UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN ORGANIZACIÓN INDUSTRIAL

DISEÑO DE UNA HERRAMIENTA BASADA EN HEURÍSTICAS Y UN GRASP PARA LA RESOLUCIÓN DEL PROBLEMA DE SECUENCIACIÓN DE RETROVISORES DE UNA LÍNEA DE MONTAJE EN BUCLE CERRADO

AUTOR: CAMPOS BELTRÁN, JAVIER

TUTOR: MAHEUT, JULIEN

COTUTOR: ANDRÉS ROMANO, CARLOS

Curso Académico: 2017-18

RESUMEN

El presente trabajo trata sobre el problema de secuenciación de pintado de retrovisores en una línea de montaje. En el trabajo se encontrarán la introducción y descripción del problema, las bases matemáticas para el posterior análisis y diseño de la herramienta. En los siguientes capítulos se implementarán las diversas herramientas para dar solución a este tipo de problemas, y se presentarán los resultados y conclusiones a los que se ha llegado tras el trabajo.

Palabras Clave: GRASP, secuenciación, heurísticas, metaheurísticas.

ÍNDICE GENERAL

DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Presupuesto

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

MEMORIA

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN.

1.1. JUSTIFICACIÓN.

Este trabajo tiene como finalidad la realización del trabajo final de grado de Javier Campos Beltrán, que permitirá completar el grado de Ingeniería de Organización Industrial. En él se pretende demostrar las capacidades del alumno para implementar en un caso real aplicado a una empresa los conceptos aprendidos durante el grado, así como la creatividad y la capacidad de aprendizaje, fundamentales en ingeniería.

1.2. OBJETIVO DEL TRABAJO.

El objeto principal de este trabajo es ofrecer una solución a un problema real de secuenciación a una línea de montaje de retrovisores proveedora de una gran firma automovilística. Para ello se va a estudiar la eficiencia y eficacia de una herramienta basada en heurísticas y un GRASP.

Las especificaciones del trabajo serán datos extraídos de un generador de instancias aleatorias dentro de unos rangos, que simulan instancias que podría recibir la línea de montaje y para las cuales necesitará una secuenciación que optimice el proceso.

Para la realización de esta herramienta, se ha utilizado el software Excel, con su programador integrado que utiliza lenguaje Visual Basic, con el que se realizarán los cálculos, y se presentará la interfaz donde se introducirán los datos de la instancia y las soluciones aportadas por la herramienta.

1.3. ESTRUCTURA DEL TRABAJO.

La estructura del trabajo consta de un primer capítulo donde se describe el problema real de la línea de montaje con sus correspondientes características. En el siguiente capítulo, se presentan los antecedentes teóricos en los que se ha basado el trabajo, y sin los cuales no se habría podido implementar esta herramienta; el primero de ellos, un estudio realizado sobre un caso similar, en el que se analizaban las complicaciones del problema, se creaba un algoritmo que resolvía y localizaba el óptimo del problema para instancias sencillas, y se planteaba que, en instancias más complejas, el tiempo de computación se elevaría exponencialmente. En el tercer capítulo, se representa el por qué utilizar un software como Excel para la implementación de la herramienta. En el siguiente, el cuarto capítulo, se trabaja con las posibles heurísticas que podrían ayudar a la resolución del problema, analizando cuáles podrían ser efectivas y cuales no y razonando los motivos, y posteriormente se realiza un estudio numérico, focalizando los resultados en la función objetivo y el tiempo de computación. En el quinto capítulo, se repite el procedimiento del punto cuarto, pero con metaheurísticas. Para concluir, se presentará un análisis de todo lo anterior, con sus consecuentes conclusiones, analizando todas las soluciones;

posteriormente, se propondrá para el caso real una herramienta que sea la más efectiva para el caso que se trata. Por último, se incluirá la bibliografía utilizada.

CAPÍTULO 2. DESCRIPCIÓN DEL PROBLEMA.

2.1. INTRODUCCIÓN.

En este capítulo se pretende introducir y explicar el problema en cuestión, y las características referentes al problema de secuenciación de retrovisores en una línea de montaje. Estas características son las que posteriormente se identificarán y se observará de qué manera afectan a la solución, a la función objetivo y al tiempo de procesamiento.

2.2. DESCRIPCIÓN DEL PROBLEMA.

El problema se centra en una línea de montaje de una planta de un proveedor de una de las principales empresas de automoción del mundo, en la operación de pintado de retrovisores.

Para este proceso, se tiene un carrusel en el que mediante perchas se cuelgan retrovisores, que posteriormente un robot de pintura pinta, y pasan por un horno que fija la pintura y evita que se deposite polvo.

La sección de pintado consta de un carrusel o cinta transportadora circular en la que se sitúan las carcasas a pintar tal y como se muestra la figura:

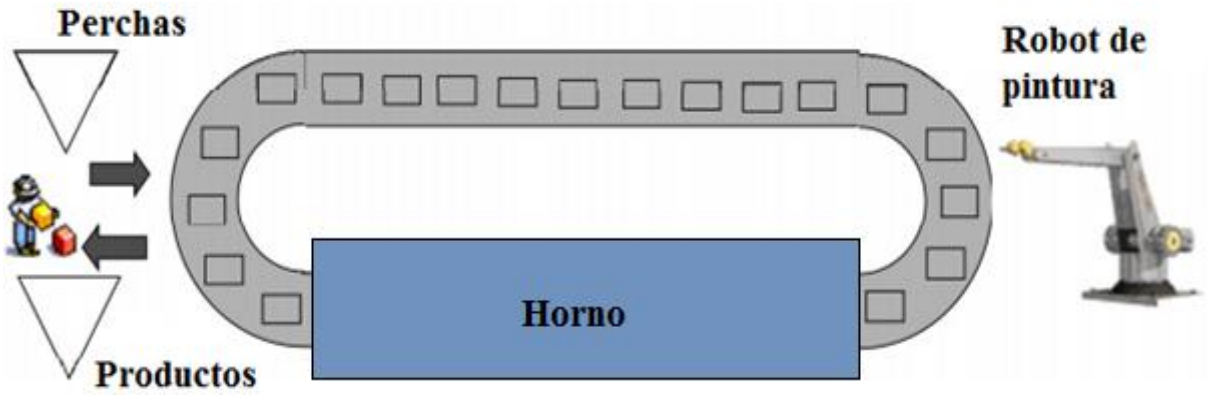

Fig. 1 Esquema del funcionamiento de la línea de pintura.

Se trata de un carrusel giratorio de longitud definida por el parámetro PPL (Parts Per Loop) y que se mueve a velocidad constante. En un extremo del carrusel hay una zona en la que un robot pinta los espejos retrovisores. Cada posición del carrusel (señalada en la imagen superior con un cuadrado) se compone de una "percha" sobre la que se cuelgan los espejos retrovisores. Esta percha se va moviendo en el carrusel hasta que llega al robot de pintura que se encarga de pintar los retrovisores que cuelgan de ella. El robot lee en la percha el tipo de producto que se trata y realiza su programa de pintado. El carrusel se sigue moviendo de tal manera que los productos vuelven a la zona opuesta a la zona de pintado.

En ese recorrido, los retrovisores pasan por un horno que fija la pintura para que no se deposite polvo en su superficie. Este horno está encendido durante todo el tiempo de funcionamiento del carrusel, ininterrumpidamente, haya o no haya producto en el interior.

Una vez sale del horno, un operario descarga la percha ya pintada y ajusta, si es necesario, la posición para colocar otra percha diferente y coloca una nueva percha con producto sin pintar.

Cada carcasa de retrovisor tiene dos atributos a considerar: geometría de la pieza y color de la pintura. En este sistema hay tres costes fundamentales a reducir, el primero es el coste asociado a pintar consecutivamente colores diferentes (debido a los costes del disolvente en el robot). El segundo es el coste asociado al tiempo de operario necesario para realizar el cambio de tipo de percha, ya que, si la geometría de la carcasa es diferente en una posición determinada para vueltas consecutivas, no solo ha de cambiarse la percha, sino también la base donde se sustenta la percha. Sin embargo, el coste que más interesa minimizar, y sobre el que se trabajará para minimizarlo, es el tiempo de funcionamiento del horno, ya que utiliza una gran cantidad de energía y cuanto menos tiempo esté encendido para hornear todos los retrovisores, menor es el gasto energético.

El objetivo del proyecto es definir la secuencia de lotes a pintar de tal manera que se cumpla el plan de producción y se intenten minimizar los costes asociados a cambio de color en posiciones consecutivas y el cambio de geometría para la misma posición en vueltas consecutivas, es decir, en el siguiente loop.

2.3 CARACTERIZACIÓN DEL PROBLEMA.

Dentro del entorno de este problema, hay varios factores que afectarían a la hora de encontrar la solución al problema planteado y que pueden tomar diversos valores.

2.3.1 Rellenado.

Se refiere a la forma en la que se rellena la matriz final (en la que cada fila corresponde a un loop (vuelta), y cada columna corresponde con cada una de las posiciones del carrusel. Se puede rellenar de varias formas:

- Horizontal monosentido.
- Horizontal bisentido.
- Vertical monosentido.
- Vertical bisentido.
- Diagonal.

La diferencia entre el rellenado bisentido y monosentido, es que, en el monosentido, la matriz visual final se completa rellenando todas las filas de izquierda a derecha, y en el bisentido la matriz visual final se completa rellenando cada fila de izquierda a derecha, y la siguiente fila de derecha a izquierda.

2.3.2 Costes de setup.

Se refiere a los costes por cambio de color o geometría entre productos, y a cómo se asigna el coste. Pueden ser de las dos siguientes formas:

- Simétricos (mismo coste pasar de 1 a 2 que de 2 a 1)
- Asimétricos (distinto coste pasar de 1 a 2 que de 2 a 1)

2.3.3 Agrupación.

Se refiere a la forma en la que se introducen los elementos. Puede ser de las siguientes formas:

- Por unidades sueltas.
- Por bloques de todos los productos iguales.
- Por bloques definidos por PPL.
- Por bloques definidos por unidades de almacenamiento: cajas, pallets… (fabricando un extra de productos, sobrepasando los pedidos recibidos, ya que no se trabaja just-in-time).

2.3.4 Consideraciones físicas.

Se trata de la presencia o no de huecos entre los productos, es decir, espacio dentro del Loop que se dejan sin producto debido a factores como el tiempo de limpieza de cabezales, cambio de color, geometría…

- Sin huecos.
- Con huecos (unidireccionales): Huecos al pasar de un producto al siguiente.
- Con huecos (bidireccionales): Huecos al pasar de un elemento al siguiente y/o de un hueco en un loop al hueco en el siguiente loop.

2.4 REPRESENTACIÓN GRÁFICA DE LAS CARACTERÍSTICAS.

Para poder clarificar las diferentes características del problema a tratar, se utiliza una representación gráfica de estas. Se representa mediante un diagrama de araña todas las posibles opciones de cada uno de estos factores.

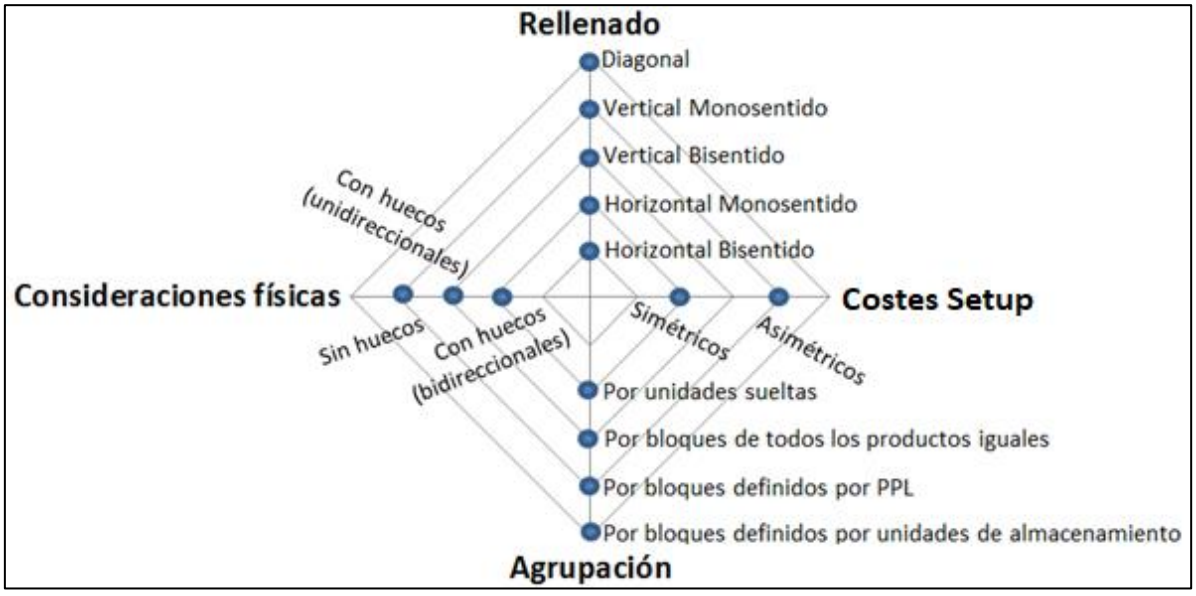

Fig. 2 Diagrama de araña de los factores que afectan al problema.

