PROPUESTA DE UNA METAHEURÍSTICA PARA EL PROBLEMA DE TAMAÑO DE LOTE MULTIPRODUCTO NO CAPACITADO CON COTAS DE INVENTARIO

MARIA CAMILA MONSALVE BLANCO 331108

DANIEL JAIMES CARVAJAL 295664

UNIVERSIDAD PONTIFICIA BOLIVARIANA

ESCUELA DE INGENIERÍAS

FACULTAD DE INGENIERÍA INDUSTRIAL

SECCIONAL BUCARAMANGA

2020

PROPUESTA DE UNA METAHEURÍSTICA PARA EL PROBLEMA DE TAMAÑO DE LOTE MULTIPRODUCTO NO CAPACITADO CON COTAS DE INVENTARIO

MARIA CAMILA MONSALVE BLANCO 331108

DANIEL JAIMES CARVAJAL 295664

Trabajo de grado para optar al título de Ingeniero Industrial

Asesor

ANDRÉS FELIPE ACEVEDO OJEDA

PhD. Ingeniería industrial

UNIVERSIDAD PONTIFICIA BOLIVARIANA

ESCUELA DE INGENIERÍAS

FACULTAD DE INGENIERÍA INDUSTRIAL

SECCIONAL BUCARAMANGA

2020

Tabla de Contenidos

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: PROPUESTA DE UNA METAHEURÍSTICA PARA EL PROBLEMA DE TAMAÑO DE LOTE MULTIPRODUCTO NO CAPACITADO CON COTAS DE INVENTARIO

AUTOR(ES): Maria Camila Monsalve Blanco
Daniel Jaimes Carvajal

PROGRAMA: Facultad de Ingeniería Industrial

DIRECTOR(A): Andrés Felipe Acevedo Ojeda

RESUMEN

En el presente proyecto, se considera un problema de tamaño de lote multiproducto no capacitado con cotas de inventario, donde se realiza un plan de producción para múltiples ítems teniendo en cuenta que el inventario de estos productos compartirá un mismo espacio de almacenamiento. En primer lugar, se establece la estructura matemática del algoritmo usando programación lineal-entera mixta para luego resolverlo usando IBM ILOG CPLEX. Posteriormente, se presentan tres versiones de una metaheurística de enfriamiento simulado. La primera versión se basa en un algoritmo encontrado en la literatura. Basado en las desventajas encontradas en la primera versión, se diseña una segunda versión utilizando una diferente variable de decisión como input para la solución inicial. A partir de modificaciones de la versión dos y del estudio de las variables de decisión de respuestas cercanas al óptimo obtenidas por CPLEX, se establece una versión final del algoritmo. Finalmente, se lleva a cabo un diseño de experimentos factorial para definir el mejor esquema de enfriamiento para cada una de las instancias estudiadas. Se presentan los resultados computacionales comparando las respuestas obtenidas con la metaheurística implementada y las respuestas obtenidas con CPLEX bajo los mismos tiempos de ejecución. Los resultados numéricos demuestran que es posible llegar a una solución óptima de manera eficiente para instancias pequeñas del problema. Para las instancias en las que no fue posible alcanzar una solución óptima se lograron variaciones promedio del 11.2% con respecto a las soluciones optimas obtenidas con CPLEX.

PALABRAS CLAVE:

Tamaño de lote, cotas de inventario, metaheurística, Investigación de operaciones

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE: PROPOSAL OF A METAHEURISTIC FOR THE MULTI-ITEM UNCAPACITATED LOT-SIZING PROBLEM WITH INVENTORY BOUNDS

AUTHOR(S): Maria Camila Monsalve Blanco
Daniel Jaimes Carvajal

FACULTY: Facultad de Ingeniería Industrial

DIRECTOR: Andrés Felipe Acevedo Ojeda

ABSTRACT

In this project, we consider the multi-item uncapacitated lot-sizing problem with inventory bounds where the production plan is for multiple items taking into account that the inventory of these products will share the same storage space. First, the mathematical structure of the algorithm is established using mixed integer-linear programming and then solved using IBM ILOG CPLEX. Subsequently, we propose three versions of a simulated annealing metaheuristic as a solution approach. We based the first version of the metaheuristic on an algorithm found in the literature. Based on the disadvantages found in the first version, we designed a second version using a different decision variable as input for the initial solution. Based on modifications of version two and the study of the decision variables of responses close to the optimal obtained by CPLEX, a final version of the algorithm is established. Finally, a factorial experiment is employed to define the best cooling scheme for each of the instances studied. We show the computational results comparing the implemented metaheuristics results with the results obtained with CPLEX under the same execution times. The numerical results demonstrate that it is possible to achieve an optimal solution efficiently for small instances of the problem. Where it was not possible to reach an optimal solution, the proposed metaheuristic had mean gaps of 11.2%.

KEYWORDS:

Lot-sizing, Inventory bounds, Matheuristic, Operations Research

V° B° DIRECTOR OF GRADUATE WORK

Introducción

En el sector industrial, es común que la producción se vea restringida por la capacidad de la bodega en la que se almacenan los productos terminados. En varios escenarios, es necesario que este espacio sea compartido entre múltiples tipos de productos, por lo que resulta relevante la adición de cotas de inventario a los problemas de planeación de la producción. Esta consideración surge en múltiples sistemas de producción en donde es común encontrar que la capacidad de almacenamiento disponible es limitada. Las restricciones que dictan cuánto espacio hay disponible para inventario pueden ser impuestas por el espacio físico de la bodega, por políticas administrativas, debido al gran volumen de los productos, o por el requerimiento de condiciones especiales de almacenamiento, como es el caso de los cuartos limpios, o aquellos con temperaturas controladas (Akbalik et al., 2014).

Con estas restricciones, la planeación de la producción de bienes busca determinar la cantidad de producto a producir que minimice el total de los costos y cumpla con el pronóstico de demanda (Melo & Ribeiro, 2017). Los problemas de tamaño de lote multiproducto con cotas de inventario han sido estudiados en contextos generales y bajo diferentes aplicaciones. Esto ha dado como resultado una variedad de propuestas de solución. En la literatura, hay numerosos ejemplos de metaheurísticas y algoritmos heurísticos que han sido utilizados para resolver diferentes variantes del problema de tamaño de lote. El algoritmo de Enfriamiento Simulado ha sido ampliamente propuesto para resolver problemas de combinatoria por ser flexible y tener la habilidad de manejar problemas grandes y complejos (Jans & Degraeve, 2007). La presente investigación busca solucionar instancias del problema de tamaño de lote multiproducto no capacitado con cotas de inventario haciendo uso de la metaheurística de Enfriamiento Simulado y evaluar su desempeño respecto a la calidad de la solución.

El documento se organiza de la siguiente forma. En el Capítulo 1 se describe el problema objeto de estudio y el interrogante que se intentará responder a través de la investigación. En el Capítulo 2 se presentan los avances principales en el modelo de optimización para el problema de tamaño de lote multiproducto no capacitado con cotas de inventarios. En el Capítulo 3 se exponen las razones que justifican la realización de la esta investigación. En el Capítulo 4 se especifica el principal objetivo de la investigación y sus respectivos logros intermedios para alcanzarlo. En el Capítulo 5 se precisan las ideas y conceptos con los cuales se aborda el problema. En el Capítulo 6 se estudia la estructura del problema de tamaño de lote multiproducto no capacitado con cotas de inventario analizando los parámetros y variables de decisión asociados al modelo matemático. En el Capítulo 7 se presenta la estructura de la metaheurística de Enfriamiento Simulado con sus respectivos parámetros y su aplicación durante la investigación, finalmente se exponen las tres versiones del algoritmo con sus respectivas variantes en el planteamiento y diseño a lo largo del estudio. En el Capítulo 8 se establecen las instancias del problema, se estudia una instancia pequeña y su respectiva solución obtenida en CPLEX. Posteriormente se presenta un diseño factorial de los parámetros del algoritmo concluyendo los mejores para cada instancia. En el Capítulo 9 se exponen los resultados en cuanto al desempeño de la metaheurística respecto a CPLEX en términos de calidad de la solución. Finalmente, en el Capítulo 10 se exponen las conclusiones del estudio y se motiva el tema para futuras investigaciones.

Lista de Tablas

Lista de Figuras

Capítulo 1

Situación problema

La planeación de la producción de bienes busca determinar la cantidad de producto a producir que minimice el total costos asociados y cumpla con la demanda. Esto también incluye decisiones relacionadas con el momento de la producción, la entrega oportuna, y los niveles de inventario. Es común encontrar industrias en donde esta producción se ve restringida por la capacidad de almacenamiento de la bodega en la que se mantendrá el inventario (Melo & Ribeiro, 2017).

Existen múltiples variantes del problema de tamaño de lote (Erromdhani et al., 2011). El caso específico del problema de tamaño de lote multiproducto no capacitado con cotas de inventario es considerado *NP-Hard* (o *NP*-Complejo) incluso cuando el horizonte de planeación considera tan solo dos periodos (Akbalik et al., 2014).

A finales del siglo XX, metaheurísticas como la Búsqueda Tabú, el Algoritmo Genético y el algoritmo de Recocido Simulado, han sido reconocidas como métodos eficientes para solucionar problemas de optimización combinatoria difíciles (Jans & Degraeve, 2007).

Teniendo en cuenta lo anterior, el presente proyecto busca responder la siguiente pregunta de investigación: ¿cómo adaptar una metaheurística para solucionar eficientemente el problema del tamaño de lote multiproducto no capacitado con cotas de inventario?

Capítulo 2

Antecedentes

A través de la base de datos ScienceDirect se realizó la búsqueda de artículos con los siguientes términos: *Multi-item*, *uncapacitated*, *lot sizing problem*, e *inventory bounds*. Con la búsqueda se obtuvieron 228 resultados, los cuales fueron filtrados por su año de publicación. Se seleccionaron los artículos con un máximo de 10 años de antigüedad (2010 en adelante) reduciendo así la cantidad a 104 artículos. De estos 104, se tomaron los 17 que coincidían con los criterios de (1) trabajar específicamente con el problema de tamaño de lote y (2) tener múltiples productos o cotas de inventario.

Los problemas de tamaño de lote multiproducto con cotas de inventario han sido estudiados en contextos generales y bajo diferentes aplicaciones. Esto ha dado como resultado una variedad de propuestas de solución.

Utilizando algoritmos de programación dinámica, (Phouratsamay & Cheng, 2019) resolvieron una versión extendida del problema de tamaño de lote que incluía restricciones de índole ambiental, donde la producción debía planearse de un modo tal que el total de las emisiones de carbón no excedieran ciertas capacidades en cada periodo. (Guan & Liu, 2010) desarrollaron y aplicaron diferentes algoritmos de programación dinámica para resolver la versión estocástica de los problemas de tamaño de lote donde las demandas, las cotas de inventario, y demás costos no son conocidos y son interdependientes a lo largo de un horizonte finito de planeación. Este mismo enfoque fue utilizado por Absi et al. (2012) para resolver un problema de tamaño de lote multiproducto con tiempos de alistamiento y ventas perdidas. Además, para mejorar las soluciones que obtuvieron con su metodología de programación dinámica, propusieron una metaheurística basada en la *Large Neighborhood Search.*

Las heurísticas basadas en algoritmos Lagrangianas también han sido aplicadas para resolver problemas de tamaño de lote. Carvalho & Nascimento (2016) definieron un problema aplicado a empresas compuestas por múltiples plantas de producción, en el que cada fábrica produce los mismos productos, suple su propia demanda y cada máquina tiene tiempos de alistamiento y una limitada capacidad de producción. En este caso se aplicó una nueva heurística Lagrangiana que no solo resolvió todas las instancias propuestas, sino que además lo hizo más eficientemente que las propuestas ya existentes en la literatura. Un problema de tamaño de lote multiproducto con tiempos de alistamiento y ventanas de producción fue propuesto por Brahimi et al. (2010), quienes lo resolvieron con heurísticas basadas en relajaciones Lagrangianas y demostraron que bajo este enfoque es posible encontrar soluciones factibles rápidamente.

Erromdhani et al. (2011) utilizaron dos enfoques colaborativos para solucionar un problema de tamaño de lote capacitado multiproducto con franjas de tiempos de producción y tiempos de alistamiento independientes del cliente. Los algoritmos aplicados usan simultáneamente la programación matemática y los procedimientos heurísticos tales como el Algoritmo de Búsqueda Local y la Búsqueda Vecindad Variable Descendiente. El problema se descompone en dos subproblemas: variables binarias y variables continuas. Los experimentos computacionales demostraron su efectividad de acuerdo con los enfoques ya expuestos en la literatura.

Otro enfoque de solución encontrado en la literatura son las heurísticas basadas en la programación lineal-entera mixta (MIP). Wu et al. (2018) consideraron un problema de tamaño de lote con cotas de inventario y restricciones de emisiones de carbón. Formularon el problema con un modelo MIP y propusieron una relajación Lagrangiana, y para resolverlo aplicaron una heurística llamada Selección Progresiva cuyos resultados demostraron ser superiores a los ya

encontrados en la literatura utilizando los mismos recursos computacionales. Cunha et al. (2019) propusieron heurísticas matemáticas basadas en la programación lineal-entera mixta para resolver una extensión del problema de tamaño de lote multiproducto que incluye remanufactura y producción capacitada para satisfacer demandas dinámicas determinísticas a lo largo de un horizonte discreto de planeación. La primera consiste en un algoritmo de generación de columnas seguido por una programación linear y la segunda consistió en el método heurístico *relax-and-fix.*

Los algoritmos basados en metaheurísticas y las metaheurísticas en sí, han sido enfoques utilizados en la literatura para resolver diferentes variantes de problemas de tamaño de lote. Sifaleras & Konstantaras (2015b) propusieron una extensión de este problema con devoluciones, como parte de los problemas que surgen en la logística inversa. Para resolverlo aplicaron un algoritmo heurístico basado en la metaheurística Búsqueda de Vecindad Variable y además aplicaron la metaheurística misma para resolver este problema siendo la primera vez en la literatura que este enfoque es utilizado para este problema (Sifaleras & Konstantaras, 2015a). Para resolver un problema de tamaño de lote no capacitado multinivel, Xiao et al. (2011) estudiaron múltiples técnicas de aplicación de la metaheurística Búsqueda de Vecindad y demostraron que los parámetros de: distancia, rango y cambio de nivel tienen una gran influencia en la eficacia de la búsqueda de esta metaheurística. Este conocimiento puede ser de gran utilidad a la hora de generar heurísticas más eficientes.

