ALGORITMO HEURÍSTICO BASADO EN LISTAS TABÚ PARA LA PLANIFICACIÓN DE LA PRODUCCIÓN EN SISTEMAS MULTINIVEL CON LISTAS DE MATERIALES ALTERNATIVAS Y ENTORNOS DE COPRODUCCIÓN

ING. ALFONSO RAFAEL ROMERO CONRADO

UNIVERSIDAD DE LA COSTA - CUC

MAESTRÍA EN INGENIERÍA

BARRANQUILLA

2018

ALGORITMO HEURÍSTICO BASADO EN LISTAS TABÚ PARA LA PLANIFICACIÓN DE LA PRODUCCIÓN EN SISTEMAS MULTINIVEL CON LISTAS DE MATERIALES ALTERNATIVAS Y ENTORNOS DE COPRODUCCIÓN

ING. ALFONSO RAFAEL ROMERO CONRADO

Trabajo de Grado como requisito para optar al título de
MAGÍSTER EN INGENIERÍA CON ÉNFASIS EN INGENIERÍA INDUSTRIAL

TUTOR: PhD. JAIRO CORONADO HERNÁNDEZ

COTUTOR: PhD. RENSO RAÚL VISBAL ACEVEDO

UNIVERSIDAD DE LA COSTA

DEPARTAMENTO DE GESTIÓN INDUSTRIAL, AGROINDUSTRIAL Y OPERACIONES

MAESTRÍA EN INGENIERÍA

BARRANQUILLA, COLOMBIA

2018

Nota de aceptación

_____

_____

_____

Jurado

_____

Jurado

_____

Barranquilla, abril de 2018

Dedicatoria

Dedico este trabajo a Dios: el motor de mi vida y la razón de todo lo que he logrado, todo lo que estoy haciendo y todo lo que lograré.

A mi principal motivación: mis padres Nelly Conrado y Alfonso Romero, mi hermana Madis y mi hermano Juan Carlos. Durante estos dos años hemos logrado mucho: hemos comprendido el verdadero valor de la familia y que el esfuerzo y dedicación tiene su recompensa.

Alfonso R. Romero-Conrado

Agradecimientos

Agradecimientos a la Universidad de la Costa, al programa de becas Opción y al señor rector Tito José Crissien por todo el apoyo recibido, la confianza depositada y las oportunidades de crecimiento personal, académico y profesional a lo largo de los últimos años.

A mis amigos de toda la vida Yulineth, Mayra, Alex, Stefania y Samir. A los compañeros de estudio y trabajo que siempre estuvieron prestos a colaborar con la investigación.

A la Ing. Lauren Castro Bolaño por su constante guía y motivación, por convertirse en un referente de profesionalismo y permitirme iniciar esta ardua y gratificante carrera en investigación.

Agradezco al Dr. Jairo Coronado Hernández y al Dr. Renso Visbal Acevedo por su ferviente apoyo en el desarrollo de este proyecto y su constante búsqueda de perfección, rigor académico y calidad científica. Muchas gracias por su voto de confianza y por su guía constante en mis inicios en el exigente mundo de la investigación operativa.

Alfonso R. Romero-Conrado

Contenido

Lista de Figuras

Lista de Tablas

Resumen

En esta investigación se presenta el desarrollo un algoritmo heurístico basado en los principios de búsqueda tabú para la solución del problema de lotificación multinivel con restricciones de capacidad, listas de materiales alternativas y entornos de coproducción, basado en la estructura del modelo de Planificación de Materiales y Operaciones Genéricas GMOP propuesto en el año 2013. El algoritmo propuesto utiliza el mecanismo de memoria a corto plazo (Lista Tabú) para la selección de Strokes alternativos para la fabricación de cada producto. La validación del algoritmo se realizó analizando la calidad y los tiempos de obtención de las soluciones. El algoritmo demostró potencial al alcanzar porcentajes de diferencia entre el 10% y 17% con respecto a las soluciones óptimas en los problemas de mayor tamaño y un equilibrio entre calidad y tiempos de solución problemas relativamente pequeños.

*Palabras clave**:** lotificación, multinivel, listas de materiales alternativas, coproducción, lista tabú, GMOP.*

Abstract

This research shows the development process of a heuristic algorithm based on the principles of taboo search for the solution of the capacitated multilevel lot sizing problem with alternate bill of materials and co-production environments, based on the structure of the Generic Materials and Operations Planning model (GMOP). The proposed algorithm uses the short-term memory mechanism (Taboo List) for the selection of alternate strokes to produce each product. The validation of the algorithm was carried out analyzing the quality and the solution times. The algorithm demonstrated potential by reaching difference percentages around 10% and 17% compared with optimal solutions in large problems and a balance between quality and solution times when is used in relatively small problems.

Keywords: lot sizing, multilevel, alternate bill of materials, coproduction, tabu list, GMOP.

Capítulo 1. Consideraciones Generales

1.1.Introducción

La definición de los tamaños de lote representa una de las decisiones más importantes en la planificación de los sistemas productivos. Los modelos de lotificación buscan garantizar el cumplimiento de los requerimientos de la demanda, estableciendo un equilibrio entre los costos asociados al inventario de producto disponible y los costos generados por los alistamientos necesarios para el procesamiento de lotes.

La complejidad de esta decisión ha aumentado junto con el nivel de flexibilidad de los sistemas productivos, cada vez más globalizados y enfocados hacia la personalización en masa. Las estructuras de producto son cada vez más elaboradas y la necesidad de contar con modelos de planificación que describan en mayor manera la realidad de los sistemas productivos conlleva considerar un gran número de restricciones.

Dependiendo del tamaño y la complejidad del problema, el uso de métodos exactos para la definición de tamaños de lote puede resultar poco eficiente en términos de tiempos de solución. Este trabajo tiene por objetivo proponer un método heurístico basado en el funcionamiento de las listas tabú para la lotificación en sistemas productivos en los que se contemplan restricciones de capacidad de los recursos, la existencia de listas de materiales alternativas y la presencia de entornos de coproducción. Esto como alternativa a la utilización del método GMOP, un modelo exacto propuesto por Garcia-Sabater et al., (2013).

El presente capítulo brinda una contextualización al problema estudiado, define los objetivos y establece el marco metodológico implementado. Los capítulos 2 y 3 presentan los

conceptos básicos referentes a los modelos de lotificación y la búsqueda tabú, así como los principales antecedentes de su aplicación en problemas de planificación de la producción.

El capítulo 4 muestra una descripción y las generalidades del algoritmo propuesto y en capítulo 5 se muestran el análisis y la discusión de los resultados obtenidos con el uso de la heurística, determinando su viabilidad para la solución eficiente del problema planteado y exponiendo las oportunidades de trabajo futuro.

1.2.Planteamiento del Problema

El tamaño de los lotes es una de las variables de decisión más importantes dentro de la planificación de un sistema productivo, debido a que su definición está condicionada y relacionada con una gran cantidad de factores como son la disponibilidad de la materia prima, las capacidades instaladas, los costos de inventario, los costos de mano de obra, entre otros. Un buen modelo de lotificación debe permitir responder de manera eficiente y oportuna a los requerimientos de la demanda buscando un equilibrio entre costos y teniendo en cuenta las restricciones presentes en el sistema.

La complejidad de los sistemas productivos aumenta junto con el nivel de flexibilidad del proceso y la cantidad de restricciones y consideraciones presentes. La existencia de listas de materiales alternativas y la presencia de coproductos son características que denotan un alto nivel de flexibilidad en un sistema productivo y a su vez añaden un mayor grado de complejidad al momento de definir tamaños de lote.

Recientemente se han propuesto modelos de lotificación que permiten considerar estos factores de complejidad y dentro de los más recientes se encuentra el Modelo de Planificación Genérico de Materiales y Operaciones GMOP (*Generic Materials and Operations Planning*) propuesto por Maheut, Garcia-Sabater, Garcia-Sabater, & Valero Herrero (2011). Es un modelo de programación entera mixta que, utilizando el concepto de *Stroke*, permite dar solución al problema capacitado de lotificación multinivel.

El problema capacitado de lotificación, en algunas de sus diversas variaciones, es considerado como *NP Hard* (Buschkühl, 2008) y como sucede en la mayoría de estos problemas, la complejidad y el tamaño de la instancia afecta de manera importante la

capacidad del modelo para obtener soluciones óptimas en tiempos de ejecución razonables para un nivel de planificación operativa.

Adicionalmente, los sistemas ERP (*Enterprise Resource Planning*) más utilizados cuentan con módulos MRP *(Materials Requirement Planning*) que en ocasiones no son lo suficientemente flexibles para responder de manera eficiente a aspectos relevantes de la planificación de la producción, especialmente en ciertas industrias en las que se involucran parámetros de coproducción y listas de materiales alternativas. En el caso de aquellos sistemas más robustos, la adquisición e implementación de estas herramientas puede resultar costosa y muchas veces no son rentables para las empresas productoras de menor tamaño (SAP, 2017).

El uso de modelos heurísticos y meta heurísticos para la solución de problemas de planificación de la producción es ampliamente estudiado en la literatura científica (Buschkühl, Sahling, Helber, & Tempelmeier, 2010). Con su utilización se busca mejorar la calidad y los tiempos de obtención de soluciones con respecto a los métodos exactos y así dar soporte a la toma de decisiones en horizontes de planificación operativos. Dentro de las metaheurísticas, los algoritmos de búsqueda tabú son frecuentemente utilizados y han resultado efectivos para gran cantidad de problemas de planificación y *scheduling (Capítulo 4)* debido a que utilizan mecanismos de memoria (Listas Tabú) que optimizan la búsqueda dentro del espacio de soluciones factibles del problema, disminuyendo los tiempos de solución. Teniendo en cuenta lo anterior, **¿Puede un algoritmo heurístico basado en la utilización de listas tabú obtener soluciones factibles y cercanas a valores óptimos para el problema de lotificación multinivel con listas de materiales alternativas y coproducción?**

1.3. Objetivos

1.3.1. Objetivo General

- Diseñar un algoritmo heurístico basado en listas tabú para la solución del problema capacitado de lotificación en sistemas de producción multinivel con listas de materiales alternativas y entornos de coproducción.

1.3.2. Objetivos Específicos

- Recopilar y describir las principales investigaciones en torno a la planificación de la producción y el problema lotificación en sistemas de producción multinivel.
- Programar un algoritmo heurístico que permita la obtención de soluciones para el problema propuesto, sujeto a restricciones de capacidad, listas de materiales alternativas y coproducción.
- Determinar la viabilidad del uso y funcionamiento del algoritmo propuesto como una alternativa eficiente en calidad y tiempos de solución para el problema definido.

1.4. Justificación

La productividad empresarial es un factor especialmente importante a nivel macroeconómico ya que afecta indicadores relacionados con el desarrollo económico de los países. La eficiencia de los procesos y las organizaciones se sostiene mayormente en sus procesos logísticos y de planificación.

"Incrementar la productividad de las empresas colombianas a partir de la sofisticación y diversificación del aparato productivo" es uno de los objetivos plasmados dentro del Plan de Desarrollo Nacional 2014-2018. La sofisticación del aparato productivo depende en gran medida y deposita su potencial de desarrollo en los aportes provenientes de proyectos de ciencia y tecnología e innovación, especialmente aquellos relacionados con gestión operativa y optimización.

El presente proyecto busca aportar al mejoramiento de las capacidades tecnológicas para la planificación de la producción a nivel industrial mediante la propuesta y validación de una alternativa de solución para la planificación de sistemas productivos multinivel, aplicable y no limitada a sectores industriales importantes como por el ejemplo, el sector automotriz o de autopartes y las industrias química y farmacéutica.

Así mismo, a nivel institucional, el trabajo se enmarca en la línea de investigación de Sostenibilidad Corporativa y las sub-líneas de "*Gestión Empresarial*" y "*Competitividad, Emprendimiento e Innovación Empresarial",* bajo el aval del grupo *Productividad y Competitividad: PRODUCOM* de la Universidad de la Costa.

1.5. Metodología

La metodología de la investigación se enmarca en un enfoque cuantitativo. Dado que no existen antecedentes sobre la utilización de métodos de solución heurísticos para la solución del problema capacitado de lotificación multinivel con listas de materiales alternativas y coproducción, la investigación tendrá un alcance exploratorio, descriptivo y correlacional.

Se desarrollaron las siguientes etapas, necesarias para la consecución de los objetivos del proyecto:

Fase 1. Revisión Bibliográfica.

En esta etapa se realiza una revisión bibliográfica basada en fuentes primarias de información (artículos científicos, libros, tesis de maestría y doctorado, etc.) con el fin de construir un marco conceptual y referencial que permita establecer de manera clara el alcance exploratorio de la utilización de algoritmos de búsqueda tabú en la solución del problema capacitado de lotificación en sistemas de producción multinivel.

La revisión bibliográfica incluye artículos y documentos presentes en las bases de datos especializadas como *Science Direct, Springer Link* y las bases citacionales de *Scopus* y *Web of Science.*

Fase 2. Definición del problema y Generación de una Solución Inicial.

En esta etapa, mediante un enfoque descriptivo, se estudia el problema específico de planeación de la producción en sistemas multinivel con listas de materiales alternativas y restricciones de capacidad en entornos de coproducción a través del modelo GMOP de (Garcia-Sabater et al., 2013). Se establecen los parámetros del problema, las variables de

decisión, las restricciones y la función de costos o función objetivo que serán considerados en el método heurístico. Se propone un mecanismo para la generación aleatoria de las instancias o casos de prueba que se utilizarán para validar el modelo. Y adicionalmente se desarrolla un método heurístico para la obtención de soluciones iniciales que permiten la inicialización del algoritmo de basado en Listas Tabú.

Fase 3. Construcción y Validación del Algoritmo Heurístico basado en Listas Tabú

En esta fase, se desarrolla un algoritmo heurístico basado en Listas Tabú para la solución del problema estudiado. A través del uso de instancias generadas aleatoriamente de manera controlada, se evalúa el impacto en términos de calidad de la solución y tiempos de ejecución en el abordaje del problema mediante el uso del algoritmo heurístico propuesto, comparando sus resultados con los obtenidos mediante un método exacto.

1.6. Estructura del trabajo de grado

El trabajo de grado se encuentra dividido en 6 capítulos, incluyendo el presente capítulo introductorio. El contenido de los capítulos siguientes se resume a continuación.

Capítulo 2: En este capítulo se expone y conceptualiza el problema de lotificación, se presentan las principales clasificaciones del problema de lotificación y los elementos que definen su complejidad. Se describen las generalidades del uso del modelo de Planificación de Materiales y Operaciones Genéricas (*GMOP*) para la solución del problema capacitado de lotificación multinivel con listas de materiales alternativas y coproducción.

Capítulo 3: El tercer capítulo conceptualiza y brinda un acercamiento a los algoritmos de búsqueda tabú. Se exponen los principales antecedentes de su aplicación en la solución de problemas de planificación de la producción, especialmente en sistemas de producción multinivel. Con los resultados y conclusiones de los capítulos 2 y 3 se evidencia el cumplimiento del primer objetivo específico definido.

Capítulo 4. Este capítulo muestra la arquitectura y el funcionamiento del algoritmo heurístico basado en listas tabú para resolver el problema de lotificación en sistemas multinivel con listas de materiales alternativas y coproducción. Se muestran las consideraciones generales para su diseño y el procedimiento utilizado para la generación de las instancias aleatorias utilizadas en el *Capítulo 5.* Mediante las conclusiones y resultados de este capítulo se evidencia el cumplimiento del segundo objetivo específico definido.

Capítulo 5. En este capítulo se exponen y analizan los resultados del análisis computacional realizado en la solución de las instancias de prueba generadas. Se describe el procedimiento para la obtención de la solución exacta para cada instancia y para la

optimización de los parámetros iniciales del algoritmo heurístico. A través de las conclusiones y resultados de este capítulo se evidencia el cumplimiento del tercer objetivo específico definido.

Por último, se exponen las principales conclusiones del trabajo de grado, así como las oportunidades identificadas de trabajo futuro.

Capítulo 2. Planificación de la producción en sistemas multinivel

2.1. Introducción

La definición de los tamaños de lote a producir es una de las decisiones más complejas e importantes dentro del proceso de la planificación de la producción, el cual se desarrolla como un proceso jerárquico que conlleva decisiones en el largo, mediano y corto plazo.

La definición del tamaño de lote para cada producto debe buscar el balance entre *costos de alistamiento* (producción de grandes lotes) y *costo de almacenamiento* (producción de lotes pequeños con mayor variedad de productos) (Drexl & Kimms, 1997). Como afirman Bahl, Ritzman, & Gupta (1987), el problema de lotificación toma mayor importancia cuando se presentan altos tiempos o costos de alistamiento y tiene sus orígenes con la definición del conocido modelo EOQ *(Economic Order Quantity*) (Harris, 1913).

Este capítulo muestra las generalidades y consideraciones fundamentales para entender la complejidad en la solución del problema de lotificación. Se exponen los conceptos fundamentales y los elementos presentes en la gran cantidad de variaciones de los modelos de lotificación. Finalmente, se expone y describe el funcionamiento del modelo GMOP como la base para solución del problema capacitado de lotificación multinivel.

2.2. Elementos del problema de lotificación

Las variaciones del problema de lotificación presentes en la bibliografía científica son innumerables. Como afirman Karimi, Fatemi Ghomi, & Wilson (2003), existen factores del

proceso o restricciones que afectan de manera directa la complejidad, la modelación y la clasificación de los problemas de lotificación:

2.2.1. Número de Niveles

Una primera clasificación de los problemas de lotificación está dada por el número de niveles de la estructura de materiales del producto. Cuando el producto final es simple y su proceso no requiere ensambles o subensambles, estamos frente a un *problema de un solo nivel o "Single-Level Problem"* (Dixon & Silver, 1981; Hindi, 1996; Karimi et al., 2003; Karimi, Ghomi, & Wilson, 2006; Maes & Wassenhove, 1988). Este tipo de problema se presenta, por ejemplo, cuando el producto es fabricado directamente de la materia prima con una simple operación. Usualmente la demanda de los productos de un solo nivel es totalmente independiente (Karimi et al., 2003).

En segundo lugar se encuentran los *problemas multinivel* o *"Multi-Level Problems*"(Chen, 2015; Gaury et al., 2016; Kimms, 1996; Toledo, de Oliveira, & Morelato França, 2013) en donde el producto final es resultado de una serie de operaciones sucesivas de ensamble o tratamiento. En este tipo de problemas, los ítems resultantes de un nivel (operación) son la entrada del siguiente nivel (Karimi et al., 2003). Por consiguiente las decisiones para la lotificación de un determinado ítem de un producto dependen de las decisiones tomadas para los ítems en el nivel superior (Bahl et al., 1987).

2.2.2. Número de productos

El número de productos terminados o productos finales es otro factor que considerar y afecta de manera importante la complejidad del problema. De acuerdo a esto, existen los problemas con un solo ítem o "*Single-Item Problems*" (Brahimi, Dauzere-Peres, Najid, & Nordli, 2006; Choudhary & Shankar, 2011; Hindi, 1995, 1996; Karimi et al., 2006; Kirca,

1990; Parsopoulos, Konstantaras, & Skouri, 2015) y los problemas con múltiples productos finales o "*Multi-Item Problems"*(Caserta & Rico, 2009; Sambasivan & Yahya, 2005; Sifaleras & Konstantaras, 2015; Toledo et al., 2013). El nivel de complejidad de los problemas de lotificación con múltiples productos suele ser mayor que la complejidad de los modelos mono producto. (Karimi et al., 2003)

2.2.3. Restricciones de capacidad o recursos

Cuando existen restricciones en alguno de los recursos productivos (maquinaria, materia prima, mano de obra, capital etc.) estamos frente a un problema con restricciones de capacidad o "*Capacitated Lotsizing Problem - CLSP*" (Boonmee & Sethanan, 2016; Caserta & Rico, 2009; Chen, 2015; Gaury et al., 2016; Quadt & Kuhn, 2008). En este tipo de problemas, la complejidad aumenta sustancialmente con respecto a un problema sin restricciones de este tipo, conocido como "*Uncapacitated Lotsizing Problem-UCLSP*"(Brahimi et al., 2006; Karimi et al., 2003). Los modelos de lotificación capacitados deben entonces buscar la satisfacción de la demanda, cumpliendo con las restricciones en los recursos.

2.2.4. Restricciones de almacenamiento

La naturaleza perecedera de ciertos productos puede aumentar la complejidad de los problemas de lotificación. Esto debido a que no es conveniente conservarlo por mucho tiempo en inventario y por consiguiente los tiempos de almacenamiento se deben minimizar (Kämpf & Köchel, 2006).

2.2.5. Características de la demanda

La demanda puede comportarse de manera estacionaria o constante durante el periodo de planeación (demanda estática) o puede variar con el paso del tiempo (demanda dinámica). Si los valores de la demanda son conocidos con anterioridad estamos frente a una demanda determinística. En cambio, si los valores de la demanda son inciertos o pueden ser descritos por una función de probabilidad, estamos hablando de una demanda probabilística. La complejidad de los problemas de lotificación es mayor para casos de demanda dinámica y/o probabilística (Karimi et al., 2003).

2.2.6. Estructura de los alistamientos

La complejidad de los alistamientos radica en la dependencia de estos con respecto a la secuencia de operaciones. Si los tiempos o costos de alistamiento no dependen de la secuencia de las operaciones, nos referimos a una estructura de alistamientos simple.

Por otra parte, si estos costos o tiempos varían considerablemente al cambiar de tipo de producto en la secuencia de producción, estamos frente a una estructura de alistamientos compleja ("*Complex Setup").* En este tipo de estructura, los valores de las variables de alistamiento pueden ser menores ("*Minor Setup*") o mayores ("*Major Setup*") y dependiendo de la similitud de los procesos entre los cambios de lote o las características dentro de familias de productos, en ocasiones se considera que el alistamiento no es necesario y se puede "arrastrar" al siguiente periodo de tiempo ("*Setup Carry-over");* en estos casos, la complejidad del problema de lotificación aumenta (Karimi et al., 2003).

2.2.7. El horizonte de planificación.

El horizonte de planificación es el periodo de tiempo en el cual el plan de producción se extiende en el futuro. Existen horizontes finitos e infinitos de planificación y la complejidad del modelo de lotificación puede variar de acuerdo con el tamaño de los intervalos de tiempo en los que se realice la planificación ("*Time buckets*": horas, días, meses etc.).

En los problemas con "*Big buckets*" (Quadt & Kuhn, 2008) (Akartunalı & Miller, 2009) el periodo de tiempo es suficientemente largo para producir múltiples ítems, mientras que en los problemas "*Small bucket*" el periodo de tiempo es tan corto que solo es posible la producción de un solo ítem (Karimi et al., 2003).

2.2.8. El manejo de inventarios

Escasez de inventario ("*Shortage*"): Si es posible cumplir parcialmente la demanda de un periodo y satisfacer los faltantes en el periodo siguiente, se dice que los pedidos atrasados "*Backlogging case*" son permitidos. Por otro lado, cuando es permitido no cumplir totalmente con los pedidos, estamos frente un caso de ventas perdidas o ("*Lost Sales Case*") (Karimi et al., 2003).

2.3. Coproducción

La coproducción es el fenómeno que se presenta en aquellos entornos de manufactura en los cuales, debido a cambios físicos o químicos en el material, se generan productos derivados o secundarios diferentes del producto principal (Öner & Bilgiç, 2008).

Un entorno de coproducción puede presentarse en sistemas de manufactura con cualquier nivel de tecnificación, que no necesariamente estén totalmente controlados, que no sean completamente entendidos y que controlarlos pueda resultar muy costoso. En este caso

se hace referencia a un entorno de coproducción no controlado y algunos de los ejemplos incluyen la industria de vidrio (Öner & Bilgiç, 2008), la producción de semiconductores (Bitran & Gilbert, 1994) y la industria alimenticia (Bravo, Rodríguez, & Medina, 2009).

Por otro lado, es posible que el fenómeno de coproducción sea mayormente entendido y sea posible controlar las salidas del sistema de manufactura (Deuermeyer & Pierskalla, 1978). En ese caso se trata de un entorno controlado de coproducción y se presenta, por ejemplo, en la industria automotriz y de autopartes (Vidal-Carreras, Garcia-Sabater, & Coronado-Hernandez, 2012), y la producción y corte de madera (Wery, Gaudreault, Thomas, & Marier, 2018).

Los denominados coproductos pueden servir como insumo para otros procesos de fabricación o ser comercializados independientemente, razón por la cual el factor de coproducción tiene un alto impacto sobre el control de inventarios y existencias de productos finales y componentes.

2.4. Las listas de materiales (*Bill of Materials – BOMs*)

Las listas de materiales son un elemento esencial en el funcionamiento de los métodos de planificación de la producción como el MRP y el MRP II. Las listas de materiales contienen la información de los componentes, partes y sub-partes que son necesarias para la fabricación de un producto.

Se emplea el concepto de Lista Directa de Materiales (*Direct BOM*) para referirse a estructuras de producto convergentes, en las cuales un producto final está compuesto por el ensamble de varias partes componentes. Por otra parte, está el concepto de Lista Inversa de Materiales (*Reverse BOM*), el cual denota una estructura en la cual, a partir de un producto o componente pueden derivarse uno o varios subproductos (Maheut, 2013). Este es el caso de

los procesos en los cuales hay presencia de coproductos, la lista de materiales será divergente ya que a partir de un material o insumo se pueden generar varios productos resultantes. El diagrama de Gozinto (Vazsonyi, 1954) es la representación gráfica más utilizada para describir la relación existente entre los componentes de un producto complejo. El factor de Gozinto representa la cantidad del respectivo componente que es necesario para elaborar el componente o producto en el nivel superior.

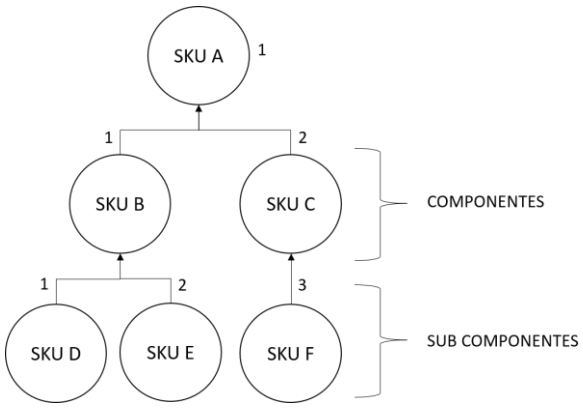


Figura 1. Representación de Gozinto.

La estructura de la lista de materiales aporta un grado de complejidad adicional al modelamiento de los sistemas productivos. La mayoría de los modelos de planificación multinivel presentes en la literatura se basan en la representación de Gozinto. Los sistemas ERP contemporáneos basan su funcionamiento en este esquema desde 1960, cuando (Kurbel, 2013). Como lo menciona Maheut (2013), la representación de Gozinto presenta limitaciones considerables en casos donde existen listas materiales inversas o la existencia de coproductos.

2.4.1. Listas de materiales alternativas

Un producto o ítem posee una lista de materiales alternativa cuando, con una combinación diferente de componentes o procesos, se puede obtener el mismo resultado. En este sentido, para una misma referencia de producto pueden existir varias listas de materiales que

pueden variar principalmente en la utilización de determinados componentes que, para efectos del proceso de fabricación, cumplen la misma función.

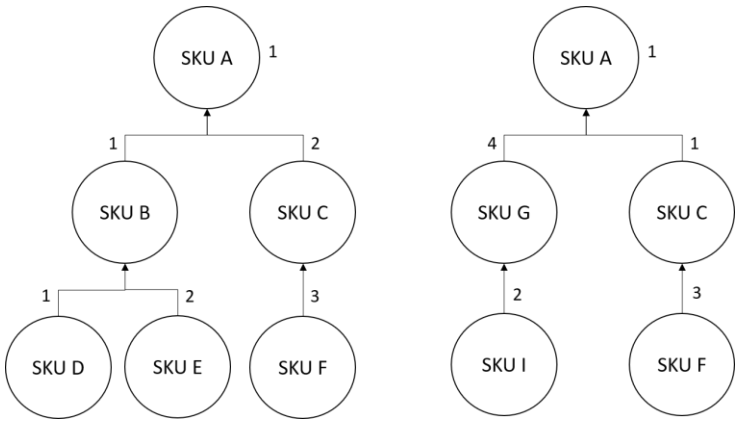


Figura 2. Listas de Materiales Alternativas para el SKU A.

2.5. Antecedentes y Clasificaciones del problema de lotificación

La nomenclatura y la clasificación de los problemas de lotificación presentes en la bibliografía científica son amplias y diversas. En la siguiente tabla se muestra un resumen de la notación relacionada con la clasificación y tipologías de los problemas de lotificación más estudiados.

Tabla 1

Criterio de Clasificación	Clasificación en la Bibliografía Científica
Según el número de niveles	Single-Level / Multi-Level Lot-sizing problems SLLP/MLSP
Según el número de productos finales	Single-Item / Multi-Item Lot-sizing problems SILSP-MILSP
Según las restricciones de capacidad	Un-Capacitated / Capacitated Lot-sizing problems UCLSP – CLSP
Según la demanda	Stationary /Dynamic Demand Deterministic/ Probabilistic Demand
Según los intervalos de planificación	Small Bucket / Big Bucket Lot-sizing Problems

Nota: Clasificaciones del Problema de Lotificación.

Una clasificación más detallada de la taxonomía de los problemas de lotificación puede encontrarse en el aporte de Kuik, Salomon, & van Wassenhove (1994). En su revisión se hace un recorrido por los aportes en la toma de decisiones de lotificación, partiendo de la formulación del modelo EOQ (*Economic Order Quantity*) de Harris (1913) hasta los modelos de lotificación multinivel con restricciones de capacidad.

Los modelos de lotificación multinivel se desarrollaron simultáneamente con los modelos mono producto y como lo menciona Billington, McClain, & Thomas (1986) hasta finales de los 70 y principios de los 80 muchos de los modelos de lotificación multinivel o multi-etapa no consideraban restricciones de capacidad (Afentakis, Gavish, & Karmarkar, 1984; Blackburn & Millen, 1982; Crowston & Wagner, 1973)

El Plan de Requerimiento de Materiales o "*Material Requirement Planning*" MRP, ha sido la principal herramienta para la definición de los tamaños de lote desde que fue introducida por Orlicky (1975). Es una técnica de planificación a mediano y corto plazo que permite el cálculo de los requerimientos de material y la programación de las órdenes para cada uno de los componentes de la lista de materiales "*Bill of Materials*" de un producto en sus diferentes niveles.

El MRP se alimenta a su vez de una planificación más general: el plan maestro de producción o "*Master Production Schedule*" MPS. El MPS consiste en un método de planeación que define las cantidades demandadas o requeridas de un producto o familia de productos finales (demanda independiente) durante un determinado periodo de planeación (Clark & Armentano, 1995). Con el fin de cumplir y satisfacer los requerimientos de la demanda provenientes del MPS, el MRP tiene en cuenta los tiempos de suministro "lead

times", los tiempos de ciclo de producción y los niveles de inventario en cada periodo (Clark & Armentano, 1995).

Steinberg & Napier (1980) presentaron los primeros avances para el problema capacitado de lotificación multinivel (MLCLSP - *Multilevel capacited Lot-Sizing Problem*) modelando un sistema como una red restringida generalizada con arcos de costos fijos en un entorno multiperiodo y multiproducto.

Luego Billington, McClain, & Thomas (1983) presentan un modelo exacto para la minimización de los costos de alistamiento e inventario en problemas multinivel con uno o varios recursos con capacidad finita. Su aporte es considerado como uno de lo más determinantes para la solución del problema capacitado de lotificación multinivel. (Kuik et al., 1994).

Blackburn & Millen (1984) exponen un estudio exploratorio en el cual utilizan y comparan las heurísticas de lotificación como EOQ, POQ, PPB, Wagner Within y Silver Meal en la resolución del problema capacitado de lotificación multinivel, destacándose buenos resultados en la utilización las heurísticas de Wagner Within (modificado) y Silver Meal (modificado). El uso de meta heurísticas tomó fuerza en los años 90 y Kimms (1999) propone el uso de algoritmos genéticos para la solución del problema y compara sus resultados con los obtenidos mediante la aplicación de algoritmos de búsqueda tabú en años anteriores (Kimms, 1996).