Para aclarar estos conceptos, se va a ver un ejemplo:

Se trata de un problema de secuenciación de 15 elementos, en 3 loops (vueltas) de tamaño 5. Las X representan los cambios, ya sean de color (horizontales, de un elemento al siguiente) o de geometría (verticales, de un loop a otro).

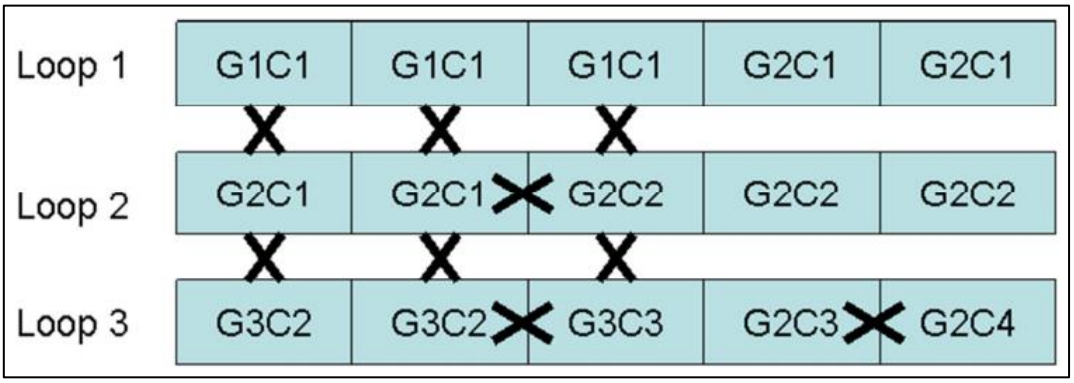

Fig. 3 Esquema de una matriz solución de un problema similar.

Este problema seguiría las mismas condiciones que el caso que se estudia, que son las siguientes:

- Costes de setup: Asimétricos.
- Agrupación: Por bloques de todos los productos iguales (Código de producto igual, es decir, mismo color y geometría).
- Consideraciones físicas: Sin huecos.
- Rellenado: Horizontal, y en este caso, monosentido.

Y su diagrama de araña sería el siguiente:

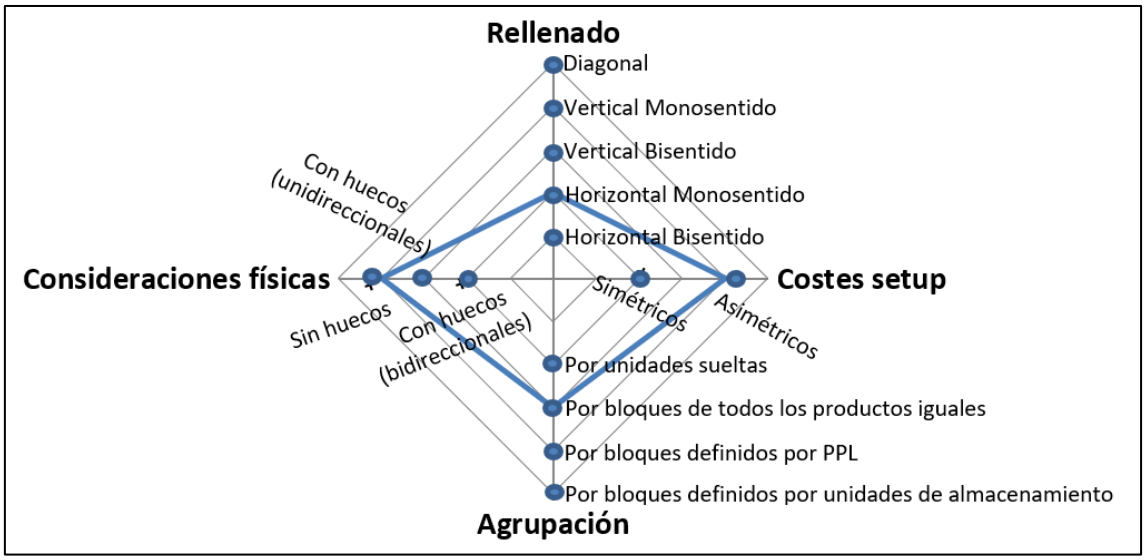

Fig. 4 Diagrama de araña de los factores que afectan al problema.

2.5 CONCLUSIONES.

Por tanto, se trata de un problema de secuenciación en el que hay que considerar diversos factores como la forma de rellenado, los costes de setup, la agrupación y las consideraciones físicas. Como cualquier caso en automoción en el que no se trata la calidad, el principal objetivo es aumentar la productividad, y esto se consigue reduciendo los tiempos de procesado. En este caso se añade además un coste asociado al horno que obliga, más si cabe, a minimizar este tiempo de proceso.

CAPÍTULO 3. ANTECEDENTES TEÓRICOS.

3.1. INTRODUCCIÓN.

En este capítulo, se va a repasar y analizar toda la base teórica sobre la cual se sustenta este trabajo, y que nos permite sentar las bases sobre las cuales podemos proponer una herramienta que pueda contribuir a la resolución del problema.

3.2. PROBLEMAS DE SECUENCIACIÓN.

El problema tratado es un problema de secuenciación. Estos problemas constan de una serie de elementos (en nuestro caso productos) que hay que ordenar de tal forma que todos queden incluidos en la solución final, y buscando minimizar una función objetivo (en nuestro caso el coste asociado al cambio de color y geometría)

3.3. PROGRAMACIÓN MATEMÁTICA.

Se trabaja con un problema complejo, que no se puede resolver ni visualmente ni con un razonamiento sencillo. Por tanto, habrá que recurrir a la programación matemática para conseguir alcanzar la solución óptima.

Para formular un problema y poder resolverlo con programación matemática, hay que definir tres características:

- Variable de decisión.
- Función objetivo.
- Restricciones.

Y a partir de ahí se formula en problema y se procede a buscar un método de resolución. En el caso estudiado, las características del problema son las siguientes:

- La **variable de decisión** de esta programación matemática será un vector en el cual se incluyan los "N" productos demandados en el orden en el que se introducirán en el carrusel.
- La **función objetivo** a minimizar será la función "Costes Totales", que suma los costes por cambios de color y los costes por cambios de geometría.
- Sin embargo, como la única **restricción,** al tratarse de un problema de secuenciación, sería que todos los ítems demandados se deben incluir en la solución final, el problema no estaría lo suficientemente acotado y por tanto no se podría encontrar el óptimo de otra forma que enumerando todas las soluciones posibles y escogiendo la que tenga menor valor en la función objetivo.

Este método de resolución se denomina enumeración completa y es todo lo eficaz que se requiere, ya que nos garantiza alcanzar el óptimo en cada problema. Sin embargo, es posible que para este caso de estudio no sea eficaz ya que se necesita que se obtenga la solución en reducido tiempo de computación.

Analizando las posibles soluciones en problemas sencillos, se puede extraer en problemas más complejos cuál sería el número total de alternativas.

Por ejemplo, un problema en el que haya dos tipos distintos de productos, las posibles soluciones, expresadas en forma de vector, serían:

(1, 2) (2, 1) → 2 posibles alternativas

En el caso de tener tres tipos distintos de productos, el número de posibles soluciones sería:

(1, 2, 3) (2, 1, 3) (3, 1, 2)

(1, 3, 2) (2, 3, 1) (3, 2, 1) → 6 posibles alternativas

En el caso de tener cuatro tipos distintos de productos, el número de posibles soluciones sería:

(1, 2, 3, 4) (2, 1, 3, 4) (3, 1, 2, 4) (4, 1, 2, 3)

(1, 2, 4, 3) (2, 1, 4, 3) (3, 1, 4, 2) (4, 1, 3, 2)

(1, 3, 2, 4) (2, 3, 1, 4) (3, 2, 1, 4) (4, 2, 1, 3)

(1, 3, 4, 2) (2, 3, 4, 1) (3, 2, 4, 1) (4, 2, 3, 1)

(1, 4, 2, 3) (2, 4, 1, 3) (3, 4, 1, 2) (4, 3, 2, 1)

(1, 4, 3, 2) (2, 4, 3, 1) (3, 4, 2, 1) (4, 3, 1, 2) → 24 posibles alternativas

Por tanto, es sencillo extraer que el número de alternativas totales depende del número total de productos demandados de la siguiente forma:

2! = 2 posibles alternativas

3! = 24 posibles alternativas

(Demanda de productos)! = Alternativas totales

En el caso estudiado, se trabaja con instancias de entre 20 y 1200 productos.

En el caso más sencillo y favorable, con una demanda de N=20 productos, el número de soluciones sería de:

20! = $2.43 * 10^{18}$ posibles soluciones. 2,43 trillones de soluciones.

Actualmente se dispone de ordenadores que pueden realizar sin problemas este tipo de operaciones, pero si se estudia un caso un poco más complejo en el que la demanda sea de N=100 productos, el número total de soluciones sería de:

100! = $9.33 * 10^{157}$ posibles soluciones.

Crece exponencialmente el número de posibles soluciones conforme se aumenta la demanda de productos, por lo que surge la necesidad de encontrar alternativas a la resolución mediante la enumeración completa.

Se conocen dos posibles alternativas a la enumeración completa:

- Enfoques optimizadores.
- Enfoques heurísticos y metaheurísticos.

3.4. ENFOQUES OPTIMIZADORES.

Con anterioridad, se realizó un estudio sobre un caso similar al descrito en este trabajo. Este estudio lo realizaron Subhamoy Ganguly y Manuel Laguna, que generaron un modelo matemático que resolvía el problema localizando la solución óptima.

El estudio lo realizaron en 2015, y en él modelaban y resolvían problemas de tipo ciclo cerrado con dos tipos de costes de setup (geometría y color).

El problema se formula como un modelo Mixed-Integer Linear Programming (MILP)

La notación utilizada es la siguiente:

- C: número de colores.
- G: número de geometrías.
- S: número de productos.
- CC: número de pares de colores distintos (C x C).
- GG: número de pares de geometrías distintas (G x G).
- LB_c: límite inferior de número de cambios de color (0 por defecto).
- LB_G: límite inferior de número de cambios de geometría (0 por defecto).
- LB: límite inferior del valor de la función objetivo.
- $h_{ii'}$: (horizontal) coste de cambio de color de i a i'.
- $v_{jj'}$: (vertical) coste de cambio de geometría de j a j'.
- PPL: partes por vuelta.
- n: número total de posiciones donde $n = \sum_{(i,j)\in S} d_{ij}$

Variables principales:

- x_{ijk}

  1 si el producto (i,j) ∈ S está en la posición k.
  0 si no lo está.

Variables auxiliares:

- y_{ik}

  1 si el producto de color i está en la posición k.
  0 si no lo está.

- z_{jk}

  1 si el producto de geometría j está en la posición k.
  0 si no lo está.

- $p_{ii'k}$

  1 si el producto cambia de i a i' ((i,i') ∈ CC) entre las posiciones k y k+1.
  0 si no lo hace.

- $q_{jj'k}$

1 si el producto cambia de j a j' ((j,j') ∈ GG) entre las posiciones k y k+PPL.
0 si no lo hace.

La **función objetivo** a minimizar es la siguiente:

$$\sum_{k=1}^{n-1} \sum_{(i,i')\in CC} h_{ii'} p_{ii'k} + \sum_{k=1}^{n-PPL} \sum_{(j,j')\in GG} v_{jj'} q_{jj'k}$$

Y las **restricciones** son:

$$\sum_{(i,j)\in S} x_{ijk} = 1, \quad k = 1, \ldots, n,$$

$$\sum_{k=1}^{n} x_{ijk} = d_{(i,j)}, \quad \forall (i,j) \in S,$$

$$y_{ik} = \sum_{j:(i,j)\in S} x_{ijk}, \quad \forall i : \exists (i,j) \in S, k = 1, \ldots, n,$$

$$z_{jk} = \sum_{i:(i,j)\in S} x_{ijk}, \quad \forall j : \exists (i,j) \in S, k = 1, \ldots, n,$$

$$p_{ii'k} \geq y_{ik} + y_{i'k+1} - 1, \quad \forall (i,i') \in CC, k = 1, \ldots, n-1,$$

$$q_{jj'k} \geq z_{jk} + z_{j'k+PPL} - 1, \quad \forall (j,j') \in GG, k = 1, \ldots, n-PPL,$$

$$x_{ijk} \in \{0,1\}, \quad \forall (i,j) \in S, k = 1, \ldots, n,$$

$$0 \leq p_{ii'k} \leq 1, \quad \forall (i,i') \in CC, k = 1, \ldots, n-1,$$

$$0 \leq q_{jj'k} \leq 1, \quad \forall (j,j') \in GG, k = 1, \ldots, n-PPL$$

Su método mejoraba el definido en 2008 por García-Sabater, pero tenía limitaciones cuando se operaba en escalas industriales. Es decir, cuando el número de colores, geometrías y piezas por lote aumentaba: |C| ≥ 10, |G| ≥ 10 y |PPL| ≥ 50

Se ha realizado un estudio generando instancias aleatorias con determinados números de C (Colores), G (Geometrías) y PPL (Piezas Por Lote), para estudiar a partir de qué punto, se supera el tiempo computacional límite fijado sin encontrar el óptimo.

