4$. Las colisiones ocurren en los repositorios 2 y 3.

Entrada: Matriz marcada

$$D = \begin{Bmatrix} - & 0 & 0 & - & 30 \\ 0 & 0 & - & - & 50 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 40 \end{Bmatrix}$$

$T = (20, 30, 40, 10)$

Salida: Matriz llena

$$D = \begin{Bmatrix} - & 30 & 0 & - & 0 \\ 20 & 0 & - & - & 30 \\ 0 & 0 & - & - & 20 \\ - & - & 40 & 0 & 0 \end{Bmatrix}$$

$T = (0, 0, 0, 10)$

4.4.3 Selección del tipo de lote

Para generar la lista de lotes σ^* se introduce el algoritmo MaxD. Es un algoritmo voraz iterativo, el cual crea G copias de la matriz $D[N, R+1]$ en cada iteración tomando en cuenta el tipo de lote. El algoritmo emplea de manera interna el algoritmo NWP como procedimiento para llenar las G copias de la matriz y después selecciona aquella que minimice la cantidad de las piezas no empaquetadas. Se toma el índice g^* de la copia seleccionada para asignar este índice en la siguiente posición de la lista σ. Este proceso se repite hasta que todos los pedidos sean asignados.

Algoritmo MaxD

Entrada:

$\pi[N]$ Permutación de pedidos;
$K[G, R]$ Matriz que contiene las capacidades para R repositorios para los G tipos de lotes;
$O[N, R]$ Matriz binaria de disponibilidad de R repositorios para N pedidos;
$S[N]$ Lista de tamaños de pedidos;
$V[N]$ Lista de códigos de productos.

Salida:

σ Lista de tipos de lotes;
C Lista de matrices D_l ; $l = 1,..., L$; L = Longitud[σ].

1 Crear la matriz $D_l[N,R+1]$; $l = 1$.
2 Asignar $d_{j,R+1} = S_j$; $j = 1,\ldots, N$.
3 Marcar las celdas (j, r) no disponibles en D_l tomando en cuenta $o_{v_j,r}$; $j = 1,\ldots, N$, $r = 1,\ldots, R$.
4 Ordenar los renglones de la matriz D_l en el orden de π.
5 Mientras $d_{j,R+1}$ no sea 0, $j = 1,\ldots, N$,
6 Crear G copias de D_l; asignar $CopyD_g = D_l$,
7 Para $g = 1$ hasta G

8 Llamara el algoritmo NWP enviando como parámetros de entrada $CopyD_g$ y las unidades de suministros k_g; asignar el resultado D_l a $CopyD_g$.

9 Seleccionar g^* con $\min \sum_{j=1}^{N} d_{g,j,R+1}$; entre $CopyD_g$, $g = 1,\ldots, G$.

10 Agregar g^* al final de la lista σ; asignar $D_l = CopyD_{g^*}$;

11 Agregar D_l al final de la lista C; $l=l+1$;

12 Hacer 0 las celdas (j, r) disponibles de acuerdo a $o_{v_j,r}$; $j = 1,\ldots, N$, $r = 1,\ldots, R$.

13 Regresar $[\sigma, C]$.

Ejemplo de la corrida del algoritmo para dos tipos de lotes:

$S_j = (50, 30, 40, 20)$, $V = [1, 2, 3, 1]$, $N = 4$, $R = 4$, $\pi = (2, 1, 4, 3)$.

$$o = \begin{Bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{Bmatrix} \qquad k = \begin{Bmatrix} 20 & 30 & 40 & 10 \\ 10 & 20 & 30 & 40 \\ 5 & 15 & 30 & 50 \\ 30 & 40 & 20 & 10 \end{Bmatrix}$$

El resultado de los pasos 1-4 es:

$$D = \begin{Bmatrix} 0 & 0 & - & - & 50 \\ - & 0 & 0 & - & 30 \\ - & - & 0 & 0 & 40 \\ 0 & 0 & - & - & 20 \end{Bmatrix}, \text{ después de ordenar } D = \begin{Bmatrix} - & 0 & 0 & - & 30 \\ 0 & 0 & - & - & 50 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 40 \end{Bmatrix}$$

Resultado de la primera iteración de los pasos 5-8:

$g = 1$

$$copyD_1 = \begin{Bmatrix} - & 0 & 0 & - & 30 \\ 0 & 0 & - & - & 50 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 40 \end{Bmatrix} \qquad copyD_1 = \begin{Bmatrix} - & 30 & 0 & - & 0 \\ 20 & 0 & - & - & 30 \\ 0 & 0 & - & - & 20 \\ - & - & 40 & 0 & 0 \end{Bmatrix}$$

$K_1 = [20, 30, 40, 10]$ $\qquad$ $K_1 = [0, 0, 0, 10]$

$g = 2$

$$copyD_2 = \begin{Bmatrix} - & 0 & 0 & - & 30 \\ 0 & 0 & - & - & 50 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 40 \end{Bmatrix} \qquad copyD_2 = \begin{Bmatrix} - & 20 & 10 & - & 0 \\ 10 & 0 & - & - & 40 \\ 0 & 0 & - & - & 20 \\ - & - & 20 & 20 & 0 \end{Bmatrix}$$

$K_2 = [10, 20, 30, 40]$ $\qquad$ $K_2 = [0, 0, 0, 20]$

$$g = 3 \quad copyD_3 = \begin{Bmatrix} - & 0 & 0 & - & 30 \\ 0 & 0 & - & - & 50 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 40 \end{Bmatrix} \qquad copyD_3 = \begin{Bmatrix} - & 15 & 15 & - & 0 \\ 5 & 0 & - & - & 45 \\ 0 & 0 & - & - & 20 \\ - & - & 15 & 25 & 0 \end{Bmatrix}$$

$$K_3 = [5, 15, 30, 50] \qquad K_3 = [0, 0, 0, 25]$$

$$g = 4 \quad copyD_4 = \begin{Bmatrix} - & 0 & 0 & - & 30 \\ 0 & 0 & - & - & 50 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 40 \end{Bmatrix} \qquad copyD_4 = \begin{Bmatrix} - & 30 & 0 & - & 0 \\ 30 & 10 & - & - & 10 \\ 0 & 0 & - & - & 20 \\ - & - & 20 & 10 & 10 \end{Bmatrix}$$

$$K_4 = [30, 40, 20, 10] \qquad K_4 = [0, 0, 0, 0]$$

En el paso 9: Se selecciona el tipo de lote g = 4 debido a que corresponde $\min \sum_{j=1}^{N} d_{g,j,R+1}$; esto quiere decir, que la selección del tipo de lote 4 permite la asignación de la máxima cantidad de elementos.

En el paso 10:

$$\sigma = [4];\ D = copyD_4 = \begin{Bmatrix} - & 30 & 0 & - & 0 \\ 30 & 10 & - & - & 10 \\ 0 & 0 & - & - & 20 \\ - & - & 20 & 10 & 10 \end{Bmatrix}$$

Los pasos 5 a 12 se ejecutan dos veces obteniendo el resultado final:

$$g = 1 \quad copyD_1 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 0 & 0 & - & - & 10 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 10 \end{Bmatrix} \qquad copyD_1 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 10 & 0 & - & - & 0 \\ 10 & 10 & - & - & 0 \\ - & - & 10 & 0 & 0 \end{Bmatrix}$$

$$K_1 = [20, 30, 40, 10] \qquad K_1 = [0, 20, 30, 10]$$

$$g = 2 \quad copyD_2 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 0 & 0 & - & - & 10 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 10 \end{Bmatrix} \qquad copyD_2 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 10 & 0 & - & - & 0 \\ 0 & 20 & - & - & 0 \\ - & - & 10 & 0 & 0 \end{Bmatrix}$$

$$K_2 = [10, 20, 30, 40] \qquad K_2 = [0, 0, 20, 40]$$

$$g = 3 \quad copyD_3 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 0 & 0 & - & - & 10 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 10 \end{Bmatrix} \quad copyD_3 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 5 & 5 & - & - & 0 \\ 0 & 10 & - & - & 10 \\ - & - & 10 & 0 & 0 \end{Bmatrix}$$

$$K_3 = [5, 15, 30, 50] \qquad K_3 = [0, 0, 20, 50]$$

$$g = 4 \quad copyD_4 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 0 & 0 & - & - & 10 \\ 0 & 0 & - & - & 20 \\ - & - & 0 & 0 & 10 \end{Bmatrix} \quad copyD_4 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 10 & 0 & - & - & 0 \\ 20 & 0 & - & - & 0 \\ - & - & 10 & 0 & 0 \end{Bmatrix}$$

$$K_4 = [30, 40, 20, 10] \qquad K_4 = [0, 40, 10, 10]$$

El resultado final es:

$$\sigma = [4,1];\ D = copyD_1 = \begin{Bmatrix} - & 0 & 0 & - & 0 \\ 10 & 0 & - & - & 0 \\ 10 & 10 & - & - & 0 \\ - & - & 10 & 0 & 0 \end{Bmatrix};$$

Esto quiere decir que el algoritmo MaxD asignó cuatro pedidos en dos lotes: uno del tipo 4 y uno del tipo 1.