Dentro de los antecedentes más recientes se encuentra la propuesta de (Garcia-Sabater et al., 2013; Maheut & Garcia-Sabater, 2011; Maheut, Garcia-Sabater, Garcia-Sabater, & Valero Herrero, 2011) con el modelo de Planificación de Materiales y Operaciones Genéricas (GMOP), el cual será descrito en la siguiente sección.

2.6. El modelo de Planificación de Materiales y Operaciones Genéricas (*GMOP*)

El modelo GMOP (*Generic Materials and Operations Planning*) fue propuesto por (Coronado Hernández, 2015; Garcia-Sabater et al., 2013; Maheut & Garcia-Sabater, 2011) proponen el modelo GMOP (*Generic Materials and Operations Planning*), como una alternativa para el modelamiento de las relaciones existentes entre los procesos y los materiales necesarios para la elaboración de un producto. A diferencia de la representación de Gozinto, donde la prioridad recae sobre el producto final y sus componentes, este enfoque se centra en la planeación de las operaciones (*Strokes*) a realizarse para la manufactura, compra o el transporte de un determinado producto o conjunto de éstos. El problema GMOP es modelado como un problema de programación entera mixta (MIP). La tabla 2 muestra la notación de índices, parámetros y variables del modelo.

Tabla 2.

Índices	
i	Índice conjunto de productos
t	Índice de periodos de planificación
r	Índice de conjunto de recursos
k	Índice de conjunto de *Strokes*
Parámetros	
$D_{i,t}$	Demanda del producto i para el periodo t
$h_{i,t}$	Costo de almacenar una unidad del producto i en el periodo t
$CO_{k,t}$	Costo del *Stroke* k en el periodo t
$CS_{k,t}$	Costo de alistamiento del *Stroke* k en el periodo t
$CB_{i,t}$	Costo de comprar el producto i en el periodo t
$SO_{i,k}$	Número de unidades del producto i que genera el *Stroke* k
$SI_{i,k}$	Número de unidades del producto i que consume el *Stroke* k
LT_k	Lead time del *Stroke* k
$KAP_{r,t}$	Capacidad disponible del recurso r en el periodo t (en unidades de tiempo)
M	Una cantidad lo suficientemente grande
$TO_{k,r}$	Capacidad del recurso r requerida para fabricar una unidad para el *Stroke* k (en unidades de tiempo)
$TS_{k,r}$	Capacidad requerida para el alistamiento del recurso r para el *Stroke* k (en unidades de tiempo)
Variables	
$Z_{k,t}$	Cantidad de *Strokes* k a ser desarrollados en el periodo t
$\delta_{k,t}$	= 1 si el *Stroke* k es desarrollado en el periodo t (0 en caso contrario)

$W_{i,t}$	Cantidad comprada del producto i en el periodo t
$X_{i,t}$	Nivel de inventario disponible del producto i al final del periodo t

Nota: Notación de índices, parámetros y variables del modelo GMOP

$$Z = min \sum_t \sum_i \left(h_{i,t} * X_{i,t}\right) + \sum_t \sum_k \left(CS_{k,t} * \delta_{k,t} + CO_{k,t} * Z_{k,t}\right) + \sum_t \sum_i \left(CB_{i,t} * w_{i,t}\right) \quad (1)$$

Sujeto a:

$$X_{i,t} = X_{i,t-1} - D_{i,t} + w_{i,t} - \sum_k \left(SI_{i,k} * z_{k,t}\right) + \sum_k \left(SO_{i,k} * z_{k,t-LT_k}\right), \quad \forall\, i,t \quad (2)$$

$$z_{k,t} - M * \delta_{k,t} \leq 0, \qquad \forall k,t \quad (3)$$

$$\sum_k \left(TS_{k,r} * \delta_{k,t}\right) + \sum_k \left(TO_{k,r} * z_{k,t}\right) \leq KAP_{r,t} \quad \forall\, r,t \quad (4)$$

$$x_{i,t} \geq 0;\ w_{i,t} \geq 0 \quad \forall\, i,t \quad (5)$$

$$z_{k,t} \in \mathbb{Z}^+;\ \delta_{k,t} \in \{0,1\} \quad (6)$$

La ecuación (1) muestra la función objetivo del modelo, la cual consiste en la minimización de la totalidad de los costos de alistamiento, costos de operación y costos de almacenamiento de los *strokes* durante el horizonte de planificación.

La ecuación (2) representa la restricción de balance de inventario para cada periodo de planificación. Se tiene en cuenta el inventario del periodo inmediatamente anterior para cada SKU, la demanda independiente de los productos finales, las unidades o partes compradas, las unidades consumidas por los *strokes* durante el periodo y las unidades producidas por la realización de los *strokes.*

La ecuación (3) muestra la restricción que permite al modelo penalizar los alistamientos de los *Strokes*. Cada *Stroke* tiene un determinado Costos de Alistamiento. En este caso no se tendrá en cuenta alistamientos dependientes de la secuencia de operaciones.

La ecuación (4) representa la restricción de capacidad para los recursos disponibles. Se tiene en cuenta que los requerimientos de los alistamientos y la operación de los *Strokes* no sobrepasen la capacidad de los recursos en cada periodo de planificación.

Las ecuaciones (5) y (6) determinan el dominio de las variables de decisión presentes en el modelo.

2.6.1. El concepto de Stroke

> "*Un stroke representa cualquier operación básica (en su sentido más genérico), tarea o actividad que pueda transformar (o transportar) un conjunto de productos (medido preferentemente como SKU) en otro conjunto de productos (también medido preferentemente en SKU) y/o que consuma (o inmoviliza) recursos*" (Maheut et al., 2011)

La información de costos y uso de recursos se asigna a cada *stroke*, y no al producto. Un puede tener asignados los siguientes atributos:

1. Entradas (*Stroke Inputs*): El producto o conjunto de productos que se consumen para la realización del stroke.
2. Salidas (*Stroke Outputs*): El producto o conjunto de productos que se obtienen al ejecutar el stroke.
3. Tiempos de entrega (*Lead Times*): es el plazo de entrega asociado al stroke.
4. Tiempos y costos de Operación (*Processing Times*)
5. Tiempos y costos de Alistamiento (*Setup Times*)
6. Recursos Asignados (*Resources*): Los recursos pueden ser maquinaria, mano de obra etc.
7. Localización (*Location*)

El modelo es capaz de incluir de manera sencilla restricciones de capacidad, listas de materiales (Bill of materiales BOM) directas o inversas, listas de materiales alternativas, producción en diferentes sitios, recursos utilizados, coproductos, configuraciones de transporte y procesos de empaque (Garcia-Sabater et al., 2013).

Un ejemplo de la representación de una lista de materiales utilizando el concepto de *Stroke* se muestra en la Figura 3:

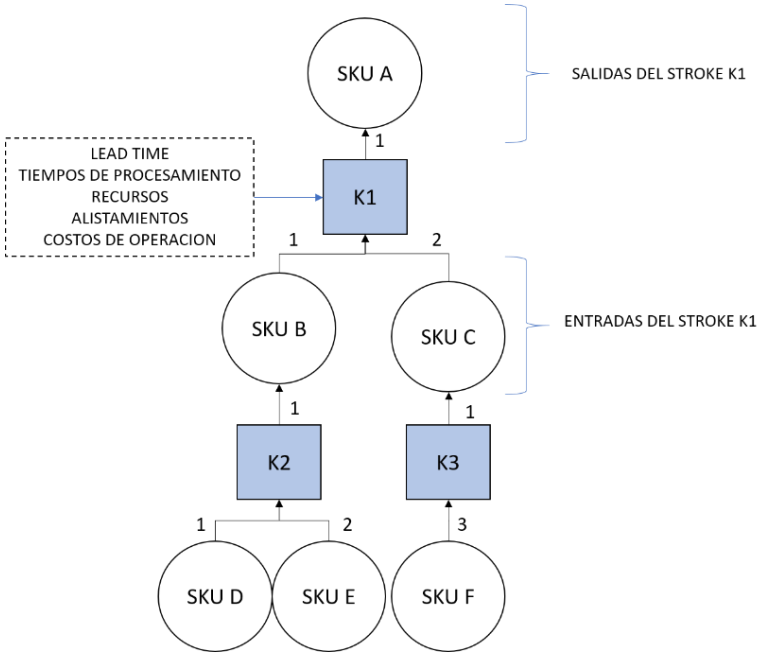
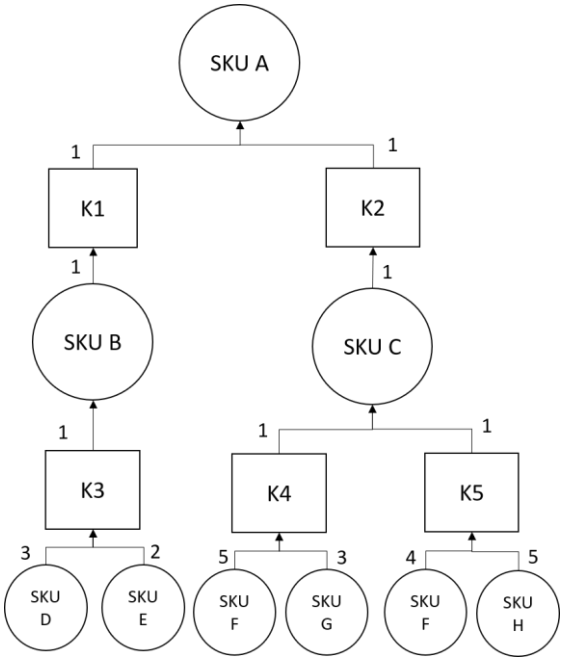


Figura 3. Representación de los Strokes en el problema GMOP.

Según Maheut & Garcia-Sabater (2011), algunos supuestos adicionales del modelo GMOP son :

a. El consumo en recursos en un *Stroke* debe ser inferior a un periodo de planificación y se limitará al primer periodo de la ejecución del *Stroke*.
b. Un recurso no podrá cambiar de localización.
c. No se asumirán *Stroke* parciales.

El modelo GMOP utiliza un conjunto de matrices que cumple las funciones de una lista de materiales y que permiten describir la relación entre las diferentes partes componentes de los productos finales. En primer lugar, está la matriz de entradas $SI(i,k)$ la cual determinará el número de unidades del $SKU\ i$ que consume el $Stroke\ k$. Luego está la matriz de salidas $SO(i,k)$, la cual determinará el número de unidades del $SKU\ i$ que genera el $Stroke\ k$. La figura 4 muestra la representación gráfica de la lista de materiales para el SKU A y su respectiva representación mediante las matrices de entrada y salida (Garcia-Sabater et al., 2013).



$SI_{i,k}$	K1	K2	K3	K4	K5
SKU A	0	0	0	0	0
SKU B	1	0	0	0	0
SKU C	0	1	0	0	0
SKU D	0	0	3	0	0
SKU E	0	0	2	0	0
SKU F	0	0	0	5	4
$SO_{i,k}$	K1	K2	K3	K4	K5
SKU A	1	1	0	0	0
SKU B	0	0	1	0	0
SKU C	0	0	0	1	1
SKU D	0	0	0	0	0
SKU E	0	0	0	0	0
SKU F	0	0	0	0	0
SKU G	0	0	0	0	0
SKU H	0	0	0	0	0

Figura 4. La representación gráfica (izquierda) y tabular (derecha) de una lista de materiales. Basado en *(Garcia-Sabater et al., 2013)*

Con el modelo GMOP se propone la creación de un *Stroke* diferente para cada proceso alternativo. Esta opción puede suponer un aumento considerable en el número de *Strokes* y por ende en la complejidad del modelo.

Como principal antecedente para la presente investigación está el aporte de (Roca Molina, 2016), en el cual se aplica un algoritmo de relajación lagrangeana a la solución del

problema de lotificación en sistemas multinivel, restricciones de capacidad, listas de materiales alternativas y entornos de coproducción, utilizando la estructura de producto GMOP de Garcia-Sabater et al., (2013). Los resultados obtenidos muestran respuestas muy cercanas al óptimo y un ahorro en los tiempos de solución de hasta un 96%.

2.7. Conclusiones

El modelo GMOP representa una alternativa robusta para la solución del problema de lotificación multinivel con listas de materiales alternativas y coproducción. El concepto de Strokes facilita el uso de listas de materiales alternativas y el cálculo de los costos asociados a la planificación. Maheut et al., (2011) afirman que la construcción de la matriz de recursos y operaciones para el problema GMOP es sencilla, pero, a medida que aumenta el número de operaciones, recursos y localizaciones, el tamaño de las matrices aumenta de manera exponencial y se incrementan en gran medida los tiempos de carga y solución del modelo.

Es necesario tener en cuenta que, las consideraciones de los autores del modelo están basadas en un entorno de solución mediante un método exacto. Al momento de implementar un método de solución heurístico, es necesario revalidar el funcionamiento de las matrices de entradas y salidas de los *Strokes* para así definir un procedimiento iterativo que permita el cálculo del número de Strokes necesarios y los niveles de inventario en cada periodo de planificación.

Capítulo 3. Búsqueda Tabú aplicada a la solución de problemas de planificación de la producción.

3.1. Introducción

Los modelos heurísticos y metaheurísticos se han convertido en alternativas para la solución de innumerables problemas de planificación de la producción debido a su eficiencia en tiempos de solución. Los algoritmos de búsqueda local o de vecindario son ampliamente utilizados en diversos problemas de optimización, pero una de sus principales desventajas es su tendencia a converger en soluciones óptimas locales.

Es por esto por lo que se implementaron métodos basados en los algoritmos básicos de búsqueda local, los cuales involucran mecanismos de aprendizaje y el uso de un registro de memoria a corto, mediano y largo plazo para optimizar el proceso de exploración de vecindarios de solución complejos. El presente capitulo expone las generalidades de los algoritmos de búsqueda tabú y se muestra un conjunto de antecedentes del uso de este tipo de algoritmos en problemas de planificación de la producción.

3.2. Búsqueda Tabú (*Tabu Search*)

El término, enfoque y metodología fueron presentados por Glover (1989, 1990).

> **"***La Búsqueda Tabú (Tabu Search - TS) es un procedimiento metaheurístico cuya característica distintiva es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas. Su filosofía se basa en la explotación de diversas estrategias inteligentes para la resolución de problemas, basadas en procedimientos de aprendizaje. El marco de memoria adaptativa de TS explota la*

> *historia del proceso de resolución del problema haciendo referencia a cuatro dimensiones principales, consistentes en la propiedad de ser reciente, en frecuencia, en calidad, y en influencia*." (Batista & Glover, 2003)

Una de las ventajas del uso de algoritmos TS es su capacidad para encontrar soluciones de mejor calidad, evitando estancarse en óptimos locales, como puede suceder con algoritmos de búsqueda local. Su funcionamiento se basa en la creación de entornos de búsqueda restringidos y el uso de mecanismos de aprendizaje (memoria a corto y largo plazo).

3.2.1. La Lista Tabú

La Lista Tabú representa el principal mecanismo de memoria a corto plazo y contiene un registro de nodos o atributos de las soluciones ya exploradas y seleccionadas como mejores dentro de un vecindario evaluado. Esto evita que el algoritmo las considere o evalúe nuevamente durante un determinado número de iteraciones (*tenure*)**,** favoreciendo la búsqueda en regiones aún no exploradas del espacio de solución.

En términos operativos, almacenar todos y cada uno de los nodos visitados representará un gasto computacional importante. Es por lo que se define un tamaño de lista, el cual puede ser constante o dinámico.

Un tamaño de lista reducido puede ocasionar que el algoritmo se comporte de forma parecida un algoritmo de búsqueda local o *hill climbing,* susceptible a quedar atrapado en comportamientos cíclicos cercanos a valores óptimos locales.

Por otro lado, un tamaño de lista elevado puede provocar que se prohíban un gran número de soluciones de calidad en el espacio de solución, además de representar un aumento considerable en el uso de memoria computacional.

3.2.2. Movimientos

A aquellos cambios en el vector o la matriz de solución que originen soluciones factibles candidatas, se les denominará Movimientos. Las modificaciones al vector de solución pueden realizarse de diversas maneras, una de las más comunes es la utilización de mecanismos de inserción, cruce, mutación o combinación, parecidos a los utilizados en algoritmos evolutivos.

Realizar movimientos en el vector solución permite la generación del vecindario de soluciones que se evalúa durante cada iteración. Se seleccionará la mejor solución dentro del conjunto de soluciones candidatas y el movimiento realizado será registrado en la lista tabú.

Si en un determinado movimiento intervienen una o más variables marcadas dentro de la lista tabú, el movimiento se considera como Movimiento Tabú y no se podrá realizar hasta tanto las variables involucradas salgan de la lista (al finalizar el conteo de su *Tabu Tenure*) o entre en funcionamiento un mecanismo de aspiración.

3.2.3. Tenencia Tabú (*Tabu Tenure)*

El valor de este parámetro representa el número de iteraciones que un movimiento permanece dentro de la lista tabú desde que es realizado. Valores altos en este parámetro ocasionarían, al igual que un tamaño elevado de lista tabú, que el algoritmo prohíba ciertas soluciones que puedan guiar hacia la exploración más amplia del vecindario de solución.

Si el *Tabu Tenure* es bajo, el algoritmo puede fácilmente reevaluar soluciones ya consideradas en iteraciones pasadas recientes.

3.2.4. Solución Inicial

La solución inicial debe ser factible y puede ser obtenida de diversas maneras. A partir de ésta se generan conjuntos de soluciones vecinas. Estas soluciones vecinas se evalúan al mismo tiempo que se actualiza la lista tabú. Es posible utilizar soluciones factibles obtenidas por otros métodos o heurísticas. La calidad de estas soluciones puede afectar la exploración del vecindario de soluciones y por ende la eficiencia del algoritmo de búsqueda tabú.

3.2.5. Criterios de Parada

El algoritmo de búsqueda tabú se caracteriza por funcionar bajo cierto número de iteraciones. Dado que muchos de los movimientos involucran procesos de selección aleatorios, el número de iteraciones es importante al momento de configurar un algoritmo de búsqueda tabú. Los criterios de parada definen el momento en el cual el algoritmo se detiene y muestra los resultados obtenidos. Algunos de los criterios de parada pueden ser:

1. Número máximo de iteraciones.
2. Hasta que se alcance un valor mínimo o límite de la función objetivo
3. Número de iteraciones sin mejora o sin mejora sustancial.

3.3.6. Mecanismos de Aspiración

Una de las principales ventajas de los algoritmos de búsqueda tabú es la utilización de mecanismos de aspiración. Si un movimiento Tabú lleva a una solución con mejor valor objetivo puede considerarse la utilización de un método de aspiración, con el fin de aceptar de manera excepcional la realización del movimiento (Batista & Glover, 2003).

3.2.7. Mecanismos de Intensificación y Diversificación.

En muchas ocasiones, el uso de la memoria a corto plazo (lista tabú) permite la obtención de soluciones de buena calidad, sin embargo, para aplicaciones con entornos de

búsqueda complejos es posible utilizar mecanismos de memoria a largo plazo (métodos de intensificación y diversificación). Estos permiten mejorar la exploración del conjunto de soluciones en un determinado vecindario y propiciar evaluación en entornos poco explorados. (Chelouah & Siarry, 2000)

Los mecanismos de intensificación permiten visitar zonas del espacio de solución prometedoras que fueron exploradas parcialmente mediante el mecanismo a corto plazo. Por otro lado, los mecanismos de diversificación permiten la visita de espacios de solución que no han sido explorados.

3.2.8. Estructura Básica de un Algoritmo de Búsqueda Tabú

La estructura básica de un algoritmo de búsqueda tabú se muestra a continuación:

Algoritmo Básico de Búsqueda Tabú

Paso 1. Inicialización
Definir $Tmax$:= Número Máximo de iteraciones
Inicializar Contador de iteraciones $t = 0$;
Generar una solución inicial S_0 factible, con valor objetivo Z_0
Definir la mejor solución disponible; $BestSol = Z_0$
Inicializar la lista tabú con tamaño n; $TabuList = Empty$
Definir $TabuTenure$
Definir movimientos posibles

Paso 2. Movimientos
Generar el conjunto de soluciones vecinas V candidatas.
Si el movimiento de $S_k \ a \ S_c$ pertenece a $TabuList$, hacer S_{k+1}
Sino, agregar S_c al conjunto V.

Paso 3. Evaluación
Evaluar las soluciones vecinas del conjunto V y seleccionar la mejor $BestSolV$
Si $BestSolV$ mejora a $BestSol$,
Hacer $BestSol = BestSolV$

Paso 4. Actualización
Actualizar solución actual y Z actual
Si $TabuList$ está completa,
Eliminar el movimiento que lleve a la "peor" solución
Añadir movimiento $S_k \ a \ S_c$ a $Tabulist$

Paso 5. Iteración

Si $t < Tmax$ Ir a Paso 2.
Sino Detener
Actualizar $TabuList$
Mostrar $BestSol$
Mostrar la solución S_c que optimiza Z.

3.3. Uso de Búsqueda Tabú en la planificación de la producción

3.3.1. Búsqueda Tabú en modelos de lotificación de un solo nivel

El uso de la búsqueda tabú en la solución de problemas de lotificación ha sido ampliamente estudiado en la literatura. A continuación, se describen algunos de los principales aportes para la solución del problema de lotificación en sistemas de planificación de un solo nivel.

Utilizando un algoritmo de búsqueda tabú, Hindi (1995) soluciona el problema de lotificación capacitado de un solo ítem con demanda dinámica teniendo en cuenta costos de inicio y costos de utilización. Más adelante, Hindi (1996) aplicó el procedimiento de búsqueda tabú en la solución de un problema capacitado de un solo nivel, demanda dinámica y múltiples ítems CLSP. Se modeló el problema mediante programación entera mixta MIP y se aplicó un método de relajación mediante generación de columnas. El resultado fue una solución factible que luego fue optimizada mediante un modelo de *minimum-cost network flow*. La solución optimizada fue el punto de partida para el algoritmo de búsqueda tabú con el mismo número de movimientos que el modelo de mínimo costo.

Años después, Gopalakrishnan, Ding, Bourjolly, & Mohan (2001) muestran la aplicación de la búsqueda tabú en un problema de lotificación multi ítem con tiempos de alistamiento (Setup Carry-over). La búsqueda tabú consistió en cinco movimientos básicos,

para secuenciación y la definición del tamaño del lote. A su vez se utilizaron listas tabú dinámicas, memoria adaptativa y penalidades autoajustadas para fortalecer la búsqueda.

Y. F. Hung, Chen, Shih, & Hung (2003) utilizan un algoritmo de búsqueda tabú para la solución de problemas de planificación de la producción con costos de alistamiento y múltiples ítems. Basados en la solución del problema mediante método simplex, los autores proponen un método mejorado para la determinación de listas de candidatos para la exploración de vecindarios de solución y la obtención de soluciones con menor gasto computacional.

Karimi et al., (2006) exponen la utilización de la búsqueda tabú para la solución del problema identificado por ellos como CLSP+: un problema de lotificación multi-ítem, con un solo nivel, restricciones de capacidad y demanda dinámica, que tenía como consideración especial la presencia de pedidos atrasados (*Backlogging*) y similitudes en los requerimientos de alistamiento (*Setup Carry-Over*). En este caso, el problema es abordado de manera similar a K S Hindi (1996), y está formulado como un modelo de programación entera mixta MIP y se obtienen soluciones iniciales factibles por medio de algoritmos heurísticos que tienen en cuenta los cambios en la demanda, las reglas de lotificación, las condiciones de factibilidad y la determinación de los ítems cuyo alistamiento se debe "arrastrar" (*carry over*).

Luego la solución inicial es mejorada aplicando un modelo de *minimum-cost network flow*. Esta solución es utilizada como inicio del procedimiento de búsqueda tabú, configurado con el número de movimientos arrojados por el modelo de mínimo costo, logrando tiempo soluciones cercanas a valores óptimos (gaps cercanos al 1,3% para ciertos casos).

Li, Baki, Tian, & Chaouch (2014a) presentan un algoritmo de búsqueda tabú para el problema dinámico de lotificación con devoluciones de producto y re-manufactura (DLRR).

Utilizando un algoritmo de cadena de bloques (*block-chains*) se generan las soluciones iniciales para el procedimiento de búsqueda tabú. El algoritmo arrojó resultados satisfactorios en el 96% de los problemas de prueba y una baja desviación estándar comparado con soluciones previas del problema estudiado.

En el caso de los modelos de *scheduling*, uno de los principales objetivos es la disminución del *makespan* o tiempo de ciclo de la programación. Existen infinidad de aplicaciones de búsqueda tabú (Oliva San Martín & Ramírez Guzmán, 2015), algunos son listados a continuación:

Dell'Amico & Trubian (1993) aplicaron búsqueda tabú en la solución de un problema de scheduling tipo Job Shop, demostrando mejorar resultados con respecto a técnicas heurísticas de mejoramiento iterativo previamente estudiadas. Otro ejemplo es el aporte de Nowicki & Smutnicki (1998), en cual dan solución a un problema del tipo *Flow Shop* en máquinas paralelas.

De manera detallada, Cesaret, Oğuz, & Sibel Salman (2012) exponen un procedimiento de búsqueda tabú soportado por un búsqueda local probabilística, para un problema de aceptación de órdenes y scheduling en un sistema de una solo máquina con tiempos de alistamiento. Los resultados obtenidos demuestran las ventajas de la búsqueda tabú frente a dos técnicas heurísticas previamente estudiadas para ese mismo problema (m-ATCS e ISFAN).

3.3.2. Búsqueda Tabú en modelos de lotificación multinivel

Aunque la complejidad de los modelos de lotificación multinivel es mayor, los algoritmos de búsqueda tabú arrojaron resultados satisfactorios en la solución de este tipo de problemas:

Kuik, Salomon, Van Wassenhove, & Maes (1993) presentan la aplicación de las metaheurísticas de recocido simulado y búsqueda tabú en la solución del problema de lotificación multinivel para sistemas de ensamble con cuello de botella *(bottleneck*). El problema es modelado mediante programación lineal LP y los resultados son comparados con los obtenidos con las dos heurísticas, demostrando ser superiores en cuanto a calidad y uso computacional. Se propone entonces la posibilidad de combinar estas técnicas con LP con el fin de optimizar su funcionamiento.

Kimms (1996) propuso dos heurísticas para la solución del problema de lotificación y scheduling en sistemas multinivel con una sola máquina, múltiples ítems y restricciones de capacidad: la primera heurística está basada en la heurística de "*randomized regrets*" y la segunda se basa en un procedimiento de búsqueda tabú. Para esta última se tiene en cuenta una estructura de producto representada mediante Gozinto, de la cual se define una estructura de arcos disyuntivos para dar solución al problema. Los resultados entre las dos heurísticas propuestas fueron cercanos al óptimo y muy similares. Se propuso como trabajo futuro la inclusión de restricciones como múltiples recursos, tiempos de alistamiento y órdenes atrasadas.

Y.-F. Hung & Chien (2000) a través de un modelo de programación entera mixta, aplicaron la búsqueda tabú, el recocido simulado y algoritmos genéticos al problema de lotificación multinivel con varias clases de demanda. Los resultados obtenidos permitieron afirmar que los algoritmos de búsqueda tabú tuvieron un mejor desempeño en problemas con demanda conformada por órdenes confirmadas, es decir, demanda determinística.

En el año 2005, Berretta, França, & Armentano proponen el uso de meta heurísticas para el problema de lotificación multinivel con restricciones de capacidad, con estructuras generales de productos, tiempos y costos de alistamiento y tiempos de entrega. Se utilizó búsqueda tabú

y recocido simulado para la solución del problema, cuyos resultados fueron comparados posteriormente con los obtenidos mediante un método de relajación lagrangeana. Las metaheurísticas arrojaron soluciones muy cercanas al óptimo para problemas pequeños.

Las tablas 3 y 4 resumen la configuración básica de los algoritmos de búsqueda tabú utilizados en los artículos consultados, para sistemas de producción de un solo nivel y de múltiples niveles respectivamente.

Tabla 3

Referencia	Solución de Referencia	Solución Inicial	Movimientos	Métodos de Intensificación	Métodos de Diversificación	Método de Aspiración
(Hindi, 1995)	Algoritmo Simplex Dual	Solución al problema de transbordo	Vecinos con mejor solución al problema de transbordo	Mostraron no ser necesarios	Mostraron no ser necesarios	Aspiración por mejora en función objetivo
(Hindi, 1996)	Programación entera mixta	Método de Generación de Columnas	Lista de candidatos con menores costos de Setup	No se especifica	Problema de transbordo sin restricciones de capacidad	Aspiración por mejora en función objetivo
(Gopalakrishnan et al., 2001)	TTM	Lote por lote y Wagner-Within	Movimiento de Setups, Movimiento de Items y Movimiento de lotes	Memoria Adaptativa	Memoria Adaptativa	No se especifica
(Karimi et al., 2006)	Optimal enumeration method	Shorstest Processing Time SPT	Insersión Aleatoria de trabajos	No se especifica	No se especifica	Aspiración por mejora en función objetivo
(Li et al., 2014a)	Silver-Meal heuristic	Método de cadena de bloques (Block Chain Method)	Movimiento de Setups (One-shift / Two-Shift moves)	No se especifica	Número de iteraciones sin mejora	Aspiración por mejora en función objetivo

Nota: Configuraciones básicas de la búsqueda tabú en modelos de lotificación de un solo nivel.

Tabla 4

Referencia	Solución de Referencia	Solución Inicial	Movimientos en la búsqueda	Método de Intensificación	Método de Diversificación	Método de Aspiración
(Kuik et al., 1993)	Programación Lineal LP	Algortimo Voraz modificado	Movimiento No tabú con mayor valor en Función Objetivo	No se especifica	No se especifica	No se especifica
(Kimms, 1996)	Programación entera mixta	Método de Arcos disyuntivos	Nodos adyacentes con mayor reducción en la función objetivo	Asignación de pesos a nodos adyacentes	No se especifica	Aspiración por mejora en función objetivo
(Berretta et al., 2005)	Límite Inferior Lagrangeano	Problemas generados aletoriamente	Movimiento No tabú con mayor valor en Función Objetivo	Matriz de Frecuencia de movimientos	Matriz de frecuencia de movimientos	Aspiración por mejora en función objetivo

Nota: Configuraciones básicas de la búsqueda tabú en modelos de lotificación multinivel.

Los métodos exactos de programación lineal y programación entera mixta son los más utilizados para comparar la eficiencia y la calidad (gap) de las soluciones arrojadas por los algoritmos de búsqueda tabú. Para problemas de gran tamaño, prevalece el uso de heurísticas y métodos de relajación matemática (Berretta et al., 2005).

Las soluciones utilizadas para inicializar la búsqueda tabú influyen en gran manera en el rendimiento de esta metaheurística. Las heurísticas de solución *Wagner-Within*, lote por lote, transbordo y SPT (en caso de problemas de lotificación con *scheduling*) son algunas de las alternativas más utilizadas. Los métodos de generación de columnas, cadena de bloques y el método de arcos disyuntivos constituyen también alternativas para su utilización en generación de soluciones iniciales para problemas de lotificación multinivel.

El máximo número de iteraciones juega también un papel determinante en el desempeño y el rendimiento computacional de un algoritmo de búsqueda tabú. Su definición en muchos casos es determinada a través de un diseño experimental, en otros casos se define a

consideración del programador y en muchos otros no se especifica cómo fue definido este parámetro.

En la mayoría de los aportes el tamaño de la lista tabú no se especifica. Generalmente se utiliza una lista tabú de tamaño finito, en la cual se permite un número máximo de movimientos tabú. Aquellos movimientos tabú más recientes reemplazan a los más antiguos.

El valor de tenencia Tabú o *Tabu Tenure* determina también el número de movimientos en la lista Tabú, esto sucede más que todo en aquellas listas con tamaño indefinido, en las cuales los movimientos tabú salen de la lista al cumplirse el tiempo de tenencia tabú. El valor de tenencia tabú generalmente depende el número de iteraciones que se establezcan, puede ser generado de manera aleatoria ((Li, Baki, Tian, & Chaouch, 2014b) o puede obtenerse mediante experimentación previa (Berretta et al., 2005).