En el caso estudiado para demostrar las limitaciones del modelo matemático, se toma un *time limit* de 10 minutos → 600 segundos

Se toman como ejemplos significativos estas instancias, de diferente nivel de complejidad, y se ejecuta el algoritmo de Laguna para obtener el valor de la FO (Función Objetivo).

(**S**) Productos	(**N**) Demanda	(**PPL**) Piezas Por Lote	(**G**) Geometrías	(**C**) Colores	**TIEMPO** (segundos) utilizado hasta alcanzar el resultado	¿**Óptimo alcanzado**? (durante el *time limit*)	**FO**
5	30	10	3	5	2	SI	18
5	30	5	5	20	14	SI	36
5	50	25	3	20	108	SI	15

5	50	25	10	20	257	SI	19
5	50	25	20	20	600	NO	29
10	60	5	10	10	600	NO	84
10	100	50	10	10	600	NO	76
50	300	25	10	5	600	NO	1144
50	300	5	3	20	600	NO	2134
100	600	100	20	5	600	NO	5103
100	600	50	20	20	600	NO	6022
200	1200	100	20	10	600	NO	12039

Tabla 1 Resultados utilizando el algoritmo de Laguna.

Dentro de las características de cada instancia, se observa claramente como la más determinante a la hora de aumentar el tiempo de computación es la demanda ("N").

A partir de N=50 no se consigue alcanzar el óptimo dentro del *time limit*, así que es posible que este modelo no nos sirva para el caso que se estudia en este trabajo.

Ahí surge un dilema, ya que con este modelo no conseguimos en todos los casos alcanzar el óptimo global dentro del *time limit*. Pero… ¿Cómo saber si hay otra forma de resolver el problema descrito que dé mejores soluciones que las que ofrece este modelo matemático?

Habrá que estudiar la segunda alternativa, los enfoques heurísticos.

3.5. ENFOQUES HEURÍSTICOS.

Una heurística es un algoritmo que explora el espacio de soluciones buscando una solución aceptable en un tiempo de computación reducido. La heurística "guía" la búsqueda en el espacio de soluciones, buscando ignorar las zonas de este espacio donde prevé que estén las peores soluciones, y centrándose en buscar en las zonas donde se prevé que estén las mejores.

De esta forma, se consiguen buenas soluciones, aunque no se asegura alcanzar el óptimo global, reduciendo considerablemente el tiempo de computación total que conllevaría la enumeración completa de todas las soluciones para localizar la mejor de ellas.

En el caso de heurísticas utilizadas para problemas de secuenciación, se trata de una heurística constructiva, y se construye paso a paso la solución, eliminando en cada paso el candidato elegido de la función Greedy (la elección del candidato depende del criterio que se elija).

¿Qué es la función Greedy? Es la función en la que se incluyen todos los candidatos posibles y de la que se van eliminando conforme se incluyen en la solución. De esta forma, no se repiten los candidatos. El algoritmo se detiene cuando la función Greedy se queda vacía y la función solución está completa.

3.5.1 Tipos de enfoques heurísticos.

- **Inserción**: Se parte de una solución incompleta que puede haberse generado con alguna heurística sencilla (como, por ejemplo, la del vecino más corto o de manera aleatoria). A continuación, se van eligiendo elementos aún no asignados a la solución usando una determinada estrategia (Se seleccionan T elementos del conjunto W en la inicialización). La posición en la que menos incremente la función objetivo es la que se elige.
- **Algoritmo de Dijkstra** (Por ejemplo: Ruta más corta – Google Maps): Ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen al resto de vértices que componen el grafo, el algoritmo se detiene.
- **Barrido** (Por ejemplo: Viajante de comercio): Se basa en tomar un eje de referencia y "barrer" la figura en un sentido (en este ejemplo es el eje de abscisas y en sentido horario) para establecer un orden.

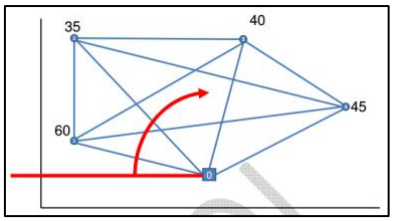

Fig. 5 Ejemplo visual heurística de barrido.

- **Clarke and Wright** (Por ejemplo: Rutas de reparto): Se busca conocer la cantidad de vehículos a usar y su ruta para atender una serie de demandas desde un depósito de distribución. Es un ejemplo de heurística constructiva en la que la función Greedy se basa en los ahorros que aparecen cuando se fusionan dos rutas.

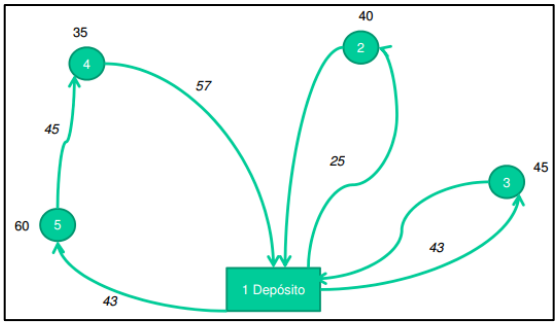

Fig. 6 Ejemplo visual heurística de Clarke and Wright.

POSIBLES MEJORAS E INCONVENIENTES.

Posible mejora:

Backtracking: En el caso de que se elija un candidato del Greedy para la solución y se pueda observar que la solución empeora, volver atrás y seleccionar otro candidato.

Posible inconveniente:

Miopía de la heurística: Es el principal problema de las heurísticas constructivas al ir añadiéndose sucesivamente elementos a la solución final de manera iterativa, es posible que las decisiones

tomadas en cada etapa, aunque buenas en ese momento según el criterio de selección, perjudiquen a la solución final.

3.6. ENFOQUES METAHEURÍSTICOS.

Una metaheurística es un procedimiento heurístico que, como su propio nombre indica, "meta" significa "más allá", busca ir más allá de los procedimientos heurísticos básicos para evitar quedar atrapados en óptimos locales. Son procedimientos heurísticos de alto rendimiento y de complejidad añadida, que se formulan para obtener mejores soluciones que las heurísticas básicas cuando estas por sí solas no son efectivas.

Las metaheurísticas crean nuevos algoritmos mixtos, combinando conceptos de diferentes campos de la biología, genética y física, entre otras.

3.6.1 Tipos de enfoques metaheurísticos.

Lo primero para entender los tipos de metaheurísticas es entender los procesos iterativos de búsqueda local, que realizan pequeñas perturbaciones en una solución, buscando mejores soluciones. En función de los tipos de perturbaciones, se clasifican en:

- **Intensificación**: Agitaciones pequeñas dentro del espacio de soluciones. Se busca alcanzar cada óptimo local o global.

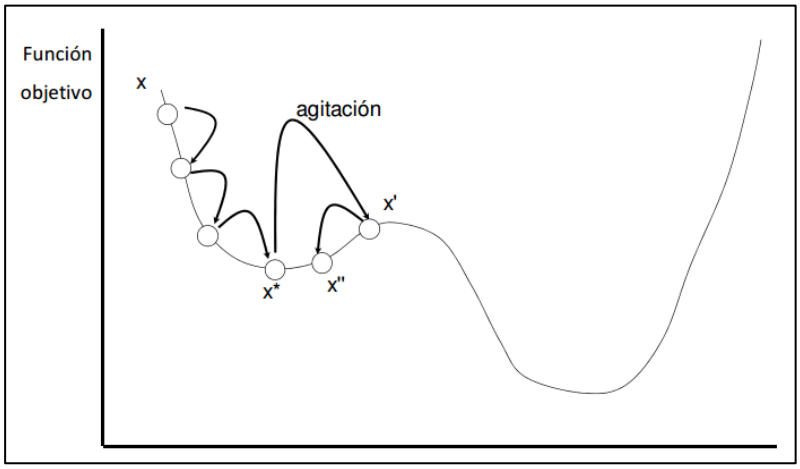

Fig. 7 Función representativa de la intensificación en los enfoques metaheurísticos.

- **Diversificación**: Agitaciones grandes dentro del espacio de soluciones. Se busca huir de los óptimos locales para explorar nuevas zonas donde se pueda encontrar el óptimo global.

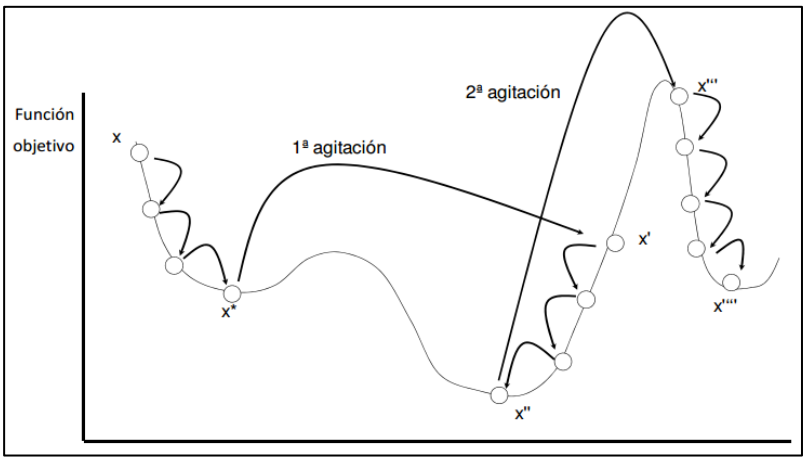

Fig. 8 Función representativa de la diversificación en los enfoques metaheurísticos.

A partir de ahí, las metaheurísticas se clasifican en

- Metaheurísticas sin memoria.
- Metaheurísticas con memoria.

La metaheurísticas sin memoria, es decir, no tienen en cuenta las decisiones previas a la hora de generar una nueva solución en del espacio de soluciones.

- **GRASP**: Se trata de una combinación de una heurística constructiva y una heurística de búsqueda local. Primero se genera una primera solución con la heurística constructiva y posteriormente, con la heurística de búsqueda local se busca mejorar esta solución con diferentes excitaciones de esta solución para moverse por el espacio de soluciones buscando el óptimo global. Tipos de búsqueda local:
  - **Búsqueda local iterativa**: Al llegar a un mínimo local, se realiza un salto brusco para salir del mínimo local y se vuelve a realizar el proceso de búsqueda del óptimo local.

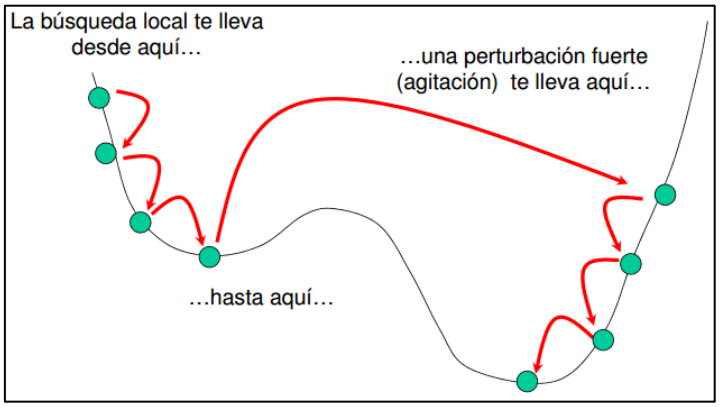

Fig. 9 Función representativa de la búsqueda local iterativa.

  - **Búsqueda por entorno variable**: Va cambiando los tipos de salto (no siempre iguales).

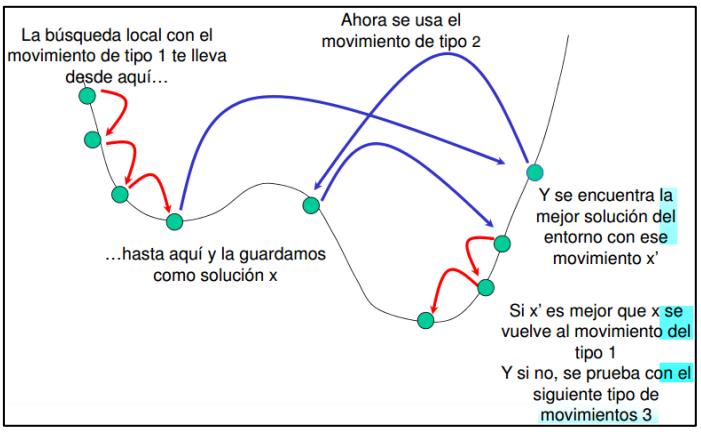

Fig. 10 Función representativa de la búsqueda local por entorno variable.