4.4.4 Minimización del número de lotes

Existen múltiples estudios sobre el problema del empaquetamiento en desarrollos offline y online. Un algoritmo *online* asigna piezas a los repositorios en el orden cómo estos se entregan al empaquetamiento (orden original), sin el uso de ningún tipo de conocimiento previo acerca de los trabajos subsecuentes en la lista; ver por ejemplo (Ye y Zhang, 2009). En el caso de desarrollo *offline*, se dispone de la lista entera de trabajos para realizar el empaquetamiento (Coffman et al., 2007). El último desarrollo corresponde exactamente con el caso de esta investigación ya que se considera que se cuenta con la información sobre los pedidos de antemano.

Para resolver el problema (4.1), se introduce el algoritmo genético MinLot-GA, el cual busca la permutación de los pedidos que minimice el número de lotes necesarios utilizando además los algoritmos MaxD y NWP como los procedimientos internos. El diagrama de flujo de un algoritmo genético general se presenta en la Figura 4.4.

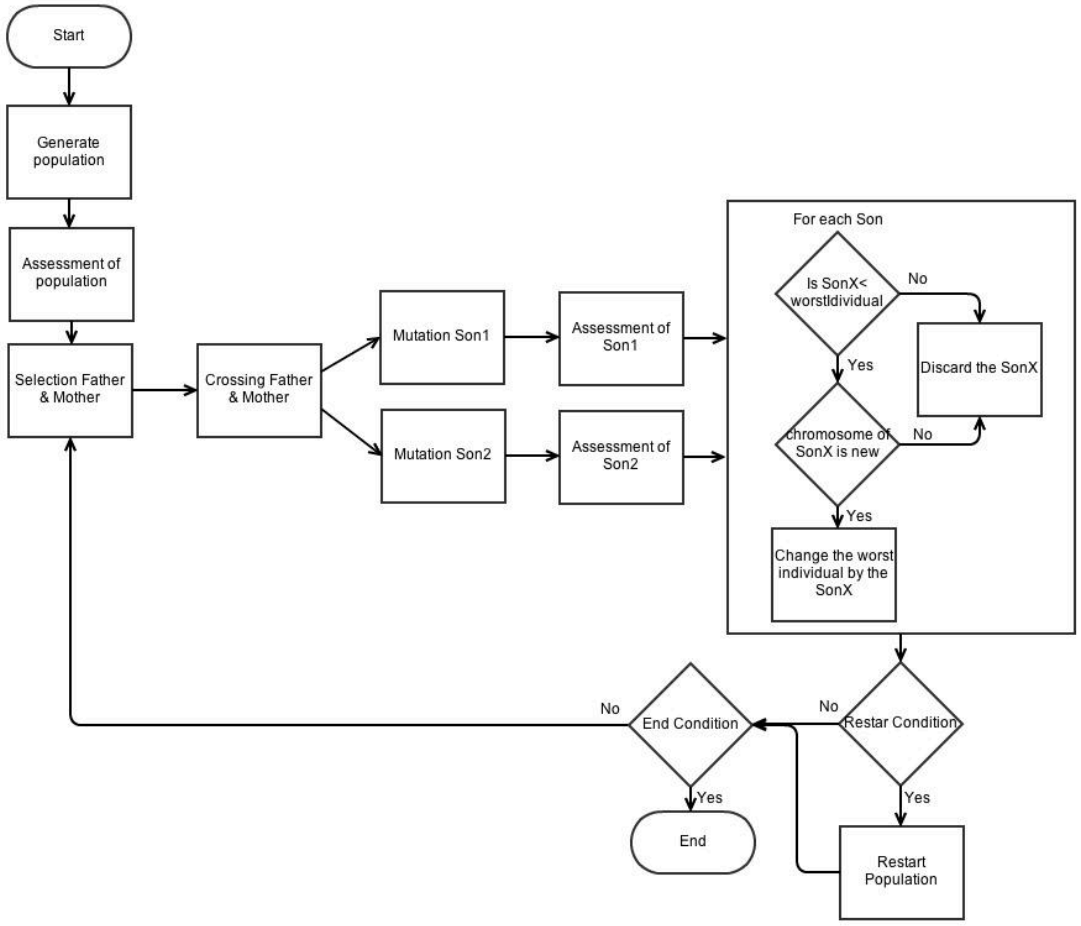

Figura 4.4 Diagrama de flujo general de un algoritmo genético

A continuación, se describen las características del algoritmo MinLot-GA.

Codificación

Se utiliza una codificación basada en orden (*order-based*). Una permutación de demandas se codifica en el cromosoma de un individuo.

Inicialización y evaluación de aptitud

Se crea la matriz de población $P[P_{size}, N]$ asignando a cada renglón, el cual representa el cromosoma del individuo en la población, de forma aleatoria sin repetición N números desde 1 a N. La longitud de la lista σ^* obtenida por el algoritmo MaxD se utiliza como una función objetivo para evaluar el valor de aptitud del cromosoma de los individuos.

Reinicio y criterio de parada

La población se renueva después de un número de iteraciones (generaciones), si el algoritmo genético no obtiene un individuo con un valor de aptitud mejor (menor); se conservan el 20% de los mejores individuos, 30% se generan a partir del cruce de los individuos previamente seleccionados del 20%, y el 50% restante se generan aleatoriamente. El contador de reinicio se establece en cero. El algoritmo genético se ejecuta hasta un número fijo asignado de iteraciones.

Selección, cruzamiento y mutación

La selección de los padres se realiza considerando el *torneo binario*: Se toman dos individuos de la población utilizando el método de ruleta y seleccionado el individuo con el mejor (menor) valor de aptitud. El proceso se repite para obtener el segundo padre.

La operación de cruzamiento genera nuevos individuos a través de la combinación de dos individuos (padres). El objetivo es generar soluciones con mejores valores de aptitud. El operador de cruce se aplica con una probabilidad P_c. Se consideran cuatro operadores de cruce, de los cuales tres se han aplicado en la búsqueda de la mejor permutación en programación de la producción y el operador de cruce TP de uso común:

Two-Point (TP) (Michalewicz, 1998): Se seleccionan dos puntos de manera aleatoria dentro de la permutación. Los elementos se copian del padre 1 al hijo desde la primera posición hasta el primer punto y del segundo punto hasta la última posición. El resto de los elementos se copian del padre 2 al hijo desde el primer punto hasta el segundo punto.

Order Based Crossover (OBX) (Gen y Cheng, 1999): el operador emplea una máscara binaria para determinar que alelo de los padres será tomado para formar el nuevo individuo (hijo). El valor de la máscara igual a 1 indica que el elemento correspondiente se copia del padre 1 al hijo. El resto de los elementos se copian del padre 2. Los valores de la máscara se generan aleatoria y uniformemente en todas las operaciones de cruzamiento.

Precedence Preservative Crossover (PPX) (Bierwirth et al., 1996): el operador se basa en una máscara binaria. El valor de la máscara igual a 1 indica que el elemento

correspondiente se copia del padre 1 al hijo y un valor igual a 0 indica que ese elemento se copia del padre 2 de acuerdo a cada valor de la máscara uno a la vez.

One Segment Crossover (OSX) (Guinet y Solomon, 1996): Se seleccionan dos puntos de manera aleatoria dentro de la permutación. Los elementos se copian del padre 1 iniciando desde la posición 1 hasta el primer punto, luego se copian los elementos del padre 2 desde el primer punto hasta el segundo. Finalmente, los elementos se copian del padre 1 desde el segundo punto hasta el último considerando solo los elementos cuyo valor no fueron copiados anteriormente.

Uno de los problemas más comunes cuando se utilizan algoritmos genéticos es que las soluciones a menudo caen en óptimos locales. Para evitar tal situación, se incorpora el uso de un operador de mutación, el cual altera el cromosoma de los individuos introduciendo una variabilidad en la población. Se emplea el operador de mutación SWAP con un valor de probabilidad P_m.

Mutación SWAP (Gen y Cheng, 1999): Se seleccionan de manera aleatoriamente dos alelos en el cromosoma del individuo y se intercambian.

Inserción de nuevos individuos: Después de una iteración del algoritmo, se debe revisar si el nuevo individuo es mejor que el peor individuo en la población, si lo es, entonces el nuevo individuo toma el lugar del peor en la población.

Control de clones: Antes de insertar el nuevo individuo en la población se verifica que la secuencia del cromosoma no exista ya en la población; de existir el nuevo individuo se descarta.

A continuación se describe el algoritmo MinLot-GA.

Entrada:

P_c	Probabilidad de cruce;
P_m	Probabilidad de mutación;
P_{size}	Tamaño de la población;
$S[N]$	Lista de tamaños de los pedidos;
$O[N, R]$	Matriz binaria de disponibilidad de los R repositorios para N códigos de productos;
$V[N]$	Lista de códigos de productos;
$K[G, R]$	Matriz de capacidades para R repositorios para G tipos de lote;
restart	El número de generaciones antes de reiniciar la población;

stop_criterion El número de generaciones hasta detener la ejecución del algoritmo genético.

Salida:

π^* Permutación de pedidos

1 Inicialización:

Crear la población $P[P_{size}, N]$,

crear $FITNESS[P_{size}]$, asignando 0 a cada posición,

hacer *generations counter* = 0.

2 Calcular el valor de aptitud para P_{size} individuos de la población llamando el algoritmo MaxD. Guardar los resultados en la posición correspondiente de *FITNESS*. $FITNESS[i] = \text{MaxD}(D, O, V, K, R, P_i)$, $i = 1,\ldots, P_{size}$.