Melo & Ribeiro (2017) propusieron diferentes formulaciones y heurísticas para el problema de tamaño de lote multiproducto con cotas de inventario. En primer lugar, muestran la *formulación de la ruta más corta* y la formulación basada en la adición previa de *desigualdades válidas*, las cuales son comparadas con las formulaciones encontradas en la literatura. Se

efectuaron experimentos computacionales para evaluar los diferentes enfoques. Los resultados demostraron que la heurística de *relax-and-fix* fue el mejor enfoque y sus soluciones estuvieron a un 4% del valor óptimo en menos de diez minutos de tiempo de ejecución. Demostraron también que las soluciones obtenidas siempre fueron mejores que aquellas obtenidas por programas comerciales para resolver problemas de optimización, en un tiempo de ejecución de una hora.

Parsopoulos et al. (2015) utilizaron un algoritmo basado en población, llamado Evolución Diferencial (DE) para resolver un problema dinámico de tamaño de lote con devoluciones y remanufactura. Para resolver el problema, utilizaron una variante de esta metaheurística que fue adaptada para mejorar su rendimiento. Realizaron una serie de pruebas para comparar su desempeño con un enfoque moderno y otras metaheurísticas. Los resultados sugieren que esta variante en específico puede ser considerada como uno de los métodos más eficientes en la solución de esta clase de problemas.

Un problema de tamaño de lote extendido a un sistema híbrido de manufactura y remanufactura, donde los productos son producidos en líneas separadas y vendidos en mercados segmentados fue resuelto por Koken et al. (2018) utilizando una heurística basada en el Algoritmo Genético. Este enfoque fue evaluado comparando los resultados obtenidos con los de otras metaheurísticas como el Enfriamiento Simulado y la Búsqueda de Vecindad Variable. Otra aplicación del Algoritmo Genético fue estudiada por Fabiano et al. (2012) para resolver un problema de tamaño de lote multinivel capacitado con acumulación. En este caso, un algoritmo híbrido con múltiples poblaciones del Algoritmo Genético fue propuesto. El método combina esta metaheurística utilizando la heurística *fix-and-optimize* y técnicas de programación matemática.

El algoritmo de Enfriamiento Simulado se encontró ampliamente en la literatura para resolver problemas de combinatoria complejos y además por ser flexible y manejar problemas grandes y difíciles (Jans & Degraeve, 2007). Para resolver un problema discreto de tamaño de lote multiproducto, Ceschia et al. (2017) propusieron un algoritmo de Enfriamiento Simulado en conjunto con un proceso estadístico de ajuste. Debido a que se le reconoce como un problema *NP*-Hard, Roshani et al. (2016) propusieron un algoritmo de Enfriamiento Simulado con una eficiente generación de búsqueda de vecindario como solución para un problema de tamaño de lote con devoluciones en un sistema cerrado de remanufactura. Para evaluar la eficiencia y demostrar la efectividad del algoritmo propuesto, realizaron una comparación entre los resultados que obtuvieron con el algoritmo y aquellos generados por el *solver* CPLEX.

Debido a su amplio uso en la literatura y a características como su flexibilidad y facilidad para manejar problemas complejos, se tomó la decisión de aplicar la metaheurística de enfriamiento simulado para resolver el problema de tamaño de lote multiproducto no capacitado con cotas de inventario propuesto en este proyecto.

Capítulo 3

Justificación del proyecto

La presente investigación se enfoca en proponer un método de solución al problema de tamaño de lote multiproducto no capacitado con cotas de inventario mediante la adaptación de una metaheurística y posteriormente evaluar su rendimiento.

Se seleccionó este problema en específico ya que en la industria es común que la producción se vea limitada por la capacidad de la bodega en la que se mantiene el inventario de uno o varios productos. Estas restricciones de cantidad se relacionan con el espacio físico de la bodega o incluso con políticas administrativas, especialmente para productos voluminosos, o productos que requieren condiciones especiales de almacenamiento (cuartos limpios, temperaturas controladas) (Akbalik et al., 2014). Por esta razón, los problemas con cotas de inventario y que además son de múltiples productos, adquieren mucha relevancia desde un punto de vista práctico (Melo & Ribeiro, 2017).

Pruebas de la teoría de la complejidad y experimentos computacionales han demostrado que la mayoría de los problemas de tamaño de lote son difíciles de resolver (Jans & Degraeve, 2007). Esto implica que los algoritmos que pretenden resolver estos problemas con exactitud tienden a ser lentos (Mann, 2017). Por esta razón se considera necesaria la búsqueda de métodos alternos de solución. Para este proyecto, se propone el uso de metaheurísticas, puesto que han demostrado importantes mejoras en la solución de problemas de optimización de alto grado de complejidad. Además, autores como Akbalik et al. (2014) han demostrado que este tipo de problemas es considerado como un problema *NP*-Hard (o *NP*-Complejo), incluso cuando se restringe tan solo a dos periodos.

Este estudio servirá para dar a conocer el rendimiento del método de solución propuesto, lo que será de utilidad para futuras investigaciones en las que se busque aplicar este problema a contextos industriales reales y para aquellas en las que requiera resolver y/o simular diferentes instancias del problema.

Capítulo 4

Objetivos del proyecto

Objetivo General

Proponer una metaheurística que permita encontrar soluciones factibles para el problema de tamaño de lote multiproducto no capacitado con cotas de inventario.

Objetivos Específicos

- Identificar los avances principales en el estudio del modelo de optimización para el problema de tamaño de lote multiproducto no capacitado con cotas de inventarios, de tal forma que se visualice los enfoques metaheurísticos para la solución de este problema.
- Analizar las propuestas de implementación de algoritmos metaheurísticos al problema de tamaño de lote multiproducto no capacitado con cotas de inventarios, lo que permite la selección de uno de los algoritmos para su posterior implementación.
- Examinar la estructura del modelo de optimización para el problema de tamaño de lote multiproducto no capacitado con cotas de inventarios, de tal forma que, a través del software IBM ILOG CPLEX, se obtenga la solución óptima para un número de instancias del problema.
- Implementar el algoritmo metaheurístico seleccionado para solucionar el problema de optimización referente al tamaño de lote multiproducto no capacitado con cotas de inventarios.
- Desarrollar un diseño de experimentos factorial, para hacer un ajuste paramétrico de la metaheurística propuesta para el problema de tamaño de lote multiproducto no capacitado; de tal forma que se obtenga un mejor rendimiento del algoritmo para cada instancia del problema.

- Evaluar la calidad de solución obtenida por la metaheurística para el problema de tamaño de lote multiproducto no capacitado con cotas de inventarios, considerando el tiempo computacional y la variación de la solución obtenida por la metaheurística con respecto a la solución exacta del modelo obtenido con IBM ILOG CPLEX.

Capítulo 5

Marco Teórico

> La optimización es el acto de obtener la mejor solución bajo circunstancias dadas. Su objetivo consiste en minimizar el esfuerzo requerido o maximizar un beneficio deseado. Este esfuerzo o beneficio puede ser expresado en función de variables de decisión por lo que es posible definir optimización como "el proceso de hallar las condiciones que darán como resultado el valor máximo o mínimo de una función". No existe un único método para resolver eficientemente todos los problemas de optimización. En respuesta a esta problemática, numerosos métodos de optimización han sido desarrollados. (Rao, 2009)

Sin embargo, una gran cantidad de problemas de optimización en áreas como la ciencia, la economía, y la industria son complejos y difíciles de resolver. Incluso podrían no tener una respuesta exacta en un rango razonable de tiempo. Los algoritmos de aproximación (*approximate algorithms*) son la principal alternativa para resolver esta clase de problemas (Talbi, 2009).

El área de la programación matemática trata con la formulación, solución y el análisis de problemas de optimización y programas matemáticos. Este análisis y solución puede incluir herramientas gráficas, algebraicas y computacionales ("Overview of optimization", 2006).

Los algoritmos de aproximación pueden dividirse en dos clases: heurísticas específicas y metaheurísticas. La primera depende del problema, se diseñan y se aplican a una situación específica. Las metaheurísticas son algoritmos generalizados que pueden ser aplicados a una gran variedad de problemas de optimización y se adaptan para resolver cualquier problema de optimización (Talbi, 2009).

En contextos industriales, la función de planeación y control de la producción es una de las áreas más importantes de una empresa de manufactura sin importar cual sea su campo de

acción. Dicha función asegura la disponibilidad de materiales o componentes en el área de producción en el momento oportuno y las cantidades apropiadas (Kiran, 2019).

En el área de la planeación de la producción, los problemas suelen servir para determinar la cantidad de producto a producir para minimizar los costos y asegurar la satisfacción del cliente, lo que busca proveer los bienes en la cantidad y en el momento requerido (Melo & Ribeiro, 2017).

La gestión de inventarios se encuentra entre las actividades operacionales más importantes de la industria. La estructura de los niveles de inventario puede influir directamente sobre la calidad del servicio al cliente en cuanto afectan la disponibilidad de producto y la velocidad de entrega, esto es indispensable para la competitividad de las empresas (Glock et al., 2014).

Un problema de tamaño de lote es un problema del área de la planeación de la producción, cuyos resultados se pueden extender para solucionar problemas más generales como el de la ruta más corta (Melo & Ribeiro, 2017). Es un problema considerado como *NP*-Hard y propone determinar la cantidad óptima a producir de cada ítem en cada periodo de tiempo y a la vez minimizar el costo total de manufactura, lo que incluye costos de inventario, costos fijos de preparación y costos de producción. Esto a su vez, considerando que todos los productos comparten una misma bodega de almacenamiento. Existen múltiples variantes del problema de tamaño de lote. La variedad depende del número de productos, la estructura de los productos, las restricciones a considerar y diferentes combinaciones de estas variantes (Erromdhani et al., 2011). El caso para el presente trabajo se refiere a un problema de tamaño de lote multiproducto no capacitado con cotas de inventario.

El término *NP* hace referencia a Tiempo Polinomial No-Determinístico, en inglés conocido como "*non-deterministic polynomial-time*" (Mann, 2017) y es usado para referirse a un conjunto de problemas considerados complejos (o muy difíciles) de resolver.

El término *metaheurística* describe un nivel más alto de heurísticas que se proponen para solucionar un amplio rango de problemas de optimización. En la actualidad, un gran número de metaheurísticas han sido exitosamente aplicadas a problemas de gran complejidad (Dokeroglu et al., 2019). Su utilidad radica en que pueden hallar soluciones más rápido de problemas más grandes. Se consideran algoritmos robustos, flexibles y fáciles de implementar (Talbi, 2009).

El algoritmo de Recocido Simulado o Enfriamiento Simulado es una metaheurística formalmente presentada en 1983 por Kirkpatrick, Gellatt y Vecchi basada en una analogía con la termodinámica del recocido de en sólidos (Tang, 2004). Esta metaheurística es ampliamente conocida como un algoritmo de búsqueda de vecindad que puede proveer soluciones óptimas o cercanas al óptimo para problemas de combinatoria. Por esta razón ha sido utilizada para resolver una gran variedad de problemas de optimización incluyendo los de tamaño de lote (Roshani et al., 2016).

Capítulo 6

Definición del problema

El problema de tamaño de lote no capacitado con cotas de inventario es un problema de optimización del área de planeación de la producción, en el cual la producción está limitada únicamente por la capacidad física de almacenamiento que esté disponible. Se busca encontrar cuándo producir y en qué cantidad de tal modo que se satisfaga una demanda y se minimice la suma de los costos asociados con la producción, los costos de preparación y de mantenimiento del inventario.

Se asume:

- No se dispone de inventario al inicio del horizonte de planeación
- La demanda es previamente establecida
- No hay tiempos de preparación

Tabla 1. Parámetros asociados con el problema de tamaño de lote

Símbolo	Definición
T	Número de periodos
I	Número de productos a producir.
M	Un número muy grande
d_{it}	Demanda para el producto *i* en el periodo *t*.
q_{it}	Costo fijo de preparación para el producto *i* en el periodo *t*.
h_{it}	Costo de mantener inventario para el producto *i* en el periodo *t*.
p_{it}	Costo de producción del producto *i* en el periodo *t*.
u_t	Capacidad de almacenamiento de inventario en el periodo *t*.

Tabla 2. Variables de decisión del problema de tamaño de lote

Símbolo	Definición
x_{it}	Tamaño de lote de producción para el producto *i* en el periodo *t*.
s_{it}	Inventario para el producto *i* al final del periodo *t*.
y_{it}	Variable binaria que indica si hay de preparación para el producto *i* en el periodo *t* ($y_{it} = 1$) o no ($y_{it} = 0$).

Utilizando esta notación, es posible definir este problema de tamaño de lote como un modelo MIP de la siguiente manera:

$$Min \sum_{i=1}^{I} \sum_{t=1}^{T} q_{it} \cdot y_{it} + h_{it} \cdot s_{it} + p_{it} \cdot x_{it} \qquad (1)$$

Sujeto a

$$s_{i(t-1)} + x_{it} = d_{it} + s_{it}, \forall\, i \in I, t \in T \qquad (2)$$

$$x_{it} \leq M \cdot y_{it}, \forall\, i \in I, t \in T \qquad (3)$$

$$\sum_{i=1}^{I} s_{it} \leq u_t\,, \forall\, i \in I \qquad (4)$$

$$x_{it}, s_{it} \geq 0, \forall\, i \in I, t \in T \qquad (5)$$

$$y_{it} \in \{0,1\}, \forall\, i \in I, t \in T \qquad (6)$$

La función objetivo (1) consiste en minimizar la suma de los costos fijos de preparación, los costos de mantenimiento de inventario y los costos de producción. La igualdad (2) representa el balance de inventario. Debido a la inigualdad (3) se generan costos de preparación para los periodos en los que hay producción. La restricción (4) limita la cantidad de inventario a almacenar en cada periodo. Las inigualdades (5) y (6) son respectivamente, restricciones de no negatividad y de integridad de las variables.

Capítulo 7

Algoritmo de Enfriamiento Simulado

El algoritmo de enfriamiento simulado es una técnica de búsqueda estocástica basada en la simulación del recocido de sólidos a alta temperatura para alcanzar el menor valor de la función objetivo en un problema de minimización (Aydin & Artem, 2017). En esta aplicación clásica del algoritmo, una perturbación aleatoria de la solución actual genera una nueva posible solución. Se supone E_a y E_n como los niveles de energía de la solución actual y de la nueva solución, respectivamente. Si la solución actual tiene un nivel de energía mayor a la nueva solución ($E_a > E_n$), quiere decir que una mejor solución ha sido alcanzada, esta solución será ahora la solución actual. Si la nueva solución tiene más energía que la actual ($E_a < E_n$), esta nueva solución será aceptada bajo una probabilidad dictada por la temperatura $p = e^{\left(-\frac{\Delta E}{T}\right)}$. El proceso continúa hasta que esta probabilidad de aceptar una peor solución disminuye (Lee & Yang, 2009).