Los criterios de finalización más utilizados en los aportes estudiados son el número máximo de iteraciones y el número máximo de iteraciones sin mejoras en la función objetivo. Estos criterios se utilizan principalmente para optimizar los tiempos de respuesta del algoritmo.

En muchas de las soluciones al problema de lotificación de un solo nivel no se utilizaron (o no se especificaron) métodos de intensificación; esto debido principalmente al tamaño y complejidad de este tipo de problemas. El uso de memoria adaptativa es uno de los métodos de intensificación utilizados. En el caso de los modelos de lotificación multinivel, los criterios de intensificación se rigen por asignación de pesos o penalizaciones a los movimientos durante la búsqueda.

La memoria adaptativa es utilizada como principal método de diversificación. La definición de nuevas soluciones iniciales después de un número máximo de iteraciones sin

mejora en la solución es una estrategia de diversificación utilizada en la literatura. Adicionalmente, la creación de matrices de frecuencia para los movimientos realizados permite una mayor exploración del espacio de soluciones en problemas de lotificación multinivel. En algunos casos no fue necesaria la utilización de métodos de diversificación o bien no se hizo mención en el artículo.

Para ambos tipos de problema, el método de aspiración más utilizado es la aspiración por objetivo, en la cual se exonera de la lista tabú el movimiento que represente una mejora en el atributo utilizado o un aumento en la función objetivo. En algunos casos no fue necesaria la aplicación de métodos de aspiración o bien no se mencionó dentro del artículo.

3.4.Conclusiones

Los algoritmos de búsqueda tabú han arrojado buenos resultados en la solución de problemas de problemas de planificación de la producción. Dado que estos algoritmos permiten una exploración eficiente de vecindarios de solución complejos, son una alternativa para la solución del problema capacitado de lotificación multinivel con listas de materiales alternativas y coproducción.

El algoritmo propuesto en el presente proyecto se formula como una aproximación a los algoritmos de búsqueda tabú, al utilizar únicamente el mecanismo de memoria a corto plazo (Lista Tabú).

Debido a que no se implementarán mecanismos de aspiración, intensificación o diversificación, el algoritmo se ha denominado como una propuesta heurística basada en listas tabú, y busca la optimizar la selección y cálculo de Strokes para cada uno de productos.

No se encontraron antecedentes de aplicación de heurísticas similares a la búsqueda tabú para el problema de lotificación modelado mediante el modelo GMOP lo cual permite evidenciar el aporte del presente proyecto.

En cuanto a la generación de soluciones iniciales, se seleccionó el método de lote por lote para calcular, a través de un proceso de planificación de requerimientos MRP, los niveles de inventario y los Strokes a desarrollarse en cada periodo.

Se trabajará con un tamaño de lista tabú definido, y los mecanismos de finalización incluirán un número máximo de iteraciones y un determinado número de iteraciones sin mejora en la solución.

Capítulo 4. Algoritmo heurístico basado en listas tabú para la solución del problema de lotificación en sistemas multinivel con listas de materiales alternativas y coproducción

4.1. Introducción

En este capítulo se presenta la propuesta heurística basada en listas tabú para la solución del problema de lotificación multinivel con listas de materiales alternativas y coproducción. El algoritmo se basa en el funcionamiento de un algoritmo básico de búsqueda tabú (Sección 3.2.8) mediante el uso del mecanismo de memoria a corto plazo. Se muestra la estructura en código del algoritmo, junto con las subrutinas de evaluación de soluciones y cálculo de Strokes.

4.2. Notación

La notación que se utilizará a lo largo del desarrollo del algoritmo se muestra a continuación:

Número de productos	i
Número de productos finales	i*
Número de periodos de planificación	t
Número de recursos	r
Número de operaciones o Strokes	k
D (i, t):	Demanda del producto i para el periodo t
h (i, t):	Costo de almacenar una unidad del producto i en el periodo t
CO (k, t):	Costo de operación del Stroke k en el periodo t
CS(k,t):	Costo de alistamiento del Stroke k en el periodo t
SO(i,k):	Número de unidades del producto i que genera el Stroke k
SI(i,k):	Número de unidades del producto i que consume el Stroke k
LT(k):	Lead time del stroke k

KAP(r,t):	Capacidad disponible del recurso r en el periodo t (en unidades de tiempo)
TO(k,r):	Capacidad del recurso r requerida para fabricar una unidad para el Stroke k (en unidades de tiempo)
TS(k,r):	Capacidad requerida para el alistamiento del recurso r para el Stroke k (en unidades de tiempo)
Zk(k,t):	Número de repeticiones de c ada Stroke en el periodo t.
δk(t):	1 si el Stroke k es desarrollado en el periodo t (0 en caso contrario)
Xi(i,t):	Inventario disponible del producto i al final del periodo t
It	Número de Iteraciones
NumVec	Número de soluciones vecinas por iteración
BestV	Mejor solución dentro del vecindario
BestSol	Mejor solución global
MaxAlt	Máximo número de Listas Alternativas
Ism	Número de Iteraciones sin mejora en la función objetivo

4.3. Arquitectura de la herramienta

La figura 5 contiene el flujograma que describe el funcionamiento y la estructura del algoritmo heurístico basado en listas Tabú y seguidamente se muestra el pseudocódigo del algoritmo.

El código programado en Matlab para de la herramienta propuesta puede ser consultado en el **Anexo 5.**

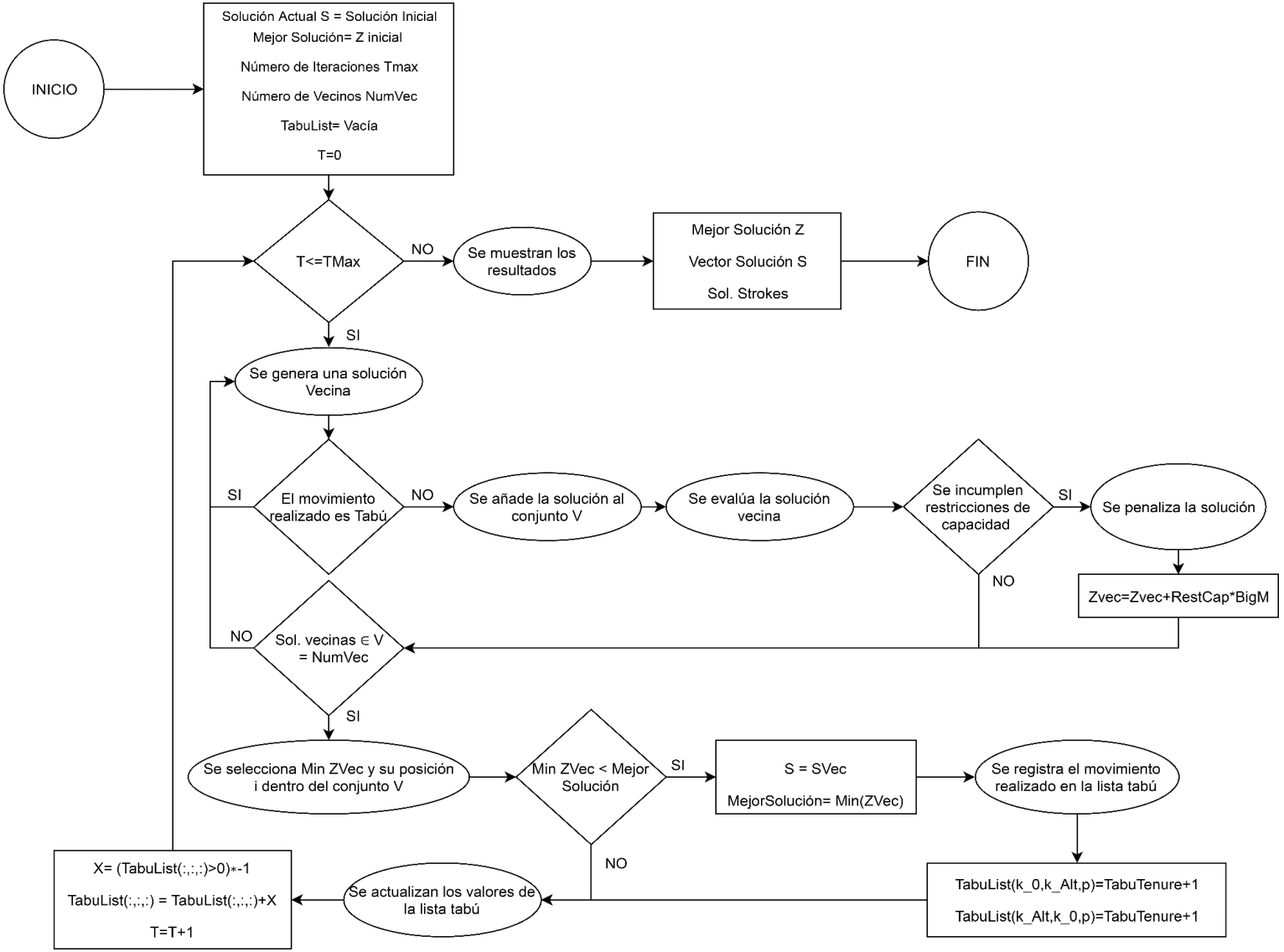


Figura 5. Flujograma del funcionamiento de la heurística basada en listas tabú.

Algoritmo 1. *Algoritmo heurístico basado en listas tabú.*

1 $Inicialización\ It, NumVec, TabuTenure.$
2 $Se\ genera\ una\ solución\ inicial\ S_0, con\ valor\ objetivo\ Z_0$
3 $S \leftarrow S_0$
4 $BestSol \leftarrow Z_0$
6 $While\ t <= It\ do$
Se seleccionan NumVec soluciones vecinas y se incluyen en el conjunto V:
7 $vec \leftarrow 1$
8 $While\ vec \leq NumVec$
Se selecciona un producto p aleatoriamente
9 $p = round(1 + (Card(i) - 1) * rand)$
Si el producto tiene por lo menos un Stroke alternativo
10 $If\ MaxAlt(p) > 1$
Se selecciona aleatoriamente un Stroke alternativo de $Alt(p,:)$
11 $kvec = round(1 + (MaxAlt(p) - 1) * rand)$
Se verifica si el movimiento no es tabú
12 If TabuList$\left(\left(\text{Alt(p, kvec)}\right), \left(\text{S(p)}\right), \text{p}\right) == 0$ AND TabuList$\left(\left(\left(\text{S(p)}\right), \text{p}\right), \text{Alt(p, kvec)}\right) == 0$
13 Se incluye la solución vecina en el conjunto V
14 $V(:, vec) = S$
15 $V(p, vec) = Alt(p, kvec)$
16 Se registran los detalles del movimiento en RegVec
17 Se evalúa la solución vecina AltVec
18 $EvaluacionSolucionVecina$
19 Si la solución incumple con restricciones de capacidad se aplica penalización
20 $ZVec = Zvec + RestCap * 100000$
21 Se registra la solución obtenida ZVec
22 $vec = vec + 1$
23 $End\ If$
24 $End\ if$
25 $End\ While$
Se selecciona la mejor solución BestV del conjunto V
26 $If\ BestV < BestSol$
Se actualiza el mejor valor objetivo y la mejor solución
27 $BestSol = BestV$
28 $S = V(:, vec)$;
29 Se actualiza la lista tabú descontando una iteración de los *tenure* registrados.
30 $X = (TabuList(:,:,:) > 0) * -1$
31 $TabuList(:,:,:) = TabuList(:,:,:) + X$
Se agrega el último movimiento realizado a la lista tabú y el movimiento inverso.
32 $TabuList(Alt(p, kvec), S(p), p) = TabuTenure$
33 $TabuList(S(p), p, Alt(p, kvec)) = TabuTenure$;
$t = t + 1$
34
36 $Else$
37 $X = (TabuList(:,:,:) > 0) * -1$
38 $TabuList(:,:,:) = TabuList(:,:,:) + X$
39 $t = t + 1$
40 $End\ If$
41 $End\ While$
42 Se muestran los resultados
43 $S, BestSol, BestSolStrokes$
44

4.3.1. Vector Solución y Vecindario de búsqueda.

Dada la presencia de listas alternativas de materiales, la definición del *Stroke* que fabrica cada uno de los SKU es una de las decisiones más importantes para la solución del problema de lotificación estudiado. Dado que cada *Stroke* posee una configuración diferente de entradas, salidas y uso de recursos, se hace necesario decidir cuál resulta más eficiente para la planificación y cuál permite a su vez la disminución del costo total de la planificación.

El vector solución del algoritmo propuesto contendrá la información del *Stroke* seleccionado para la fabricación de cada uno de los SKU (Figura 6).

$$S(i) = \begin{matrix} Bestk(SKU1) \\ Bestk(SKU2) \\ \vdots \\ Bestk(SKUi) \end{matrix}$$

Figura 6. Estructura del vector solución.

El vecindario de búsqueda está definido por el conjunto de soluciones resultantes de las posibles selecciones de *Strokes* para cada SKU, presentes en la lista de *Strokes* alternativos $Alt(i,k)$.

La matriz $Alt(i,k)$ proviene de la información registrada en la matriz de salidas $SO(i,k)$ y contiene un registro de los *Strokes* que pueden fabricar cada uno de los SKU. Esta matriz permite tener un rápido acceso a la información de las listas de materiales alternativas existentes. Si un SKU puede ser fabricado por más de 1 *Stroke*, se puede afirmar que ese SKU tiene más de una lista de materiales.

La figura 7 muestra un ejemplo de la estructura de la matriz $Alt(i,k)$. En el ejemplo, el SKU2 solo puede ser fabricado por el *Stroke* K2. En cambio, el SKU4 puede ser fabricado por los *Strokes* K1, K2 o K4.

$$\begin{array}{cccccc} SKU1 & k1 & k3 & 0 & 0 & \cdots \\ SKU2 & k2 & 0 & 0 & 0 & \cdots \\ SKU3 & k4 & k5 & 0 & 0 & \cdots \\ SKU4 & k1 & k2 & k4 & 0 & \cdots \\ SKU5 & k5 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ SKUi & kn & kn & kn & kn & kn \end{array}$$

Figura 7. Estructura de la matriz de Strokes alternativos.

4.3.2. Movimientos

En cada iteración, el vecindario V está conformado por un determinado número de soluciones vecinas: *NumVec*. Cada solución vecina se genera a partir de la realización de un cambio en alguna de las diferentes posiciones de vector solución S.

El principal movimiento del algoritmo propuesto consiste en la selección aleatoria de un producto o SKU y la asignación de uno de los *Strokes* alternativos posibles.

El movimiento es realizado únicamente para aquellos SKU que tengan por lo menos 1 *Stroke* alternativo y por cada nueva solución vecina generada se realiza un solo cambio en el vector solución. A partir de la línea 8 y hasta la línea 16 del Algoritmo1 se muestra el procedimiento para la creación del vecindario de soluciones en cada iteración.

4.3.3. La lista Tabú

La lista tabú utilizada consiste en una estructura de datos de 3 dimensiones, la cual lleva el registro de movimientos entre los *Strokes* alternativos de cada producto. Es una matriz cuadrada con k número de filas y k número de columnas. La tercera dimensión permite identificar el producto o SKU para el cual se realiza el movimiento, por lo cual existirán i listas diferentes de tamaño $k \; x \; k$. La figura 8 muestra de manera general la estructura de la lista tabú.

$$TabuList(k,k,i) = \begin{array}{c|cccc} & K1 & K2 & \cdots & k \\ K1 & M & 0 & \cdots & 0 \\ K2 & 0 & M & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ k & 0 & 0 & 0 & M \end{array}$$

Figura 8. Estructura de la Lista Tabú.

La matriz tiene asignado por defecto un valor M en las posiciones de la diagonal principal, este es un valor muy grande que impide que el algoritmo realice movimientos entre el mismo *Stroke.* Cada vez que un movimiento ingresa a la lista tabú, se le asigna el valor del *Tabu Tenure* y se penaliza de igual manera el movimiento que lo revierte.

Un ejemplo de la estructura de la lista tabú para el caso de 5 *Strokes* y 2 productos, se presenta en la figura 9:

$$TabuList(5,5,1) = \begin{array}{c|ccccc} & K1 & K2 & K3 & K4 & K5 \\ K1 & M & 0 & 0 & 0 & 3 \\ K2 & 0 & M & 0 & 0 & 0 \\ K3 & 0 & 0 & M & 0 & 0 \\ K4 & 0 & 0 & 0 & M & 0 \\ K5 & 3 & 0 & 0 & 0 & M \end{array}$$

$$TabuList(5,5,2) = \begin{array}{c|ccccc} & K1 & K2 & K3 & K4 & K5 \\ K1 & M & 0 & 0 & 0 & 0 \\ K2 & 0 & M & 4 & 0 & 0 \\ K3 & 0 & 4 & M & 0 & 0 \\ K4 & 0 & 0 & 0 & M & 5 \\ K5 & 0 & 0 & 0 & 5 & M \end{array}$$

Figura 9. Ejemplo de una lista tabú para el caso de 5 strokes y 2 productos.

En el ejemplo anterior, para el producto 1, el movimiento de cambio del *stroke* K1 al *stroke* K5 se encuentra marcado como movimiento tabú durante tres iteraciones. De igual manera la lista prohíbe el que el algoritmo evalúe el movimiento que revierte el último movimiento realizado, es decir el cambio del *stroke* K5 al *stroke* K1.

La lista Tabú podrá contener un máximo de movimientos tabú equivalentes al doble del *Tabu Tenure* seleccionado. Por cada movimiento adoptado e ingresado en la lista tabú se añade también el movimiento contrario a éste y estos permanecerán en la lista por un máximo de iteraciones definidas por el *Tabu Tenure*. Los movimientos más antiguos saldrán primero y los movimientos más recientes saldrán después.

4.3.4. Criterios de finalización

El criterio para la finalización del algoritmo es el cumplimiento de un determinado número de iteraciones It.

4.3.5. Evaluación de las soluciones vecinas

El cálculo del costo total para cada una de las soluciones vecinas se realiza a través de la subrutina *EvaluacionSolucionVecina* (Línea 18, Algoritmo1). Esta subrutina basa su funcionamiento en el modelo de lotificación de lote por lote, utilizado también para la obtención de soluciones iniciales (Capitulo 5). El algoritmo de esta subrutina puede ser consultado en el Anexo 6.

A través de la realización de un MRP para cada producto final y componente, se obtienen la cantidad de Strokes requeridos en cada uno de los periodos de planificación y de acuerdo con esto se calculan los costos de inventario, costos de alistamiento y costos de operación. Por tratarse de un problema de lotificación multinivel, el cálculo de requerimientos se realiza en 3 niveles diferentes. La estructura de la subrutina se muestra a continuación:

Algoritmo 2. Subrutina EvaluacionSolucionVecina

Definir Requerimientos en Conjunto para cada producto final $RC_{i,t} = d_{i,t}$
$For\ t = 1\ to\ Card(t)$
Se realiza el MRP para cada producto final p
For p=1 to Card (p)
$s = AltVec(p)$
Se calculan los Requerimientos en Conjunto para las entradas y del Stroke que fabrica el producto final p
$RC(y, t - lt) = ceil(RC(y, t - lt) + ((ceil(RPLAN(p, t)/strokeOut(p, s)\ )) * strokeIn(y, s)))$;
Se realiza el MRP a las entradas y del Stroke s que fabrica el producto p
$End\ For$
$End\ For$

$For\ t = 1\ to\ Card(t)$
Se realiza el MRP para cada producto final p
For p=1 to Card (p)
$s2 = AltVec(y)$
Se calculan los Requerimientos en Conjunto para las entradas $y2$ del Stroke que fabrica y
$RC(y2, t - lt) = ceil(RC(y2, t - lt) + ((ceil(RPLAN(y, t)/strokeOut(y, s2))) * strokeIn(y2, s2)))$
Se realiza el MRP a las entradas $y2$ del Stroke $s2$ que fabrica el producto y
$End\ For$
$End\ For$

$For\ t = 1\ to\ Card(t)$
Se realiza el MRP para cada producto final p
For p=1 to Card (p)
$s3 = AltVec(y2)$
Se calculan los Requerimientos en Conjunto para las entradas $y2$ del Stroke que fabrica y
$RC(y3, t - lt) = ceil(RC(y3, t - lt) + ((ceil(RPLAN(y2, t)/strokeOut(y2, s3))) * strokeIn(y3, s3)))$
Se realiza el MRP a las entradas $y3$ del Stroke $s3$ que fabrica el producto $y2$
$End\ For$
$End\ For$
Se calcula el costo total

$$CostoTotalInventario = \sum_{i=1}^{Card(i)} \sum_{t=1}^{Card(t)} BINV(i,t) * h\ (i,t)$$

$CostoTotalOperacion = \sum_{i=1}^{Card(i)} \sum_{t=1}^{Card(t)} SolStrokes(k,t) * CO(k,t)$
$CostoTotalSetup = \sum_{i=1}^{Card(i)} \sum_{t=1}^{Card(t)} \delta(k,t) * CS(k,t)$
$Z = CostoTotalInventario + CostoTotalSetup + CostoTotalOperacion$

Por último, el cálculo de los requerimientos en cada periodo se realiza a través de la subrutina de MRP. Esta subrutina, es utilizada en el algoritmo de Evaluación del vecindario, calcula los requerimientos de netos de cada SKU en cada periodo y lleva el control del

inventario al hacer el registro de las unidades consumidas y producidas por cada stroke. La subrutina MRP puede ser consultada en el Anexo 7.

4.4. Generación de las Instancias de Prueba

El entorno de trabajo para la generación de las instancias de prueba es una integración de las interfaces de Matlab y Excel. La generación de la estructura de datos se realiza en Matlab a través del algoritmo mostrado en el **Anexo 1**.

La nomenclatura utilizada para la identificación de las instancias generadas se muestra en la figura 9.

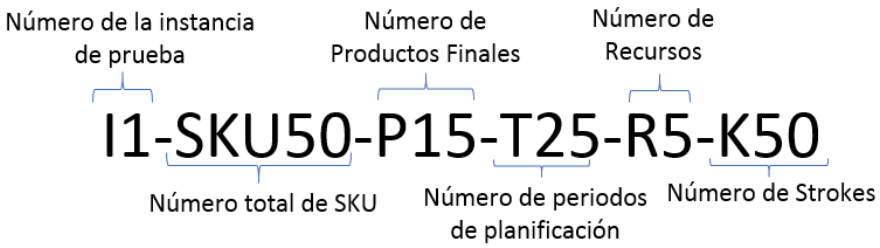


Figura 10. Nomenclatura para la identificación de instancias de prueba aleatorias.

Algunas de las consideraciones presentes en la generación de instancias de prueba son:

1. Se asegura que cada Stroke tenga por lo menos 1 entrada.
2. Los productos SKU finales no podrán ser componentes de otros SKU, por lo cual no tendrán valores asignados en la matriz de entradas $SI_{i,k.}$
3. No se considerarán Strokes de Transporte: Los SKU que entran a un Stroke no pueden ser salidas de ese mismo Stroke.
4. La matriz de demanda solamente contendrá la demanda de los SKU finales y estará distribuida uniformemente a partir del séptimo periodo de planificación con el fin de evitar que ciertas órdenes de componentes salgan del periodo de planificación.

5. El algoritmo de generación de instancias de prueba incluye el cálculo de los siguientes parámetros, utilizados dentro del algoritmo de generación de solución inicial para resumir la información del caso generado y tener acceso más rápido a la estructura de datos:
    a. *ListadoSKU(i):* es la lista que resume e identifica los SKU finales, es decir aquellos que dentro del caso generado tienen una demanda independiente mayor a 0.
    b. *StrokeInputs(k,i):* es un listado resumido de todos los SKU que son requeridos por cada Stroke.
    c. *MaxInputs:* este valor representa el número máximo de SKU diferentes que pueden entrar a un Stroke. Se utiliza para minimizar el recorrido de ciertos ciclos dentro del algoritmo.
    d. *StrokeOutputs(k,i):* es un listado resumido de todos los SKU que resultan de la ejecución de cada Stroke. Se utiliza para tener un rápido acceso a la información de coproducción del problema generado.
    e. *MaxOutputs:* este valor representa el número máximo de SKU diferentes que pueden salir de un Stroke. Se utiliza para minimizar el recorrido de ciertos ciclos dentro del algoritmo.
    f. *Alt(i,k):* es una lista resumida de los Strokes alternativos que existen para la producción de cada SKU. Se utiliza para tener un rápido acceso a la información de listas de materiales alternativas del problema generado.

4.4.1. Exportación de instancias de prueba Matlab – Excel.

El segundo paso consiste en la exportación de la instancia generada a una hoja de cálculo de Excel, esto con el fin de asegurar la reusabilidad de los conjuntos de datos generados y el acceso a la instancia de desde cualquiera de los programas utilizados.

El algoritmo utilizado para el proceso de exportación Matlab – Excel puede ser consultado en el **Anexo 2.**

La estructuración del archivo .xls utilizado para la exportación debe permitir que cada uno de los parámetros de la instancia de prueba pueda ser escrito en una pestaña diferente del archivo. Con el fin estandarizar el proceso de exportación, se definieron las siguientes pestañas dentro del archivo .xls:

Pestaña 1. DatasetDetails: A manera de identificación, esta pestaña contiene un resumen de la configuración de la instancia de prueba: Número de total de productos, Número de productos finales, número de Strokes, Número de recursos y Número de periodos de planificación.

*Pestaña 2. Demanda***:** Esta pestaña contiene las cantidades requeridas de cada uno de los productos finales en cada uno de los periodos de planificación.

*Pestaña 3. CostoAlmacenaje***.** En esta pestaña se registrará el costo de almacenaje para cada SKU en cada periodo de planificación.

*Pestaña 4. InventarioInicial***:** En esta pestaña se registrará el inventario inicial para cada SKU.

*Pestaña 5. RecepcionesProgramadas***:** En esta pestaña se registrará las recepciones programadas para cada SKU en el horizonte de planificación.

*Pestaña 6. EntradasStrokes***:** En esta pestaña se registran las cantidades requeridas de cada SKU para desarrollar cada uno de los Strokes.

*Pestaña 7. SalidasStrokes***:** En esta pestaña se registran las cantidades producidas de cada SKU después de la ejecución de cada Stroke.

*Pestaña 8. CostoStrokes***:** Esta pestaña contiene a los costos de la ejecución de cada Stroke en cada uno de los periodos.

*Pestaña 9. CostoSetupStroke***:** En esta pestaña se registran los costos de alistamiento para la ejecución de un Stroke en cada uno de los periodos de planificación. Los costos y la naturaleza de los alistamientos no dependen de la secuencia de operaciones.

Pestaña 10. LeadTimeStrokes: En esta pestaña se registran los tiempos de entrega de cada Stroke.

Pestaña 11. CapacidadRecursos: En esta pestaña se registra la capacidad (en unidades de tiempo) de cada recurso en cada periodo de planificación.

Pestaña 12. CapacidadRequerida: En esta pestaña se registra la capacidad requerida de cada recurso (en unidades de tiempo) por cada uno de los Strokes.

Pestaña 13. SetupRequerido: En esta pestaña se registra el tiempo de alistamiento requerido de cada recurso antes de realizar cada Stroke.

Es posible importar a Matlab las instancias de prueba que ya han sido exportadas a Excel. El algoritmo utilizado para este fin se encuentra disponible en el **Anexo 3.**

4.5. Conclusiones

Se mostró la estructura y las nociones del funcionamiento del algoritmo heurístico basado en listas tabú para la solución del problema de lotificación, así como las generalidades técnicas en los procesos de generación de instancias aleatorias y los procesos de exportación e importación entre las diferentes herramientas utilizadas.

A través del uso de subrutinas se logró simplificar el funcionamiento de la heurística, y se facilitó la realización de tareas y ciclos repetitivos durante su ejecución.

La implementación de procesos de importación y exportación de datos entre las interfaces de Matlab, Excel y GAMS, permite la replicabilidad de los resultados y el almacenamiento y uso de las instancias aleatorias generadas.

La regla de despacho de Lote por Lote fue seleccionada para apoyar el proceso de cálculo de requerimientos debido a su facilidad de uso y por permitir formular el modelo sin tener en cuenta costos de faltantes.

En el siguiente capítulo se muestran los resultados obtenidos para cada de las instancias de prueba generadas.

Capítulo 5. Experiencia Computacional

5.1. Introducción

Al momento de realizar comparaciones sobre el rendimiento y la eficiencia de diferentes métodos heurísticos de solución para un determinado problema de optimización, por lo general se utiliza como referencia la solución obtenida mediante algún método exacto. Se evalúa la medida en que la solución arrojada por el método heurístico se acerca o no a la solución exacta de referencia. También es importante tener en cuenta los tiempos de solución con el fin de establecer y comparar los niveles de eficiencia computacional.

En el presente capítulo se exponen en primer lugar, las consideraciones y parámetros utilizados durante las fases de experimentación. Luego se muestran los resultados obtenidos mediante diseño experimental para la optimización de los parámetros básicos del algoritmo heurístico. Por último, se exponen y comparan las medidas de desempeño en la solución de las instancias de prueba generadas.

5.2. Consideraciones y Parámetros

5.2.1. Recursos Utilizados

Para la obtención de la solución óptima y las soluciones heurísticas de las instancias de prueba seleccionadas, se utilizó la versión 24.8.2 de GAMS y la versión R2017a de Matlab. Se utilizó un computador portátil con las siguientes especificaciones:

- Memoria RAM: 12 Gb (11.5 Gb utilizables)
- Procesador AMD A6-5200 APU 2.00 GHz
- Sistema Operativo Windows 10 de 64 bits.

5.2.2. Instancias de Prueba Generadas

Los parámetros utilizados para la generación de las instancias aleatorias se basan en los utilizados por Rius Sorolla, Maheut, Coronado-Hernandez, & Garcia-Sabater (2017) y algunos fueron modificados con el fin de asegurar soluciones factibles para las instancias más grandes.

Tabla 5

Parámetro	*Valor*
i	50 referencias SKU
	100 referencias SKU
	200 referencias SKU
i^*	15 productos finales
	30 productos finales
	60 productos finales
t	25 periodos de planificación
	50 periodos de planificación
	75 periodos de planificación
r	5 recursos
	10 recursos
	20 recursos
k	50 strokes
	100 strokes
	200 strokes
$D_{i,t}$	Uniforme Discreta [1000,2000]
$h_{i,t}$	Uniforme Discreta [10,20]
$CO_{k,t}$	Uniforme Discreta [5,8]
$CS_{k,t}$	Uniforme Discreta [5,10]
$CB_{i,t}$	Uniforme Discreta [2,10]
$SO_{i,k}$	Uniforme Discreta [35,50]
$SI_{i,k}$	Uniforme Discreta [4,8]
LT_k	Uniforme Discreta [1,2]
$KAP_{r,t}$	Uniforme Discreta [2000,6000]
	Uniforme Discreta [4000,8000]
	Uniforme Discreta [12000,14000]
$TO_{k,r}$	Uniforme Discreta [2,5]
$TS_{k,r}$	Uniforme Discreta [5,10]

Nota: Parámetros utilizados en la generación de las instancias de prueba.

Para la generación de los parámetros aleatorios se utilizó la función *Rand* de Matlab y no se utilizaron semillas de generación. Se generaron instancias de prueba de 3 tamaños

diferentes, modificando el número SKU, el número de productos finales, el número de periodos de planificación, el número de recursos y el número de *strokes.* Las instancias generadas se listan a continuación:

- I1-SKU50-P15-T25-R10-K50
- I2-SKU50-P15-T25-R10-K100
- I3-SKU50-P15-T25-R10-K200
- I4-SKU100-P30-T50-R20-K50
- I5-SKU100-P30-T50-R20-K100
- I6-SKU100-P30-T50-R20-K200
- I7-SKU200-P60-T75-R30-K50
- I8-SKU200-P60-T75-R30-K100
- I9-SKU200-P60-T75-R30-K200

5.3. Optimización de parámetros iniciales utilizando diseño experimental

Con el fin de establecer los parámetros iniciales del algoritmo heurístico basado en listas tabú, se realizó un diseño experimental en el cual se definieron las siguientes variables de respuesta:

- ***El tiempo de solución de la instancia (Tiempo Solución).*** Es el tiempo utilizado por el algoritmo (en segundos) para arrojar un resultado. Dado que el funcionamiento de la heurística involucra procesos en los que intervienen variables aleatorias, el resultado en cada iteración varía y por este motivo el valor para cada tratamiento se obtuvo promediando el resultado de 5 réplicas.