3 Seleccionar aleatoriamente dos individuos como padres de la población.

4 Generar un número aleatorio {0, 1}. Si el número es mayor que P_c entonces aplicar un operador de cruzamiento (TP, OSX, OBX o PPX) a los padres para obtener los nuevos individuos (hijos). Si no entonces se pasan los padres como los hijos.

5 Generar un número aleatorio {0, 1}. Si el número es mayor que P_m, entonces aplicar el operador de mutación SWAP a los cromosomas de los hijos.

6 Calcular el valor de aptitud para los hijos e intentar insertar los nuevos individuos en la población. Incrementar *generations_counter*;

7 Si se logra insertar un nuevo individuo en la población, entonces

Si el nuevo individuo tiene un valor de aptitud mejor que el valor de él para el mejor individuo de la población, entonces hacer *bestcounter* = 0;

Si no, entonces hacer *bestcounter* = *bestcounter* +1;

Si no, hacer *bestcounter* = *bestcounter* +1.

8 Si *bestcounter* = *restart*, entonces se reinicia la población y se evalúa el valor de aptitud para los individuos de la nueva población.

9 Si *generation_counter* < *stop_criterion*, entonces ir a 3

10 Se extrae el individuo con el mejor valor de aptitud de la población y se hace π^*=*cromosoma del mejor individuo.*

11 FIN

MinLot-GA se define como un *algoritmo genético dinámico* debido a que incluye procedimientos internos de optimización: Algoritmo heurístico voraz MaxD basado en fragmentación NWP para construir la solución del problema (4.1) utilizando búsqueda genética como entrada.

4.5 ALGORITMO ALTERNATIVO

La forma de operar en la empresa requiere que desde el principio se indique un orden de procesamiento de pedidos, lo que restringe la posibilidad de encontrar condiciones en

las cuales se aprovecha la fragmentación de los pedidos y con esto se reduce el tiempo de ejecución. Con la hipótesis de que es posible encontrar una solución que no esté ligada con una permutación sino con el peso del pedido o alguna otra regla de prioridad, se buscarían mejores resultados. Como experimento adicional se propone un algoritmo que aprovecha las características de MaxD en asignación con la diferencia de que en cada iteración los pedidos se ordenan con base en algún tipo de prioridad, como el tamaño del pedido, o número de repositorios disponibles.

El algoritmo offline MinLot-Alternativo provee una solución al problema (4.1) considerando prioridades para el orden de procesamiento de pedidos después de selección de cada lote (tipo), en lugar utilizar una permutación fija.

A continuación se describe el algoritmo MinLot-Alternativo.

Entrada:

$K[G, R]$ Matriz que contiene las capacidades para R repositorios para los G tipos de lotes;

$O[N, R]$ Matriz binaria de disponibilidad de R repositorios para N pedidos;

$S[N]$ Lista de tamaños de pedidos;

$V[N]$ Lista de códigos de productos.

w Tipo de prioridad (Max ó Min número de piezas por pedido)

Salida:

σ Lista de tipos de lotes;

C Lista de matrices D_l ; $l = 1,..., L;\ L = \text{Length}[\sigma]$.

1 Crear la matriz $D_l[N,R+1]$; $l = 1$.

2 Asignar $d_{j,R+1} = S_j$; $j = 1,\ldots, N$.

3 Marcar las celdas (j, r) no disponibles en D_l tomando en cuenta $o_{v_j,r}$; $j = 1,\ldots, N$, $r = 1,\ldots, R$.

4 Ordenar los renglones de la tabla utilizando una regla de prioridad indicada por la variable *peso* (orden ascendente o descendente del tamaño de la demanda, o cantidad de repositorios disponibles mínima o máxima).

5 Mientras $d_{j,R+1}$ no sea $0, j = 1,\ldots, N$,

6 Crear G copias de D_l; asignar $CopyD_g = D_l$,

7 Para $g = 1$ hasta G

8 Llamara el algoritmo NWP enviando como parámetros de entrada $CopyD_g$ y las unidades de suministros k_g; asignar el resultado D_l a $CopyD_g$.

9 Seleccionar g^* con $\min \sum_{j=1}^{N} d_{g,j,R+1}$ entre $CopyD_g,\ g = 1,\ldots, G$.

10 Agregar g^* al final de la lista σ; asignar $D_l = CopyD_{g^*}$;

11 Agregar D_l al final de la lista C; $l=l+1$;

12 Hacer 0 las celdas (j, r) disponibles de acuerdo a $o_{v_j,r}$; $j = 1,\ldots, N$, $r = 1,\ldots, R$.

13 Regresar [σ, C].

El siguiente ejemplo se presenta para ilustrar la ejecución del algoritmo: se tienen tres pedidos de tamaño 70(2), 100, (1,2), 75(2, 3), donde los valores en los paréntesis representan los repositorios permitidos para utilizar. Se tienen dos tipos de lotes ($G1$, y $G2$) con 3 repositorios cada uno. Los tamaños de los repositorios para $G1$ son 10, 20 y 30 y para $G2$ son 20, 30 y 50.

La Figura 4.5 muestra la primera iteración del algoritmo. En cuatro iteraciones el algoritmo obtiene como resultado L = (2,2), σ = [1,1,2,2] y se generan 110 piezas sobrantes del WIP. Si se considerara solo el uso del contenedor $G1$, el resultado es L_{G1}=(5,0), σ_{G1}=[1,1,1,1,1] y 255 piezas sobrantes.

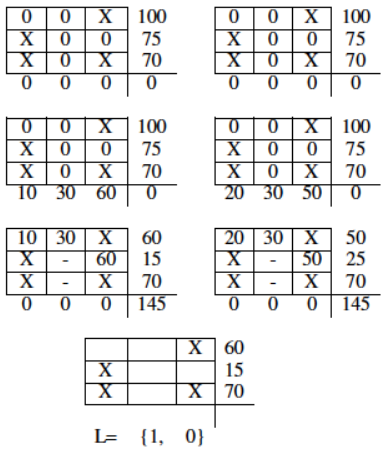

Figura 4.5 Ejemplo de la primera iteración del algoritmo

El algoritmo presentado previamente procesa los pedidos en base a un orden de prioridad que provoca que los pedidos se reordenen en cada iteración y que por consiguiente los pedidos que inician en una iteración no necesariamente siguen procesándose en la siguiente. En general, los pedidos se procesan por partes según su importancia por iteración de tal modo que el algoritmo procesa todos los pedidos de una manera homogénea.

Este algoritmo busca terminar con todos los pedidos de mayor prioridad primero y de manera colateral termina los trabajos con la menor prioridad. Sin embargo, el mecanismo de reordenación de los pedidos en cada iteración hace necesario mantener

un control de cada parte procesada de los pedidos, lo que en la realidad sería muy complicado de implementar en un proceso productivo, donde normalmente se comienza a procesar un pedido hasta terminarlo, y el único tipo de interrupción que existe es por espera de material.

4.6 CONCLUSIONES DEL CAPÍTULO

En esta sección, el problema se interpreta como una generalización del problema de empaquetamiento. Se describen algunas características del problema para diferenciarlo del problema clásico de empaquetamiento. Se presenta una revisión de la literatura, donde se describen estudios relacionados con el problema de empaquetamiento con características similares a las utilizadas en esta investigación. Se introducen las notaciones y modelo matemático que describen el problema de minimización de la cantidad de lotes necesarios para satisfacer un conjunto de pedidos. Se presenta análisis de diferentes metodologías, heurísticas y metaheurísticas que se han desarrollado para resolver generalizaciones del problema de empaquetamiento similares a la propuesta.

Se presenta el algoritmo NWP basado en la regla de esquina noroeste utilizado originalmente para obtener una solución factible al problema de trasporte. Este algoritmo resuelve el problema de asignación de las demandas a los lotes.

Se resuelve el problema de selección de tipo de lote con el algoritmo original MaxD el cual utiliza el algoritmo NWP como un procedimiento interno.

Se introduce el algoritmo genético MinLot-GA que busca la solución al problema entre las permutaciones de los pedidos a procesar. Se hace una descripción de las características del algoritmo genético. El algoritmo MinLot-GA incluye el algoritmo MaxD como un procedimiento interno para evaluar la aptitud de los resultados.

Por último, se presenta un algoritmo alternativo para solucionar el problema de minimización de lotes a través de una modificación del algoritmo MaxD. Este algoritmo busca la solución tomando en cuenta alguna regla de prioridad que ordena los pedidos en cada iteración provocando que todos los pedidos se resuelven de acuerdo a la prioridad seleccionada. Se presenta resultado comparativo de MinLot-GA contra MinLot-Alternativo; se aprecia que en la mayoría de los casos MinLot-GA obtiene mejores resultados, sin embargo existen 3 casos donde MinLot-alternativo consigue

mejorar el resultado. El algoritmo MinLot-alternativo presenta deficiencias altas con respecto a MinLot-GA, sin embargo se plantea la posibilidad de continuar los estudios del problema buscando soluciones que aprovechen la capacidad de fraccionamiento de los pedidos y no a través de la permutación de estos.

5. EXPERIMENTO COMPUTACIONAL

Los algoritmos introducidos se estudiaron y se compararon en un experimento computacional. Los algoritmos se implementaron en el lenguaje de programación Java y se ejecutaron en un WorkStation MacPro con dos procesadores Xeon Quad-Core de 2.4 GHz y 8 GB de RAM. El resultado de cada problema se almacenó y consolidó en un libro de Excel.