Definición de Parámetros

Tabla 3. Parámetros asociados del algoritmo de enfriamiento simulado

Símbolo	Definición
T_o	Temperatura inicial.
T_f	Temperatura final.
α	Factor de enfriamiento.
L	Iteraciones del algoritmo.

Temperatura Inicial y Temperatura Final

De acuerdo con Mahdi et al. (2017), la temperatura inicial debe ser lo suficientemente alta para que el algoritmo tenga suficiente libertad de explorar soluciones vecinas, pero no demasiado alta para poder tener tiempos computacionales razonables. La temperatura dicta la probabilidad de aceptar una solución con mayor energía que la actual. Esta probabilidad está

dada en el criterio metrópolis por el factor de Boltzmann $p = e^{\left(-\frac{\Delta E}{T}\right)}$, donde ΔE es el cambio de energía y T la temperatura.

De acuerdo con Roshani et al. (2016), la temperatura de la primera iteración debe ser tal que la probabilidad de aceptar una mala solución sea de al menos el 80%. Por lo que la temperatura inicial se calculará de la siguiente forma:

$$T_0 = -\frac{\Delta E}{\ln 0.80} \qquad (7)$$

De este modo, se establece una temperatura inicial diferente para cada instancia, determinada por la diferencia de energía encontrada en la primera iteración del algoritmo. La temperatura final se establece considerando que la probabilidad de aceptar una peor solución sea aproximadamente 0 (Roshani et al., 2016).

Factor de Enfriamiento

El factor de enfriamiento α es una constante entre 0 y 1. Para tener un descenso lento de la temperatura es necesario que sea un valor cercano a 1 (Mahdi et al., 2017). Un descenso lento de la temperatura evitará que la respuesta se detenga en un óptimo local y además asegurará que se logre un estado de baja energía. Los resultados del algoritmo serán un balance entre la velocidad de convergencia y la calidad de la solución (Tang, 2004). El factor de enfriamiento esta dado por un valor entre (0,85 y 0,999).

Se utiliza un enfriamiento exponencial dado por la fórmula:

$$T_{k+1} = \alpha \cdot T_k \qquad (8)$$

Aplicación del Algoritmo de Enfriamiento Simulado

El algoritmo está desarrollado en el lenguaje de programación R e inicia con una de las instancias ya establecidas. Estas instancias fueron previamente generadas y registradas en múltiples hojas de Excel. Cada instancia tiene los parámetros asociados al problema de tamaño de lote multiproducto no capacitado con cotas de inventario (véase la Tabla 1. Parámetros asociados con el problema de tamaño de lote).

El algoritmo consta de cuatro etapas principales: (1) Generación de la solución inicial, (2) Control de temperatura, (3) Generación de la solución de vecindario, y (4) Criterio metrópolis.

Tabla 4. Pseudocódigo general del Algoritmo de Enfriamiento Simulado.

Pseudocódigo Algoritmo ES. Estructura general
Paso 1. INICIO
Paso 2. Inicializar variables
Paso 3. Generar Solución Inicial Y_0
Paso 4. Ejecutar el módulo de Control de temperatura
Paso 4.1. **WHILE** $t_c > t_f$ **DO**:
Paso 4.2. **FOR** $k = 1$ **TO** L
Paso 4.2.1. Generar Solución de Vecindario
Paso 4.2.2. Calcular Función Objetivo
Paso 4.2.3. Calcular deltaZ
Paso 4.2.4. **IF** $deltaZ \leq 0$ **THEN** establecer Solución de Vecindario como Solución Actual
Paso 4.2.5. **IF** $deltaZ > 0$ **THEN** generar un número aleatorio $rand \in [1, 0]$
Paso 4.2.5.1. **IF** $rand \leq e^{\left(-\frac{deltaZ}{t_c}\right)}$ **THEN** establecer Solución de Vecindario como Solución Actual.
Paso 4.2.6. Establecer $k = k + 1$
Paso 4.3. Establecer $t_c = t_c \cdot \alpha$
Paso 4.4. Actualizar variables de probabilidad de la solución de vecindario
Paso 4.4.1. $dP = dP \cdot \alpha$
Paso 4.4.2. $aP = aP \cdot \alpha$
Paso 5. FIN

Durante el proceso de investigación, se diseñaron en total de tres versiones del algoritmo de enfriamiento simulado. En cada uno de los algoritmos, la variación se encuentra en las etapas de generación de la solución inicial (Paso 3.) y generación de la solución de vecindario (Paso 4.2.1.). Las otras dos etapas son idénticas para los tres algoritmos diseñados.

Estos algoritmos fueron diseñados de manera secuencial. Cada algoritmo nuevo se replantea y diseña teniendo en cuenta las falencias o dificultades observadas en los anteriores.

El primer algoritmo de enfriamiento simulado se basa en la teoría de Roshani et al. (2016) para el diseño de la generación de la solución inicial y el mecanismo de generación de solución de vecindario. Tras estudiar el rendimiento de este algoritmo, fue posible observar que tardaba mucho tiempo en alcanzar una solución cercana al óptimo. La razón era lo lejos que se encontraba la solución inicial de la solución óptima. Además, se observó que el "generador de solución de vecindario", a pesar de generar soluciones factibles, no cumplía con la conocida Propiedad Wagner-Whitin, requerida por la estructura de las instancias resueltas del problema. Dicha propiedad establece que, en la solución óptima del problema, para cada período del horizonte de planeación, o se tiene inventario disponible del período anterior ($s_{t-1} > 0$) , o se requiere producir una cantidad positiva ($y_t = 1$) (Wagner & Whitin, 1958).

El segundo algoritmo de Enfriamiento Simulado replantea la solución inicial. En este caso, la matriz "solución inicial" está dada por la variable de decisión *y,* creada bajo aleatoriedad controlada, asignando cierta cantidad de 1's a lo largo de los periodos para cada producto. La generación de vecindario se realiza haciendo dos cambios aleatorios en la matriz *Y*, reemplazando los 1's por 0's y viceversa en una posición aleatoria. Este algoritmo genera soluciones muy lejanas al óptimo empleando tiempos similares a los del Algoritmo 1.

Para la tercera y definitiva versión del algoritmo, se entendió que la solución inicial del algoritmo tiene un impacto significativo en el resultado final. Esta primera solución que genera el algoritmo debe ser cercana al óptimo y tener un nivel de energía bajo. Para poder tener una idea de cómo generar una solución inicial factible, que impactara positivamente en la solución final, se analizaron las soluciones óptimas obtenidas con CPLEX. De este modo, se genera una

solución inicial cercana a las obtenidas previamente con CPLEX, donde se tiene en cuenta la cantidad de periodos productivos por producto que maneja la solución óptima y cómo están distribuidos a lo largo de los periodos. A diferencia de la matriz con la que se trabajó en el algoritmo 2, la matriz binaria inicial *Y* de este nuevo algoritmo tiene una cantidad considerablemente menor de periodos productivos, lo cual reduce notablemente el valor inicial de la función objetivo.

En cuanto a la generación de vecindario, se observó en los algoritmos anteriores que cambios muy específicos (asignar más o menos producción o solo cambiar un periodo productivo) implican cambios muy mínimos en la matriz, lo que resulta en posibles soluciones de vecindario que varían muy poco y, por consiguiente, la búsqueda de soluciones con mejor valor de función objetivo se extiende. Además, tras analizar las soluciones óptimas de CPLEX se observó que a medida que la solución disminuía su nivel de energía (se acercaba al óptimo) no solo había menos periodos productivos, sino que, además, la matriz binaria cambiaba, variando la ubicación de los periodos productivos. Teniendo en cuenta esto, para el algoritmo 3 se diseña una función de generación de vecindario pensada en lograr soluciones de vecindario que no solo sean factibles, sino que además generen mayor variabilidad en las posibles soluciones encontradas, lo que facilita encontrar mejores soluciones. En esta nueva función, es posible aumentar, disminuir o mover periodos productivos. Este algoritmo logra mejorar la eficiencia de la solución al generar soluciones más cercanas al óptimo en menor tiempo.

A continuación, se profundiza en la generación de solución inicial y la generación de solución de vecindario de cada uno de los 3 algoritmos diseñados.

Algoritmo Versión 1

Generación de la Solución Inicial. De acuerdo con (Roshani et al., 2016) la solución inicial (X_0) se genera a partir de la matriz de demanda. La solución inicial, al iniciar el algoritmo pasa a ser la "solución actual" (X_a).

Mecanismo de Generación de Solución de Vecindario. La generación de la solución de vecindario se basó en el mecanismo utilizado por (Roshani et al., 2016). Este método se basa en la aplicación de dos operadores, de suma y de resta, los cuales son generados a partir de la matriz "solución actual".

Operador de Suma. A modo de ejemplo, entiéndase la matriz "solución actual" como:

$$X_a = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1T-1} & x_{1T} \\ x_{21} & x_{22} & \cdots & x_{2T-1} & x_{2T} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{I1} & x_{I2} & \cdots & x_{IT-1} & x_{IT} \end{bmatrix}$$

En primer lugar, se selecciona aleatoriamente dos elementos en la misma fila de la matriz "Solución actual". Este operador le sumará un valor V^+ al primer elemento (en el menor periodo) y le restará el mismo valor V^+ al segundo elemento (en el mayor periodo). En este ejemplo, los elementos seleccionados serán x_{12} y x_{1T-1}.

La solución de vecindario obtenida con el operador de adición será:

$$X_n = \begin{bmatrix} x_{11} & x_{12} + V^+ & \cdots & x_{1T-1} - V^+ & x_{1T} \\ x_{21} & x_{22} & \cdots & x_{2T-1} & x_{2T} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{I1} & x_{I2} & \cdots & x_{nT-1} & x_{IT} \end{bmatrix}$$

Para que la solución de vecindario generada con este operador sea factible, el valor V^+ es determinado de acuerdo con las restricciones del problema de tamaño de lote no capacitado con cotas de inventario. El valor fue seleccionado de acuerdo con los siguientes pasos.

1. Se asume que x_{it} y x_{ik}, con $t < k$ son seleccionados aleatoriamente.
2. Se establece el valor máximo de V^+, llamado V^+_{max} así:

$$V_{max}^{+} = max\left\{0, min\left\{u_l - \sum_{i=1}^{I} s_{il} \;\; l \in [t,k), x_{ik},\right\}\right\} \qquad (9)$$

3. Si $V_{max}^{+} > 1$ se generará un valor aleatorio entre 1 y V_{max}^{+}, este será el valor asignado a V^{+}.

Operador de Resta. Para ilustrar el operador resta, se utilizará la misma matriz "solución actual" a modo de ejemplo.

$$X_a = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1T-1} & x_{1T} \\ x_{21} & x_{22} & \cdots & x_{2T-1} & x_{2T} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{I1} & x_{I2} & \cdots & x_{IT-1} & x_{IT} \end{bmatrix}$$

Se selecciona aleatoriamente dos elementos en la misma fila de la matriz "Solución actual". En este caso el operador resta, restará un valor V^{-} al primer elemento (en el menor periodo) y le sumará el mismo valor V^{-} al segundo elemento (en el mayor periodo). En este ejemplo, los elementos seleccionados serán x_{12} y x_{1T-1}.

La solución de vecindario obtenida con el operador de resta será:

$$X_n = \begin{bmatrix} x_{11} & x_{12} - V^{-} & \cdots & x_{1t-1} + V^{-} & x_{1t} \\ x_{21} & x_{22} & \cdots & x_{2t-1} & x_{2t} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{nt-1} & x_{it} \end{bmatrix}$$

Para que la solución de vecindario generada con este operador sea factible, el valor V^{-} es determinado de acuerdo con las restricciones del problema. El valor fue seleccionado de acuerdo con los siguientes pasos.

1. Se asume que x_{it} y x_{ik}, con $t < k$ son seleccionados aleatoriamente.
2. Se establece el valor máximo de V^{-}, llamado V_{max}^{-} así.

$$V_{max}^{-} = max\left\{0, min\left\{u_k - \sum_{i=1}^{I} s_{ik}\,, x_{it}, s_{il}\ \ l\ \in [t,k)\right\}\right\} \qquad (10)$$

3. Si $V_{max}^{-} > 1$ se generara un valor aleatorio entre 1 y V_{max}^{-}, este será el valor asignado a V-

Generación de Vecindario. Una vez se han calculado ambos operadores (operador de suma y operador de resta), para la creación de la solución de vecindario se selecciona y se aplica uno de los dos operadores obtenidos, del siguiente modo:

1. Se seleccionan aleatoriamente dos elementos de la matriz "solución actual", x_{it} y x_{ik}, con $t < k$.
2. Se calcula el valor de V_{max}^{+} y V_{max}^{-}.
3. Se decide qué operador aplicar, así:
    a) $si\ V_{max}^{+} > 0\ y\ V_{max}^{-} = 0,\ Aplicar\ operador\ de\ suma$
    b) $si\ V_{max}^{+} = 0\ y\ V_{max}^{-} > 0,\ Aplicar\ operador\ de\ resta$
    c) $si\ V_{max}^{+} > 0\ y\ V_{max}^{-} > 0,\ Seleccionar\ operador\ aleatoriamente$
    d) $Si\ V_{max}^{+} = 0\ y\ V_{max}^{-} = 0,\ Volver\ al\ paso\ 1$

En el algoritmo, esta solución de vecindario obtenida será referida como "nueva solución" (X_n).