- **Valor Objetivo o Costo Total:** Es el costo total de la solución obtenida con la heurística basada en listas tabú para la planificación de la instancia

Dado que el funcionamiento de la heurística involucra procesos en los que intervienen variables aleatorias, el resultado en cada iteración varía y por este motivo el valor para cada tratamiento se obtuvo promediando el resultado de 5 réplicas.

Se definieron 3 factores experimentales, con 3 niveles cada uno. Los factores seleccionados se detallan a continuación:

Número de iteraciones. Este es el parámetro principal que define la terminación del algoritmo y define el número de veces que se generan y evalúan vecindarios de solución. Se definieron 3 niveles de número de iteraciones: 25, 50 y 75 iteraciones.

La tenencia Tabú (Tabu Tenure). Como se mencionaba en secciones pasadas, el valor de tenencia tabú determina el número de iteraciones que un movimiento realizado permanece en la lista tabú. El valor de tenencia impide al algoritmo deshacer el movimiento realizado y a su vez explorar más eficientemente el vecindario de soluciones. Se definieron 3 niveles de Tenencia Tabú los cuales responden a una determinada proporción del número de iteraciones que se definan: 1/4 del número de Iteraciones, ½ del número de Iteraciones y ¾ del número de Iteraciones.

Número de Vecinos: Este factor es importante para el desempeño del algoritmo ya define el número de soluciones vecinas que se generan y evalúan en cada iteración. Un número muy elevado de vecinos evaluados por iteración puede aumentar el gasto computacional de la heurística, afectando el tiempo de solución.

En la tabla 6 se muestra un resumen de los factores y los niveles experimentales utilizados:

Tabla 6

Factor	Niveles Experimentales
Número de Iteraciones	25
	50
	75
Tabu Tenure	1/4 N. Iteraciones
	½ N. Iteraciones
	¾ N. Iteraciones
Número de Vecinos	5
	10
	25

Nota: Factores y niveles experimentales

Se definieron 2 tamaños de problema diferentes para la experimentación.

Tamaño de Problema 1: El primer tamaño de problema corresponde a la instancia de prueba I1-SKU50-P15-T25-R10-K50 y fue tomado como tamaño de referencia de instancias pequeñas.

Tamaño de Problema 2: El segundo tamaño de problema corresponde a la instancia de prueba I4-SKU100-P30-T50-R20-K50 y fue tomado como tamaño de referencia de instancias grandes.

Por motivo de los elevados tiempos necesarios para la ejecución de réplicas, no se consideró la utilización de instancias de prueba más grandes para la experimentación. Los resultados arrojados para cada uno de los tamaños de problema se muestran a continuación.

5.3.1. Resultados para el problema de tamaño 1

La tabla 7 se muestra los resultados de Tiempo de Solución y Valor Objetivo para el Problema de tamaño 1 en cada uno de los tratamientos.

Tabla 7

Tabu Tenure	Iteraciones	Vecinos	Tiempo de Solución (s) Promedio	Valor Objetivo Promedio
1/4 It	25	5	283.822	18552739.4
1/4 It	25	10	506.503	16772556.2
1/4 It	25	15	820.518	11807876.2
1/4 It	50	5	268.081	14016230.8
1/4 It	50	10	893.116	8108837.4
1/4 It	50	15	1,346.348	8772643.2
1/4 It	75	5	657.966	6223119
1/4 It	75	10	1,288.821	5055230.6
1/4 It	75	15	1,985.018	5401856.4
1/2 It	25	5	283.59	17024637.6
1/2 It	25	10	504.848	15514693
1/2 It	25	15	734.847	10141966.4
1/2 It	50	5	456.300	10064700.2
1/2 It	50	10	998.242	7538917.8
1/2 It	50	15	1398.374	8183835.4
1/2 It	75	5	728.395	8362573.6
1/2 It	75	10	1224.981	4862035.4
1/2 It	75	15	2110.907	5629942.8
3/4 It	25	5	288.996	19172266.8
3/4 It	25	10	511.321	13830190.8
3/4 It	25	15	777.533	11043073.6
3/4 It	50	5	517.534	11030234.8
3/4 It	50	10	936.422	7331563.4
3/4 It	50	15	1265.6	6314529
3/4 It	75	5	840.181	7487257.6
3/4 It	75	10	1110.758	5294085.8
3/4 It	75	15	1863.22	5531267.8

Nota: Resultados de Tiempo de Solución y Valor Objetivo para el Problema de tamaño 1

5.3.1.1. Tiempo de Solución

El resultado del análisis de varianzas del experimento para la variable de Tiempo de Solución se muestra en la figura 11.

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Tenure	2	9788	9788	4894	1,09	0,382
Iteraciones	2	2801619	2801619	1400810	311,45	0,000
Vecinos	2	3544076	3544076	1772038	393,99	0,000
Tenure*Iteraciones	4	22178	22178	5545	1,23	0,370
Tenure*Vecinos	4	47939	47939	11985	2,66	0,111
Iteraciones*Vecinos	4	477581	477581	119395	26,55	0,000
Error	8	35981	35981	4498		
Total	26	6939163				

S = 67,0647 R-Sq = 99,48% R-Sq(adj) = 98,31%

Figura 11. Análisis de varianza para tiempos de solución – Problema 1.

Utilizando un valor alfa de 0,05 y teniendo en cuenta los valores P arrojados, se confirmó el efecto significativo $(P <= 0.05)$ en el tiempo de solución de los factores de *Número de Vecinos* y *Número de iteraciones* e igualmente de la interacción presentada entre estos.

En este caso, el factor de *Tenencia Tabú* no presenta un efecto estadísticamente significativo sobre el tiempo de solución $(P > 0.05)$. Tampoco lo presentan las interacciones que involucran este factor. Se excluyó el análisis de la interacción de tercer nivel entre los 3 factores dado que no se contaba con un suficiente número de grados de libertad para realizar el cálculo de los estadísticos de prueba.

En cuanto a la confiabilidad de estos resultados, vemos que los valores de R cuadrado y R cuadrado ajustado son adecuados (99,48% y 98,31% respectivamente) y permiten afirmar que el modelo generado describe de manera confiable el comportamiento de los datos.

Adicionalmente, se confirmaron los supuestos de normalidad (Prueba de Anderson-Darling, Figura 13), homocedasticidad e independencia de los datos (véase Figura 12).

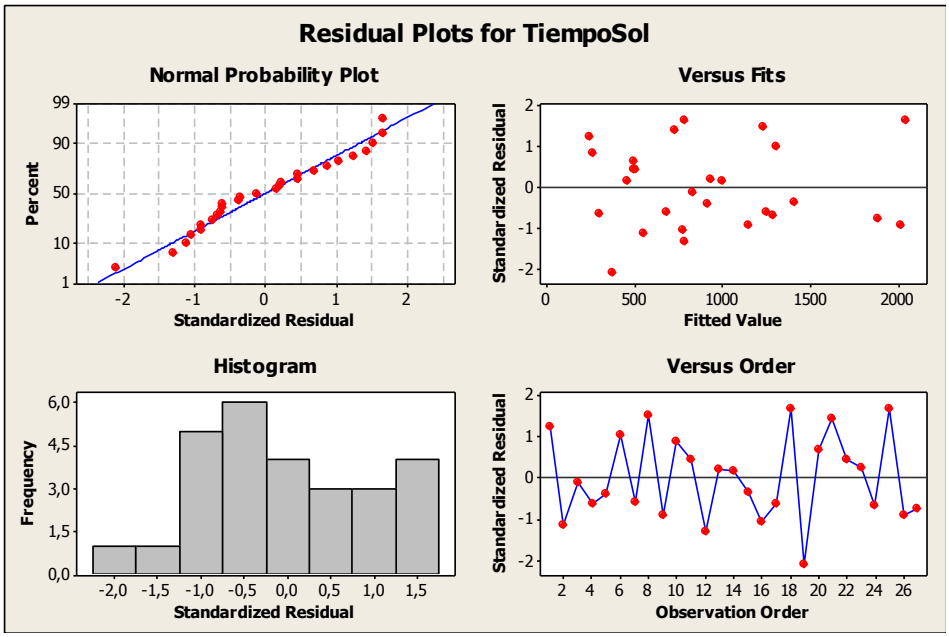


Figura 12. Comprobación de supuestos estadísticos para tiempos de solución – Problema 1.

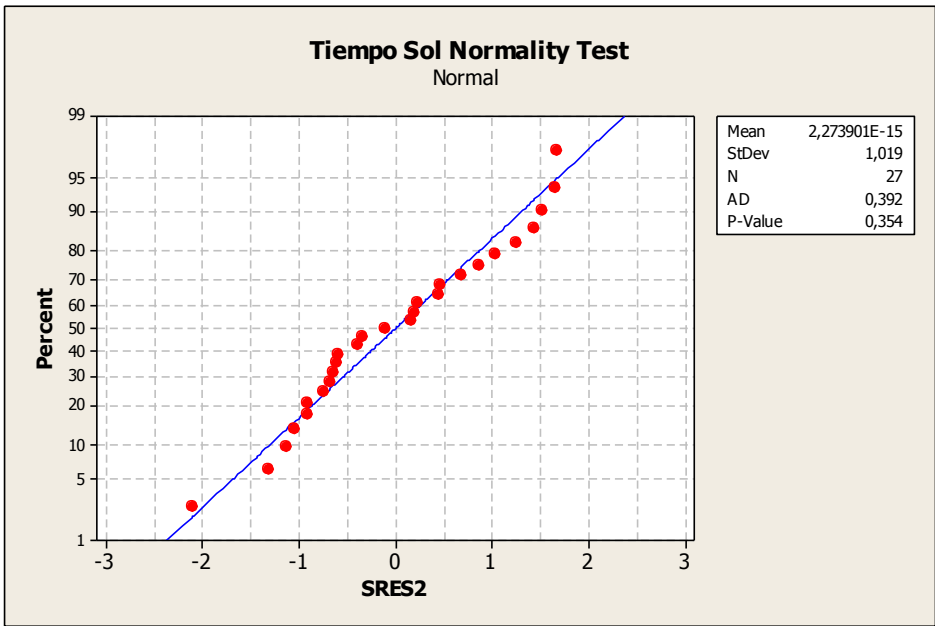


Figura 13. Prueba de Normalidad de Residuos para tiempos de solución – Problema 1.

En la Figura 14 se muestra el gráfico de efectos principales para la variable de respuesta de Tiempo de Solución y en la Figura 15 se muestra el gráfico de interacciones.

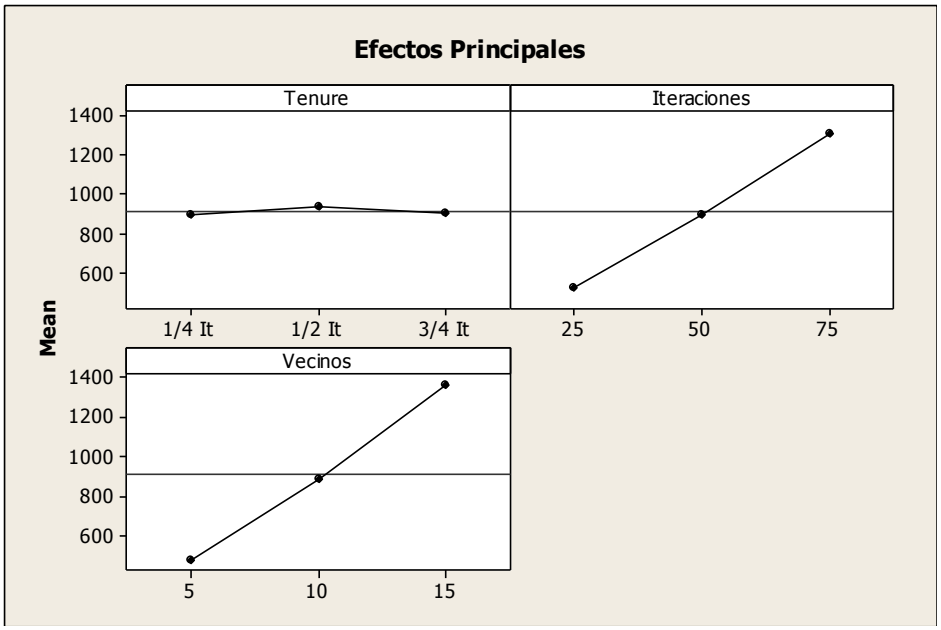


Figura 14. Gráfico de efectos principales para tiempos de solución – Problema 1.

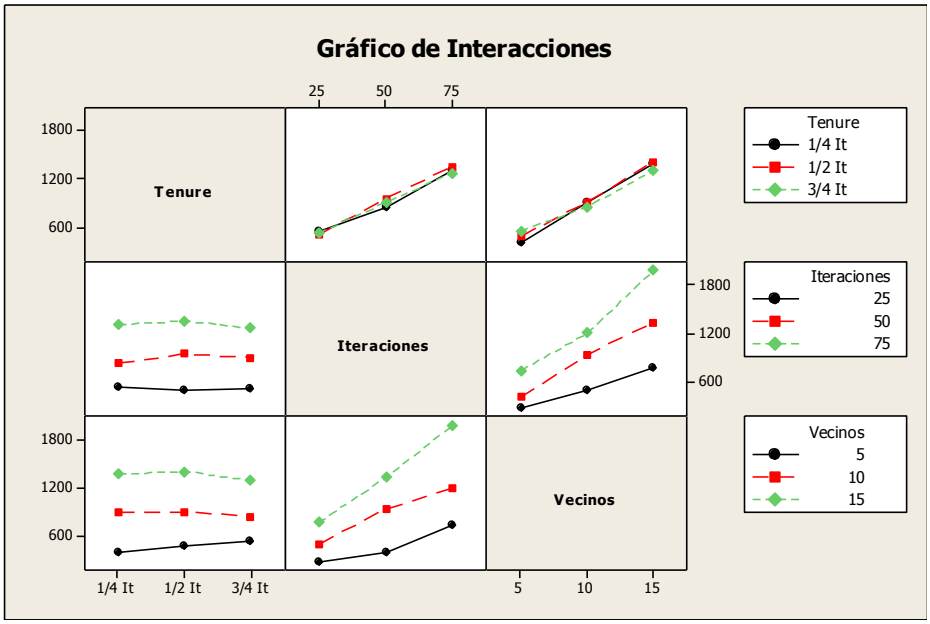


Figura 15. Gráfico de interacciones para tiempos de solución – Problema 1.

El gráfico de efectos principales permite comprobar el poco efecto estadístico de los diferentes niveles de tenencia tabú en el tiempo de solución. Como es de esperarse, el número de iteraciones y el número de vecinos determina en gran medida los tiempos de solución de la heurística. La interacción estadísticamente significativa existente entre el *Número de Vecinos* y el *Número de Iteraciones* es el resultado de la relación directamente proporcional que se presenta entre los niveles de estos factores y el tiempo de solución.

5.3.1.2. Valor Objetivo o Costo Total.

El resultado del análisis de varianzas del experimento para la variable de Valor Objetivo o Costo Total se muestra en la Figura 16:

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Tenure	2	1,13552E+13	1,13552E+13	5,67760E+12	5,13	0,037
Iteraciones	2	3,30564E+14	3,30564E+14	1,65282E+14	149,40	0,000
Vecinos	2	7,97401E+13	7,97401E+13	3,98701E+13	36,04	0,000
Tenure*Iteraciones	4	1,28211E+13	1,28211E+13	3,20527E+12	2,90	0,094
Tenure*Vecinos	4	3,33322E+12	3,33322E+12	8,33305E+11	0,75	0,583
Iteraciones*Vecinos	4	2,65571E+13	2,65571E+13	6,63927E+12	6,00	0,016
Error	8	8,85048E+12	8,85048E+12	1,10631E+12		
Total	26	4,73222E+14				

S = 1051813 R-Sq = 98,13% R-Sq(adj) = 93,92%

Figura 16. Análisis de varianza para Valor Objetivo – Problema 1

Utilizando un alfa de 0,05 y teniendo en cuenta los valores P arrojados, se puede afirmar que todos los factores principales tienen un efecto estadísticamente significativo en el valor objetivo o costo total ($P < 0.05$). Se confirmaron los supuestos de normalidad (Prueba de Anderson-Darling, Figura 18), homocedasticidad e independencia de los datos (véase figura 17).

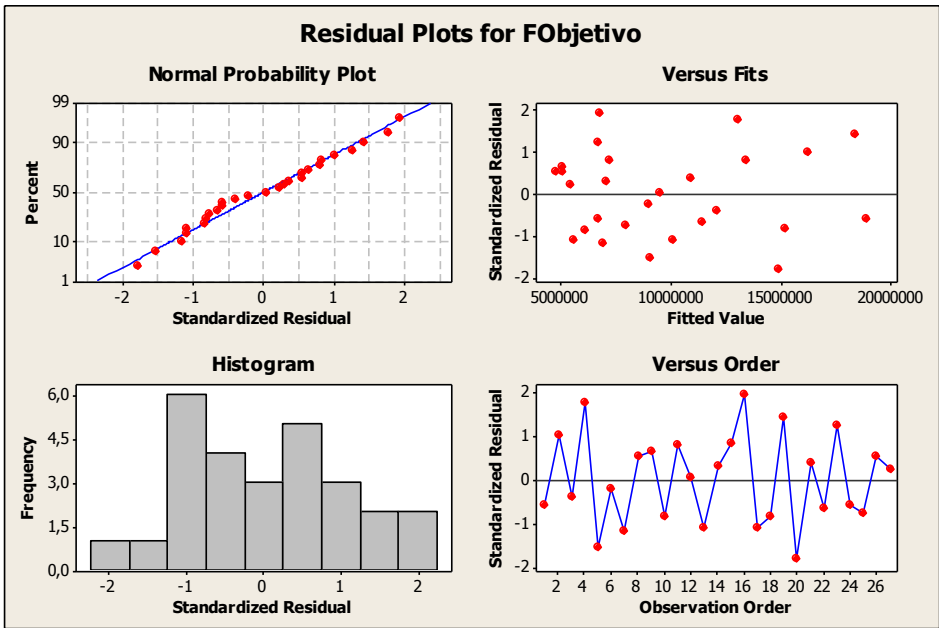


Figura 17. Comprobación de supuestos estadísticos para Valor Objetivo – Problema 1

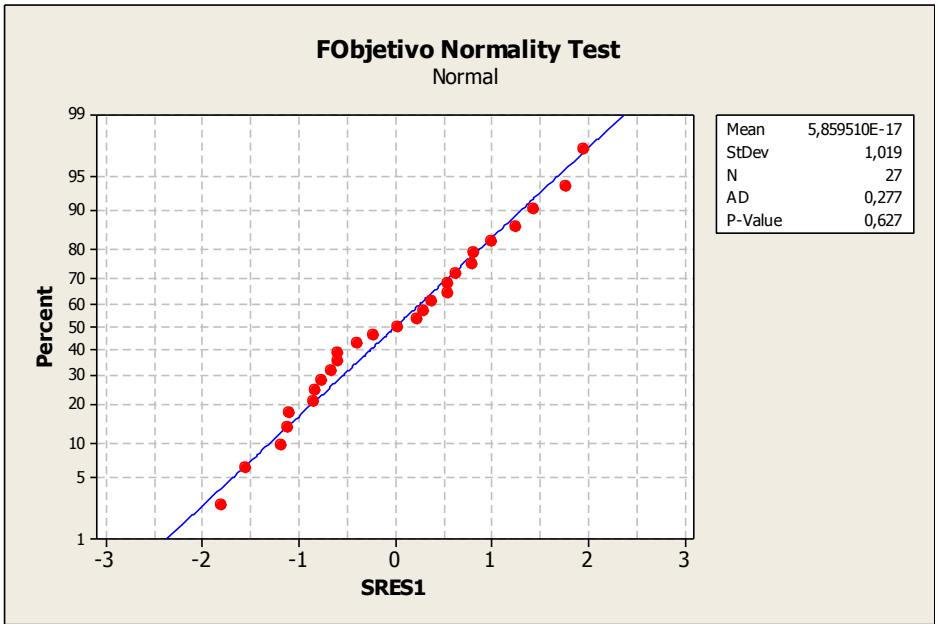


Figura 18. Prueba de Normalidad de Residuos para Valor Objetivo– Problema 1.

En la Figura 19 se muestra el gráfico de efectos principales y en la Figura 20 se muestra el gráfico de interacciones.

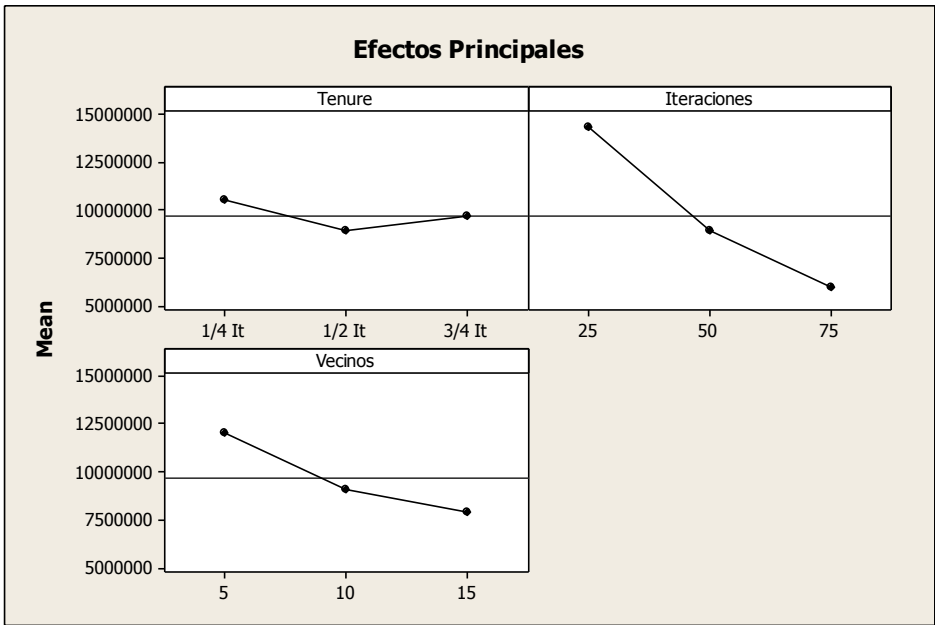


Figura 19. Gráfico de efectos principales para Valor Objetivo– Problema 1.

Según el diagrama de efectos principales, en promedio, se obtienen mejores soluciones utilizando un valor de Tenencia Tabú correspondiente a la mitad del número de iteraciones definido (1/2 It).

La calidad de la solución presenta en este caso una relación directamente proporcional al número de iteraciones y al número de vecinos, por lo que el algoritmo tiende a encontrar mejores soluciones al utilizar los niveles más altos de estos factores.

El gráfico de interacciones (Figura 20) permite asegurar que, al seleccionar el nivel más alto de número de iteraciones, se obtiene un menor promedio del valor objetivo sin importar el nivel seleccionado de los dos factores restantes (fila central del gráfico).

En cuanto al número de vecinos, seleccionar el nivel más alto de este factor no necesariamente asegura la consecución de mejores soluciones. Si observamos el recuadro central de la última fila, podemos notar que, es posible obtener soluciones de mejor calidad utilizando una combinación de 10 vecinos y 75 iteraciones.

Y, por último, seleccionar un valor de Tenencia Tabú correspondiente a 1/2 del número de iteraciones permite encontrar buenas soluciones, sin diferencias notorias en las diferentes combinaciones con los demás factores.

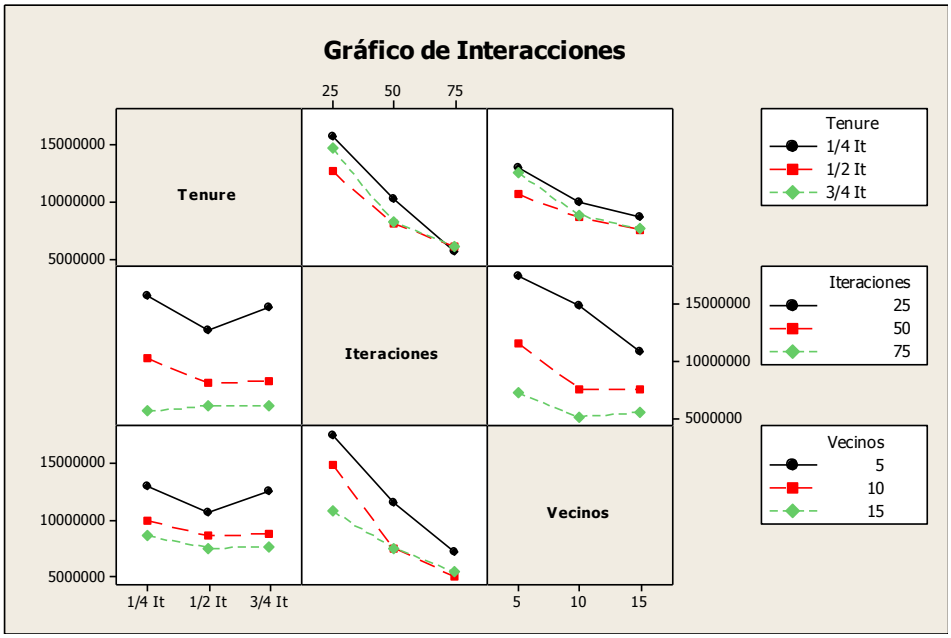


Figura 20. Gráfico de interacciones para Valor Objetivo – Problema 1.

En conclusión, los niveles de los factores que favorecen la obtención de mejores soluciones en el caso del Problema 1 son:

- Tenencia Tabú: 1/2 del número de iteraciones.
- Número de Vecinos: 10 vecinos.
- Número de Iteraciones: 75 iteraciones.

5.3.2. Resultados para el problema de tamaño 2

La tabla 8 muestra los resultados de Tiempo de Solución y Valor Objetivo para el problema de tamaño 2 en cada uno de los tratamientos.

Tabla 8.

Tabu Tenure	Iteraciones	Vecinos	Tiempo de Solución (s) Promedio	Valor Objetivo Promedio
1/4 It	25	5	824.81	157857698.4
1/4 It	25	10	1918.30	126812110.2
1/4 It	25	15	3332.56	124018894
1/4 It	50	5	1755.61	156591980
1/4 It	50	10	3750.90	130586149
1/4 It	50	15	4707.65	137488998.6
1/4 It	75	5	2902.41	131069037.6
1/4 It	75	10	5485.62	124354488.6
1/4 It	75	15	7736.08	125583299.4
1/2 It	25	5	819.66	137951686.6
1/2 It	25	10	1716.81	120946987.8
1/2 It	25	15	2503.06	120920501
1/2 It	50	5	1700.48	138717803.6
1/2 It	50	10	3453.77	138525127
1/2 It	50	15	6130.98	134299273
1/2 It	75	5	2510.78	138535623.2
1/2 It	75	10	4855.55	129618567
1/2 It	75	15	7961.27	125698316.8
3/4 It	25	5	1003.26	156885090
3/4 It	25	10	1615.81	135677831.6
3/4 It	25	15	2412.07	135363732.2
3/4 It	50	5	1851.44	127939313.6
3/4 It	50	10	3648.85	129507944.6
3/4 It	50	15	5182.18	132683921.6
3/4 It	75	5	2501.80	147343066.8
3/4 It	75	10	6257.54	132239233.8
3/4 It	75	15	7187.00	119824188.8

Nota: Resultados de Tiempo de Solución y Valor Objetivo para el Problema de tamaño 2

5.3.2.1. Tiempo de Solución

El resultado del análisis de varianzas del experimento para la variable de Tiempo de Solución en el problema 2 se muestra en la figura 21.

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Tenure	2	42537	42537	21268	0,11	0,900
Iteraciones	2	54271828	54271828	27135914	135,58	0,000
Vecinos	2	54471950	54471950	27235975	136,08	0,000
Tenure*Iteraciones	4	506729	506729	126682	0,63	0,653
Tenure*Vecinos	4	949133	949133	237283	1,19	0,387
Iteraciones*Vecinos	4	7566786	7566786	1891696	9,45	0,004
Error	8	1601151	1601151	200144		
Total	26	119410113				

S = 447,374 R-Sq = 98,66% R-Sq(adj) = 95,64%

Figura 21. Análisis de varianza para tiempo de solución – Problema 2

Utilizando un valor alfa de 0,05 y teniendo en cuenta los valores P arrojados, se confirmó el efecto significativo $(P <= 0.05)$ en el tiempo de solución de los factores de *Número de Vecinos* y *Número de iteraciones* e igualmente de la interacción presentada entre estos .

En este caso, el factor de *Tenencia Tabú* no presenta un efecto estadísticamente significativo sobre el tiempo de solución $(P > 0.05)$. Tampoco lo presentan las interacciones que involucran este factor. Se excluyó el análisis de la interacción de tercer nivel entre los 3 factores dado que no se contaba con un suficiente número de grados de libertad para realizar el cálculo de los estadísticos de prueba.

En cuanto a la confiabilidad de estos resultados, vemos que los valores de R cuadrado y R cuadrado ajustado son adecuados (98.66% y 95.64% respectivamente) y permiten afirmar que el modelo generado describe de manera confiable el comportamiento de los datos. Adicionalmente, se confirmaron los supuestos de normalidad (Prueba de Anderson-Darling, Figura 23), homocedasticidad e independencia de los datos (véase Figura 22).

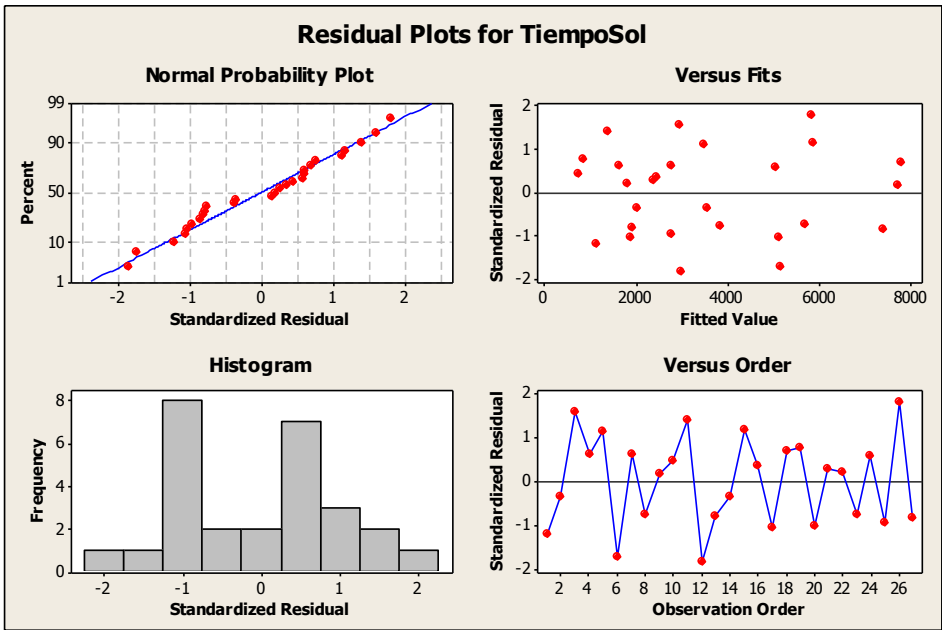


Figura 22. Comprobación de supuestos estadísticos para tiempo de solución – Problema 2

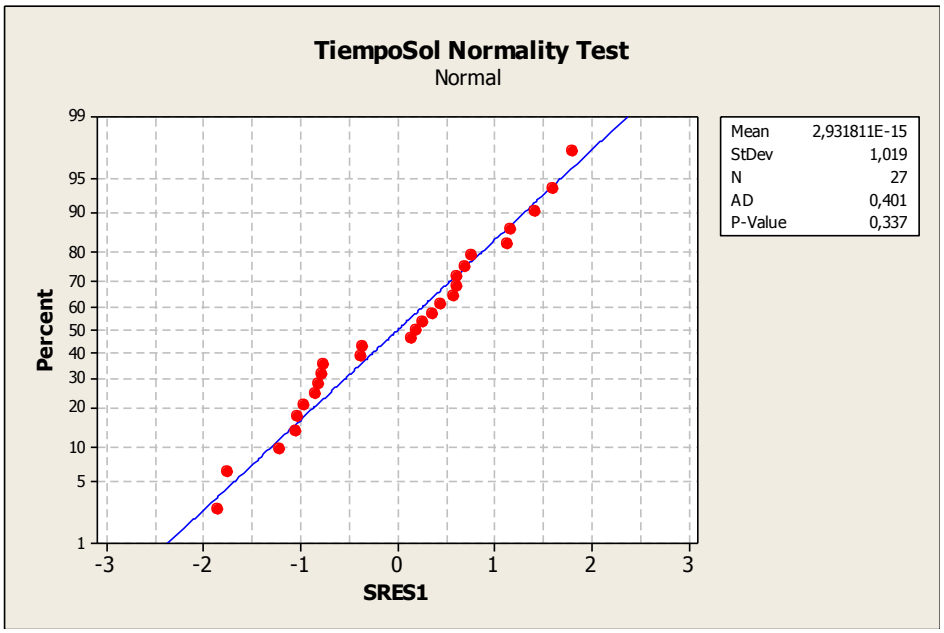


Figura 23. Prueba de Normalidad de Residuos para tiempos de solución– Problema 2

En la figura 24 se muestra el gráfico de efectos principales para la variable de respuesta de Tiempo de Solución y en la figura 25 se muestra el gráfico de interacciones.