5.1 PARÁMETROS DEL ALGORITMO GENÉTICO

Se sabe que la eficacia de un algoritmo genético depende del tipo de representación, y de la selección de los operadores de cruzamiento y mutación, así como a la probabilidad de que estos operadores se apliquen. Para realizar una selección de los parámetros se implementó un experimento factorial.

A continuación se muestran los niveles de factor utilizados en el experimento:

- Operadores de cruzamiento, cuatro niveles: TP, OSX, OBX and PPX;
- Probabilidad de cruce (P_c), tres niveles: 0.1 y 0.3;
- Probabilidad de mutación (P_m), tres niveles: 0.01 and 0.02.

Los niveles de los factores anteriormente mencionados dan como resultado un total de $4 \times 2 \times 2 = 16$ combinaciones.

Los siguientes parámetros se fijaron para el algoritmo genético:

- Criterio de paro: 100 generaciones;
- Tamaño de la población (P_s): 20 individuos;
- Selección: Ruleta;
- Generaciones antes del reinicio de la población: 25 generaciones sin mejora.

5.2 INSTANCIAS DEL PROBLEMA PARA CALIBRACIÓN

Se generaron tres grupos de 10, 20 y 30 demandas, cada uno con 50 instancias obteniendo un total de 150 problemas. El tamaño para cada demanda se creó utilizando

un generador lineal congruente para obtener números aleatorios entre 100 y 10,000. El código de producto para cada demanda se seleccionó aleatoriamente entre 20 variantes (ver Tabla 5.1).

Tabla 5.1. Lista de códigos de productos

Código	Repositorios	Código	Repositorios	Código	Repositorios	Código	Repositorios
1	1-2	**6**	5-9	**11**	10-10	**16**	14-19
2	2-3	**7**	6-7	**12**	10-11	**17**	15-17
3	3-5	**8**	8-9	**13**	11-11	**18**	16-19
4	4-5	**9**	9-9	**14**	11-13	**19**	19-21
5	4-9	**10**	9-14	**15**	12-13	**20**	20-21

La Tabla 5.2 representa la matriz *K*, cuyos renglones describen los elementos que pertenecen a un lote de cierto tipo entre los repositorios después de clasificarse. Esta tabla se creó a partir de datos reales obtenidos de una compañía dedicada a la manufactura de interruptores eléctricos (*reed switches*) como se presento en la sección 2. Los datos de la matriz *K* están normalizados para un tamaño de lote de 5000 piezas.

Tabla 5.2. Distribución de cuatro tipos de lotes entre 25 repositorios

G	Repositorios																								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
1	145	230	470	575	730	585	525	380	280	230	195	150	110	90	80	65	55	20	15	25	25	5	5	5	5
2	30	50	90	195	305	500	625	575	555	495	395	290	275	170	120	120	70	35	25	35	15	15	5	5	5
3	35	30	70	145	175	290	430	430	440	420	430	390	350	265	265	185	200	125	140	85	60	25	5	5	5
4	10	10	20	20	65	65	110	200	240	260	260	445	375	485	490	430	395	405	275	250	180	0	5	0	5

5.3 REGLAS DE PRIORIDAD PARA MAXD

Para establecer que tan buenos son los resultados obtenidos por MinLot-GA, es necesario compáralo con otras técnicas que resuelvan el mismo problema. El algoritmo MaxD crea la lista de lotes para una secuencia de demandas dada; por lo tanto puede ser adaptada para evaluar MinLot-GA empleando diferentes reglas de prioridad para ordenar las demandas.

Se aplicaron cuatro reglas de prioridad para simular las decisiones tomadas en un proceso de producción real acerca del orden en el que se deben ejecutar las demandas:

natural, aleatorio, decreciente (Max) y creciente (Min), utilizando las mismas instancias que MinLot-GA. La regla de orden natural toma las demandas en el orden en como fueron recibidas (no se realiza ningún cambio). La regla de orden aleatorio genera una permutación de tamaño *N*, donde cada posición de pedido se asigna aleatoriamente en el rango de 1 a *N* sin repetición. Para las reglas Max y Min, los pedidos se ordenan tomando en cuenta su tamaño. Se debe señalar que no existe ningún método formal en la planta para asignar los pedidos a los lotes y resolver colisiones. Por lo tanto, las cuatro reglas de prioridad mencionadas se refuerzan con el algoritmo NWP para asignar los pedidos a los lotes.

Se utilizaron las instancias empleadas en la evaluación de MinLot-GA resolviendo los mismos 150 problemas y obteniendo 150×4 = 600 resultados.

5.4 EJECUCIÓN DEL ALGORITMO

Para aumentar la precisión del experimento, y dado que en cada ejecución de MinLot-GA es posible obtener diferentes resultados, se realizan dos réplicas del diseño completo. Con esto se resuelven 150×24×3 = 10,800 problemas evaluando un total de 10,800×100 = 1,080,000 permutaciones.

Después de iniciar la corrida de MinLot-GA, se detectó que los diferentes niveles de factor obtuvieron prácticamente los mismos resultados. Esto es debido a que el mecanismo de reinicio retiene los mejores individuos y genera nuevos a partir del cruce de los mejores, causando una rápida convergencia de los resultados con la región del óptimo. Dado que solo se obtienen pequeñas diferencias entre los resultados de los diferentes niveles de factor, se establecen los siguientes parámetros para reducir el tiempo de ejecución del experimento:

- Operador de cruzamiento: OSX;
- Probabilidad de cruce (P_c): 0.3;
- Probabilidad de mutación (P_m): 0.02.

La Tabla 5.3 muestra el tiempo promedio que requieren los algoritmos para resolver una instancia.

Tabla 5.3. Tiempo empleado para la ejecución de la heurística

Cantidad de demandas	MinLot-GA (Horas:Minutos:Segundos)		MaxD (Segundos:Milisegundos)			
	Todos los niveles	Un nivel	Natural	Aleatorio	Max	Min
10	00:36:00	00:02:30	01:267	00:712	00:604	00:697
20	01:32:24	00:03:51	01:499	01:030	00:888	01:175
30	03:54:24	00:09:46	02:558	02:172	02:312	02:748

La Tabla 5.4 muestra los resultados del experimento. MinLot-GA obtiene mejores resultados en comparación con las variantes de MaxD para todas las instancias.

Tabla 5.4. Cantidad de lotes resultante para las instancias, que contienen 10, 20 y 30 demandas

10						20						30					
Instancia	MinLot-GA	Natural	Aleatorio	Max	Min	Instancia	MinLot-GA	Natural	Aleatorio	Max	Min	Instancia	MinLot-GA	Natural	Aleatorio	Max	Min
1	13	21	17	17	18	**1**	18	21	21	21	21	**1**	34	35	36	37	36
2	15	15	15	15	16	**2**	26	37	33	28	30	**2**	44	58	52	49	57
3	19	23	22	22	23	**3**	23	25	26	26	25	**3**	26	32	34	28	32
4	47	48	48	48	49	**4**	21	28	25	24	28	**4**	48	62	53	49	52
5	18	18	19	18	21	**5**	25	30	32	27	36	**5**	39	44	42	50	47
6	14	15	19	16	17	**6**	23	28	28	26	25	**6**	37	48	42	42	41
7	12	14	14	12	15	**7**	18	23	23	23	20	**7**	34	44	42	42	36
8	14	17	16	15	16	**8**	35	44	39	35	42	**8**	30	42	31	33	37
9	12	15	12	14	13	**9**	22	30	26	29	25	**9**	33	39	33	39	34
10	10	10	11	11	10	**10**	24	27	26	25	30	**10**	55	72	68	57	69
11	15	17	16	15	17	**11**	40	42	42	42	46	**11**	41	46	54	48	51
12	11	13	13	11	12	**12**	23	27	29	25	31	**12**	33	38	36	36	41
13	21	24	21	22	22	**13**	27	28	30	42	30	**13**	34	39	37	45	37
14	13	16	15	15	15	**14**	22	25	24	24	26	**14**	50	59	52	66	51
15	11	12	14	14	13	**15**	68	70	72	70	74	**15**	52	67	58	55	58
16	23	25	23	25	25	**16**	23	28	26	28	27	**16**	30	31	35	32	33
17	13	15	13	14	18	**17**	47	49	49	49	50	**17**	37	43	46	39	49
18	16	19	21	18	20	**18**	18	20	20	21	19	**18**	44	55	51	46	56
19	33	35	34	34	36	**19**	24	32	32	27	26	**19**	33	48	35	42	45
20	16	17	17	21	16	**20**	41	47	42	43	45	**20**	28	33	39	30	41
21	16	19	20	17	21	**21**	26	30	28	29	30	**21**	32	35	35	36	33
22	12	13	13	14	15	**22**	20	25	24	21	28	**22**	28	33	33	32	34
23	29	30	34	29	36	**23**	34	37	37	37	37	**23**	54	67	57	55	60
24	14	17	18	18	16	**24**	24	26	29	27	28	**24**	31	40	34	33	35
25	17	18	19	19	20	**25**	24	28	33	31	27	**25**	71	73	83	74	82
26	13	14	14	17	14	**26**	20	22	21	23	21	**26**	35	44	42	46	41
27	9	11	12	12	11	**27**	20	23	22	25	26	**27**	33	42	43	40	41
28	15	18	17	18	16	**28**	28	32	32	29	31	**28**	33	41	40	38	38
29	7	8	8	8	8	**29**	33	40	39	36	37	**29**	31	34	37	40	35
30	20	21	21	21	23	**30**	20	22	21	22	22	**30**	29	35	32	31	32
31	7	7	8	7	9	**31**	28	29	29	29	33	**31**	29	36	30	32	31
32	18	19	19	19	20	**32**	22	28	29	25	30	**32**	51	60	52	54	53
33	9	12	12	10	10	**33**	25	27	27	26	29	**33**	42	59	59	53	46
34	50	51	52	52	51	**34**	26	33	35	30	31	**34**	34	51	39	37	40
35	13	16	15	15	15	**35**	23	28	23	28	25	**35**	51	56	54	55	61
36	58	58	58	59	60	**36**	20	26	25	22	22	**36**	32	33	35	35	36
37	15	16	16	16	16	**37**	21	24	29	23	28	**37**	30	34	36	31	40
38	14	17	17	17	17	**38**	21	26	24	27	21	**38**	33	35	35	35	39