- **Recocido simulado**: Similar a los algoritmos anteriores, pero en el recocido simulado, en ocasiones se permite aceptar soluciones peores para seguir con ellas y poder mejorar las soluciones anteriores. Esta probabilidad de aceptar o rechazar soluciones peores va cambiando a lo largo de las iteraciones.

Las metaheurísticas con memoria, es decir, tienen en cuenta las decisiones previas a la hora de generar una nueva solución en el espacio de soluciones.

- **Búsqueda tabú**: Se basa en la búsqueda local, añadiéndole memoria y pudiendo realizarse movimientos de empeoramiento para escapar de óptimos locales y evitar recorridos cíclicos. Además, emplea procedimiento de reinicialización para mejorar la capacidad del algoritmo para la diversificación e intensificación de la búsqueda.
  Para ello se crea una "lista tabú" donde se incluyen las soluciones anteriores. Esta lista no mantiene eternamente las soluciones, sino que un valor denominado "tenencia tabú" marca cuánto tiempo estas soluciones permanecen en la lista antes de ser eliminadas.

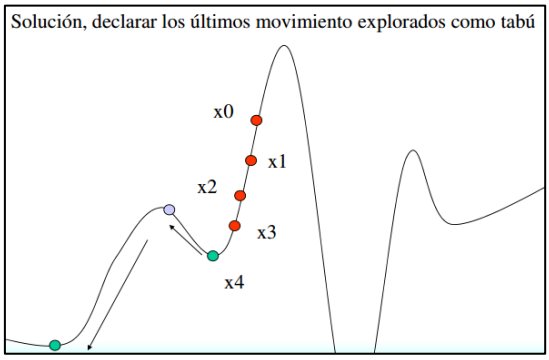

Fig. 11 Función representativa de la búsqueda tabú.

- **Algoritmo genético**: Plantean el proceso de optimización como una serie de modificaciones en paralelo sobre una población o conjunto de soluciones en vez de hacerlo sobre una solución única. A partir de estas soluciones podemos crear otras mediante modificaciones. Las modificaciones sobre las soluciones se realizan tomando características de varias soluciones para crear nuevas soluciones que tengan similitudes con las soluciones de partida.

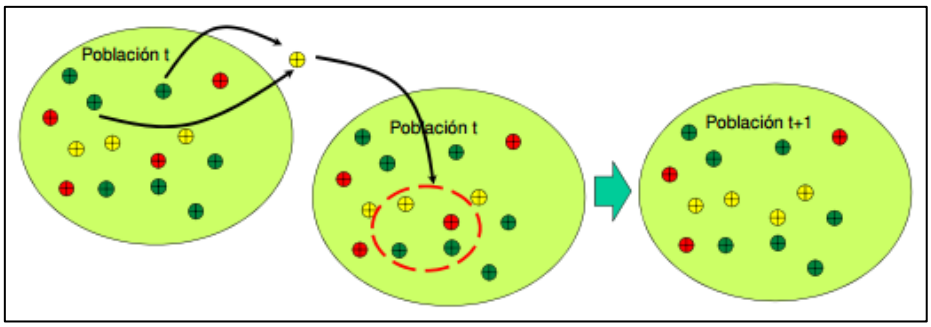

Fig. 12 Ejemplo del algoritmo genético.

POSIBLES MEJORAS E INCONVENIENTES.

Posible mejora:

Memoria: Añadiendo memoria a la heurística se puede evitar volver a las zonas del espacio de soluciones que ya se hayan explorado y de este modo, acercarse de forma más directa al óptimo global.

Posible inconveniente:

Tiempo de computación elevado: Cuanto más compleja es la heurística, y en especial si se le añade memoria a esta, mayor es el tiempo de computación.

3.7. CONCLUSIONES.

Por tanto, al tratarse de un problema de tal complejidad, y en el que el algoritmo de Laguna no permite obtener buenos resultados cuando la instancia se vuelve más elevada, habría que recurrir a métodos heurísticos y metaheurísticos que nos permitan obtener buenas soluciones, aunque no sean óptimas, en tiempos de computación reducidos.

CAPÍTULO 4. DISEÑO DE UNA HERRAMIENTA BASADA EN EXCEL PARA RESOLVER EL PROBLEMA.

4.1. INTRODUCCIÓN.

En este capítulo, se va a analizar qué software se ha seleccionado para ejecutar la herramienta que resuelva el problema, y los motivos por los cuales se ha tomado esa decisión, así como muestras del software y de la interfaz de los datos de entrada y salida.

4.2. SELECCIÓN DEL SOFTWARE.

El objetivo de este trabajo es conseguir una herramienta que se pueda utilizar sin necesidad de invertir un alto presupuesto en ninguna aplicación u ordenador. Por tanto, en el caso de trabajar con heurísticas y metaheurísticas, necesitaríamos que esta herramienta estuviese accesible prácticamente desde cualquier ordenador.

Por ello se decide trabajar con el Excel, una aplicación de Microsoft Office disponible prácticamente en cualquier puesto de trabajo que disponga de ordenador. También la facilidad de entrada y salida de datos favorece esta elección, ya que, de este modo, cualquier persona mínimamente informada sobre la herramienta podría utilizarla para este fin.

Para poder crear macros en Excel (es decir, procedimientos paso a paso escritos en Visual Basic) se debe saber programar en Visual Basic, que es un lenguaje de programación basado en eventos que se creó para facilitar la programación y hacerla más accesible.

Sin embargo, para poder utilizar las macros, no es necesario saber programar en Visual Basic, ya que se puede introducir los datos de forma sencilla en simples celdas de Excel, siempre y cuando esté bien definida la macro.

Por tanto, si se define una macro que dé buenos resultados para un amplio abanico de instancias, se puede utilizar posteriormente por personas que no sepan de programación, simplemente introduciendo datos y activando las macros. De esta forma, se consigue una herramienta eficaz para uso industrial en cualquier ámbito y con una inversión mínima.

4.3. PROCEDIMIENTO DE CREADO DEL ENTORNO DE LA HERRAMIENTA EN EL SOFTWARE SELECCIONADO.

Lo primero que se hizo fue crear un generador de instancias. Con su ayuda, se generan 2339 instancias diferentes tomando valores de las variables que afectan al problema

(S) – Tipos de productos diferentes → Desde 3 hasta 200.

(N) – Demanda total → Desde 20 a 1200.

(PPL) – Piezas por vuelta del carrusel (Parts Per Loop) → Desde 10 hasta 100.

(G) – Geometrías diferentes → Desde 3 hasta 20.

(C) – Colores diferentes → Desde 3 hasta 20.

Dentro de cada instancia, se generan aleatoriamente los costes de setup por cambio de color y geometría tomando valores de entre 1 y 10. También se generan aleatoriamente las demandas y los tipos de producto. A continuación, en el pantallazo mostrado se puede observar la **interfaz en Excel del generador de instancias** y un ejemplo.

	1	2	3	4	5	6	7	8	9	10
1	NumProd	5	Replica num	1	Name	I_S005_N0030_C020_G005_PPL005_SYFalse_Cm1_CM10_Rep001.csv				
2	NumColor	20								
3	NumGeom	5								
4	PPL	5								
5	Demand	30								
6	Cost Symmetr	False								
7	Product NÂº	Color	Geometry	Demand						
8	0	6	2	11						
9	1	3	4	9						
10	2	16	1	1						
11	3	14	3	8						
12	4	18	1	1						
13	Cost Color Changeover									
14		0	2	2	8	10	5	7	3	1
15		10	0	9	9	3	10	2	4	5
16		2	6	0	7	2	5	8	4	1
17		9	9	1	0	3	2	6	6	6
18		4	5	7	2	0	7	1	1	2
19		5	6	9	3	1	0	7	10	8
20		5	2	1	5	8	8	0	4	8
21		6	5	7	3	8	2	8	0	10
22		2	6	7	10	9	1	6	8	0
23		4	3	10	8	10	3	5	7	3
24		9	2	3	4	4	3	7	2	9
25		8	7	4	5	3	6	2	3	3
26		7	6	4	1	7	1	1	9	9
27		1	9	2	1	4	4	1	5	5
28		7	4	3	7	4	7	10	5	6
29		10	8	6	7	6	4	2	5	9
30		9	4	5	9	5	4	6	9	1
31		7	3	2	3	2	2	4	5	3
32		2	3	1	6	2	8	5	3	3
33		4	8	2	9	3	9	6	2	7
34	Cost Color Changeover of each Product									
35		0	5	9	2	5				
36		6	0	3	5	4				
37		6	9	0	7	2				
38		10	7	1	0	5				
39		5	6	10	6	0				
40	Cost Geometry Changeover									
41		0	1	9	10	6				
42		3	0	9	3	1				
43		3	1	0	6	1				
44		4	7	1	0	1				
45		7	3	5	7	0				
46	Cost Geometry Changeover per each product									
47		0	1	1	6	1				
48		5	0	3	7	3				
49		9	1	0	3	0				
50		1	1	7	0	7				
51		9	1	0	3	0				

Fig. 13 Generador de instancias.

- NumProd: Número de productos diferentes (**S**)
- NumColor: Número de colores (**N**).
- NumGeo: Número de geometrías (**G**).
- **PPL**: Piezas por vuelta del carrusel (Parts Per Loop).
- Demanda: Demanda total (**N**).

Después de estos valores, aparece la parte que genera aleatoriamente el generador de instancias: esto es la instancia en sí. El generador aleatoriamente crea una instancia en la que se incluyen 5 tipos de productos, donde en total haya 20 de demanda total, y cada producto aleatoriamente tenga un color y una geometría (no es necesario que todos los colores disponibles estén representados, y se puedan repetir). Una vez generada la instancia, aparece cada tipo de producto con su demanda, color y geometría.

Tras esto, el generador crea una matriz de costes de cambio para todas las geometrías y los colores. Y debajo de cada una de estas genera una matriz de los costes asociados para los colores y geometrías que están incluidos en nuestra instancia.

Una vez ya se tienen las instancias generadas, se pasan los datos a la herramienta fundamental del proyecto, y su interfaz es la siguiente:

	Numero de modelos	Geometrías	Colores	Posiciones a programar	PPL		Generar Solución
	5	3	5	30	10		
			Geometría	H	2	4	1 3 1
			Color	K	6	3	16 14 18
			Demanda	Q	11	9	1 8 1
			Código de modelo		1	2	3 4 5 6 7 8 9 10 11 12 13 14 15 16
	Cambios geometría	Cambios colores	Cambios Totales	Coste cambios geometría	Coste cambios colores	Coste total	COSTE MÍNIMO
	Solución						

Principal | Coste geometria | Coste colores | Enum | Hoja3

Fig. 14 Interfaz de la herramienta en Excel.

DATOS DE ENTRADA:

En la pestaña principal, la herramienta solicita los valores de la instancia (N, S, PPL, C y G), y la instancia en sí, es decir, los productos con sus colores y geometrías asociados.

Una vez lo introducimos, en las pestañas “Costes geometría” y “Costes colores” se introducen las matrices de costes por cambio de color y geometría.