Tabla 5. Pseudocódigo del Algoritmo de Enfriamiento Simulado - Versión 1

Algoritmo de enfriamiento simulado
Paso 1. Inicialización del algoritmo
Paso 1.1. Generar Solución Inicial X_0 = d, S_0, Y_0
Paso 1.2. Establecer $X_a = X_0$
Paso 1.3. Calcular $Z(X_0)$ usando la Función Objetivo (1)
Paso 2. **WHILE** $T > T_f$ **DO**:
Paso 2.1. k = 1
Paso 2.2. **FOR** k = 1 **TO** *iteraciones*
Paso 2.2.1. Definición de parámetros iniciales (Solo para la primera iteración de la ejecución de todo el algoritmo)
Paso 2.2.2. Aplicar el mecanismo de Generación de Vecindario a X_a para generar una Solución de Vecindario X_n
Paso 2.2.3. Calcular S_n y Y_n
Paso 2.2.4. Calcular $Z(X_n)$
Paso 2.2.5. Calcular deltaZ = $Z(X_n)$ - $Z(X_a)$
Paso 2.2.6. Calcular T_0 usando la formula (7)
Paso 2.2.7. Inicializar temperatura final T_f, factor de enfriamiento α. Establecer $T = T_0$;
Paso 2.2.8. **IF** deltaZ ≤ 0, establecer $X_a = X_n$, $S_a = S_n$, $Y_a = Y_n$, $Z(X_a) = Z(X_n)$
Paso 2.2.9. **IF** deltaZ > 0, generar un numero aleatorio $\alpha \in [1, 0]$
Paso 2.2.9.1. **IF** $\alpha \leq e^{\left(-\frac{\Delta E}{T}\right)}$, establecer $X_a = X_n$, $S_a = S_n$, $Y_a = Y_n$, $Z(X_a) = Z(X_n)$
Paso 2.2.10. Establecer k = k + 1
Paso 2.3. Establecer $T = T \cdot \alpha$
Paso 3. Imprimir X_a, S_a, Y_a

Algoritmo Versión 2

Generación de la Solución Inicial. A diferencia de la Versión 1 del algoritmo, la solución inicial se genera a partir de la matriz Y.

El método de generación consiste en la asignación aleatoria de cierta cantidad aleatoria uniforme de periodos productivos a lo largo del horizonte de planeación. Se inicia asignando 1's en el primer periodo de cada producto en el cual su demanda es mayor a cero.

A cada producto le es asignado una cantidad de entre 30 y 35 periodos en los que se lleva a cabo producción ($y_t = 1$). Este rango en la cantidad de periodos productivos fue designado como el mínimo y máximo valor posible con el que se logra una solución factible sin obtener un alto valor de *Z*.

Mecanismo de Generación de Solución de Vecindario. La generación de la solución de vecindario se determinar usando una distribución de probabilidad uniforme y consiste en cambiar dos posiciones en la matriz "solución actual".

Para ilustrar la generación de vecindario, entiéndase la matriz de "solución actual" como:

$$Y_a = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1T-1} & y_{1T} \\ y_{21} & y_{22} & \cdots & y_{2T-1} & y_{2T} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_{I1} & y_{I2} & \cdots & y_{IT-1} & y_{IT} \end{bmatrix}$$

Se seleccionan aleatoriamente dos elementos de la matriz "solución actual", y_{it} y y_{jk}.

Los cambios ocurren de la siguiente forma:

$$si\ y_{it}\ \text{ó}\ y_{jk}. = 0, \qquad se\ cambiará\ su\ valor\ a\ 1$$

$$si\ y_{it}\ \text{ó}\ y_{jk}. = 1, \qquad se\ cambiará\ su\ valor\ a\ 0$$

En el algoritmo, esta solución de vecindario obtenida será referida como "nueva solución" (Y_n).

Tabla 6. Pseudocódigo del Algoritmo de Enfriamiento Simulado - Versión 2

Algoritmo de enfriamiento simulado
Paso 1. Inicialización del algoritmo
Paso 1.1. Generar Solución Inicial Y_0, S_0, X_0
Paso 1.2. Establecer $Y_a = Y_0$, $S_a = S_0$, $X_a = X_0$
Paso 1.3. Calcular $Z(Y_0)$ usando la Función Objetivo (1)
Paso 2. **WHILE** $T > T_f$ **DO**:
Paso 2.1. k = 1
Paso 2.2. **FOR** k = 1 **TO** *iteraciones*
Paso 2.2.1. Definición de parámetros iniciales (Solo para la primera iteración de la ejecución de todo el algoritmo)
Paso 2.2.2. Aplicar el mecanismo de Generación de Vecindario a Y_a para generar una Solución de Vecindario Y_n
Paso 2.2.3. Calcular X_n y S_n
Paso 2.2.4. Calcular $Z(Y_n)$
Paso 2.2.5. Calcular deltaZ = $Z(Y_n) - Z(Y_a)$
Paso 2.2.6. Calcular T_0 usando la formula (7)
Paso 2.2.7. Inicializar temperatura final T_f, factor de enfriamiento α. Establecer $T = T_0$;
Paso 2.2.8. **IF** deltaZ ≤ 0, establecer $Y_a = Y_n$, $X_a = X_n$, $S_a = S_n$, $Z(Y_a) = Z(Y_n)$
Paso 2.2.9. **IF** deltaZ > 0, generar un numero aleatorio $\alpha \in [1, 0]$
Paso 2.2.9.1. **IF** $\alpha \leq e^{\left(-\frac{\Delta E}{T}\right)}$, establecer $Y_a = Y_n$, $X_a = X_n$, $S_a = S_n$, $Z(Y_a) = Z(Y_n)$
Paso 2.2.10. Establecer k = k + 1
Paso 2.3. Establecer $T = T \cdot \alpha$
Paso 3. Imprimir X_a, S_a, Y_a

Algoritmo Versión 3

Al igual que en la segunda versión del algoritmo, esta versión utiliza la matriz binaria Y como lienzo para las etapas esenciales del algoritmo.

Generación de la Solución Inicial. La solución inicial (Y_0) consta de dos etapas esenciales para ser generada, la etapa de asignación y la etapa de ajuste. Se inicia con la asignación de 1's en el primer periodo de cada producto en el cual su demanda es mayor a cero. A cada ciclo de asignación de periodos productivos se le conoce como "turno". Esto quiere decir que la asignación de 1's en el primer periodo de cada producto corresponde al primer turno.

A modo de ejemplo, entiéndase la matriz "solución inicial" (Y_0) como una matriz lienzo (vacía, correspondiente a 3 productos y 6 periodos) antes de efectuar las etapas de asignación y ajuste.

$$Y_0 = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Etapa de Asignación. La etapa de asignación consiste en fijar 1's simultáneamente para cada producto a lo largo de todos los periodos mediante un sistema de turnos. Es de gran importancia entender que el sistema de turnos obedece a la siguiente lógica:

La siguiente matriz "solución inicial" (Y_0) es una matriz lienzo en la que ya se efectuó el segundo turno de asignación.

$$Y_0 = \begin{bmatrix} 1 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 1 & 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$$

Los 1's resaltados con negrita corresponden al segundo turno, mientras que los 1's que no están resaltados con negrita corresponden al turno anterior, en este caso el primer turno.

Cabe aclarar que es normal que los 1's del mismo turno se fijen en diferentes periodos, esto se debe principalmente a que el método de fijación evaluó la matriz demanda a partir del

siguiente periodo al que fue fijado el 1 en el anterior turno y de este modo concluyó qué periodo debía ser fijado cada uno del turno actual.

El método de fijación se basa en la matriz demanda y hace uso de una variable de proceso denominada C_{max} la cual se define como un valor provisional que corresponde a la demanda acumulada máxima que puede ser producida en cada periodo productivo.

El proceso de fijación funciona de la siguiente forma:

1. Se establece que $C_{max} = \frac{u_i}{I}$.
2. Se acumula la demanda desde el periodo siguiente en que fue fijado el último 1 hasta que el acumulador sobrepase el valor de C_{max} .
3. Se asigna el 1 en el periodo en que el acumulador excedió el valor de C_{max} .

Etapa de Ajuste. La etapa de ajuste es realizada una vez finalice cada ronda durante la etapa de asignación. Su finalidad consiste en aprovechar al máximo el espacio restante entre el inventario acumulado de cada periodo y la cota.

El método de ajuste (efectuado para cada producto) consiste en evaluar la posibilidad de que el periodo productivo fijado durante la ronda abarque un periodo adicional (mover el 1 a la derecha una posición), teniendo en cuenta la alteración que podría provocar en el inventario acumulado de los periodos comprendidos entre el periodo productivo de la ronda actual y el anterior. Esto con el fin de lograr una menor fijación de 1's totales para cada producto en la matriz Y_0 final.

Durante el proceso se utiliza la variable de proceso A_{max} la cual se define como un valor provisional que corresponde al ajuste máximo, es decir el espacio disponible en inventario respecto a la cota de inventario. Esta variable de proceso debe ser calculada para cada periodo comprendido entre el anterior (sin incluirlo) y el actual periodo donde haya producción ($y_t = 1$).

A modo de ejemplo, entiéndase la matriz Y_0 como una matriz lienzo (correspondiente a 3 productos y 6 periodos) habiendo efectuado únicamente la etapa de asignación hasta el segundo turno, el vector S_a como el inventario acumulado de cada periodo y el vector u_a como la cota del inventario para cada periodo.

$$Y_0 = \begin{bmatrix} 1 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$S_a = [7 \quad 5 \quad 6 \quad 7 \quad 6 \quad 4]$$

$$u_a = [9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9]$$

Ubicándose en la posición [1,4] de la matriz Y_0 se encuentra el periodo asignado como productivo para el respectivo producto durante la primera ronda.

El proceso de ajuste busca evaluar la posibilidad de que el periodo productivo anterior (posición [1,1]) abarque la demanda del periodo productivo actual mediante el siguiente proceso de comprobación:

1. Establecer que $A_{max} = u_a - S_a$. Para este ejemplo, el valor de A_{max} corresponde a 4, 3 y 2 para los periodos 2, 3 y 4 respectivamente.
2. Identificar el valor mínimo de A_{max} entre los periodos en que fue calculado. Para este ejemplo, el valor mínimo de A_{max} corresponde a 2.
3. Si el valor de la demanda en el periodo productivo actual es menor al valor de A_{max} , es posible que el periodo productivo actual sea movido una posición a la derecha, es decir a la posición [1,5], de lo contrario, no.

Generación de la Solución de Vecindario. La generación de solución de vecindario se basa en tres funciones, sumar, restar y mover. Cada una de estas funciones tendrá una probabilidad asociada que determinará cuál va a modificar a la matriz "solución inicial" (Y_0).

Función Sumar. La función sumar adiciona un periodo productivo a la matriz "solución actual" en un periodo aleatorio, de un producto aleatorio, en este caso el elemento seleccionado será y_{13} (resaltado en negrita).

Entiéndase la matriz "solución actual" como:

$$Y_a = \begin{bmatrix} 1 & 1 & \mathbf{0} & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

La solución de vecindario obtenida con el operador de adición será obtenida así:

1. Se selecciona aleatoriamente una posición y_{it}
2. Se verifica si $y_{it} = 0$
3. Si $y_{it} = 0$, pasar al paso 4. de lo contrario, volver al paso 1.
4. Asignar $y_{it} = 1$

Siguiendo estos pasos, la solución de vecindario obtenida basada en la matriz "solución actual" ejemplo y teniendo y_{13} como la posición aleatoria seleccionada, la matriz "nueva solución" que se obtiene aplicando la función sumar, es:

$$Y_n = \begin{bmatrix} 1 & 1 & \mathbf{1} & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Función Restar. Contrario a la función sumar, la función restar disminuye la cantidad de periodos productivos en un producto aleatorio. Para este ejemplo el elemento seleccionado aleatoriamente será y_{12} (resaltado en negrita)

Entiéndase la matriz "solución actual" como:

$$Y_a = \begin{bmatrix} 1 & \mathbf{1} & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

La solución de vecindario obtenida con el operador de adición será obtenida así:

1. Se selecciona aleatoriamente una posición y_{it}
2. Se verifica si $y_{it} = 0$.
3. Si $y_{it} = 0$ pasar al paso 4. de lo contrario, volver al paso 1.
4. Asignar $y_{it} = 1$.

Siguiendo estos pasos, la solución de vecindario obtenida basada en la matriz "solución actual" ejemplo y teniendo y_{12} como la posición aleatoria seleccionada, la matriz "nueva solución" que se obtiene aplicando la función sumar, es:

$$Y_n = \begin{bmatrix} 1 & \mathbf{0} & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Función Mover. La función mover desplazara, en un producto aleatorio, un periodo productivo a la derecha o la izquierda. Para determinar cuál será el desplazamiento que se va a realizar, se establece un numero aleatorio bajo distribución uniforme que determinará con una probabilidad uniforme, en qué dirección se realizará el cambio.

La función mover, manipulara la matriz así:

1. Se genera un numero aleatorio rand
2. Se establece la dirección del desplazamiento del periodo productivo así:
    a) $si\ rand \geq 0.5$, desplazar a la derecha
    b) $si\ rand < 0.5$, desplazar a la izquierda
2. Se busca aleatoriamente una posición $Y_{it} = 1$
3. Se desplaza el periodo productivo en la dirección establecida así:

a) Si se debe desplazar a la derecha, se establece $Y_{it} = 0$ y $Y_{it+1} = 1$.

b) Si se debe desplazar a la izquierda, se establece $Y_{it} = 0$ y $Y_{it-1} = 1$.

Entiéndase la matriz "solución actual" como:

$$Y_a = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & \mathbf{1} & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

A modo de ejemplo, se selecciona el elemento y_{32} (resaltado en negrita) y se desplazara hacia la derecha, aplicando la función Mover. La matriz de solución de vecindario obtenida será:

$$Y_a = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & \mathbf{0} & \mathbf{1} & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

A diferencia de las funciones Sumar y Restar, la función Mover no cambia la cantidad de periodos productivos del producto al cual se le aplica la función.

Generación de Vecindario. Para la creación de la solución de vecindario se selecciona una de las tres funciones y se aplicará a la matriz "solución actual", así:

1. Se genera un numero aleatorio rand
2. Se establece cuál función utilizar, así:
    a. $Si\ rand < 0.15$, aplicar función Restar
    b. $Si\ rand < 0.33$, aplicar función Sumar
    c. $Si\ rand < 1$, aplicar función Mover

Las probabilidades y el orden de prioridad en las funciones (cuáles tienen mayor probabilidad de ocurrencia) fueron halladas iniciando con probabilidades de 0.33, 0.66 y 1. Posteriormente se exploró con probabilidades de 0.15, 0.33 y 1, las cuales dieron mejores resultados, y, por lo tanto, so n las que finalmente se utilizan en el algoritmo. La probabilidad de

restar decrece cuando la cantidad de periodos productivos disminuye hasta el punto de afectar la factibilidad de la solución de vecindario.

La función restar es limitada por una cantidad mínima de 1's la cual asegura la factibilidad de la solución obtenida, el nombre de esta variable de proceso es C_{min}.