39	16	17	19	17	19	**39**	42	60	57	51	49	**39**	35	37	52	50	45
40	11	13	13	12	13	**40**	20	22	27	23	24	**40**	39	50	47	43	42
41	15	19	18	15	18	**41**	28	34	32	31	32	**41**	29	33	37	41	33
42	14	16	16	14	17	**42**	57	58	58	58	58	**42**	32	36	37	35	35
43	13	14	16	13	17	**43**	47	50	48	48	49	**43**	33	41	36	38	37
44	20	24	20	20	23	**44**	23	27	25	25	25	**44**	38	41	45	51	44
45	19	24	23	20	23	**45**	33	40	37	34	41	**45**	84	89	86	90	89
46	14	16	17	17	15	**46**	20	22	23	21	29	**46**	38	43	45	48	46
47	31	36	36	33	36	**47**	22	23	25	26	26	**47**	35	37	41	40	41
48	13	13	14	18	14	**48**	25	27	26	26	31	**48**	73	77	77	74	84
49	16	19	19	16	23	**49**	22	26	26	23	27	**49**	31	41	43	38	32
50	41	44	44	44	41	**50**	20	21	20	20	20	**50**	45	51	51	61	50

Las Figuras 5.1, 5.2, y 5.3 revelan que los resultados de MinLot-GA son menores que los resultados de los otros cuatro algoritmos.

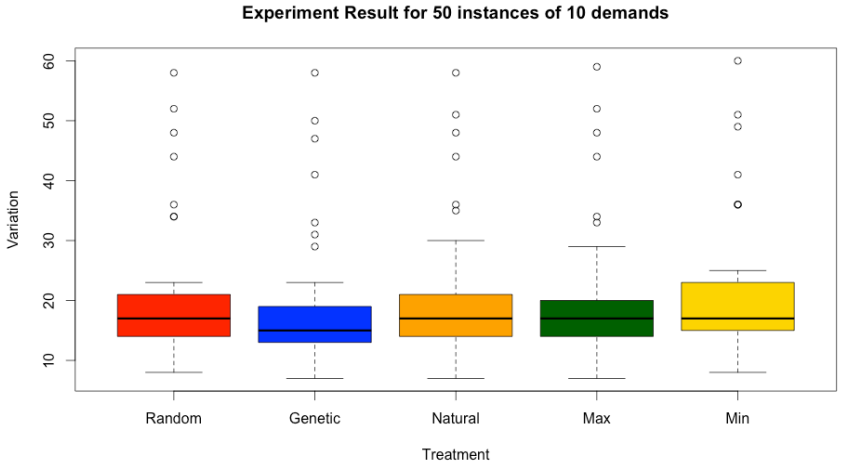

Figura 5.1. Gráfica de caja para la variación de tratamiento de 50 instancias con 10 demandas

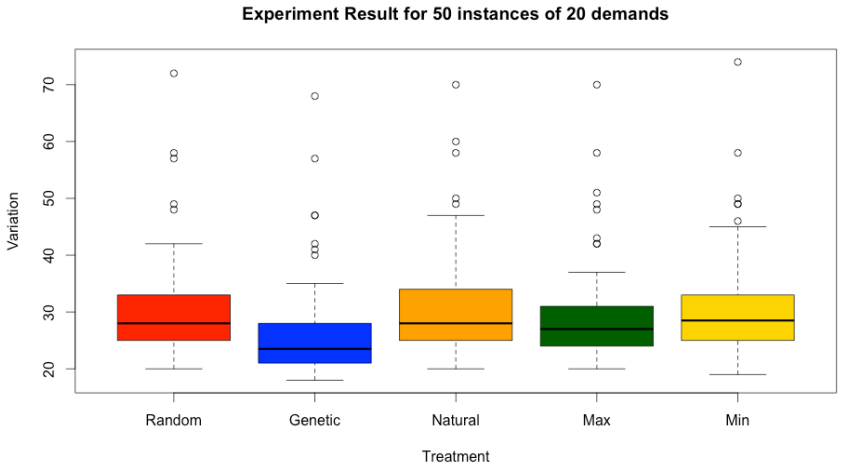

Figura 5.2. Gráfica de caja para la variación de tratamiento de 50 instancias con 20 demandas

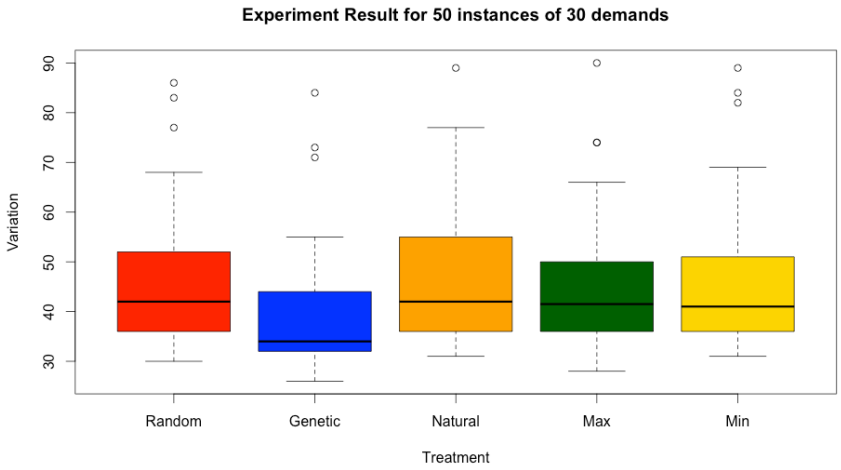

Figura 5.3. Gráfica de caja para la variación de tratamiento de 50 instancias con 30 demandas

5.5 ANÁLISIS ESTADÍSTICO DE RESULTADOS

Se realiza un análisis estadístico de los resultados del experimento. Se asume que los resultados no siguen ninguna distribución conocida y las instancias son independientes, por lo tanto se aplica la prueba ***U*** (***U-test***) no paramétrica de una cola de Mann-Whitney (Spiegel y Stephens, 2007) para comparar MinLot-GA contra los cuatro algoritmos heurísticos basados en MaxD. El análisis se realiza en el ambiente estadístico computacional R. Se prueba la hipótesis H0: $\mu_1 >= \mu_2$ contra la alternativa H1: $\mu_1 < \mu_2$ donde μ_1 y μ_2 son los valores promedio obtenidos por los algoritmos comparados.

La Tabla 5.5 muestra el resultado obtenido con la prueba ***U*** para un valor de significancia $\alpha = 0.05$.

*Tabla 5.5. Resultados de la prueba **U** para 50 instancias de 10, 20 y 30 pedidos*

No. Demands	Algorithm μ_1	Algorithm μ_2	Hypothesis	Wilcox	p-value	Accepted
10	MinLot-GA	Natural	H0: $\mu_1 > \mu_2$	950.5	0.0194	H1
		Random		946.5	0.01813	H1
		Max		996	0.0399	H1
		Min		912.5	0.009937	H1
20		Natural		950.5	0.0004914	H1
		Random		946.5	0.000887	H1
		Max		996	0.002798	H1
		Min		912.5	0.0006005	H1
30		Natural	H1: $\mu_1 < \mu_2$	950.5	0.0001339	H1
		Random		946.5	0.0003337	H1
		Max		996	0.001297	H1
		Min		912.5	0.0005282	H1

Los valores de probabilidad obtenidos son considerablemente menores a 0.05 en todos los casos, por lo tanto se rechaza la hipótesis nula H0 y se acepta la hipótesis alternativa H1, esto es, MinLot-GA tiene el mejor desempeño. Más aun, el desempeño de MinLot-GA es mejor mientras la cantidad de pedidos a asignar es mayor.

5.6 COMPARACIÓN CON ALGORITMO ALTERNATIVO

El algoritmo MinLot-GA resuelve el problema 4.1 a través de la búsqueda de la solución entre las permutaciones estableciendo una secuencia fija de procesamiento de los pedidos sin considerar ningún tipo de peso o importancia de algún pedido. En algunas situaciones se requiere que algún pedido se termine lo más pronto posible. Esto se logra buscando la solución entre los pesos o importancias tal como lo hace el algoritmo de la sección 4.5, a través de el reacomodo de la secuencia de ejecución de los pedidos. Sin embargo, al forzar el modelo a terminar algún pedido en particular, se provoca que el tiempo de ejecución se extienda para el resto de los pedidos, dado que solo se controla el tipo de material requerido para el pedido de mayor peso y el material que requieren el resto de los pedidos de la lista de ejecución.