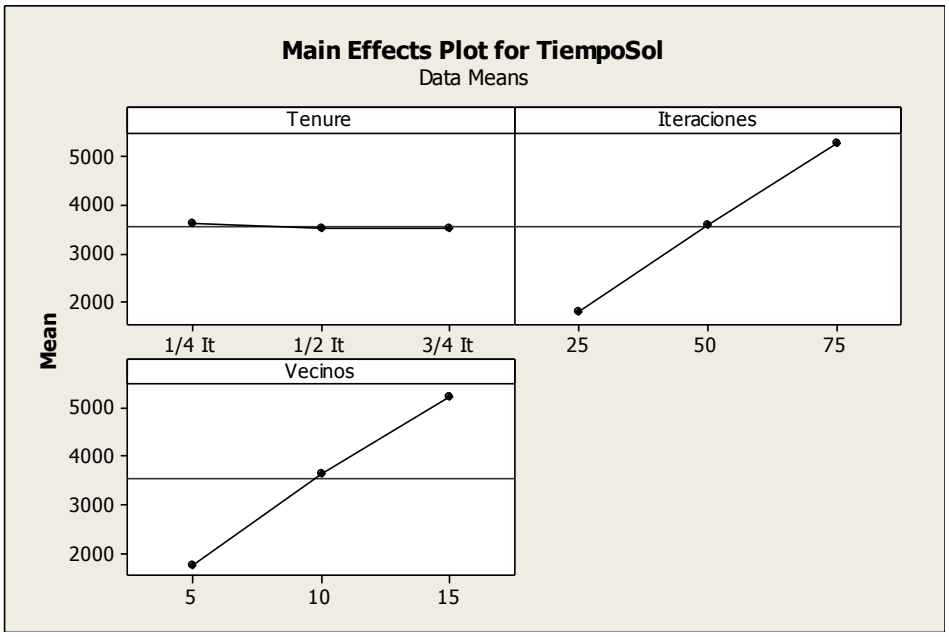


Figura 24. Gráfico de efectos principales para tiempos de solución– Problema 2

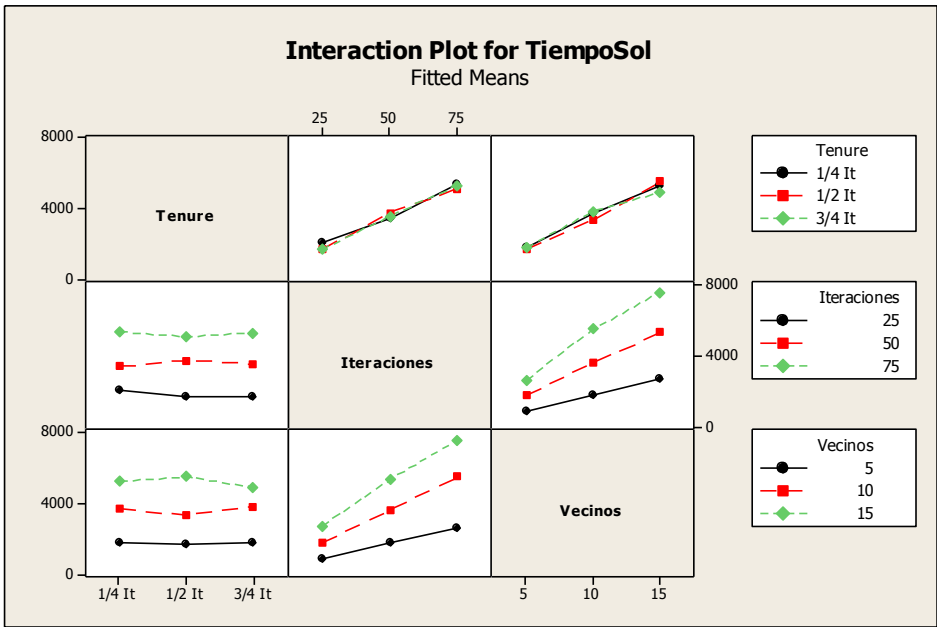


Figura 25. Gráfico de interacciones para tiempos de solución– Problema 2

Al igual que en los resultados del problema 1, el gráfico de efectos principales permite comprobar el poco efecto estadístico de los diferentes niveles de tenencia tabú en el tiempo de

solución. El número de iteraciones y el número de vecinos determina en gran medida los tiempos de solución de la heurística y nuevamente, la interacción estadísticamente significativa existente entre estos factores es el resultado de la relación directamente proporcional que existe con el tiempo de solución.

5.3.2.2. Valor Objetivo o Costo Total.

El resultado del análisis de varianzas del experimento para la variable de Valor Objetivo o Costo Total en el problema 2 se muestra en la Figura 26:

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Tenure	2	2,56563E+14	2,56563E+14	1,28281E+14	4,24	0,055
Iteraciones	2	9,82342E+14	9,82342E+14	4,91171E+14	16,24	0,002
Vecinos	2	6,25219E+14	6,25219E+14	3,12609E+14	10,34	0,006
Tenure*Iteraciones	4	1,45560E+14	1,45560E+14	3,63899E+13	1,20	0,380
Tenure*Vecinos	4	1,22394E+14	1,22394E+14	3,05985E+13	1,01	0,456
Iteraciones*Vecinos	4	1,05037E+14	1,05037E+14	2,62591E+13	0,87	0,523
Error	8	2,41914E+14	2,41914E+14	3,02393E+13		
Total	26	2,47903E+15				

S = 5499026 R-Sq = 90,24% R-Sq(adj) = 68,29%

Figura 26. Análisis de varianza para Valor Objetivo – Problema 2

Utilizando un alfa de 0,05 y teniendo en cuenta los valores P arrojados, se puede afirmar que los factores Número de Iteraciones y Número de Vecinos tienen un efecto estadísticamente significativo en el valor objetivo o costo total ($P < 0.05$). En el caso de la tenencia tabú, el valor P es cercano a al alfa seleccionado, por lo que habría que considerar su análisis utilizando un mayor valor de significancia. Se confirmaron los supuestos de normalidad (Prueba de Anderson-Darling, Figura 28), homocedasticidad e independencia de los datos (véase Figura 27).

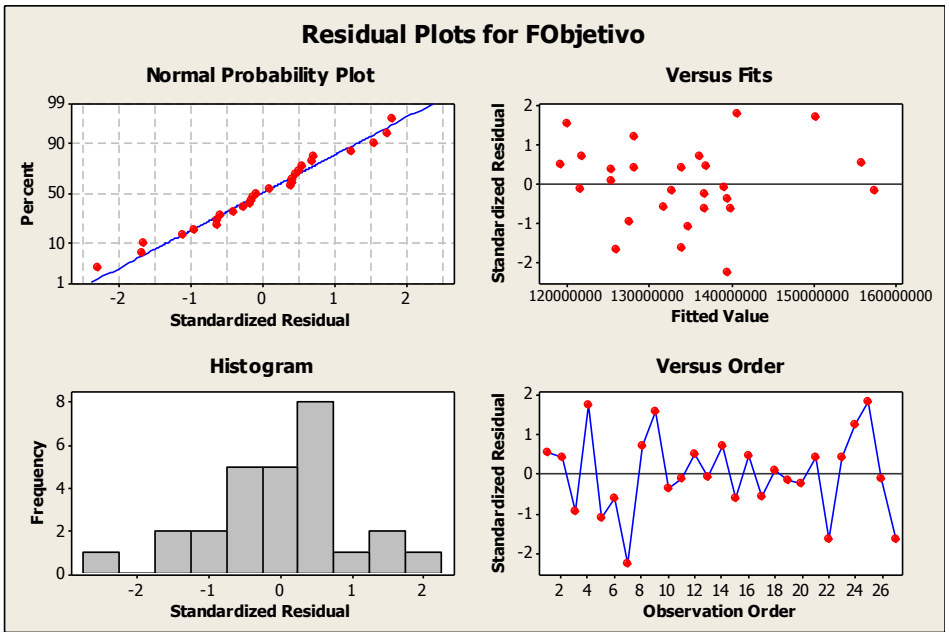


Figura 27. Comprobación de supuestos estadísticos para Valor Objetivo – Problema 2

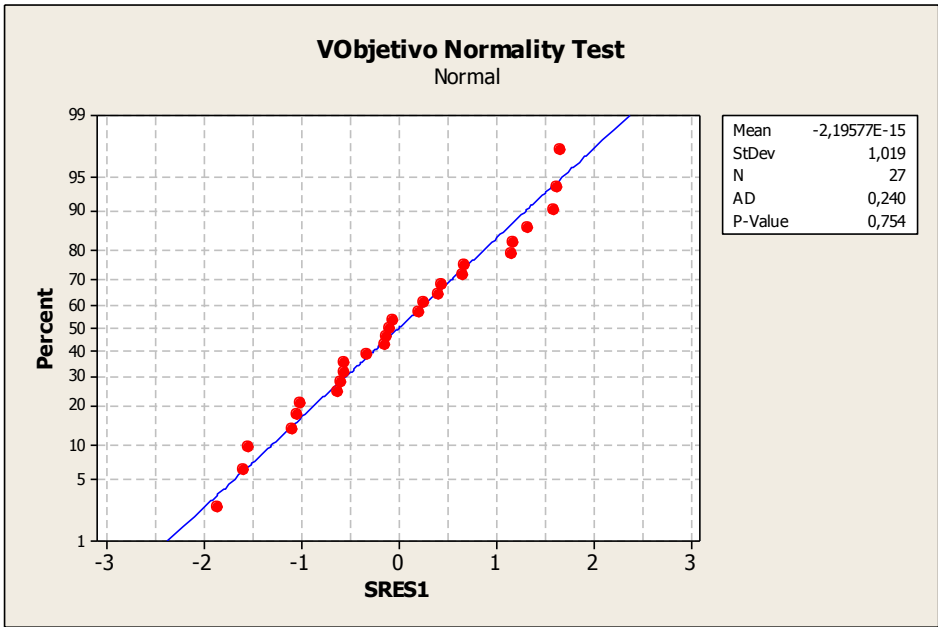


Figura 28. Prueba de Normalidad de Residuos para Valor Objetivo– Problema 2

En la Figura 29 se muestra el gráfico de efectos principales y en la Figura 30 se muestra el gráfico de interacciones.

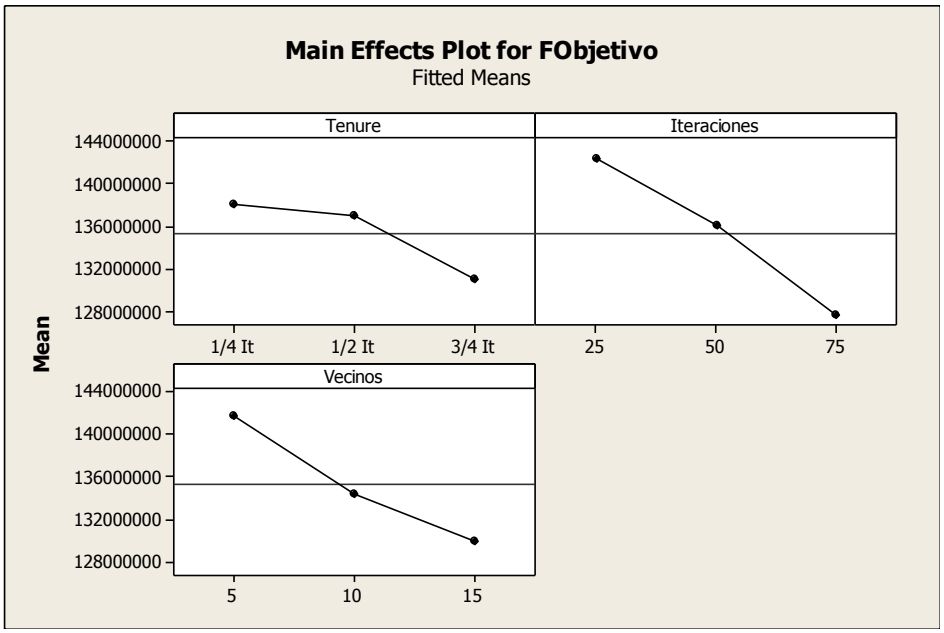


Figura 29. Gráfico de efectos principales para Valor Objetivo– Problema 2

Según el diagrama de efectos principales, en promedio, se obtienen mejores soluciones utilizando un valor de Tenencia Tabú correspondiente al 75% del número de iteraciones definido (3/4 It).

De igual manera, la calidad de la solución presenta una relación directamente proporcional al número de iteraciones y al número de vecinos, por lo que el algoritmo tiende a encontrar mejores soluciones al utilizar los niveles más altos de estos factores.

El gráfico de interacciones (Figura 30) permite asegurar que, al seleccionar el nivel más alto de número de iteraciones, se obtiene un menor promedio del valor objetivo sin importar el nivel seleccionado de los dos factores restantes (fila central del gráfico).

Del mismo modo, seleccionar el nivel más alto de número de vecinos factor asegura la consecución de mejores soluciones promedio.

Y, por último, seleccionar un valor de Tenencia Tabú correspondiente a 3/4 del número de iteraciones permite encontrar mejores soluciones, sin diferencias notorias en las diferentes combinaciones con los demás factores.

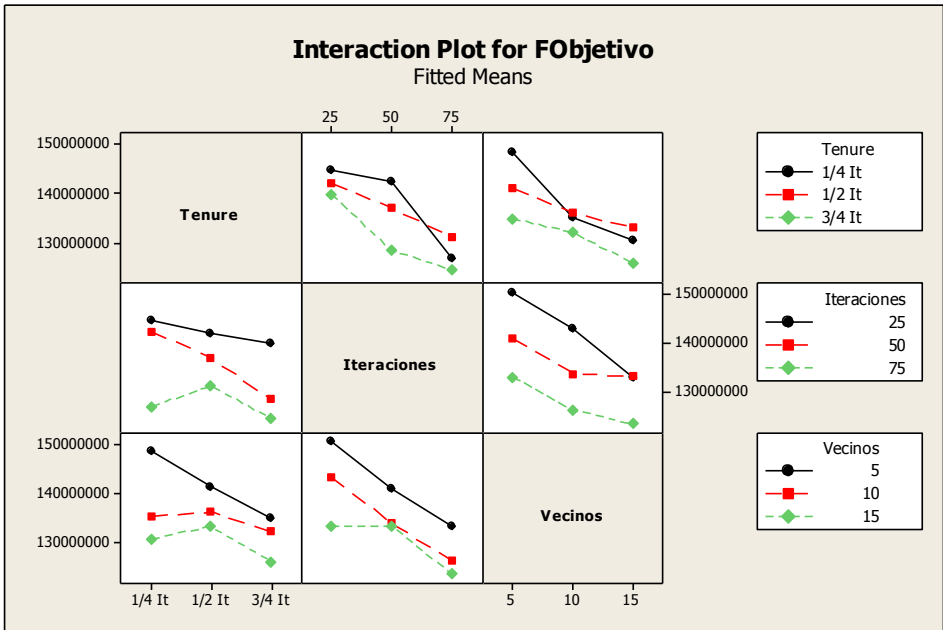


Figura 30. Gráfico de interacciones para Valor Objetivo– Problema 2

En conclusión, los niveles de los factores que favorecen la obtención de mejores soluciones en el caso del Problema 2 son:

- Tenencia Tabú: 3/4 del número de iteraciones.
- Número de Vecinos: 15 vecinos.
- Número de Iteraciones: 75 iteraciones.

5.4. Obtención de soluciones exactas y soluciones iniciales

5.4.1. Obtención de soluciones exactas

Para poder comparar el rendimiento y la eficiencia de la heurística, es necesario contar con una solución óptima para cada una de las instancias de prueba. Se utilizó una versión modificada del modelo exacto GMOP programado en lenguaje GAMS, similar al utilizado por Roca Molina (2016). Este puede ser consultado en el **Anexo 4.**

5.4.1.1. Importación de instancias de prueba Excel a Gams.

Las instancias son importadas a Gams mediante el uso de la funcionalidad GDX (Gams Data Exchange). Para cada instancia de prueba se configura un archivo con extensión .gdx, en el cual se incluirán todos los parámetros necesarios para ejecutar el modelo MIP del problema GMOP programado en GAMS.

Mediante una configuración similar a la utilizada por (Roca Molina, 2016) se generaron archivos de configuración del tipo Taskin.txt para cada una de las instancias de prueba. Estos archivos contienen la información de la ubicación y los rangos utilizados por los parámetros dentro de cada archivo de Excel y se utilizaron para la creación de los archivos GDX.

Una esquematización del proceso de generación y exportación de las instancias de prueba generadas se muestra en la figura 31.

Figura 31. Proceso de Importación y Exportación de Instancias Aleatorias.

5.4.2. Generación de una solución inicial

Con el fin de obtener una solución inicial factible como punto de partida para el algoritmo basado en listas tabú, se utilizará la subrutina de *EvaluacionSolucionVecina* (Algoritmo 2) con una modificación en el mecanismo de selección de Strokes.

El algoritmo modificado funciona mediante la política de Lote por Lote y seleccionará como stroke predeterminado el primer stroke alternativo para cada SKU.

Es decir, teniendo una matriz de strokes alternativos como la siguiente:

$$Alt\ (i,k) = \begin{matrix} k1 & k3 & 0 & 0 & \cdots \\ k2 & 0 & 0 & 0 & \cdots \\ k4 & k5 & 0 & 0 & \cdots \\ k1 & k2 & k4 & 0 & \cdots \\ k5 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix}$$

El algoritmo utilizará como vector solución la primera columna de la matriz $Alt\ (i,k)$:

$$S(i) = \begin{matrix} k1 \\ k2 \\ k4 \\ k1 \\ k5 \\ \vdots \end{matrix}$$

Aunque la solución inicial obtenida no es una solución óptima, considerar solamente el primer stroke alternativo permitirá obtener resultados en un menor tiempo y definir un valor objetivo de partida para la inicialización del algoritmo basado en listas tabú.

5.4. Resultados y Discusión.

La tabla 9 muestra los resultados obtenidos para el valor objetivo y el tiempo de solución para cada una de las instancias de prueba. Se realizaron 5 réplicas para cada una de las instancias y se registraron los valores mínimo y promedio para los resultados de valor objetivo y tiempo de solución.

Los parámetros del algoritmo utilizados durante las réplicas corresponden a los definidos para los diferentes tamaños de problema. Para las instancias I1, I2 e I3 se utilizaron los parámetros definidos mediante el diseño experimental para el problema de tamaño 1 (Sección 5.3.1.2). Para las instancias restantes I4 – I9, se utilizaron los parámetros definidos mediante el diseño experimental para el problema de tamaño 2 (Sección 5.3.2.2)

Con el fin de analizar y comparar los resultados obtenidos, la tabla muestra también las soluciones arrojadas por el modelo exacto programado en GAMS del problema GMOP y las arrojadas por el algoritmo de solución inicial. Adicionalmente se calcularon los porcentajes de diferencia entre las soluciones arrojadas por las heurísticas y el valor objetivo óptimo arrojado por el modelo exacto (Ecuación 7).

$$GAP = (Solución\ Heurística - Solución\ Exacta\ )/(Solución\ Exacta) \quad (7)$$

Así mismo se calcularon los porcentajes de diferencia para los tiempos de solución (Ecuación 8).

$$GAP\ Tiempo = \frac{Tiempo\ Solución\ Heurística - Tiempo\ Solución\ Exacta}{Tiempo\ Solución\ Exacta} \quad (8)$$

Tabla 9

	GMOP CPLEX			Solución Inicial			Heurística Lista Tabú			
	Valor Objetivo	**Gap (%)**	**Tiempo Sol. (s)**	**Valor Objetivo**	**Tiempo Sol. (s)**	**GAP vs GMOP**	**Valor Objetivo**	**GAP vs GMOP**	**Tiempo Sol. (s)**	**Tiempo Sol. Vs GMOP**
I1-SKU50-P15-T25-R10-K50	10.326.117	0,002980	1000,31	150.124.252	14,7	13,53	Mínimo 14.062.855	0,26571	Mínimo 1.424,9	0,42
							Promedio 14.650.126	0,29515	Promedio 1.482,2	0,48
I2-SKU50-P15-T25-R10-K100	2.683.534	0,017823	1000,45	63.586.982	18,5	22,69	Mínimo 3.704.761	0,27565	Mínimo 1.258,3	0,26
							Promedio 3.777.207	0,28954	Promedio 1.279,0	0,28
I3-SKU50-P15-T25-R10-K200	4.890.778	0,00234	1000,34	356.821.724	23,8	71,96	Mínimo 5.863.595	0,16590	Mínimo 1.321,5	0,32
							Promedio 5.908.060,6	0,17218	Promedio 1.350,2	0,35
I4-SKU100-P30-T50-R20-K50	106.247.444	0,000113	1001,98	3.211.497.566	47,8	29,23	Mínimo 118.817.405	0,10579	Mínimo 7.227,1	6,21
							Promedio 120.024.949,4	0,11478	Promedio 7.260,2	6,25
I5-SKU100-P30-T50-R20-K100	74.585.367	0,001660	1001,06	808.603.136	64,5	9,84	Mínimo 89.920.869	0,17054	Mínimo 7.741,5	6,73
							Promedio 90.653.853	0,17725	Promedio 7.792,7	6,78
I6-SKU100-P30-T50-R20-K200	43.278.154	0,031313	1000,77	945.203.080	64,6	20,84	Mínimo 60.710.169	0,28713	Mínimo 8.030,7	7,02
							Promedio 62.193.960,2	0,30414	Promedio 8.042,5	7,04
I7-SKU200-P60-T75-R30-K50	1.585.713.234	0,000010	1002,45	4.149.300.000	186,7	1,62	Mínimo 2.029.200.000	0,21855	Mínimo 24.959,6	23,90

							Promedio 2.066.960.000	0,23282	Promedio 25.002,3	23,94
I8-SKU200-P60-T75-R30-K100	765.235.068	0,000093	1001,95	3.568.700.000	186.4	3,66	Mínimo 994.986.057	0,23090	Mínimo 27.000,4	25,95
							Promedio 1.145.909.834.5	0,33220	Promedio 27.145,6	26,09
I9-SKU200-P60-T75-R30-K200	252.626.622	0,001546	1004,52	4.063.000.000	171,5	15,08	Mínimo 290.393.911	0,13005	Mínimo 23.741.6	22,63
							Promedio 296.727.085	0,14862	Promedio 23.980,7	22,87

Nota: Resultados obtenidos para el valor objetivo y el tiempo de solución para cada una de las instancias de prueba.

Durante la ejecución del método exacto para la obtención de las soluciones óptimas no se establecieron límites de tiempo en el código programado en GAMS. Sin embargo, se observa que la totalidad de las soluciones fueron obtenidas en un tiempo de aproximadamente 1000 segundos. El registro del proceso de solución en GAMS mostró en todos los casos un mensaje que indica que se excedieron los recursos de memoria disponibles y arrojó en todo caso el respecto porcentaje de GAP para cada solución exacta.

Aunque se utilizó un equipo de cómputo con buenas capacidades de memoria RAM (Sección 5.2.1) este inconveniente puede ser debido al tamaño de los problemas, haciendo que se excedan rápidamente los requerimientos de memoria.

Observando los resultados arrojados por el método de solución inicial, vemos que, aunque es una opción que puede ser competitiva en términos de tiempo de solución, los valores objetivo calculados se encuentran muy lejanos de los valores óptimos. Incluso estos pueden ser de hasta 73 veces mayores que los arrojados por el método exacto.

En promedio, la heurística basada en listas tabú fue capaz de arrojar soluciones cercanas al valor óptimo en porcentajes entre el 11% y el 33%. El menor porcentaje de diferencia obtenido fue del 10,58% para la instancia I4.

Los resultados del valor objetivo para el primer grupo de instancias (I1, I2 e I3) presentaron porcentajes de diferencia cercanos al 29% (instancias I1 e I2). Para la instancia I3 se obtuvo un porcentaje de diferencia mínimo de 16,56%. Los tiempos de solución para este grupo de instancias estuvieron en el orden de los 1280 segundos y los 1480 segundos. Estos valores estuvieron por encima del tiempo de solución utilizado por el modelo exacto (1000 segundos aproximadamente) en porcentajes entre 28% y 48% en promedio.

Para el segundo grupo de instancias (I4, I5 e I6) los porcentajes de diferencia promedio oscilaron entre el 11% y el 30%. La instancia I4 obtuvo un resultado mínimo de diferencia del 10,57%. Dado que el tamaño de las instancias aumentó con respecto a las del primer grupo, se observó un aumento considerable de los tiempos de solución, oscilando entre los 7200 y los 8000 segundos. Esto representa tiempos entre 600% y 700% por encima del tiempo utilizado por el método exacto.

En el tercer grupo de instancias (I7, I8 e I9) se obtuvieron soluciones con diferencias promedio que oscilaron entre el 14% y 33%. La instancia que obtuvo la solución con el menor porcentaje diferencia fue la instancia I9 con un porcentaje mínimo de 13,05%. Nuevamente, a través de los tiempos de solución se pudo evidenciar el aumento en los tamaños de las instancias. Los tiempos de solución estuvieron en el orden de los 23000 y 27000 segundos, superando los tiempos utilizados por el método exacto en porcentajes cercanos a 2600%.

La tabla 10 muestra el promedio de las diferencias mínimas para cada grupo de instancias.

Tabla 10.

	GAP Mínimo			Promedio
Grupo Instancias 1	0,26571	0,27565	0,1659	0,2358
Grupo Instancias 2	0,10579	0,17054	0,28713	0,1878
Grupo Instancias 3	0,21855	0,2309	0,13005	0,1932

Nota: Promedio de diferencias (GAP) mínimas por grupo de instancias.

Analizando el promedio de los menores GAP obtenidos, podemos observar que de manera general el algoritmo funciona mejor, en términos de calidad de la solución, en las instancias de mayor tamaño.

Pero es necesario tener en cuenta que, aunque los porcentajes de diferencia sean relativamente más altos en el primer grupo de instancias, vemos que el algoritmo es más competitivo en tiempos de solución para este conjunto de problemas. Esto nos permitiría afirmar que la heurística presenta un mejor balance entre calidad y tiempos de solución al enfrentarse a problemas relativamente pequeños.

En la Figura 32 se muestra un resumen de los tiempos de ejecución empleados en una de las réplicas realizadas para la instancia I9.

Profile Summary

Generated 13-Mar-2005 03:33:18 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
TabuSearch2	1	23741.631 s	32.546 s	
EvaluacionVecindario	1125	23709.085 s	11150.793 s	
TabuMRP	5062500	8109.104 s	8109.104 s	
TabuMRPComponentes	77544612	4449.188 s	4449.188 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

Figura 32. Resumen del perfil de tiempos del algoritmo – Matlab ®

Como se puede observar, aproximadamente la mitad del tiempo total de la ejecución corresponde al tiempo utilizado por la subrutina utilizada para la evaluación de las soluciones vecinas. También podemos observar que, el cálculo de los tamaños de lote a través de las subrutinas de MRP conlleva extensos tiempos de ejecución.

En la sección de anexos es posible encontrar un resumen del funcionamiento de la herramienta a través de la interfaz gráfica de Matlab.

Conclusiones y Recomendaciones

Se diseñó un algoritmo heurístico basado en listas tabú para la solución del problema capacitado de lotificación en sistemas de producción multinivel con listas de materiales alternativas y entornos de coproducción basados en el modelo GMOP.

Aunque la heurística no resulta eficiente en cuanto a tiempos de solución en los problemas mayor tamaño, su uso permitió encontrar soluciones cercanas a valores óptimos hasta en un 11%.

En la medida en que se optimice el gasto computacional de las subrutinas de evaluación de soluciones vecinas es posible hacer que la heurística resulte más eficiente en términos de tiempos de solución. Se propone como trabajo futuro el diseño de subrutinas que permitan el cálculo más eficiente de los tamaños de lote y el número de Strokes a realizarse por periodo.

El algoritmo basado en listas tabú representa una aproximación al funcionamiento de un algoritmo de búsqueda tabú básico. Con la inclusión de métodos de aspiración, intensificación y/o diversificación se presenta una oportunidad importante para próximas investigaciones en el estudio del problema de lotificación multinivel y especialmente el problema GMOP.

De igual manera, el perfeccionamiento de los mecanismos de generación de instancias es otra oportunidad de trabajo futuro que permitiría la simulación más realista de las condiciones de los entornos productivos y así propiciar el mejoramiento de la eficiencia de la heurística en situaciones reales.

Para el proceso de validación, es importante considerar medidas de desempeño como los tiempos de computación. Se hace necesario además la utilización de un mayor número de instancias de prueba con el fin de brindar mayor confiabilidad estadística a los resultados.

Por último, aunque Matlab es una herramienta robusta y muy eficiente en el manejo y cálculo con matrices, puede resultar interesante utilizar otras plataformas de programación. Esto no solamente para mejorar el rendimiento y eficiencia de la heurística, sino también para contar con mejores herramientas en el diseño de una interfaz gráfica que permita una mejor visualización de la información (Resultados, Listas de materiales, análisis de costos etc.)

Referencias

Afentakis, P., Gavish, B., & Karmarkar, U. (1984). Computationally Efficient Optimal Solutions to the Lot-Sizing Problem in Multistage Assembly Systems. *Management Science*, *30*(2), 222–239. http://doi.org/10.1287/mnsc.30.2.222

Akartunalı, K., & Miller, A. J. (2009). A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research*, *193*(2), 396–411. http://doi.org/10.1016/j.ejor.2007.11.033

Bahl, H. C., Ritzman, L. P., & Gupta, J. N. D. (1987). OR Practice--Determining Lot Sizes and Resource Requirements: A Review. *Operations Research*, *35*(3), 329–345. http://doi.org/10.1287/opre.35.3.329

Batista, M. B. M., & Glover, F. (2003). Búsqueda Tabú. *Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial*, *7*(19), 29–48.

Berretta, R., França, P. M., & Armentano, V. A. (2005). Metaheuristic approaches for the multilevel resource-constrained lot-sizing problem with setup and lead times. *Asia-Pacific Journal of Operational Research*, *22*(2), 261–286. http://doi.org/10.1142/S0217595905000510

Billington, P. J., McClain, J. O., & Thomas, L. J. (1983). Mathematical programming approaches to capacity-constrained MRP systems: review, formulation and problem reduction. *Management Science*, *29*(10), 1126–1141.