Capítulo 8

Diseño de Experimentos Computacionales

Instancias del Problema

Para evaluar el desempeño de la metaheurística se replicaron las instancias utilizadas por (Melo & Ribeiro, 2017). Se establecieron en total 12 instancias con cantidad de productos y cotas de inventario diferentes. Para las primeras seis instancias, el número de productos $I \in \{15, 30, 45\}$ y el número de periodos es de $T = 50$. La cota de inventario es $u_t \in \{500, 1000, 1500\}$ para las primeras tres instancias y $u_t \in \{375, 750, 1125\}$ para las tres restantes. Las siguientes seis instancias, son una versión de menor tamaño de las primeras seis. Los productos son $I \in \{3, 9, 15\}$ y el número de periodos es de $T = 18$. La cota de inventario es $u_t \in \{100, 300, 500\}$ para las primeras tres instancias y $u_t \in \{75, 225, 375\}$ para las tres restantes.

Las demandas y los costos fijos de preparación son valores enteros y aleatorios bajo distribución uniforme en los intervalos de [0, 25] y de [20, 150] respectivamente. No se considerarán costos de producción p_{it} ni de mantener inventario h_{it}, esto es constante para todos los periodos e instancias.

Las instancias están nombradas de la siguiente forma:

Numeración_número de productos_capacidad de almacenamiento

Tabla 7. Instancias trabajadas

Instancia	Productos	Capacidad de almacenamiento	Demanda aleatoria	Costo fijo de preparación
1_15_500	15	500	[0, 25]	[20, 150]
2_30_1000	30	1000	[0, 25]	[20, 150]
3_45_1500	45	1500	[0, 25]	[20, 150]
4_15_375	15	375	[0, 25]	[20, 150]
5_30_750	30	750	[0, 25]	[20, 150]
6_45_1125	45	1125	[0, 25]	[20, 150]
7_3_100	3	100	[0, 25]	[20, 150]
8_9_300	90	300	[0, 25]	[20, 150]
9_15_500	15	500	[0, 25]	[20, 150]
10_3_75	3	75	[0, 25]	[20, 150]
11_9_225	9	225	[0, 25]	[20, 150]
12_15_375	15	375	[0, 25]	[20, 150]

Con el objetivo de estudiar cómo los diferentes parámetros afectan el desempeño de la metaheurística, y por ende la calidad de la solución. Se probaron diferentes temperaturas finales, valores de factor de enfriamiento e iteraciones del algoritmo y así realizar un ajuste paramétrico de la metaheurística propuesta para el problema de tamaño de lote multiproducto no capacitado.

IBM CPLEX

Todas las instancias fueron ejecutadas en un computador con sistema operativo Windows 10, procesador Intel Core i7-8750H CPU 2.20Hhz, memoria RAM de 16GB 2400MHz usando IBM ILOG CPLEX Optimization Studio con todos sus ajustes estándar, los cuales son de 1 x 10 ^ - 6 de tolerancia del óptimo y 1 x 10^ - 5 de tolerancia de enteros. Se utilizó la formulación matemática estándar mostrada anteriormente y se implementó utilizando Optimization Programming Language (OPL) (ver Apéndice A).

Las instancias fueron ejecutadas en la versión 12.10.0 de CPLEX, en donde fue necesario establecer el *gap* como parámetro límite de cada ejecución, ya que al correr todas las instancias se observó que ninguna logra llegar al óptimo en menos de dos horas, por lo cual se decidió fijar una tolerancia relativa del *gap* en 4% y luego en 3%.

RStudio

Todas las instancias fueron ejecutadas en un computador con sistema operativo Windows 10, procesador Intel Core i7-8750H CPU 2.20Hhz, memoria RAM de 16GB 2400MHz usando en lenguaje de programación R versión 3.6.3 y la IDE RStudio 1.3.

Parámetros de Ejecución

Se utilizó la versión 3 de la metaheurística para ejecutar todas las instancias propuestas.

En el caso de la variable de proceso C_{max} se estableció dependiendo de cada instancia.

C_{min} fue establecida como 7 para las primeras 6 instancias y como 3 para las siguientes, estos valores fueron definidos con base en las soluciones obtenidas en CPLEX.

Diseño Factorial del Algoritmo

Se realizo un diseño factorial 3^3 con replicación, donde cada réplica del experimento contiene todas las posibles combinaciones de tratamientos. Se realizaron 5 réplicas, por lo que el número de observaciones es de 3 x 3 x 3 x 5 para un total de 135.

Este diseño factorial fue aplicado a cada una de las doce instancias establecidas para determinar el esquema de enfriamiento más apropiado para cada instancia.

Tabla 8. Parámetros del esquema de enfriamiento

	Bajo	Medio	Alto
Temperatura final T_f	0.001	0.01	0.1
Factor de enfriamiento α	0.85	0.95	0.99
Iteraciones L	30	50	100

Se establecieron las siguientes hipótesis, las cuales serán contrastadas en un ANOVA y así estudiar el efecto de estos factores sobre la calidad de las respuestas obtenidas en cada instancia con la metaheurística de enfriamiento simulado.

Hipótesis 1. No hay diferencia significativa en la calidad de la solución con diferentes temperaturas finales T_f.

Hipótesis 2. No hay diferencia significativa en la calidad de la solución con diferentes valores del factor de enfriamiento α.

Hipótesis 3. No hay diferencia significativa en la calidad de la solución con diferentes iteraciones del algoritmo *L*.

El análisis de varianza se realizó en el programa Minitab19 utilizando la opción ANOVA balanceado. También se usó la opción Modelo lineal general para general las comparaciones y las gráficas factoriales.

Teniendo en cuenta que se utilizó el valor del nivel de confianza estándar de 95%, el valor del estadístico de prueba corresponde a 0.05. Por lo tanto, se concluye que cualquier factor con un valor P inferior a 0.05 es significativo. Con este enfoque, las hipótesis pueden ser rechazadas o aceptadas de acuerdo con los resultados del estudio.

Instancia 1_15_500

Tabla 9. Análisis de varianza de Costo para la instancia 1_15_500

Fuente	GL	SC	MC	F	P
Temperatura final (Final)	2	194861	97430	1.32	0.272
Factor de enfriamiento (Factor)	2	41388456	20694228	279.61	0.000
Iteraciones (Iteraciones)	2	3899285	1949642	26.34	0.000
Final*Factor	4	483040	120760	1.63	0.172
Factor*Iteraciones	4	673903	168476	2.28	0.066
Final*Iteraciones	4	260683	65171	0.88	0.478
Final*Factor*Iteraciones	8	389482	48685	0.66	0.727
Error	108	7993138	74011		
Total	134	55282848			

Con la Tabla 9 es posible observar que el Factor de Enfriamiento y el Número de Iteraciones tienen un efecto significativo en el costo. Las hipótesis 2. y 3. se rechazan. La

temperatura final no tiene un efecto significativo en la calidad de la respuesta por lo que se acepta la hipótesis 1.

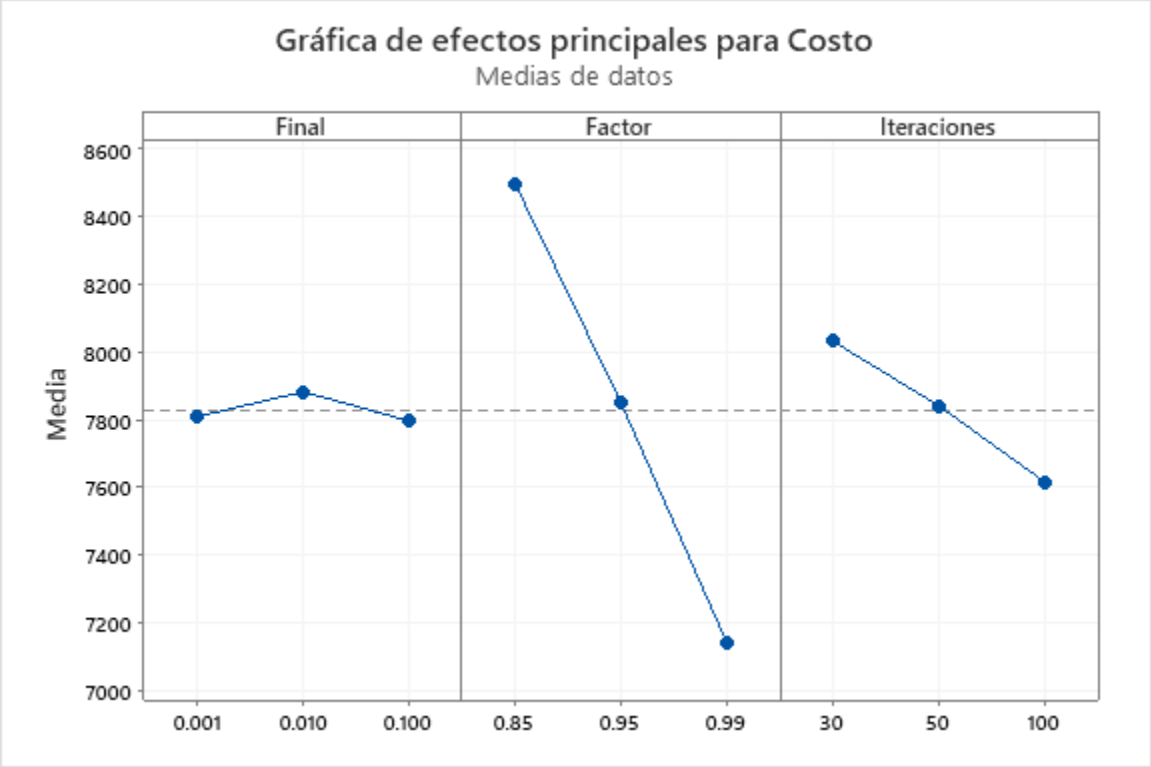

Figura 1. Gráfico de efectos principales para costo

Con la gráfica de efectos principales (Figura 1) es posible observar el efecto que tienen los diferentes factores. Se observa un efecto significativo del factor Factor de Enfriamiento y del factor Iteraciones, lo que confirma el rechazo de la hipótesis 2. y 3.

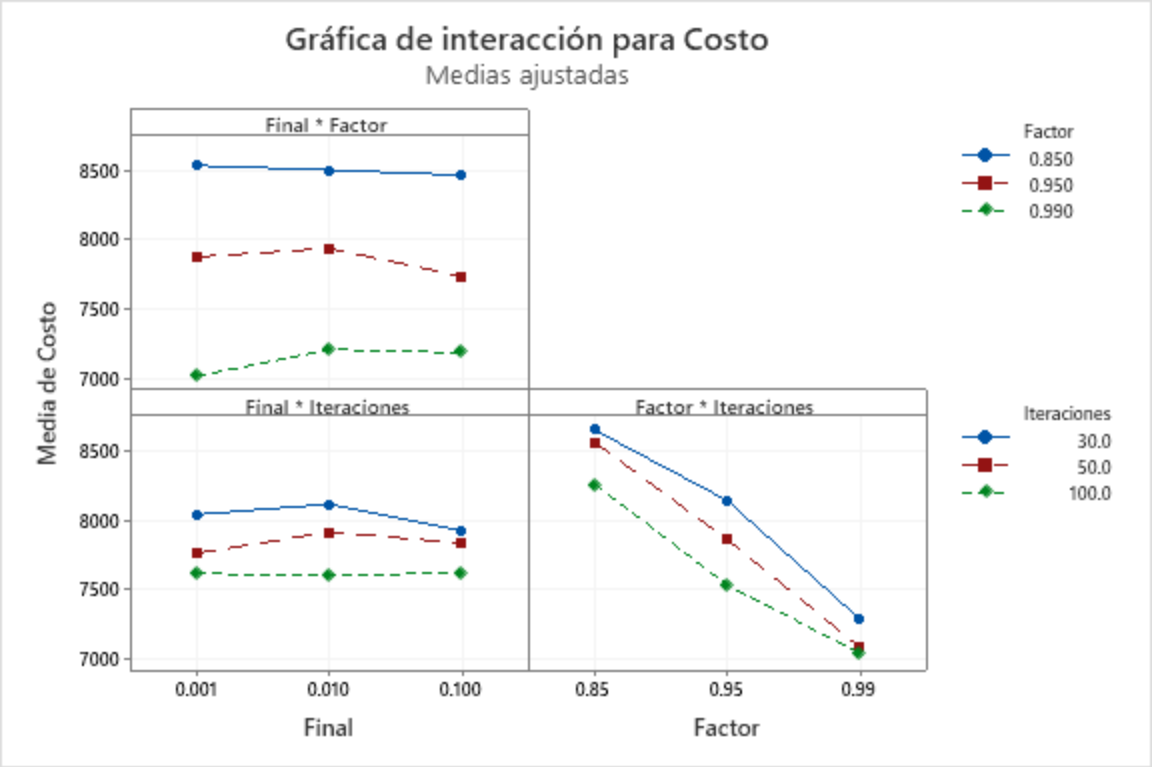

Figura 2. Gráfico de interacción para costo

Con la gráfica de interacciones (Figura 2**¡Error! No se encuentra el origen de la r eferencia.**) es posible observar el efecto que tienen los diferentes fatores cuando interactúan entre sí. También, es posible ver el poco efecto que tiene el factor Temperatura final en la media del costo. A pesar de que el valor P de la interacción entre Factor de Enfriamiento e Iteraciones es mayor a 0.05, sí posible encontrar en la gráfica, que su interacción tiene un efecto significativo en el valor de la media del costo.