Adicionalmente para ejecutar el algoritmo alternativo se requiere mantener el estado de cada parte de los pedidos que ya se ejecutaron, esto hace difícil implementarlo en una empresa de manufactura dado que normalmente se cuenta con almacenes limitados que normalmente sirven para mantener material terminado listo para envió. La Tabla 5.6 muestra una comparación de los resultados obtenidos por el algoritmo MinLot-GA y dos variantes del algoritmo MinLot-Alternativo utilizando como prioridad los pedidos en el orden descendiente (Max) y ascendente (Min).

Tabla 5.6 Comparativo de resultados entre MinLot-GA y el algoritmo MinLot-Alternativo

No.	Cantidad de trabajos por instancias								
	10			20			30		
	GA	Max	Min	GA	Max	Min	GA	Max	Min
1	13	17	18	18	43	32	34	54	43
2	15	35	35	26	45	31	44	97	92
3	19	27	24	23	34	30	26	35	33
4	47	56	56	21	46	46	48	76	65
5	18	25	23	25	53	48	39	72	68
6	14	35	26	23	54	56	37	62	56
7	12	12	11	18	47	46	34	78	65
8	14	36	36	35	55	49	30	56	54
9	12	17	16	22	30	30	33	50	43
10	10	23	23	24	60	60	55	84	76
11	15	16	17	40	75	73	41	68	67
12	11	25	26	23	43	39	33	52	51
13	21	16	16	27	56	57	34	71	60
14	13	32	28	22	61	61	50	74	64
15	11	26	26	68	80	76	52	101	92
16	23	43	44	23	38	35	30	62	56
17	13	31	31	47	57	60	37	72	62
18	16	29	28	18	38	34	44	70	68
19	33	53	52	24	36	35	33	85	77
20	16	26	26	41	75	72	28	50	49
21	16	25	20	26	36	33	32	44	36
22	12	23	23	20	39	36	28	42	40
23	29	40	40	34	69	66	54	86	76
24	14	34	28	24	57	53	31	68	56
25	17	41	36	24	53	45	71	104	101
26	13	33	30	20	37	35	35	77	71
27	9	17	17	20	34	28	33	57	53
28	15	32	28	28	80	71	33	52	56
29	7	18	16	33	58	59	31	60	60
30	20	39	34	20	46	38	29	48	45
31	7	32	32	28	80	80	29	72	70
32	18	40	40	22	33	30	51	67	66
33	9	21	21	25	41	41	42	76	66
34	50	54	54	26	54	55	34	61	45
35	13	26	25	23	60	61	51	86	78
36	58	70	70	20	53	47	32	42	40
37	15	34	34	21	48	41	30	61	53
38	14	29	29	21	47	46	33	72	75
39	16	24	25	42	68	59	35	63	62
40	11	23	22	20	37	37	39	70	67
41	15	24	23	28	36	36	29	67	66
42	14	30	27	57	114	114	32	61	57
43	13	17	16	47	56	57	33	79	76
44	20	29	29	23	51	46	38	76	64
45	19	49	42	33	72	64	84	111	101
46	14	24	24	20	36	30	38	75	69
47	31	39	36	22	40	37	35	54	52
48	13	23	20	25	48	49	73	100	95
49	16	33	32	22	33	34	31	74	71
50	41	45	49	20	32	33	45	75	69

6 CONCLUSIONES, CONTRIBUCIONES Y TRABAJO FUTURO

Se estudia un problema real de manufactura de interruptores eléctricos. Este se interpreta como una generalización del problema de empaquetamiento, al que se refiere en esta investigación como el problema FOBP-VRAC. Este representa un problema complejo de optimización combinatoria, el cual combina algunos aspectos interesantes: división de contenedores de igual capacidad en volúmenes más pequeños llamados repositorios, división de objetos a empaquetar, restricciones de asignación de objetos a los repositorios. Hasta el momento, no se ha encontrado alguna publicación que considere estas condiciones juntas. Además, se estudian aspectos no investigados anteriormente, como repositorios (subcontenedores) de capacidad variable, mientras el tamaño del contenedor es fijo, las colisiones en asignación entre los objetos. La interpretación de un lote como un contenedor también es original.

El estado del arte muestra que la generalización del problema de empaquetamiento que permite división de objetos no ha sido lo suficientemente estudiada; mientras que no es un problema exclusivo de la manufactura de interruptores eléctricos. Las aplicaciones se encuentran en diferentes sistemas de fabricación modernos con procesamiento por lotes. El termino *ítem*, que comúnmente se emplea en problemas de empaquetamiento para indicar los objetos a empaquetar, no se recomienda utilizar, si se permite la división de estos, dado que se supone indivisibilidad. El termino *objeto* utilizado por Mandal et al. (1998) para el FOBP, es apropiado para este tipo de problemas.

Se presenta un modelo matemático con componentes dinámicos. Se introduce por primera vez una metaheurística, en particular un algoritmo genético, para la solución del problema de empaquetamiento donde los objetos se permiten fraccionar arbitrariamente.

El algoritmo MinLot-GA es un algoritmo genético dinámico basado en orden. Contiene procedimientos internos de optimización: el algoritmo voraz MaxD para evaluar la aptitud y el algoritmo NWP, el cual representa una adaptación original de la regla de esquina noroeste para el empaquetado de objetos fragmentables a través de resolución de colisiones. Las pruebas estadísticas mostraron que MinLot-GA obtiene

mejores resultados en comparación con cuatro heurísticas propuestas. La complejidad computacional de FOBP-VRAC no puede deducirse directamente de NP-difícil problema FOBP (Mandal et al., 1998), debido a que el costo de la fragmentación no se considera en las suposiciones del problema FOBP-VRAC. La complejidad del problema considerado proviene de las colisiones, que ocurren en los repositorios de capacidades variables, el cual requiere de un estudio especial, como un caso particular de este problema.

Las contribuciones de la investigación son las siguientes:

1. Se introduce un nuevo tipo de división de trabajos (*splitting jobs*) en problemas de planificación que ocurre de una manera natural después de la clasificación de piezas de un lote de acuerdo al valor de un parámetro (o varios).
2. Por primera vez se investiga un problema que tiene dos tipos de división (*spliting*) a la vez: división de pedidos y división de lotes. El problema es real, y proviene de una planta que manufactura interruptores eléctricos.
3. Una nueva generalización del problema de empaquetamiento FOBP-VRAC, que se caracteriza por:
    - División de contenedores de capacidad equivalente en volúmenes más pequeños llamados repositorios.
    - Repositorios de capacidad variable.
    - División (fragmentación) de objetos a empaquetar.
    - Restricciones de asignación de objetos a depositar a ciertos repositorios.
    - Colisiones de asignación entre objetos.
4. Se introduce por primera vez una metaheurística para solucionar el problema de empaquetamiento, donde se permite fragmentar los objetos arbitrariamente.
5. Una estructura original del algoritmo genético que evalúa la aptitud, a través del algoritmo MaxD y de manera interna utiliza el algoritmo NWP para fraccionar los objetos y resolver colisiones.
6. Se propone una nueva generalización del problema de Johnson para dos máquinas, junto con la interpretación en manufactura con procesamiento por lotes, y algoritmo exacto para el caso considerado.

Se considera el siguiente trabajo futuro:

- Investigación de casos particulares del problema FOBP-VRAC que tienen importancia teórica y práctica.
- Comprobación de la complejidad del problema FOBP-VRAC.
- Búsqueda de algoritmos más eficientes para resolver el problema.
- Investigación de la complejidad de la generalización de Johnson.

ACRÓNIMOS

A

AFPTAS - Asymptotic Fully Polynomial Time Approximation Scheme 46, 49, 55
APTAS - *Asymptotic Polynomial Time Approximation Scheme* 49, 50, 55, 4

B

BFD - Best Fit Decreciente. 45, 50
BPC - Bin Packing problem with Conflicts ... 51
BP-IF - Bin Packing problem with Item Fragmentation 48
BPOS - Bin Packing Oversized Item..... 45
BP-SIF - Bin Packing with Size-Increasing Fragmentation 48
BP-SPF - Bin Packing with Size Preserving Fragmentation 48

C

CATV - Community Antenna Television 48, 49
CCABP - Class Constrained Shelf Bin Packing Problem..... 50
CCBP - Class-Constrained Bin Packing problem 47, 50

D

DDFF - Data-dependent Dual-feasible Function 51

F

FF - First Fit..... 46, 48, 51, 55
FFD - First Fit Decreciente.... 44, 45, 48, 50, 52, 55
FFDLR - First Fit Decreasing using Largest bins, at end Repack to smallest possible bins. 44
FFDLS - First Fit Decreasing using Largest bins, but Shifting if necessary..... 44
FFL- Flexible flow line, FFL; 18
FOBP - Fragmentable Object Bin Packing.... 48, 78, 79
FOBP - Fragmentable Object Bin Packing problem. 48, 78, 79
FOBP-VRAC - Fragmentable Object Bin Packing Problem with Container Repositories of Variable Capacity and Allocation Constrains (problema bajo investigación)..... V, VII, 52, 78, 79, 80
FSMP - Flow shop with multiple processors . 18