Billington, P. J., McClain, J. O., & Thomas, L. J. (1986). Heuristics for Multilevel Lot-Sizing with a Bottleneck. *Management Science*, *32*(8), 989–1006. http://doi.org/10.1287/mnsc.32.8.989

Bitran, G. R., & Gilbert, S. M. (1994). Co-Production Processes with Random Yields in the Semiconductor Industry. *Operations Research*, *42*(3), 476–491. http://doi.org/10.1287/opre.42.3.476

Blackburn, J. D., & Millen, R. A. (1982). Improved Heuristics for Multi-Stage Requirements Planning Systems. *Management Science*, *28*(1), 44–56. http://doi.org/10.1287/mnsc.28.1.44

Blackburn, J. D., & Millen, R. A. (1984). Simultaneous lot-sizing and capacity planning in multi-stage assembly processes. *European Journal of Operational Research*, *16*(1), 84–93. http://doi.org/10.1016/0377-2217(84)90315-1

Boonmee, A., & Sethanan, K. (2016). A GLNPSO for multi-level capacitated lot-sizing and scheduling problem in the poultry industry. *European Journal of Operational Research*, *250*(2), 652–665. http://doi.org/10.1016/j.ejor.2015.09.020

Brahimi, N., Dauzere-Peres, S., Najid, N. M., & Nordli, A. (2006). Single item lot sizing problems. *European Journal of Operational Research*, *168*(1), 1–16. http://doi.org/10.1016/j.ejor.2004.01.054

Bravo, D., Rodríguez, E., & Medina, M. (2009). Nisin and lacticin 481 coproduction by Lactococcus lactis strains isolated from raw ewes' milk. *Journal of Dairy Science*, *92*(10), 4805–4811. http://doi.org/10.3168/JDS.2009-2237

Buschkühl, L. (2008). *Multi-level capacitated lotsizing with setup carryover*. Kölner Wiss.-Verl.

Buschkühl, L., Sahling, F., Helber, S., & Tempelmeier, H. (2010). *Dynamic capacitated lot-sizing problems: A classification and review of solution approaches*. *OR Spectrum* (Vol. 32). http://doi.org/10.1007/s00291-008-0150-7

Caserta, M., & Rico, E. Q. (2009). A cross entropy-Lagrangean hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Computers and Operations Research*, *36*(2), 530–548. http://doi.org/10.1016/j.cor.2007.10.014

Cesaret, B., Oğuz, C., & Sibel Salman, F. (2012). A tabu search algorithm for order acceptance and scheduling. *Computers and Operations Research*, *39*(6), 1197–1205. http://doi.org/10.1016/j.cor.2010.09.018

Chelouah, R., & Siarry, P. (2000). Tabu Search applied to global optimization. *European Journal of Operational Research*, *123*(2), 256–270. http://doi.org/10.1016/S0377-2217(99)00255-6

Chen, H. (2015). Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega*, *56*, 25–36. http://doi.org/10.1016/j.omega.2015.03.002

Choudhary, D., & Shankar, R. (2011). Modeling and analysis of single item multi-period procurement lot-sizing problem considering rejections and late deliveries. *Computers & Industrial Engineering*, *61*(4), 1318–1323. http://doi.org/10.1016/j.cie.2011.08.005

Clark, A. R., & Armentano, V. A. (1995). A Heuristic for a Resource-Capacitated Multi-Stage Lot-Sizing Problem with Lead Times. *The Journal of the Operational Research Society*, *46*(10), 1208. http://doi.org/10.2307/2584617

Coronado Hernández, J. R. (2015). *Análisis del efecto de algunos factores de complejidad e*

incertidumbre en el rendimiento de las Cadenas de Suministro. Propuesta de una herramienta de valoración basada en simulación. Universitat Politècnica de València, Valencia (Spain). http://doi.org/10.4995/Thesis/10251/61467

Crowston, W. B., & Wagner, M. H. (1973). Dynamic Lot Size Models for Multi-Stage Assembly Systems. *Management Science*, *20*(1), 14–21. http://doi.org/10.1287/mnsc.20.1.14

Dell'Amico, M., & Trubian, M. (1993). Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, *41*(1–4), 231–252.

Deuermeyer, B. L., & Pierskalla, W. P. (1978). A By-Product Production System with an Alternative. *Management Science*, *24*(13), 1373–1383. http://doi.org/10.1287/mnsc.24.13.1373

Dixon, P., & Silver, E. A. (1981). A heuristic solution procedure for the multi-item, single-level, limited capacity, lot-sizing problem. *Journal of Operations Management*, *2*(1), 23–39. http://doi.org/10.1016/0272-6963(81)90033-4

Drexl, A., & Kimms, A. (1997). Lot sizing and scheduling — Survey and extensions. *European Journal of Operational Research*, *99*(2), 221–235. http://doi.org/10.1016/S0377-2217(97)00030-1

Garcia-Sabater, J. P., Maheut, J., & Marin-Garcia, J. A. (2013). A new formulation technique to model materials and operations planning: the generic materials and operations planning (GMOP) problem. *European J. of Industrial Engineering*, *7*(2), 119. http://doi.org/10.1504/EJIE.2013.052572

Gaury, A. E. G. A., Kleijnen, J. P. C., Pierreval, H., Gaury, E. G. A., Kleijnen, J. P. C., & Pierreval, H. (2016). A Multi-Class Multi-Level Capacitated Lot Sizing Model, *52*(7), 789–

799. Retrieved from http://www.jstor.org/stable/254215

Glover, F. (1989). Tabu Search—Part I. *ORSA Journal on Computing*, *1*(3), 190–206. http://doi.org/10.1287/ijoc.1.3.190

Glover, F. (1990). Tabu Search—Part II. *ORSA Journal on Computing*, *2*(1), 4–32. http://doi.org/10.1287/ijoc.2.1.4

Gopalakrishnan, M., Ding, K., Bourjolly, J.-M., & Mohan, S. (2001). A Tabu-Search Heuristic for the Capacitated Lot-Sizing Problem with Set-up Carryover. *Management Science*, *47*(6), 851–863. http://doi.org/10.1287/mnsc.47.6.851.9813

Harris, F. W. (1913). How Many Parts to Make at Once. *Operations Research*, *38*(6), 947–950. http://doi.org/10.1287/opre.38.6.947

Hindi, K. S. (1995). Solving the single-item, capacitated dynamic lot-sizing problem with startup and reservation costs by tabu search. *Computers and Industrial Engineering*, *28*(4), 701–707. http://doi.org/10.1016/0360-8352(95)00027-X

Hindi, K. S. (1996). Solving the CLSP by a Tabu Search Heuristic. *The Journal of the Operational Research Society*, *47*(1), 151–161. http://doi.org/10.2307/2584259

Hung, Y.-F., & Chien, K.-L. (2000). A Multi-Class Multi-Level Capacitated Lot Sizing Model. *The Journal of the Operational Research Society*, *51*(11), 1309. http://doi.org/10.2307/254215

Hung, Y. F., Chen, C. P., Shih, C. C., & Hung, M. H. (2003). Using tabu search with ranking candidate list to solve production planning problems with setups. *Computers and Industrial Engineering*, *45*(4), 615–634. http://doi.org/10.1016/j.cie.2003.09.006

Kämpf, M., & Köchel, P. (2006). Simulation-based sequencing and lot size optimisation for a production-and-inventory system with multiple items. *International Journal of Production Economics*, *104*(1), 191–200. http://doi.org/10.1016/j.ijpe.2006.02.008

Karimi, B., Fatemi Ghomi, S. M. T., & Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, *31*(5), 365–378. http://doi.org/10.1016/S0305-0483(03)00059-8

Karimi, B., Ghomi, S. M. T. F., & Wilson, J. M. (2006). A Tabu Search Heuristic for Solving the CLSP with Backlogging and Set-up Carry-over. *The Journal of the Operational Research Society*, *57*(2), 140–147. http://doi.org/10.1016/j.ejor.2011.04.029

Kimms, A. (1996). Competitive methods for multi-level lot sizing and scheduling: tabu search and randomized regrets. *International Journal of Production Research*, *34*(8), 2279–2298. http://doi.org/10.1080/00207549608905025

Kimms, A. (1999). A genetic algorithm for multi-level, multi-machine lot sizing and scheduling. *Computers and Operations Research*, *26*(8), 829–848. http://doi.org/10.1016/S0305-0548(98)00089-6

Kirca, O. (1990). An Efficient Algorithm for the Capacitated Single Item Dynamic Lot Sizing Problem. *European Journal of Operational Research*, *45*, 15–74. http://doi.org/10.1016/0377-2217(90)90152-2

Kuik, R., Salomon, M., & van Wassenhove, L. N. (1994). Batching decisions: structure and models. *European Journal of Operational Research*, *75*(2), 243–263. http://doi.org/10.1016/0377-2217(94)90072-8

Kuik, R., Salomon, M., Van Wassenhove, L. N., & Maes, J. (1993). Linear Programming,

Simulated Annealing and Tabu Search Heuristics for Lotsizing in Bottleneck Assembly Systems. *IIE Transactions*, *25*(1), 62–72. http://doi.org/10.1080/07408179308964266

Kurbel, K. E. (2013). MRP: Material Requirements Planning (pp. 19–60). http://doi.org/10.1007/978-3-642-31573-2_2

Li, X., Baki, F., Tian, P., & Chaouch, B. A. (2014a). A robust block-chain based tabu search algorithm for the dynamic lot sizing problem with product returns and remanufacturing. *Omega*, *42*(1), 75–87. http://doi.org/10.1016/j.omega.2013.03.003

Li, X., Baki, F., Tian, P., & Chaouch, B. A. (2014b). A robust block-chain based tabu search algorithm for the dynamic lot sizing problem with product returns and remanufacturing, *42*(1), 75–87. http://doi.org/10.1016/j.omega.2013.03.003

Maes, J., & Wassenhove, L. Van. (1988). Multi-Item Single-Level Capacitated Dynamic Lot-Sizing Heuristics: A General Review. *The Journal of the Operational Research Society*, *39*(11), 991–1004. http://doi.org/10.2307/2583198

Maheut, J. (2013, May 13). *Modelos y Algoritmos Basados en el Concepto Stroke para la Planificación y Programación de Operaciones con Alternativas en Redes de Suministro*. Universitat Politècnica de València, Valencia (Spain). http://doi.org/10.4995/Thesis/10251/29290

Maheut, J., & Garcia-Sabater, J. P. (2011). La matriz de operaciones y materiales y la matriz de operaciones y recursos, un nuevo enfoque para resolver el problema GMOP basado en el concepto del stroke. *Direccion Y Organizacion*, *45*, 46–57.

Maheut, J., Garcia-Sabater, J. P., Garcia-Sabater, J. J., & Valero Herrero, M. (2011). El Stroke y la Matriz de Operaciones y Materiales, nuevo enfoque para resolver el problema GMOP. In

5th International Conference on Industrial Engineering and Industrial Management (pp. 884–893). Cartagena. Retrieved from http://adingor.es/congresos/web/uploads/cio/cio2011/metodos_cuantitativos/884-893.pdf

Nowicki, E., & Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research*, *106*(2–3), 226–253. http://doi.org/10.1016/S0377-2217(97)00260-9

Oliva San Martín, C. D., & Ramírez Guzmán, G. M. (2015). Algoritmo de tipo búsqueda tabú para un problema de programación de horarios universitarios vespertinos. *INGE CUC*, *9*(2), 58–65. Retrieved from http://revistascientificas.cuc.edu.co/index.php/ingecuc/article/view/7

Öner, S., & Bilgiç, T. (2008). Economic lot scheduling with uncontrolled co-production. *European Journal of Operational Research*, *188*(3), 793–810. http://doi.org/10.1016/j.ejor.2007.05.016

Orlicky, J. (1975). *Material Requirements Planning*. New York, New York, USA: McGraw-Hill.

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. http://doi.org/10.1016/j.cie.2015.02.014

Quadt, D., & Kuhn, H. (2008). Capacitated lot-sizing with extensions: A review. *4or*, *6*(1), 61–83. http://doi.org/10.1007/s10288-007-0057-1

Rius Sorolla, G., Maheut, J., Coronado-Hernandez, J., & Garcia-Sabater, J. P. (2017). Lagrangian relaxation of the GMOP model. In *11th International Conference on Industrial Engineering and Industrial Management*. Valencia (Spain). Retrieved from http://www.cigip.upv.es/cio2017/wp-content/uploads/2017/07/CIO17_Full_Programme.pdf

Roca Molina, A. (2016). *Construcción de algoritmo aplicando relajación lagrangeana para la obtención de un límite inferior para el problema de lotificación en sistemas multinivel en entornos de coproducción y listas de materiales alternativas*. Universidad Tecnológica de Bolívar.

Romero-Conrado, A. R., Suárez-Agudelo, E. A., Macías-Jiménez, M. A., Gómez Charris, Y., & Lozano-Ayarza, L. P. (2017). Experimental design for obtaining compost suitable for agricultural use from Kraft paper sludge. Espacios, 38(28).

Sambasivan, M., & Yahya, S. (2005). A Lagrangean-based heuristic for multi-plant, multi-item, multi-period capacitated lot-sizing problems with inter-plant transfers. *Computers & Operations Research*, *32*(3), 537–555. http://doi.org/10.1016/j.cor.2003.08.002

SAP. (2017). Listas de materiales (PP-BD-BOM). Retrieved from http://help.sap.com/saphelp_470/helpdata/es/ea/e9bcc04c7211d189520000e829fbbd/frameset.htm

Sifaleras, A., & Konstantaras, I. (2015). Variable neighborhood descent heuristic for solving reverse logistics multi-item dynamic lot-sizing problems. *Computers & Operations Research*, 1–8. http://doi.org/10.1016/j.cor.2015.10.004

Steinberg, E., & Napier, H. A. (1980). Optimal Multi-Level Lot Sizing for Requirements Planning Systems. *Management Science*, *26*(12), 1258–1271. http://doi.org/10.1287/mnsc.26.12.1258

Toledo, C. F. M., de Oliveira, R. R. R., & Morelato França, P. (2013). A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. *Computers and Operations Research*, *40*(4), 910–919.

http://doi.org/10.1016/j.cor.2012.11.002

Vazsonyi, A. (1954). The Use of Mathematics in Production and Inventory Control. I. *Management Science*, *1*(1), 70–85. Retrieved from http://www.jstor.org.consultaremota.upb.edu.co/stable/2627076

Vidal-Carreras, P. I., Garcia-Sabater, J. P., & Coronado-Hernandez, J. R. (2012). Economic lot scheduling with deliberated and controlled coproduction. *European Journal of Operational Research*, *219*(2), 396–404. http://doi.org/10.1016/J.EJOR.2011.12.020

Wery, J., Gaudreault, J., Thomas, A., & Marier, P. (2018). Simulation-optimisation based framework for Sales and Operations Planning taking into account new products opportunities in a co-production context. *Computers in Industry*, *94*, 41–51. http://doi.org/10.1016/J.COMPIND.2017.10.002

Anexos

Anexo 1. Código para la generación de las instancias de prueba.

Anexo 2. Código para exportación de instancias de prueba Matlab a Excel.

Anexo 3. Código para importación de instancias de prueba Matlab.

Anexo 4. Código programado en GAMS para solución del modelo GMOP.

Anexo 5. Algoritmo heurístico basado en listas tabú.

Anexo 6. Subrutina de evaluación de soluciones vecinas.

Anexo 7. Subrutina MRP

Anexo 8. Ejemplo de interfaz gráfica y visualización de resultados

Con el fin de facilitar la reproducibilidad del trabajo presentado, las instancias de pruebas utilizadas y algoritmos programados están disponibles para descarga en el enlace:

http://drive.google.com/open?id=19JjIGgX7E7Q-YSDMn2uZdHZKVqjvY20v

ANEXO 1.

Algoritmo para la generación de las instancias de prueba

```
%Heuristica Basada en Listas Tabú

%%%%%%%%% GENERACION DE INSTACIA ALEATORIA  %%%%%%%%%%
%%
%%Numero de SKUs
productmax=50;
%%
%Numero de Productos Finales
finalproducts=15 ;
%%
%Numero de Periodos de Planificacion
tmax=25;
%%
%Numero de Recursos
r=10;
%%
%Numero de Strokes
k=100;
kmax=k;
%%
%Demanda
minimo=1000;
maximo=2000;
demand(productmax,tmax)=zeros();
X(finalproducts,1)=zeros();
a=0;
while a<finalproducts
    productofinal= round(1 + (productmax-1).*rand());
    W= find(X==productofinal);
    Y= isempty(W);
        if Y==1
            a=a+1;
            pedidos= round((tmax/2) + (tmax-(tmax/2).*rand()));
            for b=1:pedidos
            periodo= round(7 + (tmax-7).*rand());
            demand(productofinal,periodo)= round(minimo + (maximo -
minimo).*rand());
            end
            X(a,1)= productofinal;
            else
            end
end

%%
%Costo de Almacenamiento
minimo=10;
maximo=20;
storageCost(productmax,tmax)= zeros();
for a=1:productmax
```

```
storageCost(a,:) = round(minimo + (maximo - minimo).*rand());
end

%%
%Costo de Operacion del Stroke
strokeCost(kmax,tmax)= zeros();
minimo=5;
maximo=8;
for a=1:kmax
strokeCost(a,:) = round(minimo + (maximo - minimo).*rand());
end

%Costo de Faltantes
backlogCost(productmax,tmax)=zeros();
minimo=1;
maximo=10;
randcost=round(minimo + (maximo - minimo).*rand());
backlogCost(:,:)=randcost;


%%
%Costo de Setup del Stroke
strokeSetup(k,tmax)= zeros();
minimo=5;
maximo=10;

for a=1:kmax
strokeSetup(a,:) = round(minimo + (maximo - minimo).*rand());
end


%%
%Salidas Strokes
minimo=35;
maximo=50;
strokeOut(productmax,k)= zeros();
for i=1:kmax
    %Cuantos productos de salida tiene el stroke i
    productin= round(1 + (5 - 1).*rand());
    for j=1: productin
        randp= round(1 + (productmax - 1).*rand());
        strokeOut(randp,i)= round(minimo + (maximo - minimo).*rand());
    end
end

%%
%Entradas Strokes
minimo=4;
maximo=8;
strokeIn(productmax,k)=zeros();
for i=1:k
    %Cuantos productos entran al stroke
    prod= ceil(1 + (5 - 1).*rand());
    for j=1:prod
        randp= round(1 + (productmax - 1).*rand());
```

```
        strokeIn(randp,i)= floor(minimo + (maximo - minimo).*rand());
    end
end

%%
%Se asegura que cada Stroke tenga por lo menos una entrada.

for i=1:kmax
    if sum(strokeIn(:,i))== 0
    minimo=4;
    maximo=8;
    randp= ceil(1 + (productmax- 1).*rand());
    strokeIn(randp,i)= round(minimo + (maximo - minimo).*rand());
    end
end
%%
%Se impide que los productos finales sean SKU entrantes.
for i=1:finalproducts
    m= X(i);
  strokeIn(m,:)=0;
 end
%%
%Se impide que los SKU que entren a un Stroke sean los mismos que salgan

h= [strokeIn]==0;
strokeOut = strokeOut.*h;
%%
%Se asegura que cada SKU tenga por lo menos un stroke asignado.

for i=1:productmax
    if sum(strokeOut(i,:))== 0
    minimo=35;
    maximo=50;
    randk= ceil(1 + (kmax- 1).*rand());
    strokeOut(i,randk)= round(minimo + (maximo - minimo).*rand());
    end
end


%%
%LeadTime Strokes
minimo=1;
maximo=2;
strokeLeadTime= round(minimo + (maximo - minimo).*rand(k,1));
%%
%Capacidad de los Recursos
minimo=12000;
maximo=14000;
resourceCapacity(r,tmax)=zeros();
for i=1:r
    resourceCapacity(i,:) = round(minimo + (maximo - minimo).*rand());
end
%%
%Capacidad requerida por los strokes
minimo=2;
maximo=5;
```

```
requiredCapacity(k,r) = zeros();
for i=1:kmax
    %Cuantos recursos usa el stroke
    Q(r)=zeros;
    a=0;
    ResourcesToUse=1; %round(1 + (r - 1).*rand());
    while a < ResourcesToUse
        ResourceUsed= round(0 + (r - 0).*rand());
        W= find(Q==ResourceUsed);
        Y= isempty(W);
        if Y==1
          a=a+1;
          Q(a)= ResourceUsed;
          requiredCapacity(i,ResourceUsed) = round(minimo + (maximo -
minimo).*rand());
        else

        end

    end

end

%%
%Tiempos de Setup de los Strokes
minimo=5;
maximo=10;
requiredSetup = floor(minimo + (maximo - minimo).*rand(k,r));
%%
M=5000;
%%
%Estructura MRP
 IINIC(productmax,1)= zeros(); %INVENTARIO INICIAL (i)
 RC(productmax,tmax)= zeros();%REQUERIMIENTOS EN CONJUNTO RC(i,t)
 RPROG(productmax,tmax)= zeros(); %RECEPCIONES PROGRAMADAS OP(i,t)
 BINV(productmax,tmax)= zeros(); %BALANCE DE INVENTARIO BINV(i,t)
 REQNETOS(productmax,tmax)= zeros(); %REQUERIMIENTOS NETOS  REQNETOS(i,t)
 RPLAN(productmax,tmax)= zeros();%RECEPCIONES PLANEADAS   RECP(i,t)
 LIBORD(productmax,tmax)= zeros();%LIBERACIÓN DE ORDENES  LIBORD(i,t)
 %%
 %Consumo de Capacidad

 consumo(r,tmax)= zeros();

  %Solucion Inicial

 SolStrokes(k, tmax)= zeros();
 Strokes(k, tmax)= zeros();
 %%
%Listado de Productos con demanda Independiente.
%ReqSKU(SKU) identifica los SKU finales que tienen demanda.
ReqSKU(productmax,1) = zeros();

for i=1:productmax
    for j=1:tmax
```

```
        if  demand(i,j)>0
            ReqSKU(i,1)= 1;
            continue
        else
        end
    end
end
kc=1;
SKUCounter=0;
%%
%Crea el listado de los SKU finales que se demandan.
for i=1:productmax
    if ReqSKU(i,1)>0
        ListadoSKU(kc,1)=i;
        SKUCounter=SKUCounter+1;
        kc=kc+1;
    end
end
%%
%Creación del listado de Entradas para cada Stroke
%StrokeInputs(Stroke,SKU) identifica qué SKUs entran a cada Stroke
StrokeInputs(kmax,productmax)= zeros();
kmaxc=1;
MaxInputs=0;
for j=1:kmax
    kc=1;
    for i=1:productmax
        if strokeIn(i,j)>0
           StrokeInputs(j,kc)= i;
           if kc>kmaxc
               kmaxc=kc;
               MaxInputs=kmaxc;
           end
            kc=kc+1;
        else
        end
    end
end
%%

%Creación del listado de Salidas para cada Stroke
%StrokeOutputs(Stroke,SKU) identifica qué SKUs entran a cada Stroke
StrokeOutputs(kmax,productmax)= zeros();
kmaxc=1;
for j=1:kmax
    kc=1;
    for i=1:productmax
        if strokeOut(i,j)>0
           StrokeOutputs(j,kc)= i;
           if kc>kmaxc
               kmaxc=kc;
               MaxOutputs=kmaxc;
           end
            kc=kc+1;
        else
        end
    end
```

```
end

%%
% Identifica las alternativas de Stroke para cada SKU
Alt(productmax,kmax)= zeros();
kmaxc=0;
MaxAlt(productmax,1)=zeros();
for i=1:productmax
    if kc>kmaxc
        kmaxc=kc-1;
    end
    kc=1;
    for j=1:k
        if strokeOut(i,j)>0
            Alt(i,kc)= j;
            MaxAlt(i,1)=kc;
            MaxMaxAlt= max(MaxAlt);
            kc=kc+1;

        else
        end
    end
end
```

ANEXO 2.

Código para exportación de instancias de prueba Matlab a Excel.

```
%ExportDatasetToExcel

filename='I3-SKU50-P15-T25-R10-K200.xlsx';
xlswrite(filename,productmax,'DatasetDetails','B2');
xlswrite(filename,finalproducts,'DatasetDetails','B3');
xlswrite(filename,tmax,'DatasetDetails','B4');
xlswrite(filename,r,'DatasetDetails','B5');
xlswrite(filename,kmax,'DatasetDetails','B6');

xlswrite(filename,demand,'Demanda','B2');
xlswrite(filename,strokeOut,'SalidasStroke','B2');
xlswrite(filename,strokeIn,'EntradasStroke','B2');
xlswrite(filename,storageCost,'CostoAlmacenaje','B2');
xlswrite(filename,backlogCost,'CostoFaltante','B2');
xlswrite(filename,strokeCost,'CostoStroke','B2');
xlswrite(filename,strokeSetup,'CostoSetupStroke','B2');
xlswrite(filename,strokeLeadTime,'LeadTimeStroke','B1');
xlswrite(filename,resourceCapacity,'CapacidadRecursos','B2');
xlswrite(filename,requiredCapacity,'CapacidadRequerida','B2');
xlswrite(filename,requiredSetup,'SetupRequerido','B2');
xlswrite(filename,IINIC,'InventarioInicial','B1');
xlswrite(filename,RPROG,'RecepcionesProgramadas','B2');
```

ANEXO 3.

Código para importación de instancias de prueba Matlab.

```
%Heuristica basada en listas tabú - Importacion de instancias

%%%%%%%%% CARGA DE DATOS DEL PROBLEMA  %%%%%%%%%

filename='I7-SKU200-P60-T75-R30-K50.xlsx';
productmax= xlsread(filename,'DatasetDetails','B2');
finalproducts= xlsread(filename,'DatasetDetails','B3');
tmax= xlsread(filename,'DatasetDetails','B4');
r= xlsread(filename,'DatasetDetails','B5');
k= xlsread(filename,'DatasetDetails','B6');
kmax= xlsread(filename,'DatasetDetails','B6');

%Demanda
demand = xlsread(filename,'Demanda');

%Costo de Almacenamiento
storageCost = xlsread(filename,'CostoAlmacenaje');

%Costo de Faltante
backlogCost = xlsread(filename,'CostoFaltante');

%Entradas Strokes
strokeIn = xlsread(filename,'EntradasStroke');

%Salidas Strokes
strokeOut = xlsread(filename,'SalidasStroke');

%Costo Strokes
strokeCost = xlsread(filename,'CostoStroke');

%LeadTime Strokes
strokeLeadTime= xlsread(filename,'LeadTimeStroke');

%Costo Setup Strokes
strokeSetup= xlsread(filename,'CostoSetupStroke');

%Capacidad de los Recursos
resourceCapacity = xlsread(filename,'CapacidadRecursos');

%Capacidad requerida por los strokes
requiredCapacity = xlsread(filename,'CapacidadRequerida');

%Tiempos de Setup de los Strokes
requiredSetup = xlsread(filename,'SetupRequerido');
```

```
%Estructura MRP
%INVENTARIO INICIAL (i,0)
IINIC= xlsread(filename,'InventarioInicial');
%RECEPCIONES PROGRAMADAS OP(i,t)
RPROG= xlsread(filename,'RecepcionesProgramadas');;

RC(productmax,tmax)= zeros();%REQUERIMIENTOS EN CONJUNTO RC(i,t)

BINV(productmax,tmax)= zeros(); %BALANCE DE INVENTARIO BINV(i,t)
 REQNETOS(productmax,tmax)= zeros(); %REQUERIMIENTOS NETOS  REQNETOS(i,t)
 RPLAN(productmax,tmax)= zeros();%RECEPCIONES PLANEADAS   RECP(i,t)
 LIBORD(productmax,tmax)= zeros();%LIBERACIÓN DE ORDENES  LIBORD(i,t)

 %Consumo de Capacidad

 consumo(r,tmax)= zeros();


 %Solucion Inicial
 SolStrokes(k, tmax)= zeros();
 Strokes(k, tmax)= zeros();
%Listado de Productos con demanda Independiente.
%ReqSKU(SKU) identifica los SKU finales que tienen demanda.
ReqSKU(productmax,1) = zeros();

for i=1:productmax
    for j=1:tmax
        if  demand(i,j)>0
            ReqSKU(i,1)= 1;
            continue
        else
        end
    end
end
kc=1;
SKUCounter=0;
%Crea el listado de los SKU finales que se demandan.
for i=1:productmax
    if ReqSKU(i,1)>0
        ListadoSKU(kc,1)=i;
        SKUCounter=SKUCounter+1;
        kc=kc+1;
    end
end

%Creación del listado de Entradas para cada Stroke
%StrokeInputs(Stroke,SKU) identifica qué SKUs entran a cada Stroke
StrokeInputs(productmax,kmax)= zeros();
kmaxc=1;
for j=1:kmax
    kc=1;
    for i=1:productmax
        if strokeIn(i,j)>0
           StrokeInputs(j,kc)= i;
```

```
            if kc>kmaxc
                kmaxc=kc;
                MaxInputs=kmaxc;
            end
             kc=kc+1;
        else
        end
    end
end

%%

%Creación del listado de Salidas para cada Stroke
%StrokeOutputs(Stroke,SKU) identifica qué SKUs entran a cada Stroke
StrokeOutputs(kmax,productmax)= zeros();
kmaxc=1;
for j=1:kmax
    kc=1;
    for i=1:productmax
        if strokeOut(i,j)>0
           StrokeOutputs(j,kc)= i;
           if kc>kmaxc
               kmaxc=kc;
               MaxOutputs=kmaxc;
           end
            kc=kc+1;
        else
        end
    end
end

%%
% Identifica las alternativas de Stroke para cada SKU
Alt(productmax,kmax)= zeros();
kmaxc=0;
for i=1:productmax
    if kc>kmaxc
        kmaxc=kc-1;
    end
    kc=1;
    for j=1:k
        if strokeOut(i,j)>0
            Alt(i,kc)= j;
            MaxAlt(i,1)=kc;
            MaxMaxAlt= max(MaxAlt);
            kc=kc+1;

        else
        end
    end
end
```

ANEXO 4

Anexo 4. Código programado en GAMS para solución del modelo GMOP.

```
$ontext Modelo matematico del GMOP
$offtext
********* Declaracion de conjuntos *******
set
    i(*) Referencias SKU
    r(*) Recursos
    k(*) Strokes
    t(*) Horizonte de planificacion ;
$load i r k t
display i, r, k, t;
Parameter
 LT(k) Lead time
 D(i,t) Demanda del componente i en el tiempo t
 CO(k,t) Costo del stroke
 CSU(k,t) Costo set up
 CH(i,t) Costo de inventario
 CA(i,t) Costo de faltante
 M(i,k) Unidades del componente i requerido en el stroke k
 N(i,k) Unidades generadas del componente i requerido en el stroke k
 RE(k,r) Tiempo de ciclo de k en r
 TS(k,r) Tiempo set up k en r
 CAP(r,t) Capacidad del recurso r en t
 RPL(i,t) Recepciones programadas
 INI(i) Inventario inicial;
```

```
$loaddc LT D CO CSU CH CA M N RE TS CAP RPL INI
display LT, D, CO, CSU, CH, CA, M, N, RE, TS, CAP, RPL, INI;
$gdxin
*******  Declaracion de variables *******
variable
z(k,t) numeros de stroke k para ser desarrollados en el periodo t
y(i,t) invenario final del producto i al final del periodo t
w(i,t) faltante al final del perido t del sku i
x(i,k,t) numero de sku i generados por el stroke k en el periodo t
con(i,k,t) consumos del stroke k del sku i en el perido t
consumos(i,t) consumos del sku i en el periodo t
produccion(i,t) produccion del sku i en el periodo t
delta(k,t) 1 si el stroke k es desarrollado y 0 lo contrario
CostoDelPlan Funcion objetiva ;
positive variable y, w, x, con, consumos, produccion;
integer variable z;
binary variable delta;
************** Ecuaciones **************
equations
FUNOBJE
ConsumosUnit2(i,k,t)
cosumB(i,t)
GeneraUnit2b(i,k,t)
GeneraUnit2(i,k,t)
GeneraB(i,t)
inv1(i)
inv(i,t)
Recursos(r,t)
bigm(k,t);
****Funcion objetivo****
```

```
FUNOBJE.. CostoDelPlan =e= sum((i,t), CH(i,t)*y(i,t)+CA(i,t)*w(i,t)) +
sum((k, t), CO(k,t)*z(k,t) + CSU(k,t)*delta(k,t));

*Consumos de componentes por Stroke

ConsumosUnit2(i,k,t)..  con(i,k,t)=e=M(i,k)*z(k,t);

*Consumo total de cada componente

cosumB(i,t).. consumos(i,t)=e=sum(k, con(i,k,t));

*Generacion de componentes por stroke

GeneraUnit2b(i,k,t)$(ord(t) lt LT(k)).. x(i,k,t)=e=0;

*GeneraUnit2b(i,k,t)$(ord(t) lt LT(k)).. x(i,k,t)$(LT(k)>0)=e=0;

GeneraUnit2(i,k,t)$(ord(t) ge LT(k)-1).. x(i,k,t)=e=N(i,k)*z(k, t-
LT(k));

*Generacion total de cada componente

GeneraB(i,t).. produccion(i,t)=e=sum(k, x(i,k,t));

*Ecuacion de continuidad de inventario

inv1(i).. y(i,'T1')=e=ini(i)+RPL(i,'T1')-
consumos(i,'T1')+produccion(i,'T1')+w(i,'T1')-D(i, 'T1');

inv(i,t)$(ord(t) ge 2).. y(i,t)=e=y(i,t-1)+RPL(i,t)-
consumos(i,t)+produccion(i,t)+w(i,t)-w(i,t-1)-D(i, t);

*Capacidad de produccion

Recursos(r,t).. sum(k, RE(k,r)*z(k,t) + TS(k,r)*delta(k,t)) =l=
CAP(r,t);

bigm(k,t).. z(k,t)=l= 900000000*delta(k,t);

model gmop /FUNOBJE, ConsumosUnit2, cosumB, GeneraUnit2b, GeneraUnit2,
GeneraB, inv1, inv, Recursos, bigm/;

option optcr=0;

solve gmop using mip minimizing CostoDelPlan;

display z.l;

display delta.l;

display produccion.l;
```