*Tabla 10. Comparaciones por parejas de Fisher: Final*Factor*Iteraciones - Costo*

Final*Factor*Iteraciones	N	Media	Agrupación											
0.001 0.85 30	5	8818.8	A											
0.010 0.85 50	5	8679.6	A											
0.010 0.85 30	5	8594.0	A	B										
0.100 0.85 30	5	8553.6	A	B	C									
0.001 0.85 50	5	8521.2	A	B	C	D								
0.100 0.85 50	5	8501.8	A	B	C	D								
0.100 0.85 100	5	8329.0		B	C	D								
0.010 0.95 30	5	8293.4		B	C	D	E							
0.001 0.85 100	5	8248.2			C	D	E	F						
0.010 0.85 100	5	8220.6			C	D	E	F						
0.001 0.95 30	5	8211.6				D	E	F						
0.010 0.95 50	5	7963.6					E	F	G					
0.100 0.95 30	5	7935.2						F	G					
0.001 0.95 50	5	7831.4							G	H				
0.100 0.95 50	5	7801.2							G	H	I			
0.001 0.95 100	5	7583.4								H	I	J		
0.010 0.95 100	5	7552.2								H	I	J	K	
0.100 0.95 100	5	7474.0									I	J	K	
0.010 0.99 30	5	7472.0									I	J	K	
0.100 0.99 30	5	7298.6										J	K	L
0.100 0.99 50	5	7212.8											K	L
0.010 0.99 50	5	7111.0												L
0.001 0.99 30	5	7101.6												L
0.100 0.99 100	5	7057.6												L
0.010 0.99 100	5	7052.0												L
0.001 0.99 100	5	7019.4												L
0.001 0.99 50	5	6939.6												

		M
		M
		M
		M
		M
		M

*Tabla 11. Comparaciones por parejas de Fisher: Final*Factor*Iteraciones - Tiempo*

Final*Factor*Iteraciones	N	Media	Agrupación							
0.001 0.99 100	5	1815.46	A							
0.010 0.99 50	5	1107.60		B						
0.010 0.99 100	5	1075.16		B	C					
0.001 0.99 30	5	939.36			C	D				
0.100 0.99 50	5	917.56				D				
0.100 0.99 100	5	731.16					E			
0.010 0.99 30	5	730.62					E			
0.100 0.99 30	5	636.36					E			
0.001 0.99 50	5	600.35					E			
0.001 0.95 100	5	234.98						F		
0.100 0.95 100	5	195.18						F	G	
0.010 0.95 100	5	174.82						F	G	H
0.001 0.95 30	5	146.72						F	G	H
0.010 0.85 100	5	121.92						F	G	H
0.010 0.95 30	5	121.90						F	G	H
0.001 0.95 50	5	112.84						F	G	H
0.100 0.85 100	5	98.22						F	G	H
0.100 0.95 30	5	90.14							G	H
0.010 0.95 50	5	89.64							G	H
0.100 0.95 50	5	75.36							G	H
0.001 0.85 100	5	65.20							G	H
0.010 0.85 50	5	59.54							G	H
0.100 0.85 50	5	50.68								H
0.001 0.85 30	5	41.33								H
0.001 0.85 50	5	34.30								
0.100 0.85 30	5	27.31								
0.010 0.85 30	5	15.23								

En las tablas anteriores se agrupa la información utilizando el método LSD de Fisher y una confianza de 95%. Debido a que se busca el menor costo posible, con la Tabla 10 es posible concluir que el mejor esquema de enfriamiento para la iteración 1_15_500 es aquel cuyos factores son: Temperatura Final 0.001, Factor de Enfriamiento 0.99 y Numero de Iteraciones 50. Según la Tabla 11 este esquema de enfriamiento tarda una media de 600.35 segundos en alcanzar un costo medio de 6939.6.

Para el caso de las instancias 3_45_1500, 5_30_750 y 6_45_1125 no fue posible realizar el algoritmo con tres niveles para el Factor de enfriamiento, debido a que con un factor de 0.99, el algoritmo superaba el tiempo límite por ejecución de 3600s, algunos alcanzando tiempos de incluso dos horas. Por esta razón, para estas 3 instancias, se realizó un diseño factorial de 3

factores, dos de ellos con 3 niveles y el Factor de Enfriamiento con 2 niveles. Sin embargo, se realizaron las mismas cinco repeticiones, dando un total de observaciones de 3 x 2 x 3 x 5 = 90. Los parámetros del esquema de enfriamiento son:

Tabla 12. Parámetros del esquema de enfriamiento para las instancias 3_45_1500, 5_30_750, 6_45_1125

	Bajo	Medio	Alto
Temperatura final T_f	0.001	0.01	0.1
Factor de enfriamiento α	0.85	0.95	-
Iteraciones L	30	50	100

A continuación, se resume el diseño factorial. En la Tabla 13 se resume, para cada una de las instancias el efecto de los factores. Se rechaza o se acepta una hipótesis basado en el valor P obtenido en el estudio ANOVA realizado para cada instancia.

Tabla 13. Decisión de acuerdo con el análisis de varianza de costo

Instancia	Hipótesis 1.	Hipótesis 2.	Hipótesis 3.
1_15_500	Se acepta	Se rechaza	Se rechaza
2_30_1000	Se acepta	Se rechaza	Se rechaza
3_45_1500	Se rechaza	Se rechaza	Se rechaza
4_15_375	Se acepta	Se rechaza	Se rechaza
5_30_750	Se acepta	Se rechaza	Se rechaza
6_45_1125	Se acepta	Se rechaza	Se rechaza
7_3_100	Se acepta	Se rechaza	Se acepta
8_9_300	Se acepta	Se rechaza	Se acepta
9_15_500	Se acepta	Se rechaza	Se rechaza
10_3_75	Se acepta	Se rechaza	Se acepta
11_9_225	Se acepta	Se rechaza	Se rechaza
12_15_375	Se acepta	Se rechaza	Se rechaza

Tabla 14. Esquema de enfriamiento resultado del ajuste paramétrico para cada instancia

Instancia	Valor medio de la función objetivo	Tiempo medio (s)	Esquema de enfriamiento		
			Temperatura final	Factor de Enfriamiento	Numero de Iteraciones
1_15_500	6939.6	600.35	0.001	0.99	50
2_30_1000	13528.8	2023.85	0.01	0.99	100
3_45_1500	22665.2	522.62	0.01	0.95	100
4_15_375	8788.6	930.32	0.1	0.99	100
5_30_750	18657.2	517.83	0.001	0.95	100
6_45_1125	28165.0	481.22	0.1	0.95	100
7_3_100	744.8	70.509	0.01	0.99	100
8_9_300	2184.4	40.931	0.001	0.95	100
9_15_500	3327.0	249.835	0.01	0.99	100
10_3_75	874.0	31.8782	0.001	0.95	100
11_9_225	2517.2	208.799	0.001	0.99	50
12_15_375	3811.0	358.404	0.001	0.99	50

Estabilización del Algoritmo

A continuación, se muestra la gráfica de estabilización para la instancia 7_3_100. Con esta gráfica es posible determinar que la solución sí se estabiliza con el esquema de enfriamiento establecido. A pesar de que las configuraciones iniciales tienen un costo y una variabilidad muy alta, se demuestra que rápidamente converge a un valor mínimo. Es posible observar el algoritmo tarda más de la mitad del tiempo estabilizado.

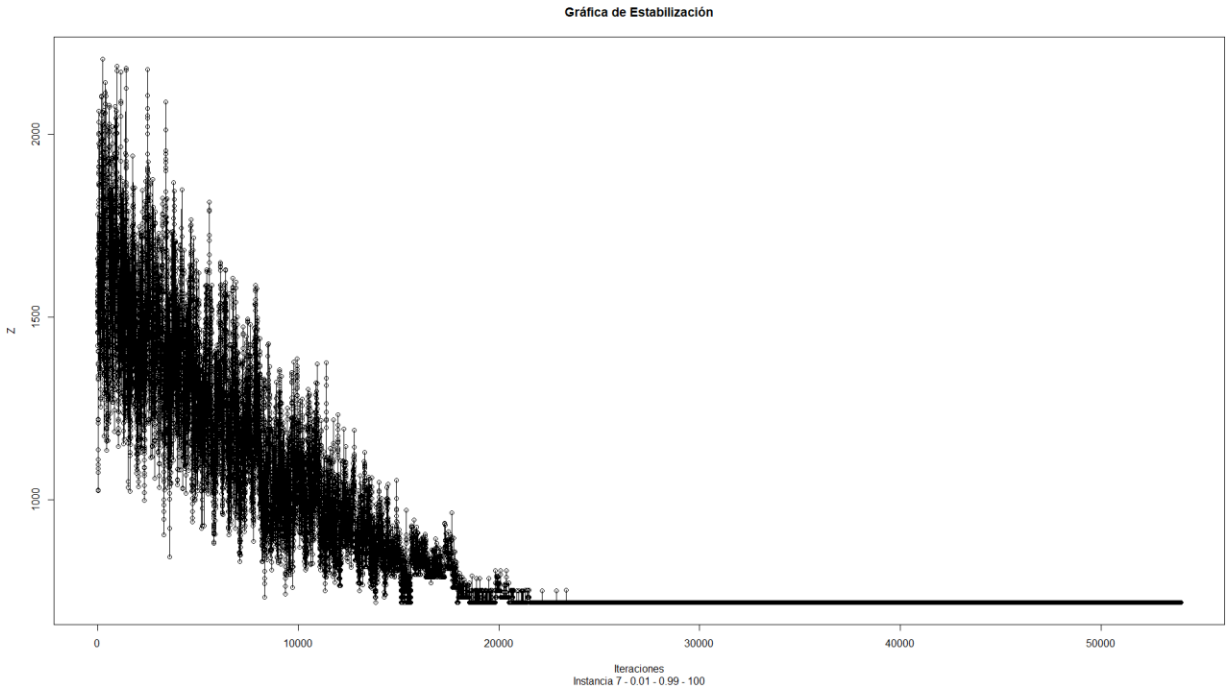

Figura 3. Gráfica de Estabilización Instancia 7_3_100

Capítulo 9

Resultados y Discusión

Estructura de la solución óptima

Las tablas Tabla 15, Tabla 16 y Tabla 17 muestran en detalle la solución óptima para un problema multiproducto de tamaño de lote con cotas de inventario para la instancia más pequeña que se estudió, la instancia 7_3_100. Esto, con el propósito de ilustrar con detalle los datos paramétricos y la solución óptima de un problema de tamaño de lote multiproducto con cotas de inventario.

Tabla 15. Frecuencia de producción óptima (variables y_{it})

Periodos (t)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total	Costo
y_{1t}	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	4	219
y_{2t}	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	5	312
y_{3t}	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	3	187
Total	3	0	1	0	1	0	2	0	1	0	1	0	2	0	0	0	1	0	12	718

Tabla 16. Tamaño de lote de producción óptimo (variables x_{it})

Periodos (t)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total	Costo
x_{1t}	58	0	0	0	0	0	23	0	80	0	0	0	40	0	0	0	0	0	201	0
x_{2t}	16	0	65	0	0	0	37	0	0	0	58	0	0	0	0	0	36	0	212	0
x_{3t}	43	0	0	0	88	0	0	0	0	0	0	0	42	0	0	0	0	0	173	0
Total	117	0	65	0	88	0	60	0	80	0	58	0	82	0	0	0	36	0	586	0

Tabla 17. Nivel de inventario óptimo (variables s_{it})

Periodos (t)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total	Costo
s_{1t}	57	51	37	24	4	0	13	0	62	55	31	23	58	35	21	13	9	0	493	0
s_{2t}	5	0	52	39	19	1	24	13	8	0	37	16	6	6	6	0	17	0	249	0
s_{3t}	35	15	11	0	71	63	58	35	30	22	18	0	35	35	25	20	15	0	488	0
Total	97	66	100	63	94	64	95	48	100	77	86	39	99	76	52	33	41	0	1230	0

Tabla 18. Resumen de los costos asociados con la solución óptima

Costos relacionados	Total
Costos Fijos de Preparación (q_{it})	718
Costos de Mantener Inventario (h_{it})	0
Costos de Producción (p_{it})	0
Costos Totales	718

Esta instancia consiste en 18 periodos productivos (T = 18), con tres productos (I = 3). La demanda para cada producto es:

Tabla 19. Demanda de la instancia 7_3_100 (variables d_{it})

Periodos (t)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total
d_{1t}	1	6	14	13	20	4	10	13	18	7	24	8	5	23	14	8	4	9	201
d_{2t}	11	5	13	13	20	18	14	11	5	8	21	21	10	0	0	6	19	17	212
d_{3t}	8	20	4	11	17	8	5	23	5	8	4	18	7	0	10	5	5	15	173
Total	20	31	31	37	57	30	29	47	28	23	49	47	22	23	24	19	28	41	586

Los costos fijos de preparación, de mantener inventario y de producción son los mismos para todas las instancias. Se asume una capacidad de almacenamiento de inventario de $u_t = 100$ constante para todos los periodos.

En términos producción y de nivel de inventario de producto terminado, el plan óptimo es el de producir en los periodos $t = \{1,7,9,13\}$, $t = \{1,3,7,11,17\}$ y $t = \{1,5,13\}$ para los tres productos respectivamente. La Tabla 16 muestra el tamaño de lote que dan como resultado los niveles de inventario de la Tabla 17. Esto representa un costo total de 718.

Desempeño de la metaheurística propuesta

Los indicadores de desempeño para definir la calidad de la solución en términos de la precisión están basados en medir la distancia de la solución obtenida a la solución óptima global obtenida con CPLEX. Esto permite una evaluación más absoluta del desempeño de la metaheurística. El gap está determinado por la diferencia absoluta entre la solución obtenida por la metaheurística y la solución óptima de CPLEX $|f(s) - f(s^*)|/f(s^*)$, donde s* es la solución óptima global (Talbi, 2009).

Las instancias que no pudieron ser resueltas hasta un valor óptimo con CPLEX fueron limitadas utilizando el parámetro Límite de Tiempo Global el cual tomo como valor el mismo tiempo que se reportó para la mejor cota obtenida por la metaheurística.

La mejor cota conseguida por la metaheurística se obtuvo de realizar 10 ejecuciones con los parámetros especificados en la Tabla 14 y seleccionando la que tuviera el menor valor de función objetivo.

Tabla 20. Resultados comparando la metaheurística con la solución óptima

Instancia	Mejor cota obtenida (CPLEX)	Tiempo (s)	Gap	Mejor cota obtenida (metaheurística)	Tiempo (s)	Gap
1_15_500	5871	435.72	2.55%	6260	435.69	9.34%
2_30_1000	11039	1805.92	1.39%	13082	1803.65	20.15%
3_45_1500	16372	594.05	1.93%	19918	593.2	24.01%
4_15_375	7659	929.19	2.93%	8597	927.96	15.54%
5_30_750	14373	497.5	2.11%	18068	493.58	28.36%
6_45_1125	21.265	504.25	2.10%	25003	503.62	20.05%
					Media geométrica	19.58%

Los resultados en la Tabla 20 muestran que, para instancias con 50 periodos, los gaps alcanzados por la metaheurística muestran mejores resultados para las instancias con cotas de inventario más altas, dando un gap promedio de 17.83%, las instancias con cotas más ajustadas obtuvieron un promedio de 21.32%. Se encontraron las mejores soluciones para la instancia 1_15_500 que corresponde a la instancia de menor tamaño de este grupo, con 15 productos y 500 de cota de inventario y un gap de 9.34%.