G

GBPP - Generalized Bin Packing Problem 46
GDDFF - Generalized Data-dependent Dual-feasible Functions 51
GT - Group Technology 19, 20

H

HFFL - Hybrid flexible flow shop or Flexible hybrid flow line 18
HFS - Hybrid Flow Shop ... I, 12, 13, 14, 15, 16, 17, 18, 19, 28, 34

J

JIT - Just-In-Time.....25

L

LB - Lower bound 37, 51, 55

M

max-batch - parallel batching 21
MaxD – Maximum Distribution Algorithm I, V, VII, 56, 58, 61, 62, 65, 66, 68, 71, 73, 75, 78, 79
MinLot-GA - Minimum Lot Genetic Algorithm IV, V, VII, 56, 61, 62, 64, 65, 68, 71, 72, 73, 74, 75, 76, 77, 78
MPFS - Flow shop with multiple processors ..18
MRP - Material requirements planning.....25

N

NF - Next Fit 44, 48, 55, 56
NFL - Next Fit Larges.....44
NWP - North West Packing Algorithm... V, VII, 56, 57, 58, 59, 61, 65, 66, 68, 72, 78, 79

O

OBX - Order Based Crossover.....63, 65, 70
OPT - Optimized Production Technology 25
OSX - One Segment Crossover.....64, 65, 70, 72

P

p-batch - parallel batching.....21
PCB - Printed Circuit Board,.....13
PHFS - Parallel hibrid flow shop 18
Pm - Máquinas paralelas identicas 64, 65, 70, 72
PPX - Precedence Preservative Crossover 63, 65, 70

S

s-batch - serial batching.....21
sum-batch - serial batching 21

T

TVSBPP - Type-constrained and Variable Sized Bin Packing Problem 45, 50

V

W

LISTA DE PUBLICACIONES

1. R. Romero Parra and Larysa Burtseva. Implementation of Bin Packing Model for Reed Switch Production Planning. IAENG Transactions on Engineering Technologies. American Institute of Physics Publisher, Editor: Sio-Iong Ao, *Melville, NY,* ISSN 0094-243X, ISBN: 978-0-7354-0794-7, V. 1247, Pp. 403-412.
2. Romero Parra R., Burtseva L. Problema de Elección de Material en la Planificación de la Producción de Interruptores Electrónicos. Congreso Nacional de Estudiantes de Posgrado del Instituto de Ingeniería, UABC Mexicali B.C, 25-27 de Noviembre de 2009 Programa de Maestría y Doctorado en Ciencias e Ingeniería, 8 p., Editorial UABC, ISBN: 978-607-7753-33-9.
3. Romero Parra R., Burtseva L. Bin packing model implementation for material election in read switches production planning. Lecture Notes in Engineering and Computer Science. Publisher: Newswood Limited International Association of Engineers, Hong Kong. S.I. Ao Craig Douglas, W.S. Grunfest Jon Burgstone (Eds.), 2009, V. 2179, I. 1, P. 1095-1100, ISSN: 20780958, ISBN: 978-988-18210-2-7.
4. Romero R., Burtseva L. Problema de cumplimientos de pedidos en un modelo basado en procesamiento por lotes: Complejidad. 2do Congreso Nacional de Estudiantes de Posgrado del Instituto de Ingeniería, Instituto de Ingeniería – UABC, Maestría y Doctorado en Ciencias e Ingeniería, Nov. 2010, ISBN:, 8 p.
5. L. Burtseva, R. Garcia, R. Romero. Herramienta Informática de Realidad Aumentado para ubicación de objetos en ambientes industriales (HIRA). Registro Público del Derecho de Autor. Certificado de SEP, No. De Registro 03-2012-10051 1372800-01, 19 Octubre 2012.
6. Larysa Burtseva, Rainier Romero, Salvador Ramirez, Victor Yaurima, Fernando Gonzalez-Navarro & Pedro Flores Perez. Lot Processing in Hybrid Flow Shop Scheduling Problem. In “Production Scheduling”, InTech Publisher, Croatia, Edited by: Dr. Rodrigo Righi, ISBN 978-953-307-935-6. 2011, pp. 65-96.
7. Ramirez Rodriguez S., Romero Parra R., Burtseva L. Modelado de operaciones estocásticas en producción de dispositivos electrónicos. 32 Congreso internacional de ingeniería electrónica ELECTRO 2010, 13-15 de Octubre del 2010, IT Chihuahua, Chihuahua, Chih., V. XXXII, ISSN 1405-2172. p. 464-469.
8. Larysa Burtseva, Victor Yaurima and Rainier Romero Parra. Scheduling Methods for Hybrid Flow Shops with Setup Times, In: “Future Manufacturing Systems”, SCIYO, Edited by Dr. Tauseef Aized, ch. 7, pp. 137-162, 2010, Croatia. ISBN: 978-953-307-128-2, Sciyo.

REFERENCIAS

Ahuja, R.K., Magnanti, T.L. y Orlin, J.B. (1993). Network Flows: Theory, Algorithms and applications. New Jersey: Prentice Hall.

Allahverdi, A., Ng, C.T., Cheng, T.C.E. y Kovalvov, M.Y. (2008). A survey of scheduling problems with setup times or costs. European Journal of Operational Research, 187(3), 985-1032.

Andrés, C., Albarracin, J., Tormo, G., CVicente, E. y Garcia-Sabaster, J. (2005). Group technology in a hybrid flowshop environment: a case study. European Journal of Operational Research, 168(1), 272-281.

Baker, K.R. y Jia, D. (1993). A comparative study of lot streaming procedures. OMEGA International Journal of Management Sciences, 21(5), 561-566.

Baker, K.R. y Pyke, D.F. (1990). Solution procedures for the lot- streaming problem. Decision Sciences, 21(3), 475-491.

Baker, K.R. y Trietsh, D. (2009). Principles of Sequencing and Scheduling (1 ed.): Willey & Sons, Inc Publicaction.

Baldi, M.M., Crainic, T.G., Perboli, G. y Tadei, R. (2012). The generalized bin packing problem. Transportation Research Part E, 48(6), 1205-1220.

Bang-Jensen, J. y Larsen, R. (2012). Efficient algorithms for real-life instances of the variable size bin packing problem. Computers & Operations Research, 39(11), 2848-2857.

Bierwirth, C., Mattfeld, D. y Kopfer, H. (1996). On permutation representations for scheduling problems. In: H.-M. Voigt, W. Ebeling, I. Rechenberg y H.-P. Schwefel, Parallel Problem Solving from Nature — PPSN IV (Vol. 1141, pp. 310-318): Springer Berlin Heidelberg.

Błażewich, J., Ecker, K., Pesch, E., Schmidt, G. y Węglarz, J. (2007). Handbook on scheduling: From Theory to Applications: Springer.

Brucker, P. (2004). Scheduling Algorithms: Springer.

Burbidge, J.L. (1975). The introduction of group technology. London: Heinemann.

Çetinkaya, F.C. y Duman, M. (2010). Lot streaming in a two-machine mixed shop. International Journal of Advanced Manufacturing Technology, 49, 1161-1173.

Chen, L., Bostel, N., Dejax, P., Cai, J.C. y Xi, L.F. (2007). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. European Journal of Operational Research, 181(1), 40-58.

Chung, F., Graham, R., Mao, J. y Varghese, G. (2006). Parallelism versus Memory Allocation in Pipelined Router Forwarding Engines. Theory of Computing Systems, 39(6), 829-849.

Coffman, E.G. y Csirik, L.J. (2007). Performance Guarentees for One-Dimensional Bin Packing. In: Handbook of Approximation Algorithms and Metaheuristics (pp. 1,432). University of California, Santa Barbara, USA: Chapman and Hall/CRC.

Coffman, E.G., Joseph, Y.T. y Csirik, L.J. (2007). Variable-Sized Bin Packing and Bin Covering. In: Handbook of Approximation Algorithms and Metaheuristics (pp. 1432). University of California, Santa Barbara, USA: Chapman and Hall/CRC.

Correia, I., Gouveia, L. y Saldanha-da-Gama, F. (2008). Solving the variable size bin packing problem with discretized formulations. Computers & Operations Research, 35(6), 2103-2113.

Crainic, T.G., Perboli, G., Rei, W. y Tadei, R. (2011). Efficient lower bounds and heuristics for the variable cost and size bin packing problem. Computers & Operations Research, 38(11), 1474-1482.

Crama, Y. (1997). Combinatorial Optimization Models for Production Scheduling in Automated Manufacturing Systems. European Journal of Operational Research, 99(1), 136-153.

Dawande, M., Kalagnanam, J. y Sethuraman, J. (2001). Variable Sized Bin packing with Color Constraints. Electronic Notes in Discrete Mathematics, 7, 154-157.

Epstein, L., Imreh, C. y Levin, A. (2010). Class constrained bin packing revisited. Theoretical Computer Science, 411(34-36), 3073-3089.

Epstein, L. y Levin, A. (2008). An APTAS for generalized cost variable sized bin packing. SIAM Journal on Computing, 38(1), 411-428.

Epstein, L., Levin, A. y van Stee, R. (2012). Approximation Schemes for Packing Splittable Items with Cardinality Constraints. Algorithmica, 62(1-2), 102-129.

Epstein, L. y van Stee, R. (2007). Online bin packing with resource augmentation. Discrete Optimization, 4(3-4), 322-333.