ANEXO 5

Algoritmo heurístico basado en listas tabú.

```
%Hueuristica Basada en Listas Tabu%

%Inicializacion
t= 1;
TabuTenure=38; %Tenencia Tabu
TabuTmax=75; %Número Maximo de Iteraciones
VecNum=15; %Numero de vecinos encontrados por iteracion
Z0= Z;%Valor Objetivo Inicial
BestSol=Z0; %Mejor Solución = Solucion Inicial
S0= Alt(:,1); %Vector Solución Solución Inicial
S=S0; %Solución Actual
BestSolStrokes(kmax,tmax)=zeros();
BestBINV(productmax,tmax)=zeros();
figure(1);
H=Z;
plot(H);
xlim([1 TabuTmax]);
ylim([1 Z]);
drawnow;

V(productmax,VecNum)= zeros(); %Conjunto de Soluciones Vecinas
RegVec(VecNum,6,TabuTmax)= zeros();
CapacityFit=zeros();
RegBINV(productmax,tmax,(TabuTmax*VecNum))= zeros();
RegSolStrokes(kmax,tmax,(TabuTmax*VecNum))= zeros();
TabuList(kmax,kmax,productmax)= zeros(); %TabuList
RegTime(TabuTmax,2)=zeros();

%Se configura la diagonal de la lista tabu para impedir que se realicen
%movimientos nulos con el mismo stroke.
for p=1:productmax
    for i=1:kmax
        TabuList(i,i,p)= 9999999;
    end
end

registro=1;

while t <= TabuTmax

    %Se crea el conjunto de soluciones vecinas V().
    vecinos=1;

    while vecinos <= VecNum
    %Se selecciona un producto al azar
```

```
minimo=1;
maximo=productmax;
prod = round(minimo + (maximo - minimo).*rand());
%Se escoge una de las soluciones vecinas
if MaxAlt(prod)>=2
    vec= round(1 + (MaxAlt(prod) - 1).*rand());
    %Se verifica si el movimiento es Tabu.
              if TabuList((Alt(prod,vec)),(S(prod)),prod)==0 %Si el
        movimiento no esta en la lista tabu.
            %Se incluye en el conjunto de soluciones vecinas.
            %Se registra la informacion de la solución vecina incluida
               RegVec(vecinos,1,t)= prod;
               V(:,vecinos)= S;
               RegVec(vecinos,2,t)= S(prod) ;
               RegVec(vecinos,3,t)= Alt(prod,vec);
               V(prod,vecinos)= Alt(prod,vec);
               AltVec(:,1)= V(:,vecinos);
               RestCap=0;
               EvaluacionVecindario
               RegVec(vecinos,4,t)= ZVec;
               RegVec(vecinos,5,t)=RestCap;
        if RestCap >0
        RegVec(vecinos,4,t)=ZVec+(RestCap*100000)
        end
               RegBINV(:,:,registro)= BINV;
               RegSolStrokes(:,:,registro)= SolStrokes;
               RegVec(vecinos,6,t)= registro;
               registro=registro+1;
               vecinos=vecinos+1;
        else

        end
else

end

end

 %Se selecciona la mejor solución del conjunto V
 BestV= min(RegVec(:,4,t));

 if BestV<BestSol
     H(t)=BestSol;
     plot(H);
     xlim([1 TabuTmax]);
     ylim([1 Z]);
     drawnow;
 RegTime(t,1)=round(cputime);
 RegTime(t,2)=BestSol;
 [m,i]= min(RegVec(:,4,t));
 CapacityFit= RegVec(i,5,t);
 BestSolStrokes= RegSolStrokes(:,:,RegVec(i,6,t));
 BestBINV= RegBINV(:,:,RegVec(i,6,t));
 %Se actualiza la mejor Solución hasta el momento
 BestSol=BestV;
 %BestSolStrokes=RegSolStrokes(:,:,)
```

```
    S=V(:,i);
    %Se actualiza la lista tabu
    X= (TabuList(:,:,:)>0)*-1;
    TabuList(:,:,:) = TabuList(:,:,:)+X;
    %Se agrega el ultimo movimiento realizado a la lista tabu
    TabuList(RegVec(i,2),RegVec(i,3),RegVec(i,1))=TabuTenure;
    TabuList(RegVec(i,3),RegVec(i,2),RegVec(i,1))=TabuTenure;
    t=t+1;
    else
     %Se actualiza la lista tabu
     H(t)=BestSol;
        plot(H);
        xlim([1 TabuTmax]);
        ylim([1 Z]);
        drawnow;

     X= (TabuList(:,:,:)>0)*-1;
     TabuList(:,:,:) = TabuList(:,:,:)+X;
     t=t+1;
     end

end

CapacityFit
BestSolStrokes
S
BestSol
```

ANEXO 6.

Subrutina de Evaluación de Soluciones Vecinas

```
%Evaluacion de soluciones vecinas

 RC(:,:)= zeros();
 BINV(:,:)= zeros();
 REQNETOS(:,:)= zeros();
 RPLAN(:,:)= zeros();
 LIBORD(:,:)= zeros();
 CostoTotalInventario=0;
 CostoTotalSetup=0;
 CostoTotalOperacion=0;
 consumo(:,:)=zeros();
 SolStrokes(:,:)=zeros();

 for i=1:finalproducts
    p= ListadoSKU(i,1);
    RC(p,:)= demand(p,:);
   end

 %////////////////////////////////////////////%
 %///////////      NIVEL 1          ///////////%
 %////////////////////////////////////////////%
 %Se realiza el MRP inicial para los productos finales
 for j=1:tmax
     for i=1:finalproducts
         p= ListadoSKU(i,1);
         %display(p)
         z=p;
         TabuMRP;


         %Se realiza el MRP para cada producto final. Se define de manera
         %predeterminada el primer Stroke Alternativo (s).

         %Se realiza el MRP para los componentes del primer stroke
         if smrp>0
             s=AltVec(p,1);
                %display (s)
                for x=1:MaxInputs
                    y=StrokeInputs(s,x);
                    if y>0
                    %display (y)
                        if j>lt
                            RC(y,j-lt)= ceil(RC(y,j-lt)+
((ceil(RPLAN(p,j)/strokeOut(p,s)))* strokeIn(y,s)));
                            z=y;
                            TabuMRPComponentes;
                        end
                    end
                end
         end
      end
```

```
end
%disp ('NIVEL 2')

%////////////////////////////////////////////%
%///////////      NIVEL 2          ///////////%
%////////////////////////////////////////////%
for j=1:tmax
    for i=1:finalproducts
        p= ListadoSKU(i,1);
        %display(p)
        smrp=AltVec(p,1);

        if smrp>0
            s=AltVec(p,1);
               %display (s)
               for x=1:MaxInputs
                   y=StrokeInputs(s,x);
                   if y>0
                   %display (y)
                   s2=AltVec(y,1);
                   if s2>0
                       for x2=1:MaxInputs
                       y2=StrokeInputs(s2,x2);
                           if y2>0
                               %display (y2)
                               if j>lt
                                   RC(y2,j-lt)= ceil(RC(y2,j-lt)+
((ceil(RPLAN(y,j)/strokeOut(y,s2)))* strokeIn(y2,s2)));
                                   z=y2;
                                   TabuMRPComponentes;
                               end
                           else
                               continue
                           end
                       end
                   end

                   end
               end
        end
     end
end

%disp ('NIVEL 3')

%////////////////////////////////////////////%
%///////////      NIVEL 3          ///////////%
%////////////////////////////////////////////%
for j=1:tmax
    for i=1:finalproducts
        p= ListadoSKU(i,1);
        %display(p)
        smrp=AltVec(p,1);

        if smrp>0
            s=AltVec(p,1);
```

```
                %display (s)
                for x=1:MaxInputs
                    y=StrokeInputs(s,x);
                    if y>0
                    %display (y)
                    s2=AltVec(y,1);
                    if s2>0
                        for x2=1:MaxInputs
                        y2=StrokeInputs(s2,x2);
                            if y2>0
                                %display (y2)
                                s3=AltVec(y2,1);
                                    if s3>0
                                        for x3=1:MaxInputs
                                        y3=StrokeInputs(s3,x3);
                                            if y3>0
                                                %display (y3)
                                                if j>lt
                                                    RC(y3,j-lt)=
ceil(RC(y3,j-lt)+ ((ceil(RPLAN(y2,j)/strokeOut(y2,s3)))* strokeIn(y3,s3)));
                                                    z=y3;
                                                    TabuMRPComponentes;
                                                end
                                            else
                                                continue
                                            end
                                        end
                                    end
                            else
                                continue
                            end
                        end
                    end

                    end
                end
            end
        end
    end

    %Setups
    CostoTotalSetup=0;
    for i=1:kmax
        for j=1:tmax
            if SolStrokes(i,j)>0
            CostoTotalSetup = CostoTotalSetup + strokeSetup(i,j);
                for w=1:r
                   if requiredCapacity(i,r)>0
                      consumo(w,j)= consumo(w,j) + requiredSetup(i,w);
                   else
                       continue
                   end
                end
            end
        end
    end
```

```
%Costo de Inventario
CostoTotalInventario= sum(sum(BINV.*storageCost))


%Costo de Operacion
 CostoTotalOperacion= sum(sum(SolStrokes.*strokeCost))

 %Costo Total de la Solucion

 ZVec =  CostoTotalInventario + CostoTotalSetup + CostoTotalOperacion
```

ANEXO 7.

Subrutina MRP

```
%REQUERIMIENTOS EN CONJUNTO RC(i,t)
%RECEPCIONES PROGRAMADAS OP(i,t)
%BALANCE DE INVENTARIO BINV(i,t)
%REQUERIMIENTOS NETOS REQNETOS(i,t)
%RECEPCIONES PLANEADAS RPLKL(i,t)
%LIBERACIÓN DE ORDENES LIBORD(i,t)


if z>0 %Si no hay productos entrantes al stroke (stroke de compra) finalice.
    smrp = Alt(z,1);
    if smrp>0
        lt= strokeLeadTime(smrp,1);

        if j==1

            BINV(z,j)= BINV(z,j)+ IINIC(z,1)+ ceil(RPROG(z,1))-
ceil(RC(z,1)) ;
            if BINV(z,j)<0
               BINV(z,j)=0;
            end
            REQNETOS(z,j)= ceil(RC(z,j))- (IINIC(z,j)+ RPROG(z,j));
            if REQNETOS(z,j)<0
               REQNETOS(z,j)=0;
            else
            RPLAN(z,j)=(ceil(REQNETOS(z,j)/strokeOut(z,smrp))*
strokeOut(z,smrp));
            BINV(z,j)= BINV(z,j)+ IINIC(z,1)+  ceil(RPLAN(z,1))+RPROG(z,1)-
ceil( RC(z,j)) ;
            if j>lt
               LIBORD(z,j-lt)= LIBORD(z,j-lt) + ceil(RPLAN(z,j));
               Strokes(smrp,j-lt)= ceil(REQNETOS(z,j)/strokeOut(z,smrp));
               SolStrokes(smrp,j-lt)= SolStrokes(smrp,j-lt)+
ceil(REQNETOS(z,j)/strokeOut(z,smrp));
               %Se registran los SKU coproducidos
               for i=1:MaxOutputs
                   output=StrokeOutputs(smrp,i)
                   if output >0
                       if output==z
                          continue
                       else
                       BINV(output,j)= BINV(output,j)+ Strokes(smrp,j-
lt)*strokeOut(output,smrp);
                       end
                   end
               end
               %Se registran los SKU consumidos
               for i=1:MaxInputs
                   input=StrokeInputs(smrp,i)
                   if input >0
                       BINV(input,j)= BINV(input,j)+
Strokes(smrp,j)*strokeOut(input,smrp);
                   end
```

```
                end
                %Se registra la capacidad de los recursos utilizada
                for i=1:r
                     if requiredCapacity(smrp,i)>0
                        consumo(i,j-lt)= consumo(i,j-lt) + Strokes(smrp,j-
lt)* requiredCapacity(smrp,i);
                        if consumo(i,j-lt)>resourceCapacity(i,j-lt)
                            disp ('Se excede la capacidad del recurso')
                        end
                     else
                         continue
                     end
                end
             else
                LIBORD(z,j)=0;
             end
             end

         else


             BINV(z,j)= BINV(z,j)+ BINV(z,j-1)+ ceil(RPROG(z,j))-
ceil(RC(z,j));
             if BINV(z,j)<0
                BINV(z,j)=0;
             end

             REQNETOS(z,j)= ceil(RC(z,j))- (BINV(z,j-1)+ RPROG(z,j));
             if REQNETOS(z,j)<0
                REQNETOS(z,j)=0;
             else
             RPLAN(z,j)=(ceil(REQNETOS(z,j)/strokeOut(z,smrp))*
strokeOut(z,smrp));
             BINV(z,j)= BINV(z,j)+ BINV(z,j-1)+ ceil(RPLAN(z,j))+ RPROG(z,j)-
ceil(RC(z,j)) ;
             if j>lt
                LIBORD(z,j-lt)= LIBORD(z,j-lt) + ceil(RPLAN(z,j));
                Strokes(smrp,j-lt)= ceil(REQNETOS(z,j)/strokeOut(z,smrp));
                SolStrokes(smrp,j-lt)= SolStrokes(smrp,j-lt)+
ceil(REQNETOS(z,j)/strokeOut(z,smrp));
                %Se registran los SKU coproducidos
                for i=1:MaxOutputs
                    output=StrokeOutputs(smrp,i)
                    if output >0
                        if output==z
                           continue
                        else
                        BINV(output,j)= BINV(output,j)+ Strokes(smrp,j-
lt)*strokeOut(output,smrp);
                        end
                    end
                end
                %Se registran los SKU consumidos
                for i=1:MaxInputs
                    input=StrokeInputs(smrp,i)
                    if input >0
```

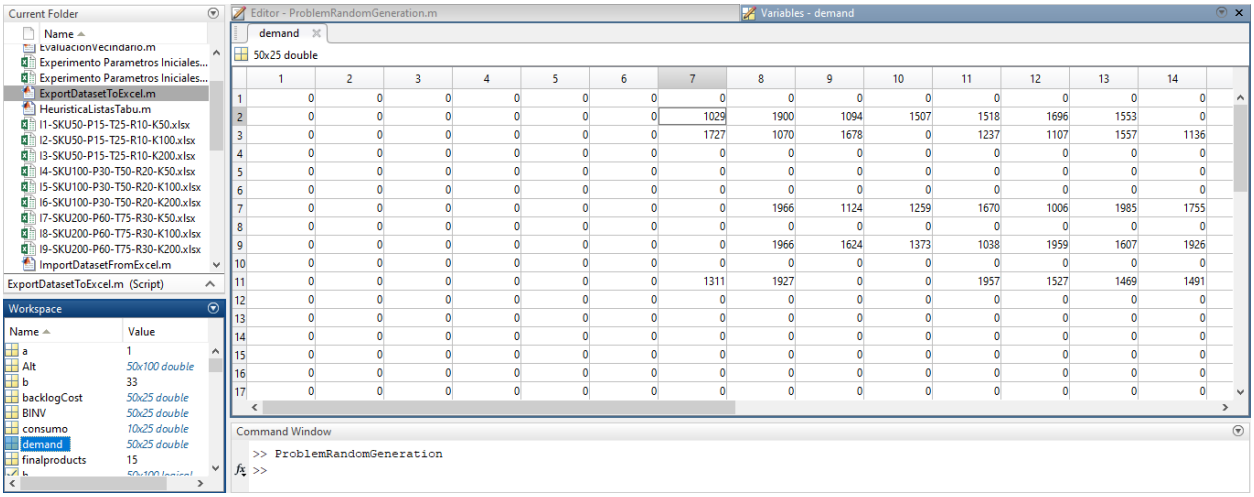
```
                        BINV(input,j)= BINV(input,j)+
Strokes(smrp,j)*strokeOut(input,smrp);
                    end
                end
                %Se registra la capacidad de los recursos utilizada
                for i=1:r
                     if requiredCapacity(smrp,i)>0
                        consumo(i,j-lt)= consumo(i,j-lt) + Strokes(smrp,j-
lt)* requiredCapacity(smrp,i);
                        if consumo(i,j-lt)>resourceCapacity(i,j-lt)
                            disp ('Se excede la capacidad del recurso')
                        end
                     else
                         continue
                     end
                end


            else
               LIBORD(z,j) =0;
            end

            end

         end
     end
 else
     smrp=0;
 end
```

ANEXO 8

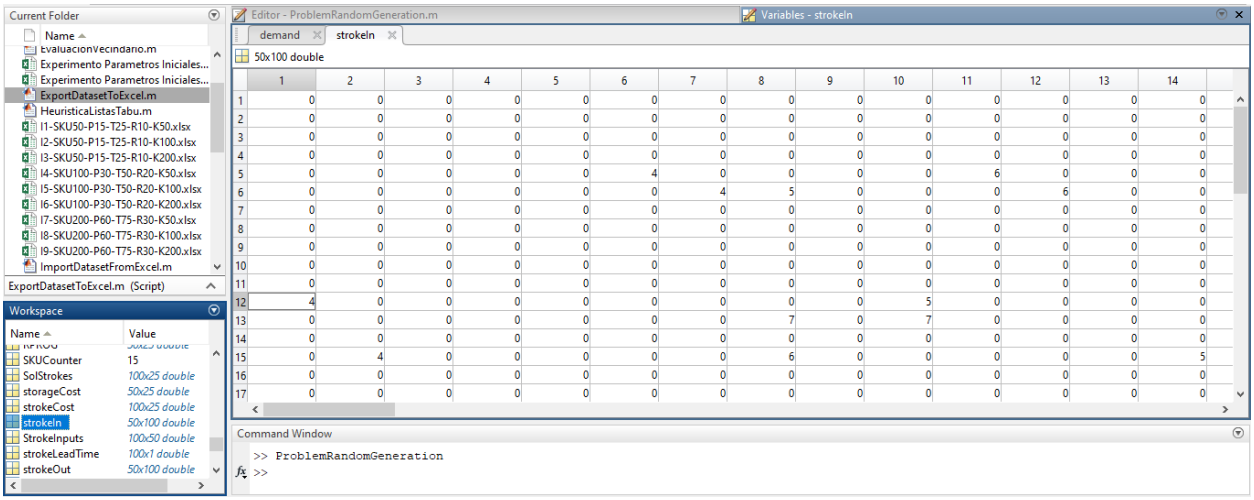
Ejemplo de interfaz gráfica y visualización de resultados

EJEMPLO 1. SKU50-P15-T25-R10-K100 – Generación de instancia aleatoria.

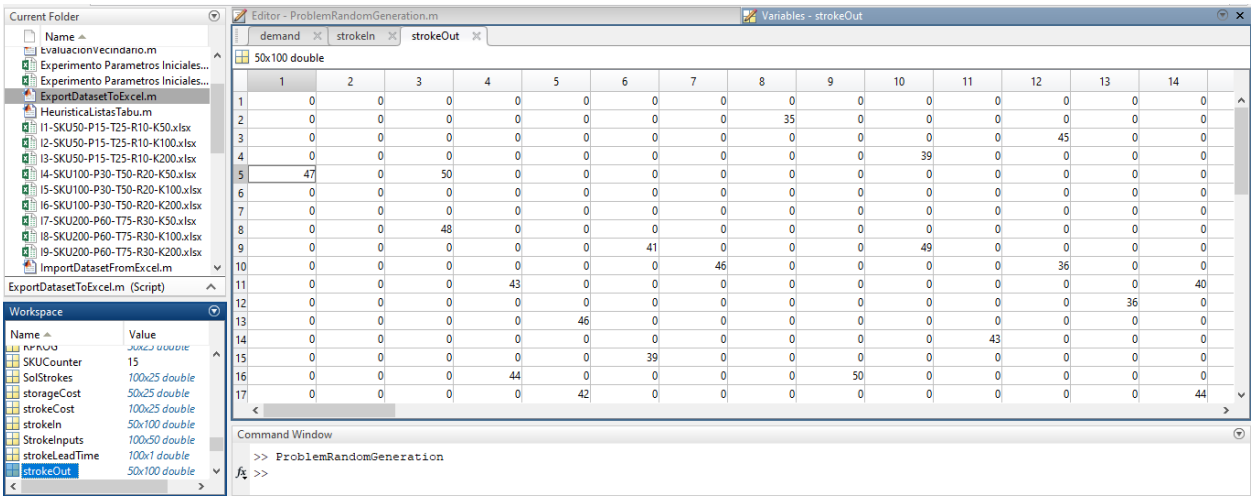
Demanda Generada Aleatoriamente



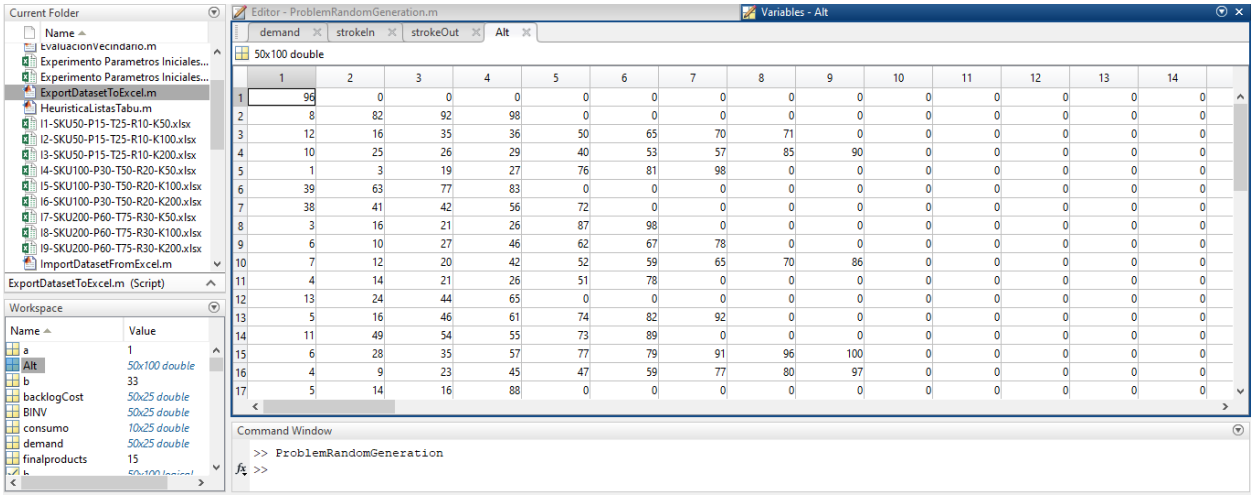
Matriz de Entradas SI(i,k)



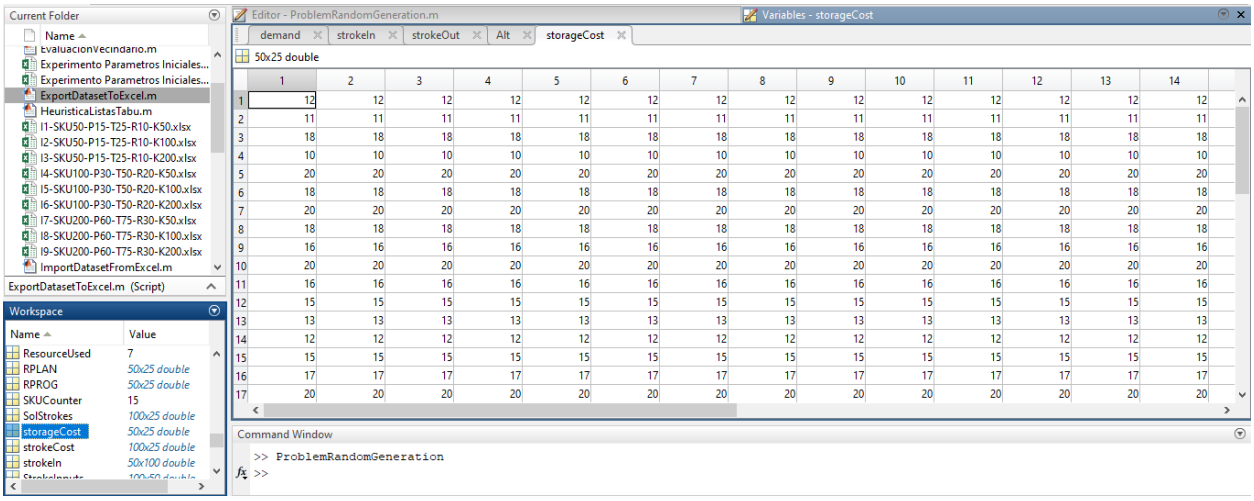
Matriz de Salidas SO(i,k)



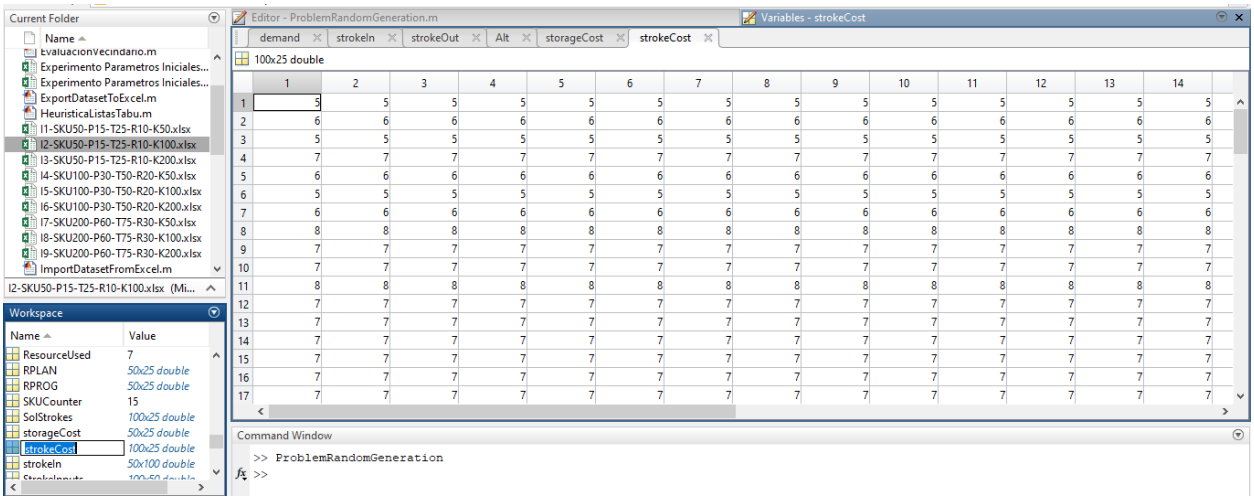
Matriz de Strokes Alternativos Alt(i,k)