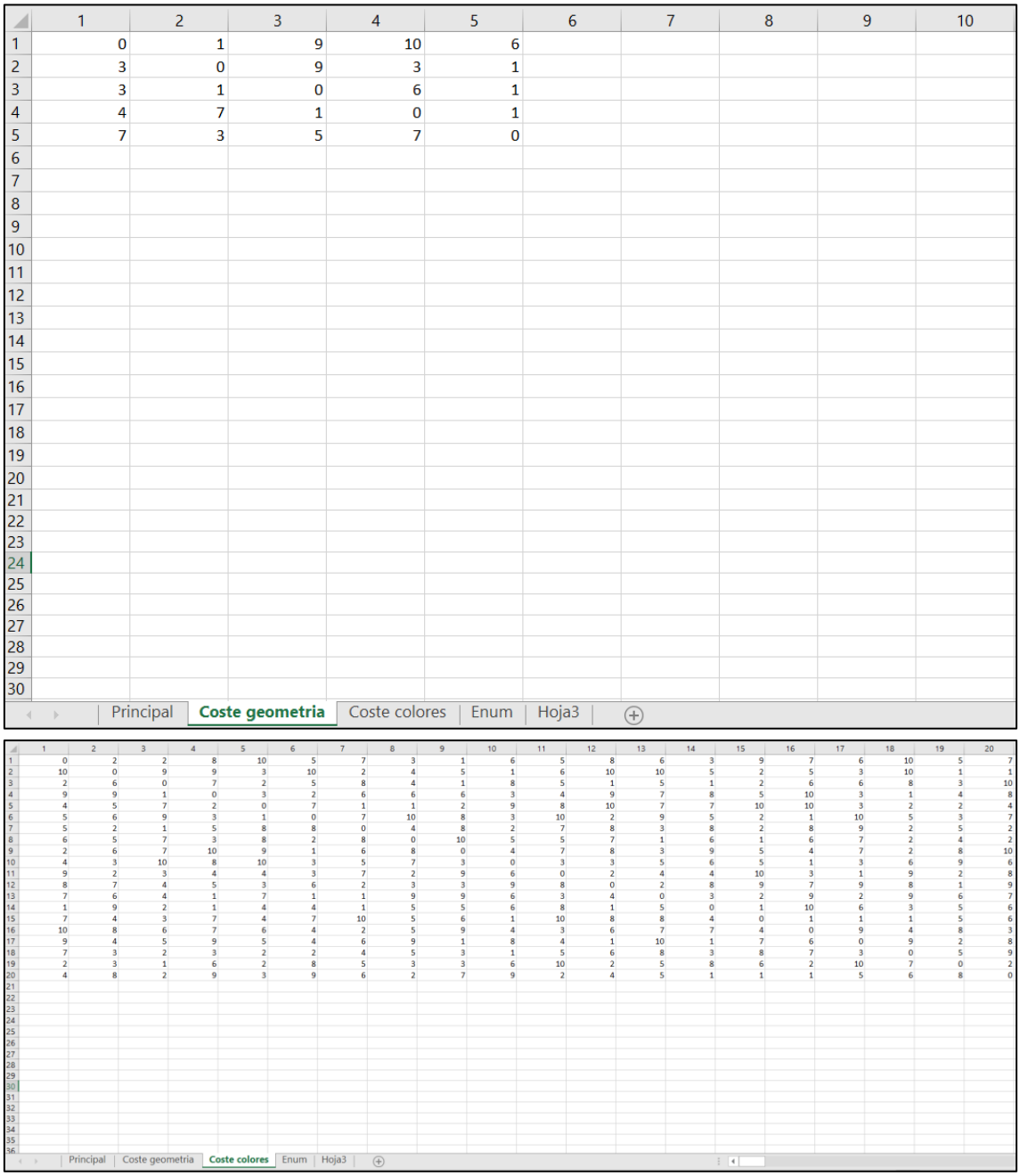

Fig. 15 Matrices en Excel de costes por cambio de color y geometría.

Y el paso posterior ya es ejecutar la herramienta pulsando el botón de “GENERAR SOLUCIÓN”, y los resultados se muestran de forma visual de la siguiente forma:

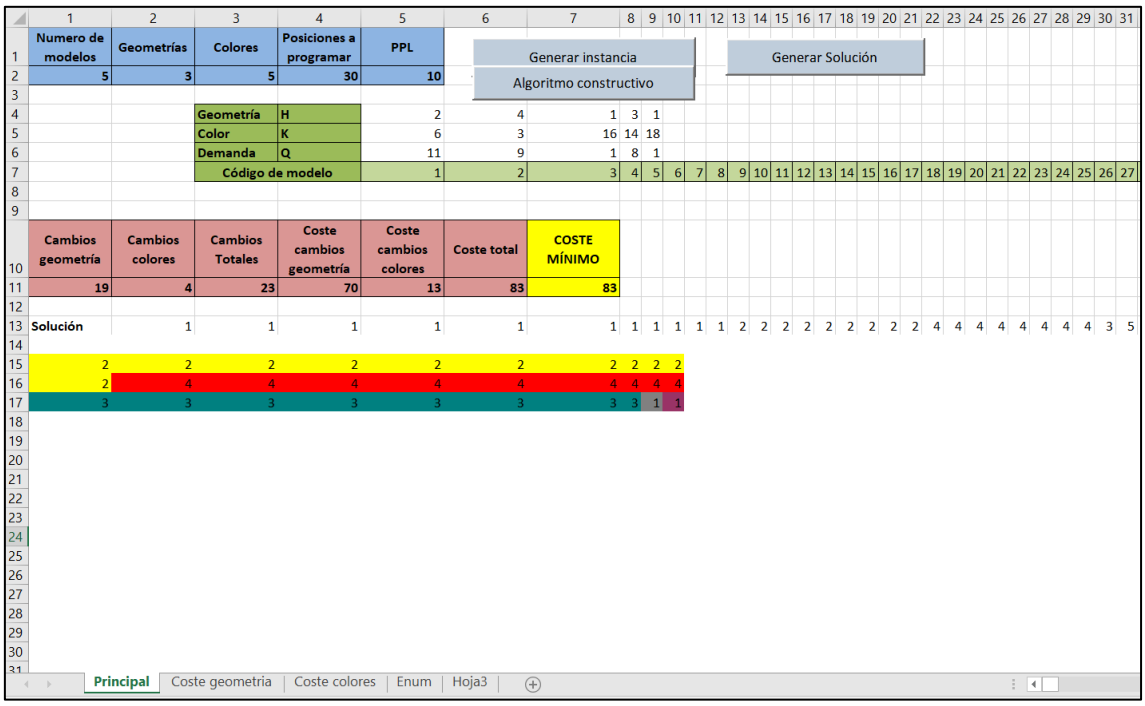

Fig. 16 Interfaz con la salida de datos de la herramienta en Excel.

DATOS DE SALIDA

- **Tabla de cambios y costes**: En primer lugar, aparece una tabla en la que se identifican los cambios de colores y geometrías totales, junto con sus costes de cambio asociados en total. A la derecha aparece el coste total, y a la derecha del todo, aparece el coste mínimo, que refleja el coste total obtenido en todas las iteraciones realizadas sobre esa instancia, aunque en este caso, al no haber iterado, aparece el mismo que el total.
- **Solución por códigos de productos**: En segundo lugar, vemos en la fila 13 "Solución", todos los productos con su orden de montaje
- **Solución por geometrías y colores (matriz)**: A partir de la fila 15, vemos los productos ya ordenados por cada loop. Muestra el color asociado a cada producto y muestra el número de geometría.

De este modo, es muy sencillo identificar la distribución, los cambios de color y los cambios de geometría de forma visual.

En el caso real, en el que la instancia no viene generada por un generador aleatorio sino por una solicitud del cliente o de la propia empresa, la introducción de la instancia se haría directamente en la interfaz de la herramienta. En el caso de este proyecto se utiliza el generador para poder tener instancias aleatorias que cubran todo el espacio de posibles solicitudes de este problema.

Como se ha observado, la herramienta presenta una interfaz muy sencilla de cara a la introducción de datos e interpretación de resultados. Esta, junto con la accesibilidad global al Excel desde casi cualquier ordenador, son las razones que han supuesto la decisión de utilizar Excel para implementar la herramienta.

4.4. CONCLUSIONES.

Por tanto, decidimos emplear este software para implementar la herramienta por las razones siguientes:

- Software disponible en prácticamente cualquier ordenador.
- Interfaz sencilla, y con la que cualquier persona se familiarizaría sin necesidad de periodo de adaptación.
- Facilidad de entrada de datos.
- Facilidad de lectura de los datos de salida.
- Lectura gráfica y visual de la solución.
- Facilidad de transferencia de datos a aplicaciones externas, y viceversa.

CAPÍTULO 5. DISEÑO DE HEURÍSTICAS Y ESTUDIO EXPERIMENTAL.

5.1. INTRODUCCIÓN.

En este capítulo se va a buscar una herramienta basada en heurísticas que nos permita encontrar buenas soluciones en tiempos de computación reducidos, mejorando los resultados del algoritmo de Laguna en instancias elevadas.

Una vez conocidos los 4 tipos de heurísticas que se han mencionado anteriormente, el algoritmo de Dijkstra, el de Clarke and Wright y el de barrido, no resultan adecuados teniendo en cuenta que se trata de un problema de secuenciación. Por tanto, se trabajará con el cuarto tipo de heurística, la de inserción. Se trata, por tanto, de una heurística constructiva.

Es importante recordar que cuando se habla de "vector solución", se habla de un vector en el que se introducen en orden todos los valores de la solución que aporta la heurística. Sin embargo, cuando se habla de "matriz solución", se habla de la distribución que tendrá la matriz solución de la forma en la que se haya rellenado esta con el vector solución. Es decir, dos vectores solución iguales podrían tener diferentes matrices solución si la forma de rellenado de dicha matriz es diferente (monosentido/bisentido).

En este tipo de heurísticas se va rellenando el vector solución siguiendo un criterio definido, que se basa en las características de cada instancia (color, geometría y demanda), y sus dos variantes de rellenado de la matriz de la solución final (monosentido/bisentido) marcan la distribución que tendrá este vector solución en la matriz solución final.

Ejemplos visuales de dos matrices solución. Una con rellenado monosentido y otra con rellenado bisentido:

MONOSENTIDO

3	3	3	3	3
3	3	3	1	2
2	2	2	2	2
2	2	2	2	2
1	4	4	4	4
4	4	4	4	4

Tabla 2 Ejemplos de una matriz solución con rellenado monosentido.

BISENTIDO

→	4	4	4	4	4
←	1	4	4	4	4
→	3	3	3	3	3
←	2	2	3	3	3
→	2	2	2	2	2
←	1	2	2	2	2

Tabla 3 Ejemplos de una matriz solución con rellenado monosentido.

Las condiciones tanto para las instancias como para la solución son las siguientes:

- Costes de setup: Asimétricos.
- Agrupación: Por bloques de todos los productos iguales (código de producto igual, es decir, mismo color y geometría).
- Consideraciones físicas: Sin huecos.
- Rellenado: Es lo que va variando entre cada una de las 6 heurísticas que se definen. Tanto el rellenado del vector solución como el rellenado de la matriz solución.

En primer lugar, se definen 6 diferentes heurísticas sin saber, a priori, su eficiencia. De este modo, posteriormente se analizarán los resultados y se extraerán conclusiones.

Heurística Nº1 "MS-Dem" – **Monosentido** y por orden de **demanda.**

Heurística Nº2 "MS-Col"– **Monosentido** y por orden de **colores.**

Heurística Nº3 "MS-Geo" – **Monosentido** y por orden de **geometría.**

Heurística Nº4 "BS-Dem" – **Bisentido** y por orden de **demanda.**

Heurística Nº5 "BS-Col" – **Bisentido** y por orden de **colores.**

Heurística Nº6 "BS-Geo" – **Bisentido** y por orden de **geometría.**

Cuando se trata de orden de **demanda**, se insertan los productos en el vector solución de la siguiente forma: De productos de mayor de manda a menor demanda. En caso de misma demanda, se sigue el orden cronológico.

1. De mayor a menor por volumen de demanda de cada producto.
2. Orden lexicográfico por código de producto.

Cuando se trata de orden de **colores**, se insertan los productos en el vector solución de la siguiente forma: En primer lugar, se crea un vector auxiliar en el que se introduce la demanda de cada color, posteriormente se introducen los productos que tengan colores con mayor demanda progresivamente. Dentro de cada color, se ordenan los productos de mayor a menor demanda, y en el caso de haber coincidencia en alguno de estos dos criterios, se recurre al orden cronológico. Por tanto, el orden de los criterios para seleccionar las preferencias es el siguiente:

1. De mayor a menor por volumen de demanda de cada color.
2. De mayor a menor por volumen de demanda de cada producto.
3. Orden lexicográfico por código de producto.

Cuando se trata de orden de **geometrías**, se insertan los productos en el vector solución de la siguiente forma: En primer lugar, se crea un vector auxiliar en el que se introduce la demanda de cada geometría, posteriormente se introducen los productos que tengan geometrías con mayor demanda progresivamente. Dentro de cada geometría, se ordenan los productos de mayor a menor demanda, y en el caso de haber coincidencia en alguno de estos dos criterios, se recurre al orden cronológico. Por tanto, el orden de los criterios para seleccionar las preferencias es el siguiente:

1. De mayor a menor por volumen de demanda de cada geometría.
2. De mayor a menor por volumen de demanda de cada producto.
3. Orden lexicográfico por código de producto.