Epstein, L. y van Stee, R. (2011). Improved Results for a Memory Allocation Problem. Theory of Computing Systems, 48(1), 79-92.

Friesen, D.K. y Lagnston, M.A. (1986). Variable sized bin packing. SIAM Journal on Computing, 15(1), 222-230.

Gen, M. y Cheng, R. (1999). Genetic algorithms & engineering optimization: Wiley & Sons.

Glass, C.A., Gupta, J.N.D. y Pott, C.N. (1994). Lot streaming in three-stage production processes. European Journal of Operational Research, 75(2), 378-394.

Glenberg, A.M. y Andrzejewki, M.E. (2008). Learning From Data: An Introduction to Statistical Reasoning (3ra ed.). New York: Lawrence Erlbaum Associates.

Glover, F. y Laguna, M. (1997). Tabu search. Boston: Kluwer Academic Publishers.

Graves, S.C. y Kostreva, M.M. (1986). Overlapping operations in material requirements planning. Journal of Operations Management, 6(3), 283-294.

Guinet, A.G.P. y Solomon, M.M. (1996). Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. International Journal of Production Research, 36(6), 1643-1654.

Gunasekaran, A., McNeil, R., McGaughey, R. y Ajasa, T. (2001). Experiences of a small to medium size enterprise in the design and implementation of manufacturing cells. International Journal of Computer Integrated Manufacturing, 14(2), 212-223.

Gupta, J.N.D. (1988). Two-stage, hybrid flowshop scheduling problem. Journal of the Operational Research Society, 39(4), 359-364.

Hadjinicola, G.C. y Kumar, K.R. (1993). Cellular manufacturing at champion irrigation products. International Journal of Operations and Production Management, 13(9), 53-61.

Haouari, M. y Serairi, M. (2009). Heuristics for the variable sized bin-packing problem. Computers & Operations Research, 36(10), 2877-2884.

Hemmelmayr, V.C., Schmid, V. y Blum, C. (2012). Variable neighbourhood search for the variable sized bin packing problem. Computers & Operations Research, 39(5), 1097–1108.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P. y Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. Computers & Operations Research, 36(2), 358-378.

Kang, J. y Park, S. (2003). Algorithms for the variable sized bin packing problema. European Journal of Operational Research, 147(2), 365-372.

Khanafer, A., Clautiaux, F. y Talbi, E.-G. (2010). New lower bounds for bin packing problems with conflicts. European Journal of Operational Research, 206(2), 281-288.

Kusiak, A. (1987). The generalized group technology concept. International Journal of Production Research, 25, 561-569.

Langston, M.A. (1984). Performance of heuristics for a computer resource allocation problem. SIAM Journal on Algebraic Discrete Methods, 5(2), 154-161.

Lewis, R. (2009). A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. Computers & Operations Research, 36(7), 2295-2310.

Li, S. (1997). A hybrid two-stage flowshop with part family, batch production, major and minor setups. European Journal of Operations Research, 102(1), 142-156.

Lin, H.-T. y Liao, C.-J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. International Journal of Production Economics, 86(2), 133-143.

Liu, C.-Y. y Chang, S.-C. (2000). Scheduling Flexible Flow Shops with Sequence-Dependent Setup Effects. IEEE Transactions on Robotics and Automation, 16(4), 408-419.

Liu, J. (2008). Single-job lot streaming in m-1 two-stage hybrid flowshops. European Journal of Operational Research, 187(3), 1171-1183.

Lushchakova, I.N. y Strusevich, V.A. (2010). Strusevich. Scheduling incompatible tasks on two machines. European Journal of Operational Research, 200(2), 334-346.

Mandal, C.A., Chakrabarti, P.P. y Ghose, S. (1998). Complexity of fragmentable object bin packing and an application. Computers & Mathematics with Applications, 35(11), 91-97.

Menakerman, N. y Rom, R. (2001). Bin Packing with Item Fragmentation. In: Proceedings of the 7th International Workshop on Algorithms and Data Structures (pp. 313-324): Springer-Verlag.

Michalewicz, Z. (1998). Genetic Algorithms + Data Structures = Evolution Programs: Springer.

Mitrofanov, S.P. (1966). Scientific Principles of Group Technology (Harris E. Yorkshire Trans.). Boston Spa: National Lending Library for Science and Technology.

Montgomery, D.C. y Runger, G.C. (2011). Applied Statistics and Probability for Engineers (5th ed.): John Wiley & Sons.

Murgolo, F.D. (1987). An efficient approximation scheme for variable-sized bin packing. SIAM Journal on Computing, 16(1), 149-161.

Naderi, B., Zandieh, M., Khaleghi, A., Ghoshe, B. y Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-

dependent setup and transportation times to minimize total completion time and total tardiness. Expert Systems with Applications, 36(6), 9625-9633.

Namman, N. y Rom, R. (2001). Analysis of Transmission Scheduling with Packet Fragmentation. Discrete Mathematics and Theoretical Computer Science, 4, 139-156.

Pan, Q.-K., Suganthan, P.N., Liang, J.J. y Tasgetiren, M.F. (2011). A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem. Expert Systems with Applications, 38(4), 3252-3259.

Petrov, V.A. (1966). Flowline Group Production Planning. London: Business Publications.

Pinedo, M.L. (2008). Scheduling: Theory Algorithms, and Systems. New York: Systems, Springer Science+Business Media.

Pott, C.N. y Baker, K.R. (1989). Flow shop scheduling with lot streaming. Operations Research Letters, 8, 297-303.

Potts, C.N. y Kovalyov, M.Y. (2000). Shceduling with batching: A review. European Journal of Operational Research, 120, 228-249.

Potts, C.N. y Van Wassenhove, L.N. (1992). Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. Journal of the Operations Research Society, 43, 395-406.

Reiter, S. (1966). A system for managing job shop production. Journal of Business, 34, 371-393.

Ruiz, R., Sirvrikaya, F. y Şerifoğlu, T.U. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. Computers & Operations Research, 35(4).

Ruiz, R. y Vázquez-Rodríguez, J.A. (2010). The hybrid flow shop scheduling problem. European Journal of Operational Research, 205(1), 1-18.

Shachnai, H. y Tamir, T. (2004). Tight bounds for online class-constrained packing. Theoretical Computer Science, 321(1), 103-123.

Spiegel, M. y Stephens, L. (2007). Schaum's Outline of Statistics (4th ed.): McGraw Hill Professional.

Tahar, D.N., Yalaoui, F. y Amodeo, L. (2006). A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times. International Journal of Production Economics, 99, 63-73.

Tatikonda, M.V. y Wemmerlöv, U. (1992). Adoption and implementation of group technology classification and coding systems: Insights from seven case studies. International Journal of Production Research, 30, 2087-2110.

Truscott, W.G. (1986). Production scheduling with capacity constraned trasportation activities. Journal of Operations Management, 6, 333-348.

Umble, M.M. y Srikanth, L. (1990). Synchronous Manufacturing: Principles for World Class Excellence. Wallingford, CT, USA: The Spectrum Publishing Company.

Vairaktarakis, G. (2004). Flexible Hybrid Flowshop,. In: J.Y.-T. Leung, Handbook of Scheduling: Algorithms, Models, and Performance Analysis (pp. 1120). Boca Raton, Florida: Chapman and Hall/CRC.

Vickson, R.G. y Alfredson, B.E. (1992). Two- and three-machine flow shop scheduling problem with equal seized transfer batches. International Journal of Production Research, 20, 1551-1574.

Vignier, A., Billaut, J.-C. y Proust, C. (2010). Les problèmes d'ordonnancement de type flow-shop hybride : état de l'art. RAIRO - Operations Research, 33(2), 117-183.

Wemmerlöv, U. y Hyer, N.L. (1989). Cellular manufacturing in the US industry, a survey of current practices. International Journal of Production Research, 27, 1511-1530.

Winston, W.L. y Goldberg, J.B. (2004). Operations Research: Applications and Algorithms (4th ed.): Thomson/Brooks/Cole.

Xavier, E.C. y Miyazawa, F.K. (2008). A one-dimensional bin packing problem with shelf divisions. Discrete Applied Mathematics, 156, 1083-1096.

Xing, W. (2002). A bin packingproblem with over-sized ítems. Operations Research Letters, 30(2), 83-88.

Xing, W. y Zhang, J. (2000). Parallel machine scheduling with splitting jobs. Discrete Applied Mathematics, 103, 259-269.

Yazdani, M.T.S. y Jolai, F. (2010). Optimal methods for bathc processing problem with makespan and maximum lateness objetives. Applied Mathematical Molelling, 34, 314-324.

Ye, D. y Zhang, G. (2009). On-line extensible bin packing with unequal bin sizes. Discrete Mathematics and Theoretical Computer Science, 11(1), 141-152.

Zandieh, M., Ghomi, S.M.T.F. y Husseini, S.M.M. (2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Applied Mathematics and Computation, 180(1), 111-127.

Zhang, W., Yin, C., Liu, J. y Linn, R. (2005). Multi-job lot streaming to minimize the mean completion time in m - 1 hybrid flowshops. International Journal of Production Economics, 96, 189-200.

Zhou, C., Wu, C. y Feng, Y. (2009). An exact algorithm for the tupe-constrained and variable sized bin packing problem. Ann Oper Res, 172(1), 193-202.