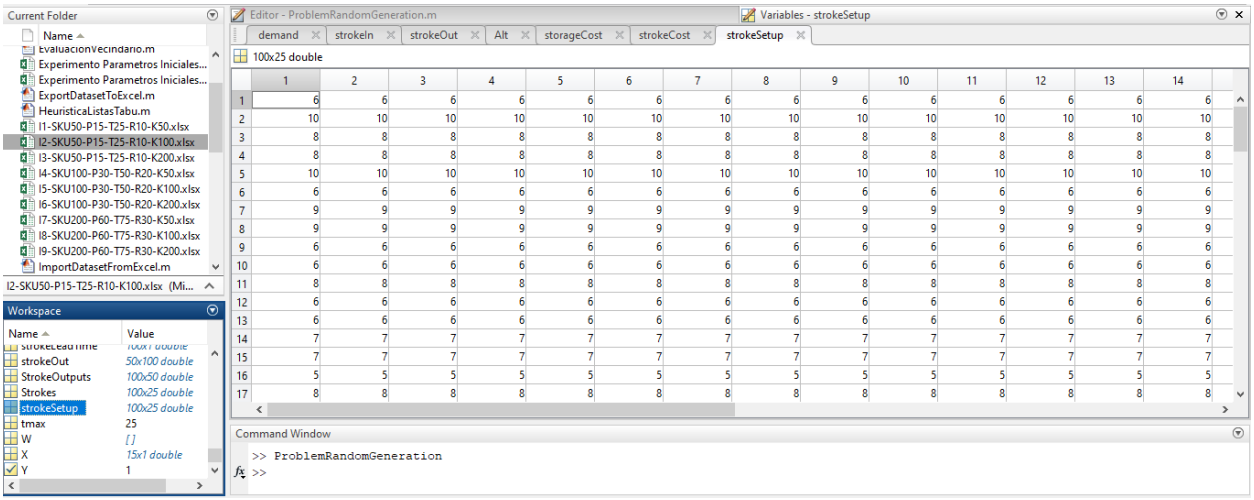
Costos de Almacenamiento



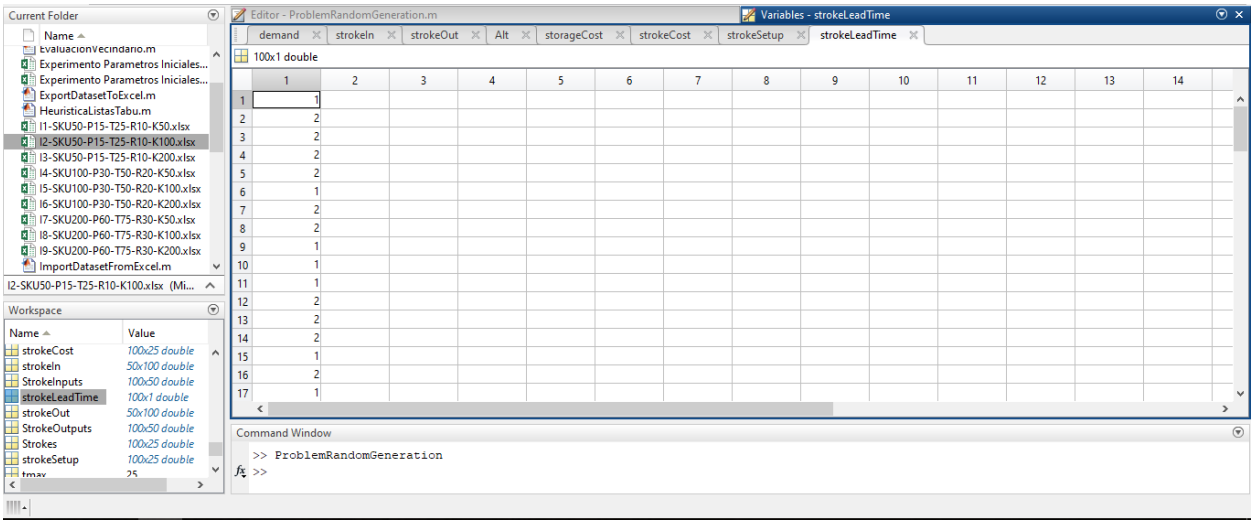
Costos de Operación de Strokes



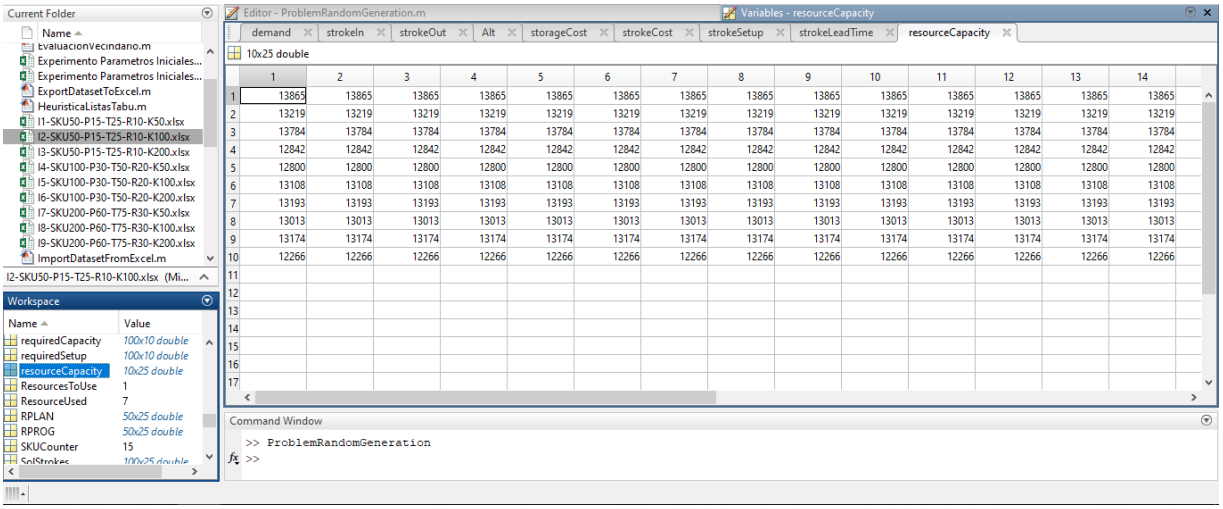
Costos de Alistamiento



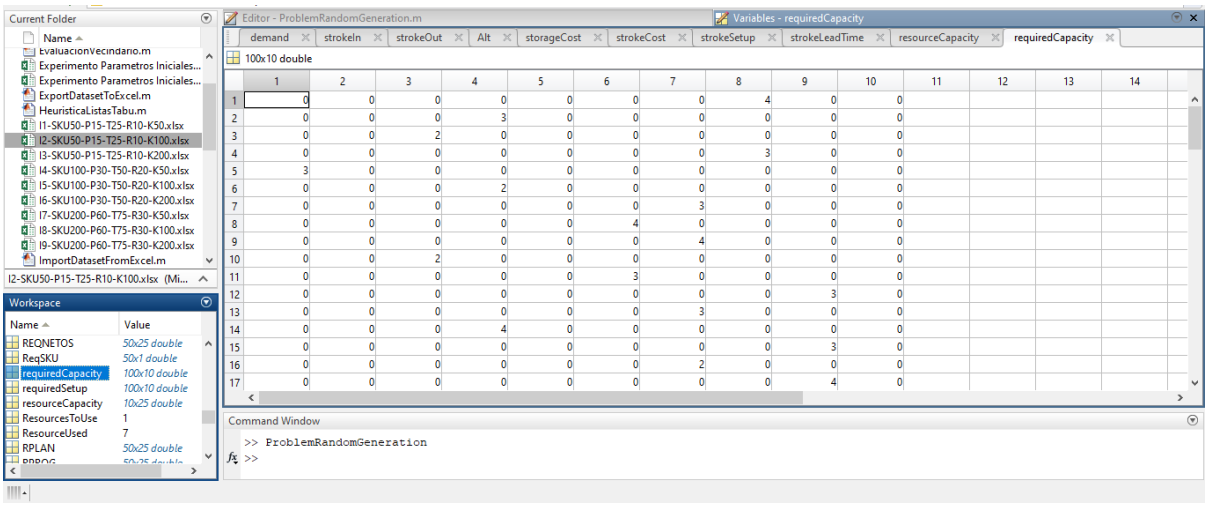
Lead times de los strokes



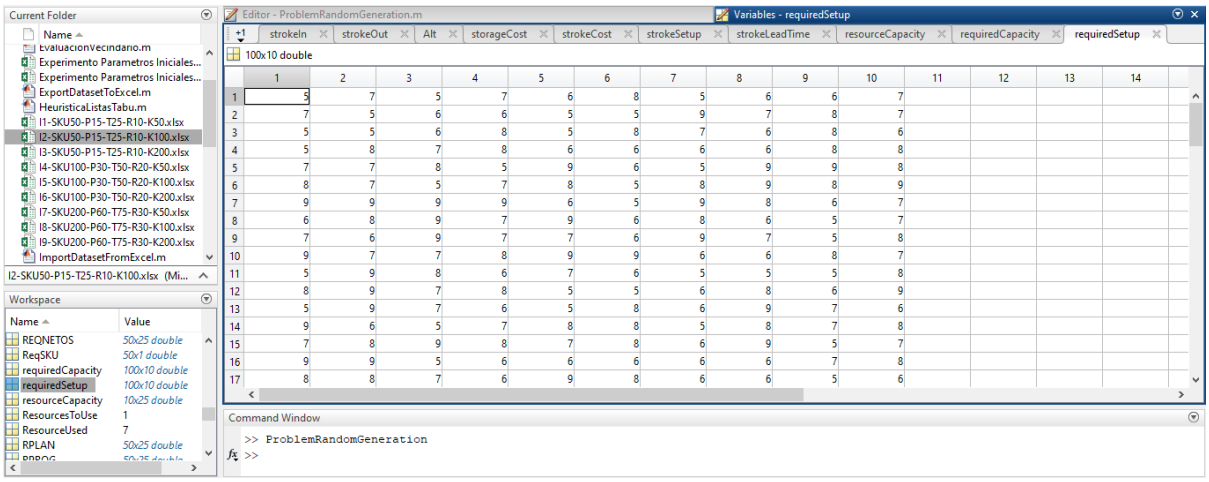
Capacidad de los recursos (tiempo)



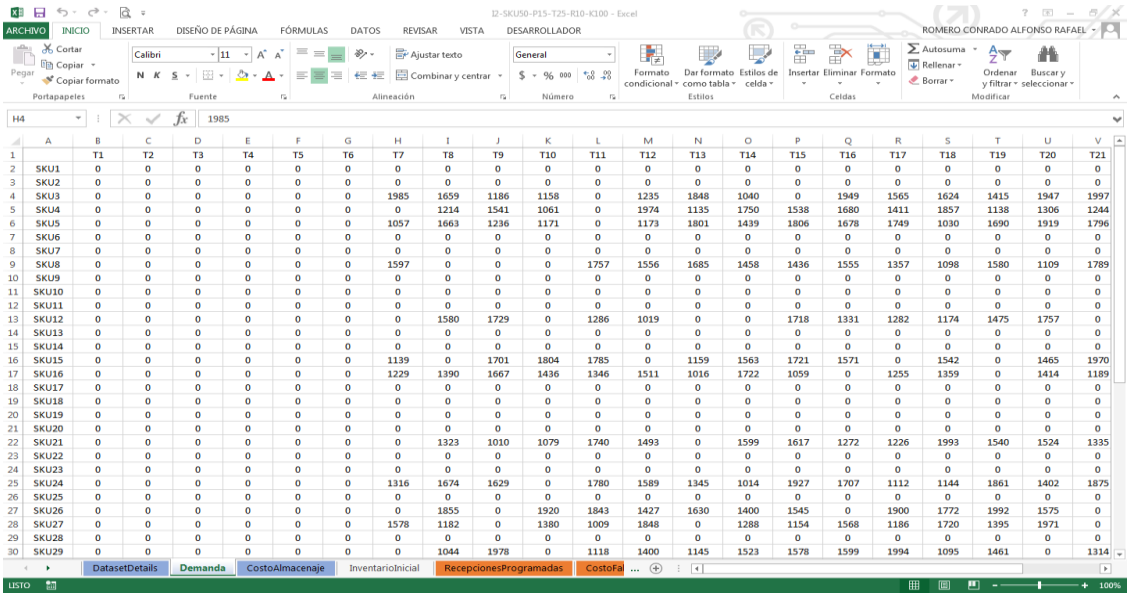
Capacidad requerida por Stroke (tiempo)



Alistamientos requeridos por Stroke (tiempo)

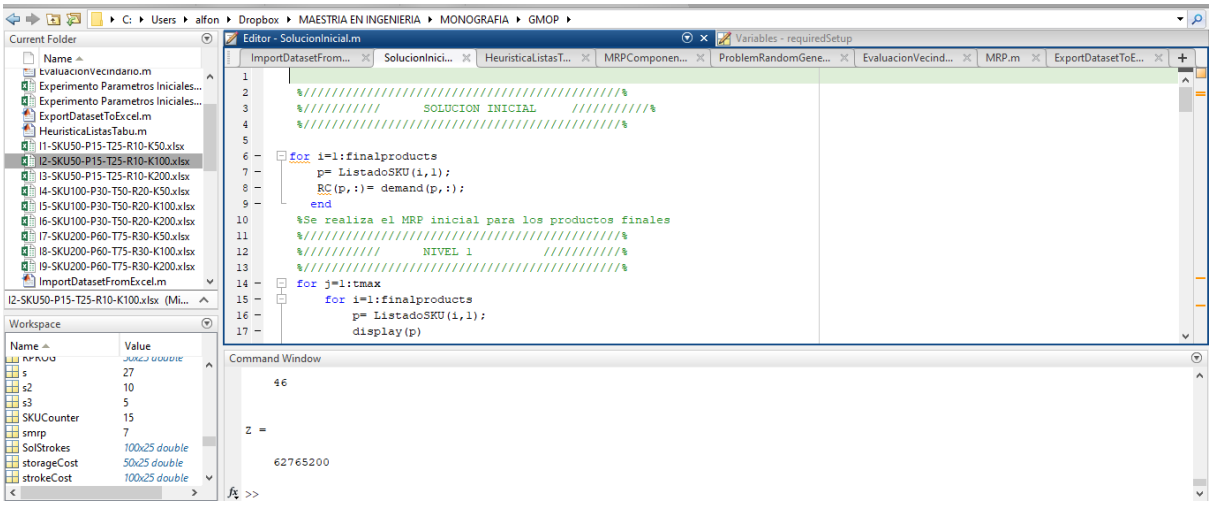


Instancia Aleatoria exportada a Excel

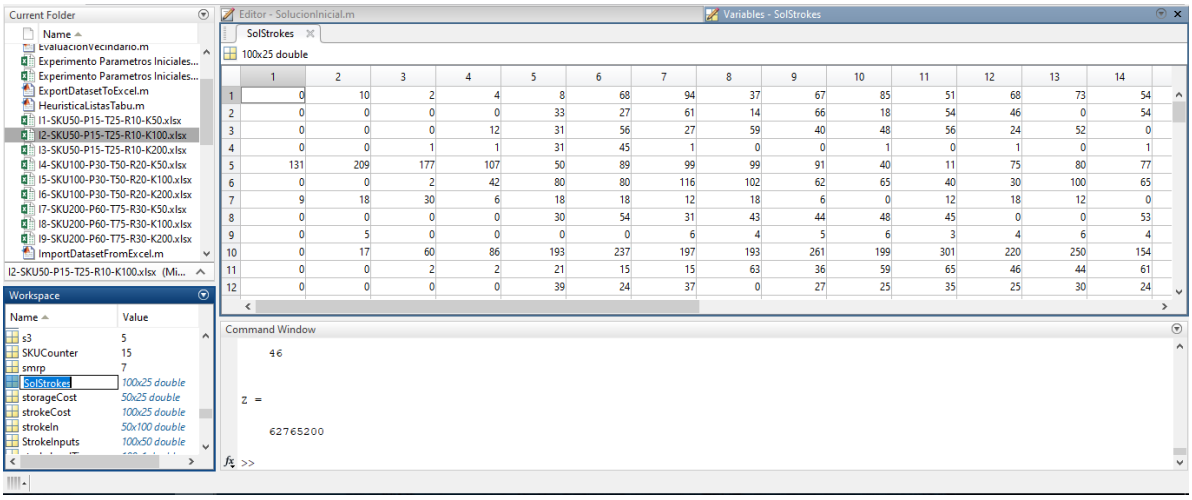


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1		T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21
2	SKU1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	SKU2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	SKU3	0	0	0	0	0	0	1985	1659	1186	1158	0	1235	1848	1040	0	1949	1565	1624	1415	1947	1997
5	SKU4	0	0	0	0	0	0	0	1214	1541	1061	0	1974	1135	1750	1538	1680	1411	1857	1138	1306	1244
6	SKU5	0	0	0	0	0	0	1057	1663	1236	1171	0	1173	1801	1439	1806	1678	1749	1030	1690	1919	1796
7	SKU6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	SKU7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	SKU8	0	0	0	0	0	0	1597	0	0	0	1757	1556	1685	1458	1436	1555	1357	1098	1580	1109	1789
10	SKU9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	SKU10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	SKU11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	SKU12	0	0	0	0	0	0	0	1580	1729	0	1286	1019	0	0	1718	1331	1282	1174	1475	1757	0
14	SKU13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	SKU14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	SKU15	0	0	0	0	0	0	1139	0	1701	1804	1785	0	1159	1563	1721	1571	0	1542	0	1465	1970
17	SKU16	0	0	0	0	0	0	1229	1390	1667	1436	1346	1511	1016	1722	1059	0	1255	1359	0	1414	1189
18	SKU17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	SKU18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	SKU19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	SKU20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	SKU21	0	0	0	0	0	0	0	1323	1010	1079	1740	1493	0	1599	1617	1272	1226	1993	1540	1524	1335
23	SKU22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	SKU23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	SKU24	0	0	0	0	0	0	1316	1674	1629	0	1780	1589	1345	1014	1927	1707	1112	1144	1861	1402	1875
26	SKU25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	SKU26	0	0	0	0	0	0	0	1855	0	1920	1843	1427	1630	1400	1545	0	1900	1772	1992	1575	0
28	SKU27	0	0	0	0	0	0	1578	1182	0	1380	1009	1848	0	1288	1154	1568	1186	1720	1395	1971	0
29	SKU28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	SKU29	0	0	0	0	0	0	0	1044	1978	0	1118	1400	1145	1523	1578	1599	1994	1095	1461	0	1314

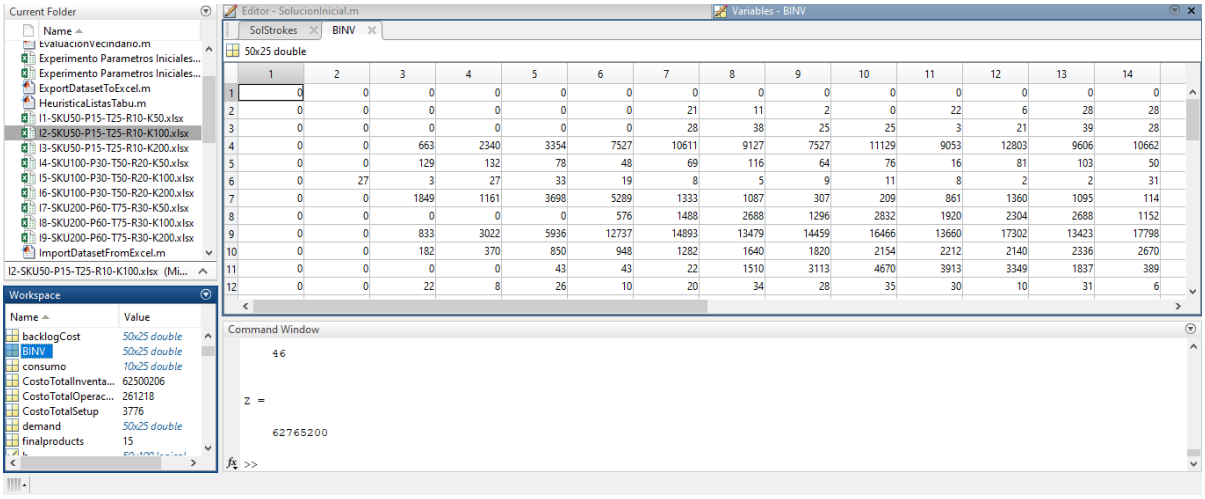
EJEMPLO 1. SKU50-P15-T25-R10-K100 – Obtención de Solución Inicial



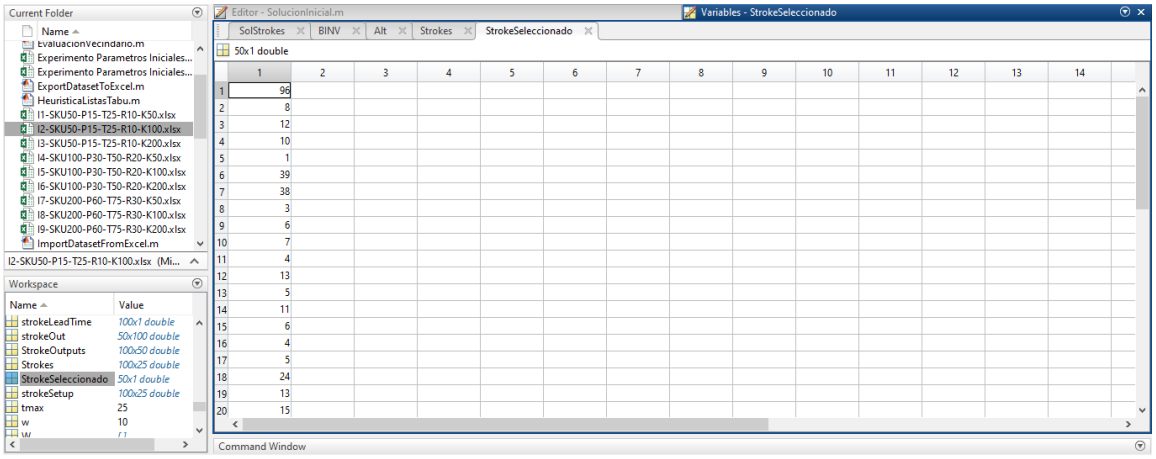
Solución de Strokes y costo inicial obtenidos



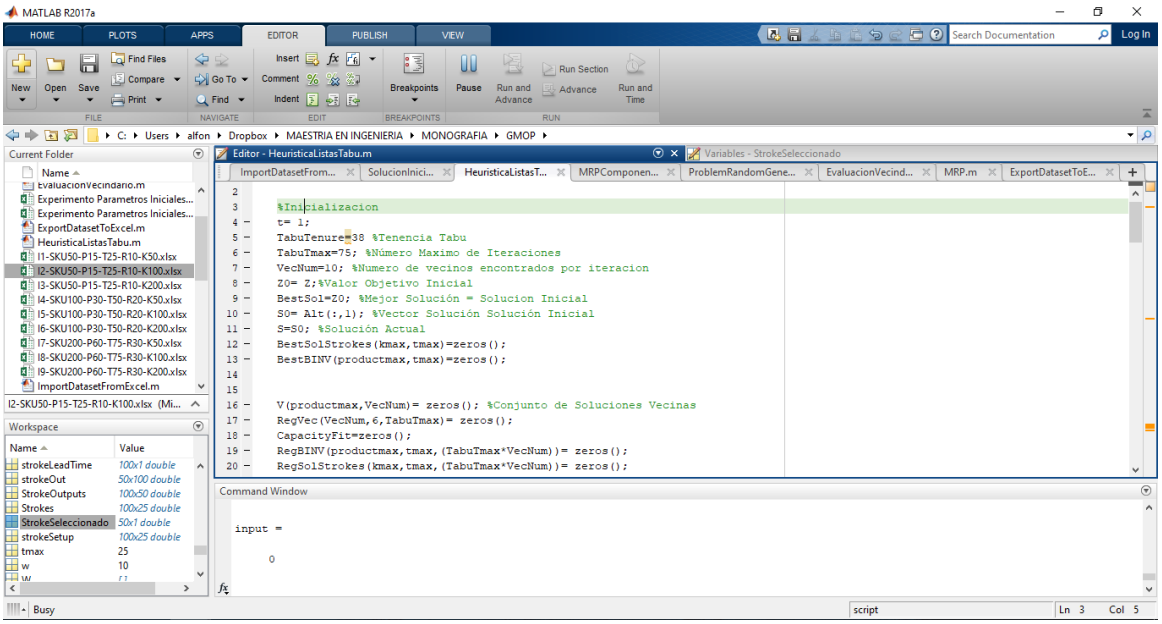
Balance de inventario de la solución inicial



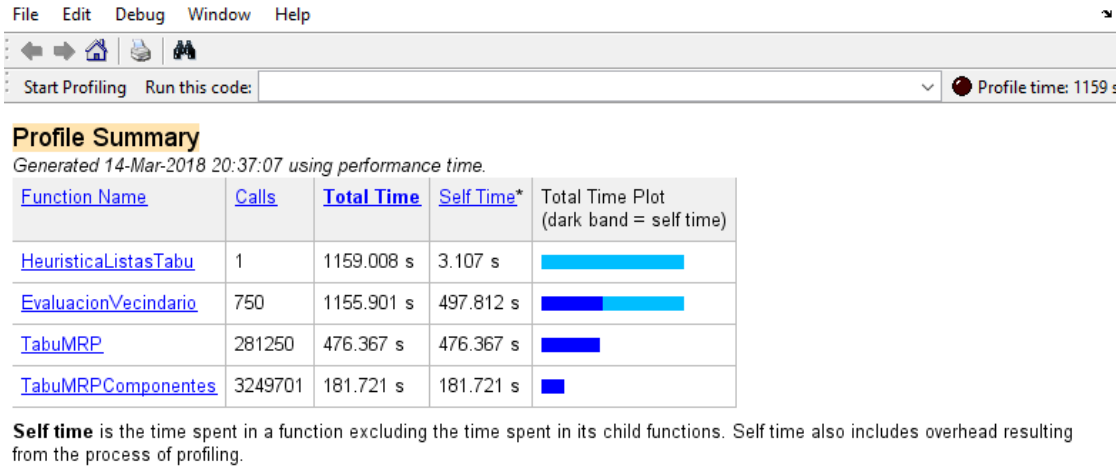
Vector solución



EJEMPLO 1. SKU50-P15-T25-R10-K100 – Solución mediante heurística basada en listas tabú



Informe de tiempos de solución



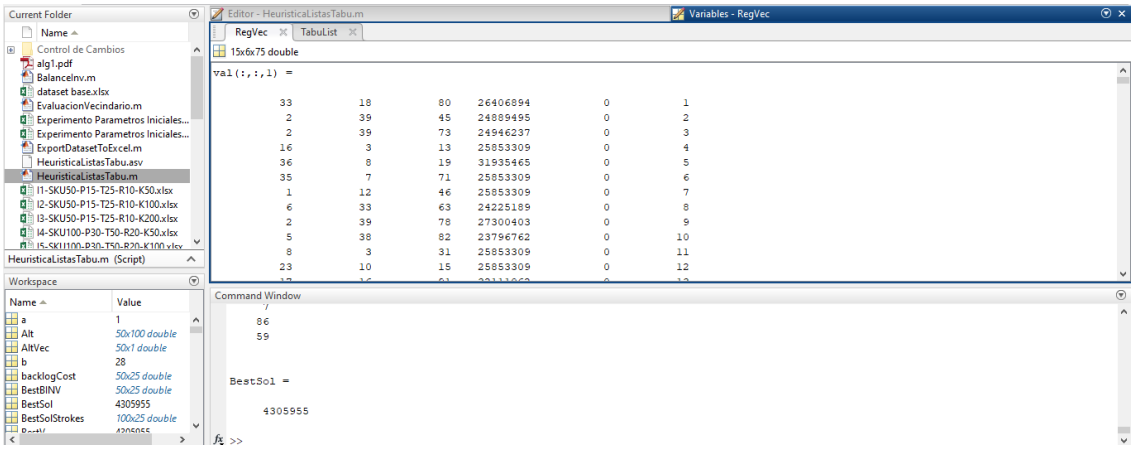
Profile Summary

Generated 14-Mar-2018 20:37:07 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
HeuristicaListasTabu	1	1159.008 s	3.107 s	
EvaluacionVecindario	750	1155.901 s	497.812 s	
TabuMRP	281250	476.367 s	476.367 s	
TabuMRPComponentes	3249701	181.721 s	181.721 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

Registro de movimientos y evaluaciones de costo por iteración- RegVec



El registro de movimientos y evaluaciones de costo para cada solución vecina generada son almacenados en la matriz RegVec. Esta matriz está conformada por 6 columnas y permite almacenar y utilizar de manera sencilla la información y los resultados para los movimientos realizados.

Columna 1: Producto seleccionado en el movimiento

Columna 2: Stroke al inicio del movimiento

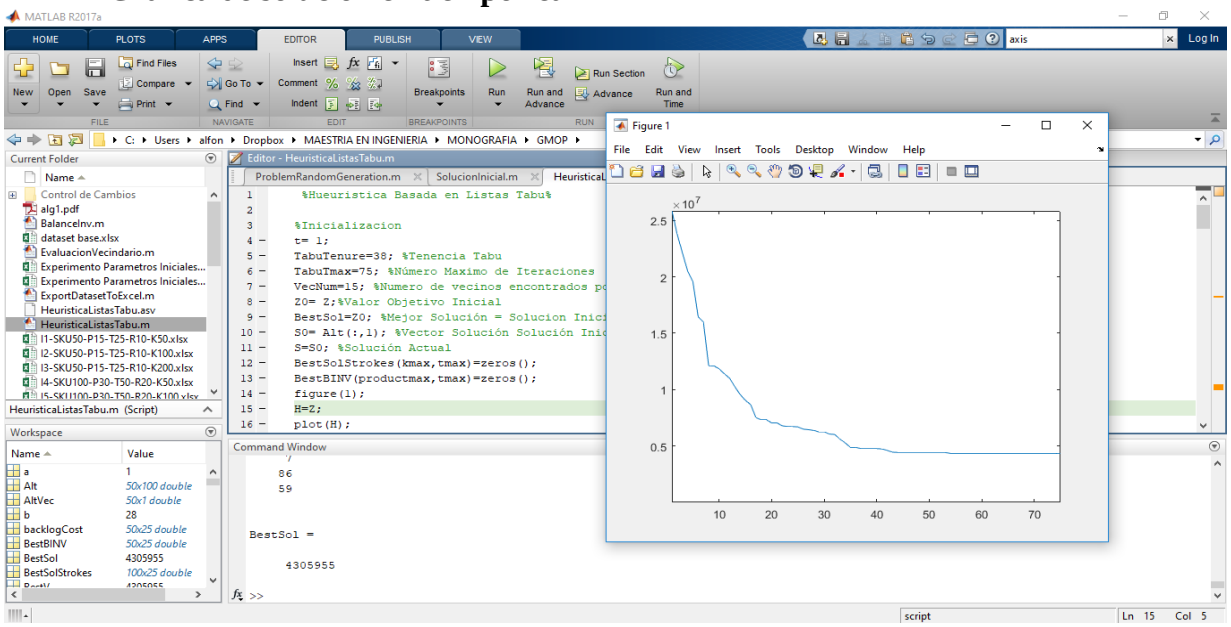
Columna 3: Stroke al final del movimiento

Columna 4: Costo de la función objetivo para la solución generada

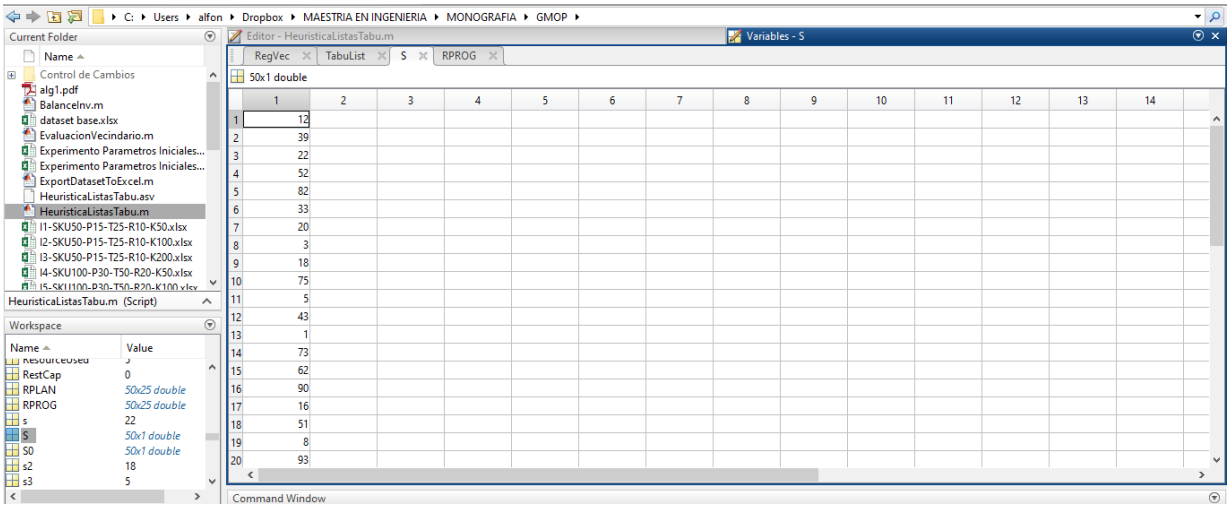
Columna 5: Número de restricciones de capacidad violadas por la solución.

Columna 6: Identificación del registro.

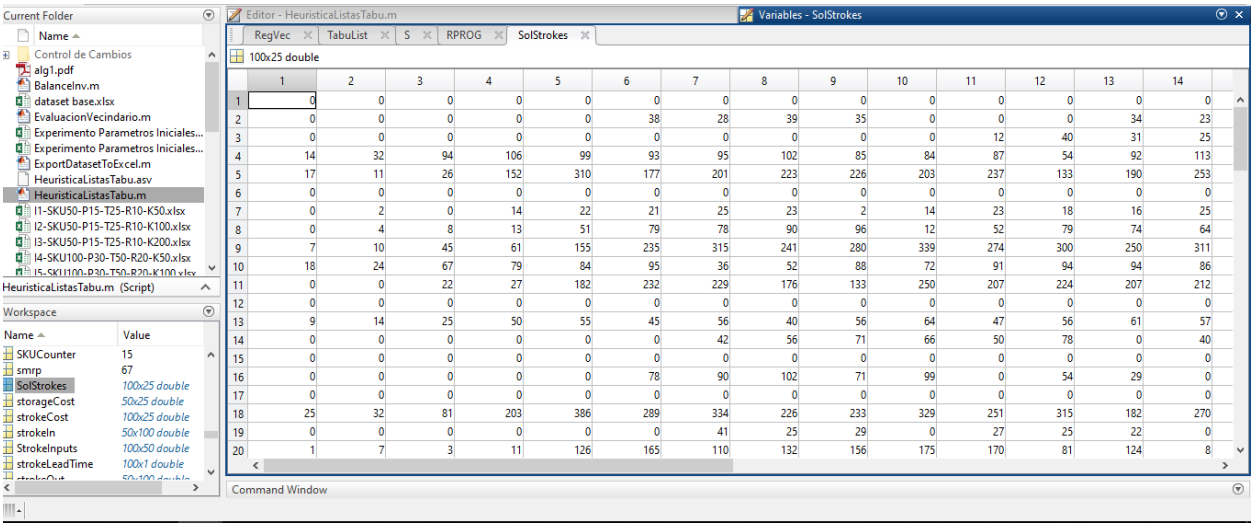
Gráfica de solución en tiempo real



Vector Solución S



Solución de Planificación de Strokes



Balance de inventario