5.2. DIAGRAMA DE PROCESO.

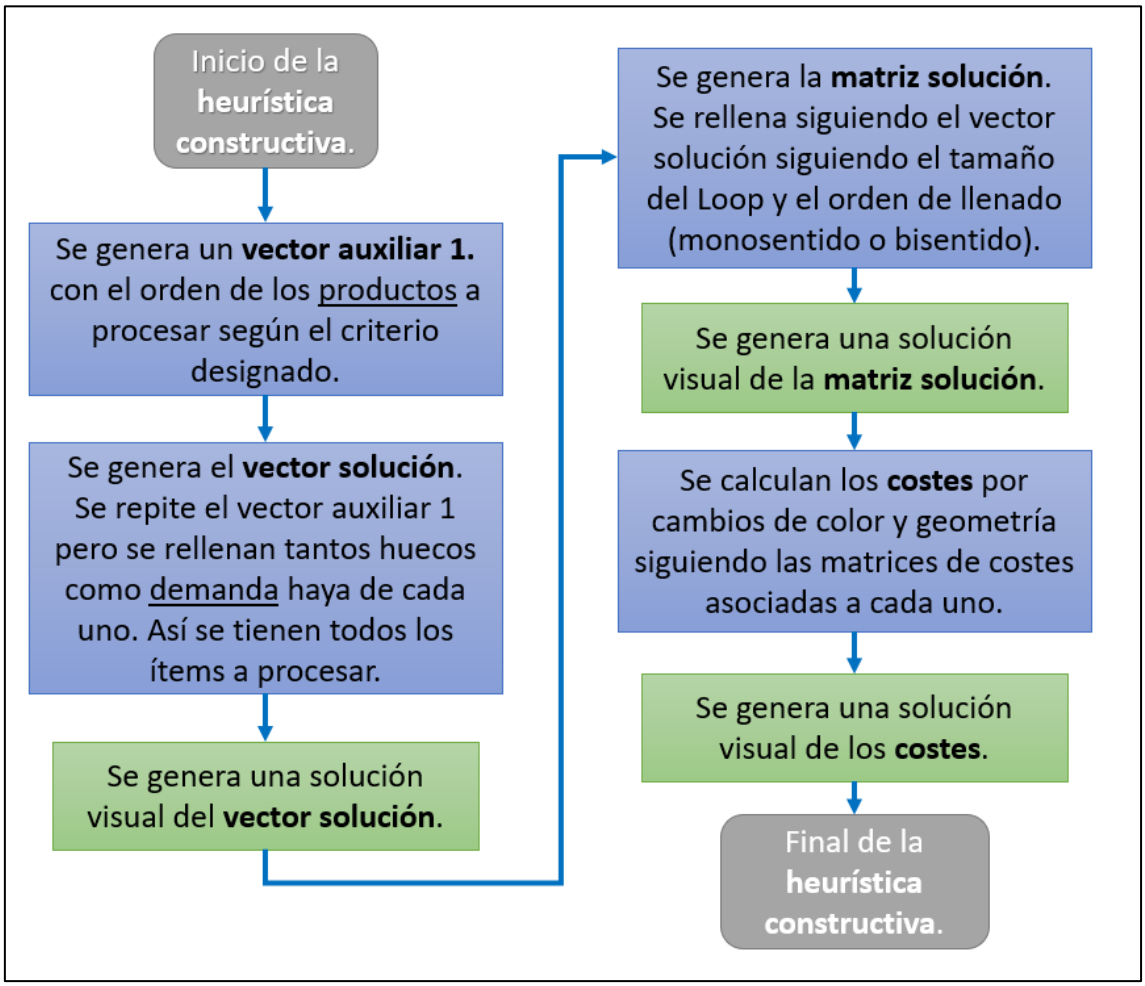

Fig. 17 Diagrama de proceso de las heurísticas diseñadas.

5.3. DISEÑO DE LAS HEURÍSTICAS EMPLEADAS.

Una vez conocido el diagrama de proceso y qué es lo que hace la herramienta y cómo debe hacerlo, se va a proceder a diseñar estas heurísticas.

Como las 6 heurísticas propuestas son muy similares, se va a tratar el diseño de una de ellas, y en las partes del código donde haya variación entre ellas, se mencionará.

a) Se introducen los datos de la instancia como "datos de entrada".
b) Se almacena en un vector auxiliar las demandas de cada color, para luego poder detectar el mínimo y el máximo de demanda de cada color y así poder posteriormente "barrer" entre esos dos valores seleccionando los colores de mayor a menor demanda y establecer en otro vector auxiliar el orden de productos por colores de mayor a menor demanda. En este caso se trata de la heurística cuyo primer criterio es la de criterio por colores de mayor demanda. En las otras sería igual, pero cambiando los criterios.

```
' 1º Mayor DEMANDA por COLOR
' 2º Mayor DEMANDA por PRODUCTO
' 3º Lexicográfico
        i = 0
        j = 0
        valorcriterio = demmax
        hueconumero = 1

' Almacenar en el vector Demandacolores() las demandas acumuladas de cada color
        For i = 0 To N_Colores - 1
            Demandacolores(i + 1) = 0
        Next

        For j = 0 To N_Colores - 1
            For i = 1 To N_Modelos
                If Instancia(2, i) = j Then
                        Demandacolores(j + 1) = Demandacolores(j + 1) + Instancia(3, i)
                End If
            Next
        Next

' Bucle que detecta el MIN y el MÁX de los COLORES
        For i = 0 To N_Colores - 1
            If Demandacolores(i + 1) < colmin Then
                    colmin = Demandacolores(i + 1)
            End If
            If Demandacolores(i + 1) > colmax Then
                    colmax = Demandacolores(i + 1)
            End If
        Next

'Bucle donde se colocan los COLORES por orden de mayor a menor DEMANDA
        color = colmax
        colorhueco = 1
        For i = colmin To colmax
            For j = 0 To N_Colores - 1
                If Demandacolores(j + 1) = color Then
                        Ordencolor(colorhueco) = j
                        colorhueco = colorhueco + 1
                End If
            Next
            color = color - 1
        Next
```

Fig. 18 Código de la heurística diseñada (I).

c) Dentro de cada color, se subdivide por la demanda de cada producto, ordenándose de esta forma de mayor a menor, como indica el 2º criterio. En el caso de que haya varios iguales, se sigue el tercer criterio, el lexicográfico por número de producto. Posteriormente se genera el **vector solución** y se representa visualmente.

```
'colmin es la demanda total de cada color y colmin2 es la demanda local de cada producto
        For i = 1 To N_Colores
            'Bucle para detectar el colmin2 y colmax2 de cada color
                For j = 1 To N_Modelos
                        If Instancia(3, j) < colmin2 And Instancia(2, j) = Ordencolor(i) Then
                                colmin2 = Instancia(3, j)
                        End If
                        If Instancia(3, j) > colmax2 And Instancia(2, j) = Ordencolor(i) Then
                                colmax2 = Instancia(3, j)
                        End If
                Next

'Bucle que escribe en Solución2
                valorcriterio = colmax2
                For l = colmin2 To colmax2
                    For k = 1 To N_Modelos
                        If Instancia(3, k) = valorcriterio And Instancia(2, k) = Ordencolor(i) Then
                            Solucion2(hueconumero) = k
                            hueconumero = hueconumero + 1
                        End If
                    Next
                    valorcriterio = valorcriterio - 1
                Next

        Next

        For hij = colmin To colmax
            For i = 1 To N_Modelos
                If Instancia(3, i) = valorcriterio Then
                        Solucion2(hueconumero) = i
                        hueconumero = hueconumero + 1
                End If
            Next
            valorcriterio = valorcriterio - 1
        Next
```

Fig. 19 Código de la heurística diseñada (II).

d) A través del vector solución y siguiendo los criterios de rellenado de la matriz, se crea un bucle que genera la solución visual, es decir, la **matriz solución**. En este caso, sí que hay diferencia de unas heurísticas a otras, ya que en las bisentido se rellena de forma distinta. Se muestran ambos códigos con su diferenciación.

Monosentido:

```
Dim NLoops
NLoops = N_Posiciones / PPL '(que sea entero)

x = 0

    For a = 1 To NLoops
        For b = 1 To PPL
            x = x + 1
            Cells(14 + a, b) = Instancia(1, solucion(x))
            Cells(14 + a, b).Interior.ColorIndex = Instancia(2, solucion(x))
        Next
    Next
```

Fig. 20 Código de la heurística diseñada (III).

Bisentido:

```
For a = 1 To NLoops
    If abc = 1 Then
    'Fila impar
            For b = 1 To PPL
                Cells(14 + a, b) = Instancia(1, solucion(x))
                Cells(14 + a, b).Interior.ColorIndex = Instancia(2, solucion(x)) + 1
                x = x + 1
            Next
        abc = 0
        n = n + 1
    Else
    'Fila par (RELLENAR AL REVÉS)
            For c = 0 To (PPL - 1)
                JAVI = PPL - c
                Cells(14 + a, JAVI) = Instancia(1, solucion(x))
                Cells(14 + a, JAVI).Interior.ColorIndex = Instancia(2, solucion(x)) + 1
                x = x + 1
            Next
        abc = 1
        n = n + 1
    End If
Next
```

Fig. 21 Código de la heurística diseñada (IV).

e) Una vez ya se tiene la matriz solución, se procede a calcular los cambios de geometrías, colores y totales, y sus respectivos costes, obteniendo así el **coste total**. En la interfaz se representan estos valores para que sean fácilmente identificables por el usuario.
Los cambios de geometría se detectan sobre la propia matriz solución gráficamente, ya que son costes por posiciones, en posiciones de filas (loops) consecutivas, y al haber dos tipos de rellenado, sería muy complejo realizarlo analizando el vector solución. Los cambios de color se detectan analizando el vector solución, ya que son costes por posiciones consecutivas.

```
'COSTES
Dim CamCol As Integer
Dim CamGeo As Integer
CamGeo = 0
CamCol = 0
CosteGeo = 0
CosteCol = 0

For d = 1 To (N_Posiciones - 1)
    If Instancia(2, solucion(d)) <> Instancia(2, solucion(d + 1)) Then
        CamCol = CamCol + 1
        CosteCol = CosteCol + Worksheets("Coste colores").Cells(1 + Instancia(2, solucion(d)), 1 + Instancia(2, solucion(d + 1))).Value
    End If
Next

For e = 1 To (N_Posiciones - PPL)
    If Instancia(1, solucion(e)) <> Instancia(1, solucion(e + PPL)) Then
        CamGeo = CamGeo + 1
        CosteGeo = CosteGeo + Worksheets("Coste geometria").Cells(1 + Instancia(1, solucion(e)), 1 + Instancia(1, solucion(e + PPL))).Value
    End If
Next

    Worksheets("Principal").Cells(11, 1).Value = CamGeo
    Worksheets("Principal").Cells(11, 2).Value = CamCol
    Worksheets("Principal").Cells(11, 3).Value = CamCol + CamGeo
    Worksheets("Principal").Cells(11, 4).Value = CosteGeo
    Worksheets("Principal").Cells(11, 5).Value = CosteCol
    Worksheets("Principal").Cells(11, 6).Value = CosteGeo + CosteCol

Dim SolAuxiliar()
ReDim SolAuxiliar(N_Posiciones)

    If Cells(11, 6) < Costemin Then
        Costemin = Cells(11, 6)
        Cells(11, 7) = Costemin
        f = 1
        For f = 1 To N_Posiciones
            SolAuxiliar(f) = solucion(f)
        Next
        contador = 0
    Else
        contador = contador + 1
        r = r + 1
    End If
```

Fig. 22 Código de la heurística diseñada (V).

f) Se reflejan en la interfaz todos los datos relevantes (**vector solución**, **matriz solución**, **cambios** y **costes**)

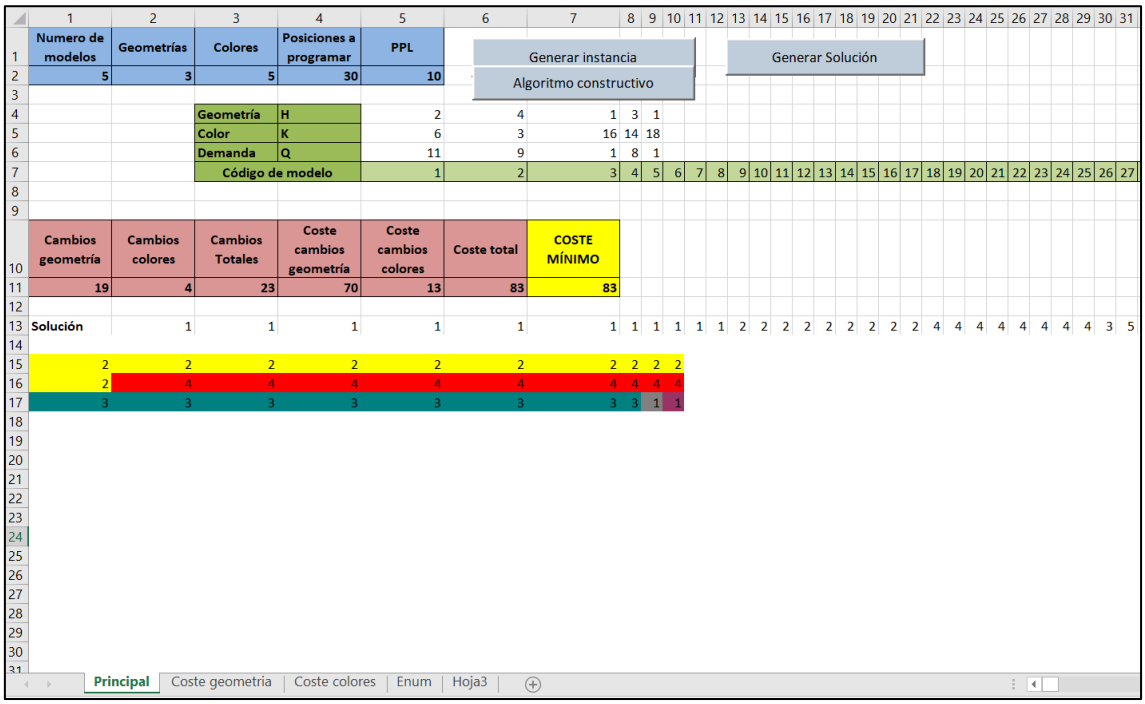

Fig. 23 Interfaz con la salida de datos de la heurística diseñada.