Tabla 21. Resultados comparando la metaheurística con la solución para las instancias de 18 periodos

Instancia	Mejor cota obtenida (CPLEX)	Tiempo (s)	Gap	Mejor cota obtenida (metaheurística)	Tiempo (s)	Gap
7_3_100	718	0.13	0%	718	81.75	0%
8_9_300	2033	2.2	0%	2134	47.633	4.97%
9_15_500	2937	3.86	0%	3092	469.49	5.28%
10_3_75	874	0.13	0%	874	30.485	0%
11_9_225	2401	2.28	0%	2456	138.459	2.29%
12_15_375	3535	29.61	0%	3694	190.073	4.50%
					Media geométrica	2.84%

La metaheurística alcanzó soluciones óptimas en dos ocasiones. Su desempeño con respecto a CPLEX para las instancias más pequeñas (Tabla 12.) fue notablemente mejor alcanzando gaps promedio de 2.84%.

Capítulo 10

Conclusiones y Futuras Investigaciones

Se realizó una revisión de la literatura que reúne los principales avances del estudio del problema de tamaño de lote y se analizaron las propuestas de implementación de algoritmos metaheurísticos. Esto permitió identificar la metaheurística de Enfriamiento Simulado como un enfoque apropiado para la solución del problema de tamaño de lote multiproducto no capacitado con cotas de inventario. Esta metaheurística es ampliamente conocida por proveer soluciones óptimas o cercanas al óptimo para una gran variedad de problemas de optimización, además es una metaheurísticas sencilla de aplicar por requerir de pocos parámetros y tener una gran cantidad de referencias en la literatura.

Se estudió la estructura del problema de tamaño de lote identificando los parámetros, las variables, la función objetivo y sus respectivas restricciones. Aportando al entendimiento y facilitando la lógica de programación del problema. Posteriormente el planteamiento se tradujo al lenguaje de programación OPL de forma que utilizando el software IMB ILOG CPLEX se hallaron las soluciones para las instancias planteadas.

Se implementó el algoritmo metaheurístico en el lenguaje de programación R, de forma que generara una solución factible para cada instancia del problema. Se estableció el esquema de enfriamiento apropiado para cada instancia mediante el desarrollo de un diseño de experimentos factorial, de forma que favoreciera la calidad de la solución.

Se midió el desempeño de la metaheurística comparando las soluciones obtenidas con las soluciones de CPLEX. Se concluyó que CPLEX presenta un mejor desempeño en cuanto a la calidad de la solución y su respectivo tiempo computacional. Sin embargo, cabe aclarar que para instancias de mayor tamaño (de 50 periodos) CPLEX requiere de horas de ejecución y tiene una

exigencia computacional muy alta para hallar el óptimo. Razón por la cual, para estas instancias, se utilizó el tiempo computacional de la metaheurística para realizar la comparación. A pesar de que la metaheurística no logra gaps comparables con los de CPLEX, este algoritmo demuestra su potencial al lograr soluciones factibles en un tiempo razonable. Además de alcanzar respuestas óptimas en las instancias más pequeñas.

Esta clase de metaheurísticas, por su facilidad de implementación y de adaptabilidad para diferentes tipos de problemas de optimización, da paso a un sinnúmero de avances en el área de investigación de operaciones, donde se espera que a futuro se puedan diseñar propuestas de metaheurísticas en combinación con otras técnicas de optimización que reduzcan su tiempo de ejecución y además reduzcan el gap con respecto a la solución óptima, facilitando así su aplicación en contextos industriales.

Para futuras investigaciones se propone ampliar los métodos de generación de solución inicial y de vecindario, de forma que reduzca su tiempo computacional y contribuya a la calidad de la solución. Se plantea aprovechar la estructura de separación de variables para profundizar en la aplicación de estructuras de construcción de soluciones factibles mediante Algoritmos Dinámicos de Vecindad (VNS). También, se propone realizar una investigación aplicada del problema de tamaño de lote con cotas de inventario en las múltiples industrias en las que se ha demostrado su aplicación y resolverlo aplicando esta metaheurística.

Referencias

Absi, N., Detienne, B., & Phane Dauz Ere-Pé, S. (2012). *Heuristics for the multi-item capacitated lot-sizing problem with lost sales*. https://doi.org/10.1016/j.cor.2012.06.010

Akbalik, A., Penz, B., & Rapine, C. (2014). Multi-item uncapacitated lot sizing problem with inventory bounds. *Optimization Letters*, *9*(1), 143–154. https://doi.org/10.1007/s11590-014-0746-6

Aydin, L., & Artem, H. (2017). *Fiber Technology for Fiber-Reinforced Composites* (O. Seydibeyoglu, A. Mohanty, & M. Misra (Eds.)). Woodhead Publishing. https://doi.org/https://doi.org/10.1016/C2015-0-05497-1

Brahimi, N., Dauzere-Peres, S., & Wolsey, L. A. (2010). Polyhedral and Lagrangian approaches for lot sizing with production time windows and setup times. *Computers & Operations Research*, *37*, 182–188. https://doi.org/10.1016/j.cor.2009.04.005

Carvalho, D. M., & Nascimento, M. C. V. (2016). *Lagrangian heuristics for the capacitated multi-plant lot sizing problem with multiple periods and items*. https://doi.org/10.1016/j.cor.2016.01.019

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2017). Solving discrete lot-sizing and scheduling by simulated annealing and mixed integer programming. *Computers and Industrial Engineering*, *114*, 235–243. https://doi.org/10.1016/j.cie.2017.10.017

Cunha, J. O., Kramer, H. H., & Melo, R. A. (2019). Effective matheuristics for the multi-item capacitated lot-sizing problem with remanufacturing. *Computers and Operations Research*, *104*, 149–158. https://doi.org/10.1016/j.cor.2018.12.012

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). *A survey on new generation metaheuristic algorithms*. https://doi.org/10.1016/j.cie.2019.106040

Erromdhani, R., Jarboui, B., Eddaly, M., & Rebaï, A. (2011). Metaheuristics for the multi-item capacitated lot-sizing problem with time windows and setup times. *2011 4th International Conference on Logistics, LOGISTIQUA'2011*, *January 2014*, 520–525. https://doi.org/10.1109/LOGISTIQUA.2011.5939453

Fabiano, C., Toledo, M., Resende, R., De Oliveira, R., & Morelato Franc -A C, P. (2012). *A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging*. https://doi.org/10.1016/j.cor.2012.11.002

Glock, C. H., Grosse, E. H., & Ries, J. M. (2014). The lot sizing problem: A tertiary study. *International Journal of Production Economics*, *155*, 39–51. https://doi.org/10.1016/J.IJPE.2013.12.009

Guan, Y., & Liu, T. (2010). Stochastic lot-sizing problem with inventory-bounds and constant order-capacities. *European Journal of Operational Research*, *207*(3), 1398–1409. https://doi.org/10.1016/j.ejor.2010.07.003

Jans, R., & Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, *177*(3), 1855–1875. https://doi.org/10.1016/j.ejor.2005.12.008

Kiran, D. R. (2019). Elements of production planning and control. In *Production Planning and Control* (pp. 1–20). Butterworth-Heinemann. https://doi.org/10.1016/b978-0-12-818364-9.00001-9

Koken, P., Raghavan, V. A., & Yoon, S. W. (2018). *A genetic algorithm based heuristic for dynamic lot sizing problem with returns and hybrid products*. https://doi.org/10.1016/j.cie.2018.03.040

Lee, I., & Yang, J. (2009). Comprehensive Chemometrics. In S. Brown, B. Walczak, & R.

Tauler (Eds.), *Comprehensive Chemometrics*. Elsevier. https://www.elsevier.com/books/comprehensive-chemometrics/brown/978-0-444-52701-1

Mahdi, W., Medjahed, S. A., & Ouali, M. (2017). Performance analysis of simulated annealing cooling schedules in the context of dense image matching. *Computacion y Sistemas*, *21*(3), 493–501. https://doi.org/10.13053/CyS-21-3-2553

Mann, Z. A. (2017). The Top Eight Misconceptions about NP-Hardness. *Computer*, *50*(5), 72–79. https://doi.org/10.1109/MC.2017.146

Melo, R. A., & Ribeiro, C. C. (2017). Formulations and heuristics for the multi-item uncapacitated lot-sizing problem with inventory bounds. *International Journal of Production Research*, *55*(2), 576–592. https://doi.org/10.1080/00207543.2016.1215567

Overview of optimization. (2006). *Process Systems Engineering*, *7*(C), 285–314. https://doi.org/10.1016/S1874-5970(06)80012-3

Parsopoulos, K. E., Konstantaras, I., & Skouri, K. (2015). Metaheuristic optimization for the Single-Item Dynamic Lot Sizing problem with returns and remanufacturing. *Computers and Industrial Engineering*, *83*, 307–315. https://doi.org/10.1016/j.cie.2015.02.014

Phouratsamay, S.-L., & Cheng, T. C. E. (2019). The single-item lot-sizing problem with two production modes, inventory bounds, and periodic carbon emissions capacity. *Operations Research Letters*, *47*, 339–343. https://doi.org/10.1016/j.orl.2019.06.003

Rao, S. S. (2009). *Engineering Optimization: Theory and Practice, Fourth Edition*.

Roshani, A., Giglio, D., & Paolucci, M. (2016). A simulated annealing approach for the capacitated dynamic lot sizing problem in a closed remanufacturing system. *IFAC-PapersOnLine*, *49*(12), 1496–1501. https://doi.org/10.1016/j.ifacol.2016.07.783

Sifaleras, A., & Konstantaras, I. (2015a). General variable neighborhood search for the multi-

product dynamic lot sizing problem in closed-loop supply chain. *Electronic Notes in Discrete Mathematics*, *47*, 69–76. https://doi.org/10.1016/j.endm.2014.11.010

Sifaleras, A., & Konstantaras, I. (2015b). *Variable neighborhood descent heuristic for solving reverse logistics multi-item dynamic lot-sizing problems*. https://doi.org/10.1016/j.cor.2015.10.004

Talbi, E.-G. (2009). *Metaheuristics From Design To Implementation*. Jhon Wiley & Sons, Inc.

Tang, O. (2004). Simulated annealing in lot sizing problems. *International Journal of Production Economics*, 173–181. https://doi.org/10.1016/j.ijpe.2003.11.006

Wagner, H. M., & Whitin, T. M. (1958). Dynamic Version of the Economic Lot Size Model Stable URL : http://www.jstor.org/stable/2626974 DYNAMIC VERSION OF THE ECONOMIC LOT SIZE MODEL * t. *Management Science*, *5*(1), 89–96.

Wu, T., Xiao, F., Zhang, C., He, Y., & Liang, Z. (2018). The green capacitated multi-item lot sizing problem with parallel machines R. *Computers and Operations Research*, *98*, 149–164. https://doi.org/10.1016/j.cor.2018.05.024

Xiao, Y., Kaku, I., Zhao, Q., & Zhang, R. (2011). *Neighborhood search techniques for solving uncapacitated multilevel lot-sizing problems*. https://doi.org/10.1016/j.cor.2011.06.004

Apéndice A. Código CPLEX

```
//Parámetros
int t = ...;                    //Numero de Periodos
range T = 1..t;                 //Set de Periodos
int i = ...;                    //Número de Productos
range I = 1..i;                 //Set de Productos
int d[I][T]=...;                //Demanda
float q[I][T]=...;              //Costo Fijo de Preparación
float h[I][T]=...;              //Costo de Mantener Inventario
float p[I][T]=...;              //Costo de Producción
float u[T]=...;                 //Capacidad de Almacenamiento
int dT[I][T]=...;               //Demanda Acumulada Restante

//Variables de decisión
dvar float+ x[I][T];            //Tamaño Lote de Producción
dvar float+ s[I][T];            //Inventario
dvar boolean y[I][T];           //Variable Binaria
```

```
//Función objetivo
minimize sum(i in I, t in T) ((q[i][t]*y[i][t])+(h[i][t]*s[i][t])+(p[i][t]*x[i][t]));
subject to {
//Restricciones
forall (i in I)
        setI:x[i][1]==d[i][1]+s[i][1];
forall(t in 2..t)
        forall (i in I)
                setII:s[i][t-1]+x[i][t]==d[i][t]+s[i][t];
forall(t in T)
        forall (i in I)
                setIII:x[i][t]<=dT[i][t]*y[i][t];
forall(t in T)
        forall (i in I)
                setIV:sum(i in I)(s[i][t])<=u[t];
}
```

Apéndice B. Pseudocódigo Algoritmo de enfriamiento simulado versión 3

Parámetros y variables

Para facilitar el entendimiento del pseudocódigo, se muestra a continuación una explicación detallada de los parámetros y variables utilizadas en el algoritmo. En la Tabla 22 se dan a conocer los tipos de dato de las variables.

Tabla 22. Algoritmo ES – Tipos de dato

Nombre	Definición
num	Tipo numérico
$logic$	Tipo lógica
$matNum$	Tipo matriz numérica
$matBin$	Tipo matriz binaria
$vecNum$	Tipo vector numérico

Tabla 23. Algoritmo ES – Parámetros y variables de ejecución

Símbolo	Definición	Tipo de dato	Etapa de uso
t_o	Temperatura inicial.	num	Global
t_c	Temperatura actual	num	Global
t_f	Temperatura final.	num	Global
α	Factor de enfriamiento.	num	Global
L	Iteraciones del algoritmo.	num	Global

Tabla 24. Algoritmo ES – Parámetros y variables del problema

Símbolo	Definición	Tipo de dato	Etapa de uso
T	Número de periodos	num	Global
I	Número de productos a producir.	num	Global
d_{it}	Demanda para el producto i en el periodo t.	$matNum$	Global
q_{it}	Costo fijo de preparación para el producto i en el periodo t.	$matNum$	Global
h_{it}	Costo de mantener inventario para el producto i en el periodo t	$matNum$	Global
p_{it}	Costo de producción del producto i en el periodo t.	$matNum$	Global
u_t	Capacidad de almacenamiento de inventario en el periodo t.	$vecNum$	Global
y_{it}	Variable binaria que indica si hay de preparación para el producto i en el periodo t ($y_{it} = 1$) o no ($y_{it} = 0$).	$matBin$	Global
x_{it}	Tamaño de lote de producción del producto i en el periodo t.	$matNum$	Global
s_{it}	Inventario para el producto i al final del periodo t.	$matNum$	Global

Tabla 25. Algoritmo ES – Parámetros y variables de proceso del algoritmo

Símbolo	Definición	Tipo de dato	Etapa de uso
$deltaZ$	Diferencia entre $z_{actual} - z_{vecindario}$	num	Global
C_{max}	Demanda acumulada máxima (valor de referencia)	num	Global
C_{min}	Cantidad mínima de periodos productivos (1's)	num	Global
bN	Número muy grande, generalmente 9999999999	num	Global
sN	Número pequeño, generalmente es negativo	num	Global
rI	Producto aleatorio	num	Global
rT	Periodo aleatorio a partir del segundo	num	Global
iT	Posición del periodo productivo	num	Global / Generar X
fA	Factor A de la ecuación	num	Global / Generar Z
fB	Factor B de la ecuación	num	Global / Generar Z
fC	Factor C de la ecuación	num	Global / Generar Z
pT_i	Posición del 1 asignado en el turno anterior para el producto i	$vecNum$	S. Inicial
cT_i	Posición del 1 asignado en el turno actual para el producto i	$vecNum$	S. Inicial
dA	Acumulador de la demanda de cierto producto en ciertos periodos	num	S. Inicial / Asignación
sA	Absoluto, acumulador del inventario actual de todos los productos en cierto periodo	num	S. Inicial / Ajuste
dN	Diferencia entre capacidad de almacenamiento y almacenamiento utilizado	num	S. Inicial / Ajuste
oC	Cantidad exacta de periodos productivos de cierto producto (1's)	num	S. Vecindario
rP	Probabilidad aleatoria uniforme	num	S. Vecindario
aP	Probabilidad de añadir un periodo productivo	num	S. Vecindario
dP	Probabilidad de quitar un periodo productivo	num	S. Vecindario
mP	Probabilidad de mover un periodo productivo	num	S. Vecindario
$exit$	Variable lógica de cumplimiento de condición	$logic$	S. Vecindario

Pseudocódigos

A continuación, se presenta la estructura general del algoritmo y sus pasos principales.

Estructura General

Tabla 26. Algoritmo ES – Estructura general del algoritmo

Pseudocódigo Algoritmo ES. Estructura general
Paso 6. INICIO Paso 7. Inicializar variables Paso 8. Generar solución inicial Y_0 Paso 9. Ejecutar el módulo de Control de temperatura Paso 9.1. **WHILE** $t_c > t_f$ **DO**: Paso 9.2. **FOR** $k = 1$ **TO** L Paso 9.2.1. Generar Solución de Vecindario Paso 9.2.2. Calcular función objetivo Paso 9.2.3. Calcular deltaZ Paso 9.2.4. **IF** $deltaZ \leq 0$ **THEN** establecer Solución de Vecindario como Solución Actual Paso 9.2.5. **IF** $deltaZ > 0$ **THEN** generar un número aleatorio $rand \in [1, 0]$ Paso 9.2.5.1. **IF** $rand \leq e^{\left(-\frac{deltaZ}{t_c}\right)}$ **THEN** establecer Solución de Vecindario como Solución Actual. Paso 9.2.6. Establecer $k = k + 1$ Paso 9.3. Establecer $t_c = t_c \cdot \alpha$ Paso 9.4. Actualizar variables de probabilidad de la solución de vecindario Paso 9.4.1. $dP = dP \cdot \alpha$ Paso 9.4.2. $aP = aP \cdot \alpha$ **Paso 10. FIN**

Pasos principales

Tabla 27 . Algoritmo ES – Generación de la solución inicial

Pseudocódigo Algoritmo ES. Estructura general. Paso 3 – Generar Solución Inicial

Paso 1. INICIO
Paso 2. Declarar variables de proceso
Paso 2.1. $y = [[\quad]]$
Paso 2.2. $x = [[\quad]]$
Paso 2.3. $s = [[\quad]]$
Paso 2.4. $pT = [\quad]$
Paso 2.5. $cT = [\quad]$
Paso 2.6. $dA = 0$
Paso 2.7. $sA = 0$
Paso 2.8. $bN = 99999999$
Paso 2.9. $sN = -99999999$
Paso 3. Asignar primeros unos
Paso 3.1. **FOR** $i = 1$ **TO** I
Paso 3.1.1. **IF** $d_{i,1} > 0$ **THEN**
Paso 3.1.1.1. Establecer $Y_{i,1} = 1$
Paso 3.1.1.2. Establecer $pT_i = 1$
Paso 3.1.2. Establecer $i = i + 1$
Paso 4. Asignar y ajustar el resto de unos
Paso 4.1. **WHILE** $pT_I <> T$ **DO**
Paso 4.1.1. **FOR** $i = 1$ **TO** I
Paso 4.1.1.1. **FOR** $t = pT_i$ **TO** T (Inicio de la etapa de asignación)
Paso 4.1.1.1.1. **IF** $pT_i < T$ **THEN**
Paso 4.1.1.1.1.1. **IF** $dA < C_{max}$ **THEN**
Paso 4.1.1.1.1.1.1. Establecer $dA = dA + d_{i,t}$
Paso 4.1.1.1.1.1.2. Establecer $t = t + 1$
Paso 4.1.1.1.1.2. **ELSE**
Paso 4.1.1.1.1.1.1. Establecer $cT_i = t - 1$
Paso 4.1.1.1.1.1.2. Establecer $y_{i,t-1} = 1$
Paso 4.1.1.1.1.1.3. Establecer $dA = 0$
Paso 4.1.1.1.1.1.4. **BREAK**
Paso 4.1.1.1.2. **ELSE**
Paso 4.1.1.1.1.1. **BREAK**
Paso 4.1.1.2. Establecer $i = i + 1$
Paso 4.1.2. **FOR** $i = 1$ **TO** I
Paso 4.1.2.1. Generar x a partir de y
Paso 4.1.2.2. Generar s a partir de x
Paso 4.1.2.3. **FOR** $t = pT_i$ **TO** cT_i (Inicio de la etapa de ajuste)
Paso 4.1.1.1.1. Calcular sA
Paso 4.1.1.1.2. Establecer $dN = u_t - sA$
Paso 4.1.1.1.3. **IF** $dN < bN$ **THEN**
Paso 4.1.1.1.1.1. Establecer $sN = dN$
Paso 4.1.1.1.4. Establecer $t = t + 1$
Paso 4.1.2.4. **IF** $sN < u_t$ **THEN**
Paso 4.1.1.1.1. **IF** $d_{i,cT_i} < u_t$ **THEN**
Paso 4.1.1.1.1.1. Establecer $y_{i,cT_i} = 0$
Paso 4.1.1.1.1.2. Establecer $cT_i = cT_i + 1$
Paso 4.1.1.1.1.3. Establecer $y_{i,cT_i} = 1$
Paso 4.1.2.5. **ELSE**
Paso 4.1.1.1.1. Establecer $i = i + 1$
Paso 4.1.3. Establecer $pT_i = cT_i$
Paso 5. FIN

Tabla 28. Algoritmo ES – Generación de la solución inicial – Generar X

Pseudocódigo Algoritmo ES. Estructura general. Paso 3 – Generar Solución Inicial – Paso 4.1.2.1 – Generar X
Paso 1. **INICIO** Paso 2. **FOR** $i = 1$ **TO** I Paso 2.1. Establecer $iT = 1$ Paso 2.2. Establecer $dA = d_{i,1}$ Paso 2.3. **FOR** $t = 2$ **TO** T Paso 2.3.1. **IF** $t <> T$ **THEN** Paso 2.3.1.1. **IF** $y_{i,t} = 0$ **THEN** Paso 4.1.1.1.1. Establecer $dA = dA + d_{i,t}$ Paso 2.3.1.2. **ELSE** Paso 4.1.1.1.1. Establecer $x_{i,iT} = dA$ Paso 4.1.1.1.2. Establecer $iT = t$ Paso 4.1.1.1.3. Establecer $dA = d_{i,iT}$ Paso 2.3.2. **ELSE** Paso 2.3.2.1. **IF** $y_{i,t} = 0$ **THEN** Paso 4.1.1.1.1. Establecer $dA = dA + d_{i,t}$ Paso 4.1.1.1.2. Establecer $x_{i,iT} = dA$ Paso 2.3.2.2. **ELSE** Paso 4.1.1.1.1. Establecer $x_{i,iT} = dA$ Paso 4.1.1.1.2. Establecer $x_{i,t} = d_{i,t}$ Paso 2.3.3. Establecer $t = t + 1$ Paso 2.4. Establecer $i = i + 1$ Paso 3. **FIN**

Tabla 29. Algoritmo ES – Generación de la solución inicial – Generar S

Pseudocódigo Algoritmo ES. Estructura general. Paso 3 – Generar Solución Inicial – Paso 4.1.2.2 – Generar S
Paso 1. **INICIO** Paso 2. **FOR** $i = 1$ **TO** I Paso 2.1. Establecer $s_{i,1} = x_{i,1} - d_{i,1}$ Paso 2.2. **FOR** $t = 2$ **TO** T Paso 2.2.1. Establecer $s_{i,j} = x_{i,j} - d_{i,j} + s_{i,j-1}$ Paso 2.2.2. Establecer $t = t + 1$ Paso 2.3. Establecer que $i = i + 1$ Paso 3. **FIN**

Tabla 30. Algoritmo ES – Generación de la solución de vecindario

Pseudocódigo Algoritmo ES. Estructura general. Paso 4.2.1 – Generar Solución de Vecindario
Paso 1. **INICIO** Paso 2. Declarar variables de proceso Paso 2.1. $oC = 0$ Paso 2.2. $aP = 0.33$ Paso 2.3. $dP = 0.15$ Paso 2.4. $mP = 1$ Paso 3. Generar rI Paso 4. Calcular la cantidad de periodos productivos de rI Paso 4.1.1. **FOR** $t = 1$ **TO** T Paso 4.1.1.1. **IF** $y_{rI,t} = 1$ **THEN** Paso 4.1.1.1.1. Establecer $oC = oC + 1$ Paso 4.1.1.2. Establecer $t = t + 1$ Paso 5. Generar rP Paso 6. Efectuar operación Paso 6.1. **IF** $oC > C_{min}$ **THEN** Paso 6.1.1. **IF** $rP < dP$ **THEN** Paso 6.1.1.1. Ejecutar Función Restar Paso 6.1.2. **ELSE IF** $rP = aP$ **THEN** Paso 6.1.2.1. Ejecutar Función Sumar Paso 6.1.3. **ELSE IF** $rP = mP$ **THEN** Paso 6.1.3.1. Ejecutar Función Mover Paso 6.2. **ELSE IF** $oC = C_{min}$ **THEN** Paso 6.2.1. **IF** $rP < aP$ **THEN** Paso 6.2.1.1. Ejecutar Función Sumar Paso 6.2.2. **IF** $rP < mP$ **THEN** Paso 6.2.2.1. Ejecutar Función Mover Paso 7. **FIN**

Tabla 31. Algoritmo ES – Generación de la solución de vecindario – Función Sumar

Pseudocódigo Algoritmo ES. Estructura general. Paso 4.2.1 – Generar Solución de Vecindario – Función Sumar
Paso 1. **INICIO** Paso 2. Declarar variables de proceso Paso 2.1. rI (Referencia) Paso 2.2. $exit = false$ Paso 3. **WHILE** $exit = false$ **DO** Paso 3.1. Generar rT Paso 3.2. **IF** $y_{rI,rT} = 0$ **THEN** Paso 3.2.1. Establecer $y_{rI,rT} = 1$ Paso 3.2.2. Establecer $exit = true$ Paso 3.3. **ELSE** Paso 3.3.1. Establecer $exit = false$ Paso 4. **FIN**

Tabla 32. Algoritmo ES – Generación de la solución de vecindario – Función Restar

Pseudocódigo Algoritmo ES. Estructura general. Paso 4.2.1 – Generar Solución de Vecindario – Función Restar
Paso 1. **INICIO** Paso 2. Declarar variables de proceso Paso 2.1. rI (Referencia) Paso 2.2. $exit = false$ Paso 3. **WHILE** $exit = false$ **DO** Paso 3.1. Generar rT Paso 3.2. **IF** $y_{rI,rT} = 1$ **THEN** Paso 3.2.1. Establecer $y_{rI,rT} = 0$ Paso 3.2.2. Establecer $exit = true$ Paso 3.3. **ELSE** Paso 3.3.1. Establecer $exit = false$ Paso 4. **FIN**

Tabla 33. Algoritmo ES – Generación de la solución de vecindario – Función Mover

Pseudocódigo Algoritmo ES. Estructura general. Paso 4.2.1 – Generar Solución de Vecindario – Función Mover
Paso 1. **INICIO** Paso 2. Declarar variables de proceso Paso 2.1. rI (Referencia) Paso 2.2. $exit = false$ Paso 3. **WHILE** $exit = false$ **DO** Paso 3.1. Generar rT_1 (Variación de rango de 1 hasta $T - 1$) Paso 3.2. Generar rT_2 (Variación de rango de 3 hasta T) Paso 3.3. Generar rP Paso 3.4. **IF** $rP \geq 0.5$ **THEN** Paso 3.4.1. **IF** $y_{rI,rT} = 1$ **THEN** Paso 3.4.1.1. Establecer $y_{rI,rT} = 0$ Paso 3.4.1.2. Establecer $y_{rI,rT+1} = 1$ Paso 3.4.1.3. Establecer $exit = true$ Paso 3.5. **ELSE IF** $rP < 0.5$ **THEN** Paso 3.5.1. **IF** $y_{rI,rT_2} = 1$ **THEN** Paso 3.5.1.1. Establecer $y_{rI,rT_2} = 0$ Paso 3.5.1.2. Establecer $y_{rI,rT_2-1} = 1$ Paso 3.5.1.3. Establecer $exit = true$ Paso 3.5.2. Paso 3.6. Paso 3.6.1. Establecer $exit = false$ Paso 4. **FIN**

Tabla 34. Algoritmo ES – Estructura general del algoritmo – Calcular función objetivo

Pseudocódigo Algoritmo ES. Estructura general. Paso 4.2.2 – Calcular función objetivo
Paso 1. **INICIO**
Paso 2. Declaración de variables
Paso 2.1. $fA = 0$
Paso 2.2. $fB = 0$
Paso 2.3. $fC = 0$
Paso 3. **FOR** $i = 1$ **TO** I
Paso 3.1. **FOR** $t = 1$ **TO** T
Paso 3.1.1. Establecer que $fA = fA + \left((q_{i,t}) \cdot (y_{i,t})\right)$
Paso 3.1.2. Establecer que $fB = fB + \left((h_{i,t}) \cdot (s_{i,t})\right)$
Paso 3.1.3. Establecer que $fC = fC + \left((p_{i,t}) \cdot (x_{i,t})\right)$
Paso 3.1.4. Establecer que $t = t + 1$
Paso 3.2. Establecer que $i = i + 1$
Paso 4. **FIN**