5.4. ESTUDIO EXPERIMENTAL.

El primer análisis de resultados se realiza comparando los resultados de las heurísticas en conjunto con los resultados obtenidos con el algoritmo de Laguna. Posteriormente, se compararán los resultados de cada heurística entre ellos.

Se observan en la tabla inferior los resultados de las heurísticas constructivas realizadas y los resultados ofrecidos para cada una de las 12 instancias.

En cuanto al tiempo de computación, en todos los casos es menor de 5 segundos, por lo que el tiempo es muy inferior a los 3600 segundos (*time limit*) que se tarda con el algoritmo de Laguna en alcanzar los resultados anteriormente mencionados.

						(MS-Dem)	(MS-Col)	(MS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
5	**30**	**10**	3	5	**18**	**89**	**91**	**80**
5	**30**	**5**	5	20	**36**	**67**	**67**	**67**
5	**50**	**25**	3	20	**15**	**205**	**205**	**241**
5	**50**	**25**	10	20	**19**	**125**	**125**	**123**

5	**50**	**25**	20	20	**29**	**189**	**189**	**189**
10	**60**	**5**	10	10	**84**	**156**	**138**	**98**
10	**100**	**50**	10	10	**76**	**341**	**300**	**297**
50	**300**	**25**	10	5	**1144**	**1482**	**1648**	**1229**
50	**300**	**5**	3	20	**2134**	**836**	**748**	**310**
100	**600**	**100**	20	5	**5103**	**2689**	**2340**	**3195**
100	**600**	**50**	20	20	**6022**	**3496**	**3090**	**3056**
200	**1200**	**100**	20	10	**12039**	**6468**	**6056**	**5969**

Tabla 4 Resultados del estudio experimental de las heurísticas monosentido diseñadas.

						(BS-Dem)	(BS-Col)	(HS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
5	**30**	**10**	3	5	**18**	**107**	**77**	**98**
5	**30**	**5**	5	20	**36**	**86**	**86**	**86**
5	**50**	**25**	3	20	**15**	**199**	**199**	**244**
5	**50**	**25**	10	20	**19**	**60**	**60**	**123**
5	**50**	**25**	20	20	**29**	**167**	**167**	**167**
10	**60**	**5**	10	10	**84**	**160**	**147**	**119**
10	**100**	**50**	10	10	**76**	**360**	**332**	**214**
50	**300**	**25**	10	5	**1144**	**1387**	**1351**	**1209**
50	**300**	**5**	3	20	**2134**	**797**	**808**	**474**
100	**600**	**100**	20	5	**5103**	**2528**	**2068**	**3024**
100	**600**	**50**	20	20	**6022**	**3286**	**2825**	**2628**
200	**1200**	**100**	20	10	**12039**	**7050**	**5840**	**5401**

Tabla 5 Resultados del estudio experimental de las heurísticas bisentido diseñadas

Se identifica que en las instancias de a partir de N=300, es decir, 300 de demanda, los resultados ofrecidos por las heurísticas constructivas mejoran los resultados del algoritmo de Laguna, ya que este dentro del *time limit* no consigue alcanzar el óptimo global, y además conforme aumenta la N, los resultados que da el algoritmo dentro del *time limit* cada vez se van alejando más del óptimo global.

Tras ese análisis global de resultados en el que se ha comprobado que hay un límite a partir del cual estas heurísticas mejoran los resultados del algoritmo de Laguna, se van a analizar los resultados de cada una de las instancias (las que mejoran al algoritmo citado) para resolver cuáles son los factores que hacen que unas heurísticas ofrezcan mejores soluciones que otras.

Instancia Nº 9

						(MS-Dem)	(MS-Col)	(MS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
50	**300**	**5**	3	20	**2134**	**836**	**748**	**310**

Tabla 6 Resultados del estudio experimental de las heurísticas monosentido diseñadas para la instancia 9.

						(BS-Dem)	(BS-Col)	(HS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
50	**300**	**5**	3	20	**2134**	**797**	**808**	**474**

Tabla 7 Resultados del estudio experimental de las heurísticas bisentido diseñadas para la instancia 9.

La matriz solución sería una matriz de 60 filas por tan sólo 5 columnas. Analizando esta instancia antes de haber ejecutado la heurística, podríamos haber llegado a la conclusión de que, al tratarse de loops muy cortos, los lotes de productos de mismas geometrías y distintos colores, si fuesen consecutivos podrían minimizar en gran medida los costes por cambio de geometría. De este modo, al ejecutar la heurística podemos comprobar experimentalmente que así es, ya que la mejor heurística en este caso es la de ordenación de mayor a menor de productos por demanda de geometría (MS-Geo y BS-Geo).

Sin embargo, a priori habría sido difícil haber deducido entre la monosentido o la bisentido cuál iba a ser más efectiva. Experimentalmente se comprueba que la monosentido, ya que la bisentido, al ser loops muy cortos, produce mucho corte de lotes de mismo color al pasar de una fila a la siguiente, y no lo compensa reduciendo tanto coste por cambio de geometría como podría hacer en otras instancias.

Instancia Nº 10

						(MS-Dem)	(MS-Col)	(MS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
100	**600**	**100**	20	5	**5103**	**2689**	**2340**	**3195**

Tabla 8 Resultados del estudio experimental de las heurísticas monosentido diseñadas para la instancia 10.

						(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
100	**600**	**100**	20	5	**5103**	**2528**	**2068**	**3024**

Tabla 9 Resultados del estudio experimental de las heurísticas bisentido diseñadas para la instancia 10.

La matriz solución sería una matriz de 100 columnas por tan sólo 6 filas. En este caso, analizando esta instancia antes de ejecutar la heurística, podríamos observar que los loops son muy largos, es decir, que la mayoría de cambios serán en horizontal, por colores. Y experimentalmente se comprueba esto mismo, ya que las heurísticas cuyos criterios de ordenación son por productos de mismos colores, de mayor a menor demanda, es la que ofrece mejores resultados. En este caso, ofrece mejores resultados la heurística bisentido dentro de las de ordenación por color.

Instancia Nº 11

						(MS-Dem)	(MS-Col)	(MS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
100	**600**	**50**	20	20	**6022**	**3496**	**3090**	**3056**

Tabla 10 Resultados del estudio experimental de las heurísticas monosentido diseñadas para la instancia 11.

						(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**

100	**600**	**50**	20	20	**6022**	**3286**	**2825**	**2628**

Tabla 11 Resultados del estudio experimental de las heurísticas bisentido diseñadas para la instancia 11.

En este caso no se trata de una matriz que destaque especialmente por tener muchas más filas que columnas o viceversa, y que una de ellas sea muy reducida, ya que tendría 12 filas y 50 columnas. Por tanto, difícilmente podríamos predecir previamente al estudio experimental cuál podría ser más efectiva.

Una vez realizado el estudio se observa que la heurística más efectiva es la de ordenación por demanda de geometría bisentido.

Instancia Nº 12

						(MS-Dem)	(MS-Col)	(MS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
200	**1200**	**100**	20	10	**12039**	**6468**	**6056**	**5969**

Tabla 12 Resultados del estudio experimental de las heurísticas monosentido diseñadas para la instancia 12.

						(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO**	**FO**	**FO**
200	**1200**	**100**	20	10	**12039**	**7050**	**5840**	**5401**

Tabla 13 Resultados del estudio experimental de las heurísticas bisentido diseñadas para la instancia 12.

En este caso, tampoco se trata de una matriz que destaque especialmente por tener muchas más filas que columnas o viceversa, y que una de ellas sea muy reducida, ya que tendría 100 filas y 12 columnas. Por tanto, difícilmente podríamos predecir previamente al estudio experimental cuál podría ser más efectiva.

Una vez realizado el estudio, se observa que la heurística más efectiva es la de ordenación por demanda de geometría bisentido.

5.5. CONCLUSIONES.

Se ha observado que, con estas heurísticas, se mejoran los resultados del algoritmo de Laguna en instancias extensas; en este caso a partir de 300 de demanda. Reduciendo de los 3600 segundos que tarda como máximo el algoritmo en el caso de no alcanzar el óptimo hasta

menos de 5 segundos, por lo que con muy poco coste computacional se consigue mejores resultados que el citado algoritmo, y se podría utilizar esta herramienta para instancias extensas, fijando un límite a partir del cual se podría pasar de utilizar el algoritmo de Laguna a estas heurísticas.

CAPÍTULO 6. DISEÑO DE METAHEURÍSTICAS Y ESTUDIO EXPERIMENTAL.

6.1. INTRODUCCIÓN.

De los 4 tipos de metaheurísticas mencionados anteriormente, se descartan los dos más complejos, que son los que tienen memoria: la búsqueda tabú y el algoritmo genético. Se descartan porque el tener memoria, cada vez que el algoritmo realiza la búsqueda de otra solución, realiza un análisis de la memoria almacenada y esto conlleva un coste computacional muy elevado. Y como en el caso tratado se requiere que la herramienta pueda ser utilizada en un puesto de trabajo convencional con un ordenador estándar, no se dispone de servidores especializados dedicados a esa herramienta, y se quiere obtener el resultado de forma rápida.

Por tanto, quedan dos, el GRASP y el Recocido Simulado. Por descarte, vamos a elegir el GRASP, ya que da en menor tiempo mejores soluciones. Al permitir en el Recocido Simulado aceptar peores soluciones para luego alcanzar mejores, se emplea un tiempo de computación mayor. Se pueden alcanzar mejores soluciones al salir con mayor facilidad de los óptimos locales y encontrar el óptimo global, pero al requerir mayor tiempo de computación y en el caso estudiado tenerlo limitado, puede ofrecer peores soluciones en tiempo reducido que el GRASP, por lo que se selecciona trabajar con un GRASP.

6.2. DIAGRAMA DE PROCESO.

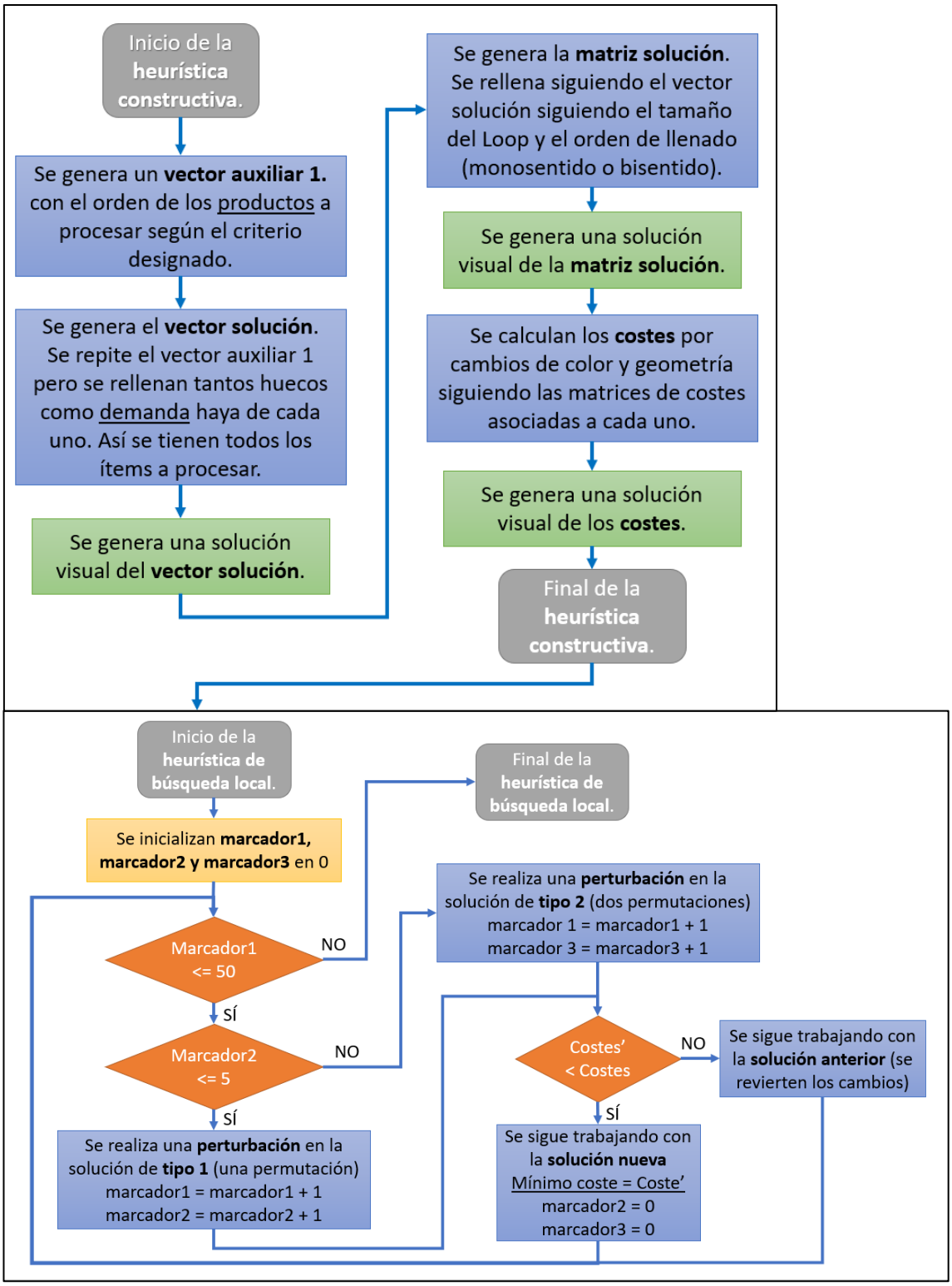

Fig. 24 Diagrama de proceso de las metaheurísticas diseñadas.

6.3. DISEÑO DE LAS METAHEURÍSTICAS.

Una vez decidido que se va a trabajar con un GRASP, se va a profundizar más en qué es un GRASP y de qué consta.

Un GRASP sigue un proceso iterativo, en el cual hay dos fases:

- Primera fase, o fase constructiva.
- Segunda fase, o fase de búsqueda local.

En la primera de ellas se sigue una heurística constructiva para generar una solución, y en la segunda fase, se sigue una heurística de búsqueda local iterativa. Tras este proceso, la mejor solución de entre todas las iteraciones permanece como mejor solución global.

Tras conocer los conceptos fundamentales del GRASP, hay que seleccionar el tipo de búsqueda que se va a realizar. En este caso se opta por la búsqueda local por entorno variable, para asegurar que una vez alcancemos un óptimo local, se hagan excitaciones mayores y no se vuelva a buscar otro óptimo local hasta que estas excitaciones mejoren la solución obtenida anteriormente. De este modo, conseguimos buenas soluciones en periodos reducidos y con pocas iteraciones.

El formato de esta búsqueda dentro del código, es definir dos tipos de excitaciones diferentes, siendo ambas en forma de permutaciones de bloques de productos. En los movimientos de tipo 1, se realiza una permutación entre 2 bloques, y se sigue con estas permutaciones hasta que se realizan 5 permutaciones sin mejorar la mejor solución obtenida. Una vez esto ocurre, se realizan movimientos del tipo 2, que son permutaciones de 3 bloques, que nos permiten explorar el espacio de soluciones de forma más diversificada. Se hacen movimientos de este tipo hasta que la solución encontrada mejora la mejor solución obtenida. Una vez sucede esto, se realizan movimientos del tipo 1 otra vez. Por supuesto, cada vez que se hacen permutaciones y no mejoran la mejor solución almacenada, se vuelve a la solución anterior y se realizan las permutaciones a partir de ella.

Como las 6 metaheurísticas propuestas son muy similares, se va a tratar el diseño de una de ellas, y en las partes del código donde haya variación entre ellas, se mencionará.

La primera fase de la metaheurística es la heurística constructiva, que sigue el mismo procedimiento que el mencionado en el capítulo anterior. Posteriormente es cuando actúa la heurística de búsqueda local, que completa el GRASP.

a) Se introducen los datos de la instancia como “datos de entrada”.
b) Se almacena en un vector auxiliar las demandas de cada color, para luego poder detectar el mínimo y el máximo de demanda de cada color y así poder posteriormente “barrer” entre esos dos valores seleccionando los colores de mayor a menor demanda y establecer en otro vector auxiliar el orden de productos por colores de mayor a menor demanda. En este caso se trata de la heurística cuyo primer criterio es la de criterio por colores de mayor demanda. En las otras sería igual, pero cambiando los criterios.

```
' 1º Mayor DEMANDA por COLOR
' 2º Mayor DEMANDA por PRODUCTO
' 3º Lexicográfico
        i = 0
        j = 0
        valorcriterio = demmax
        hueconumero = 1

' Almacenar en el vector Demandacolores() las demandas acumuladas de cada color
        For i = 0 To N_Colores - 1
            Demandacolores(i + 1) = 0
        Next

        For j = 0 To N_Colores - 1
            For i = 1 To N_Modelos
                If Instancia(2, i) = j Then
                        Demandacolores(j + 1) = Demandacolores(j + 1) + Instancia(3, i)
                End If
            Next
        Next

' Bucle que detecta el MIN y el MÁX de los COLORES
        For i = 0 To N_Colores - 1
            If Demandacolores(i + 1) < colmin Then
                    colmin = Demandacolores(i + 1)
            End If
            If Demandacolores(i + 1) > colmax Then
                    colmax = Demandacolores(i + 1)
            End If
        Next

'Bucle donde se colocan los COLORES por orden de mayor a menor DEMANDA
        color = colmax
        colorhueco = 1
        For i = colmin To colmax
            For j = 0 To N_Colores - 1
                If Demandacolores(j + 1) = color Then
                        Ordencolor(colorhueco) = j
                        colorhueco = colorhueco + 1
                End If
            Next
            color = color - 1
        Next
```

Fig. 25 Código de la metaheurística diseñada (I).

c) Dentro de cada color, se subdivide por la demanda de cada producto, ordenándose de esta forma de mayor a menor, como indica el 2º criterio. En el caso de que haya varios iguales, se sigue el tercer criterio, el lexicográfico por número de producto. Posteriormente se genera el **vector solución** y se representa visualmente.

```
'colmin es la demanda total de cada color y colmin2 es la demanda local de cada producto
        For i = 1 To N_Colores
            'Bucle para detectar el colmin2 y colmax2 de cada color
                For j = 1 To N_Modelos
                        If Instancia(3, j) < colmin2 And Instancia(2, j) = Ordencolor(i) Then
                                colmin2 = Instancia(3, j)
                        End If
                        If Instancia(3, j) > colmax2 And Instancia(2, j) = Ordencolor(i) Then
                                colmax2 = Instancia(3, j)
                        End If
                Next

'Bucle que escribe en Solución2
                valorcriterio = colmax2
                For l = colmin2 To colmax2
                    For k = 1 To N_Modelos
                        If Instancia(3, k) = valorcriterio And Instancia(2, k) = Ordencolor(i) Then
                            Solucion2(hueconumero) = k
                            hueconumero = hueconumero + 1
                        End If
                    Next
                    valorcriterio = valorcriterio - 1
                Next

        Next

        For hij = colmin To colmax
            For i = 1 To N_Modelos
                If Instancia(3, i) = valorcriterio Then
                        Solucion2(hueconumero) = i
                        hueconumero = hueconumero + 1
                End If
            Next
            valorcriterio = valorcriterio - 1
        Next
```

Fig. 26 Código de la metaheurística diseñada (II).

d) A través del vector solución y siguiendo los criterios de rellenado de la matriz, se crea un bucle que genera la solución visual, es decir, la **matriz solución**. En este caso, sí que hay diferencia de unas heurísticas a otras, ya que en las bisentido se rellena de forma distinta. Se muestran ambos códigos con su diferenciación.

Monosentido:

```
Dim NLoops
NLoops = N_Posiciones / PPL '(que sea entero)

x = 0

    For a = 1 To NLoops
        For b = 1 To PPL
            x = x + 1
            Cells(14 + a, b) = Instancia(1, solucion(x))
            Cells(14 + a, b).Interior.ColorIndex = Instancia(2, solucion(x))
        Next
    Next
```

Fig. 27 Código de la metaheurística diseñada (III).

Bisentido:

```
For a = 1 To NLoops
    If abc = 1 Then
    'Fila impar
            For b = 1 To PPL
                Cells(14 + a, b) = Instancia(1, solucion(x))
                Cells(14 + a, b).Interior.ColorIndex = Instancia(2, solucion(x)) + 1
                x = x + 1
            Next
        abc = 0
        n = n + 1
    Else
    'Fila par (RELLENAR AL REVÉS)
            For c = 0 To (PPL - 1)
                JAVI = PPL - c
                Cells(14 + a, JAVI) = Instancia(1, solucion(x))
                Cells(14 + a, JAVI).Interior.ColorIndex = Instancia(2, solucion(x)) + 1
                x = x + 1
            Next
        abc = 1
        n = n + 1
    End If
Next
```

Fig. 28 Código de la metaheurística diseñada (IV).

a) Una vez ya se tiene la matriz solución, se procede a calcular los cambios de geometrías, colores y totales, y sus respectivos costes, obteniendo así el **coste total**. En la interfaz se representan estos valores para que sean fácilmente identificables por el usuario.
Los cambios de geometría se detectan sobre la propia matriz solución gráficamente, ya que son costes por posiciones, en posiciones de filas (loops) consecutivas, y al haber dos tipos de rellenado, sería muy complejo realizarlo analizando el vector solución. Los cambios de color se detectan analizando el vector solución, ya que son costes por posiciones consecutivas.

```
'COSTES
Dim CamCol As Integer
Dim CamGeo As Integer
CamGeo = 0
CamCol = 0
CosteGeo = 0
CosteCol = 0

For d = 1 To (N_Posiciones - 1)
    If Instancia(2, solucion(d)) <> Instancia(2, solucion(d + 1)) Then
        CamCol = CamCol + 1
        CosteCol = CosteCol + Worksheets("Coste colores").Cells(1 + Instancia(2, solucion(d)), 1 + Instancia(2, solucion(d + 1))).Value
    End If
Next

For e = 1 To (N_Posiciones - PPL)
    If Instancia(1, solucion(e)) <> Instancia(1, solucion(e + PPL)) Then
        CamGeo = CamGeo + 1
        CosteGeo = CosteGeo + Worksheets("Coste geometria").Cells(1 + Instancia(1, solucion(e)), 1 + Instancia(1, solucion(e + PPL))).Value
    End If
Next

    Worksheets("Principal").Cells(11, 1).Value = CamGeo
    Worksheets("Principal").Cells(11, 2).Value = CamCol
    Worksheets("Principal").Cells(11, 3).Value = CamCol + CamGeo
    Worksheets("Principal").Cells(11, 4).Value = CosteGeo
    Worksheets("Principal").Cells(11, 5).Value = CosteCol
    Worksheets("Principal").Cells(11, 6).Value = CosteGeo + CosteCol

Dim SolAuxiliar()
ReDim SolAuxiliar(N_Posiciones)

    If Cells(11, 6) < Costemin Then
        Costemin = Cells(11, 6)
        Cells(11, 7) = Costemin
        f = 1
        For f = 1 To N_Posiciones
            SolAuxiliar(f) = solucion(f)
        Next
        contador = 0
    Else
        contador = contador + 1
        r = r + 1
    End If
```

Fig. 29 Código de la metaheurística diseñada (V).

e) Se reflejan en la interfaz todos los datos relevantes (**vector solución**, **matriz solución**, **cambios** y **costes**)

Numero de modelos	Geometrías	Colores	Posiciones a programar	PPL
200	20	10	1200	100

Generar instancia | Algoritmo constructivo | Generar Solución

Geometría	H	11	3	15	14	3	2	16	7	8	7	10	4	7	10	9	16	13	17	17	15
Color	K	8	3	9	3	6	8	1	7	3	3	0	3	6	2	7	6	2	9	0	3
Demanda	Q	11	2	1	11	4	4	17	16	1	2	9	1	1	10	1	13	7	5	4	1
Código de modelo		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Cambios geometría	Cambios colores	Cambios Totales	Coste cambios geometría	Coste cambios colores	Coste total	COSTE MÍNIMO
1018	49	1067	5336	239	5575	5369

Solución	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54
	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
	3	3	3	3	3	11	11	11	11	11	11	11	11	11	11	11	11	11	7	7	7	7	7	
	3	3	3	3	3	3	14	14	14	14	14	14	14	14	14	14	14	9	9	9	9	9	9	
																		5	5	5	5	5	5	5
	10	10	10	10	10	10	10	10	10	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
	10	10	10	10	10	10	10	10	10	10	10	11	11	11	11	11	11	11	11	11	11	11	11	11
	5	5	5	18	18	18	18	0	0	0	0	0	0	0	15	15	15	15	15	15	15	10	10	
	3	7	2	2	2	2	2	2	2	2	2	2	2	8	11	6	8	14	13	13	13	13	13	13
	13	13	13	13	13	13	13	14	14	14	14	14	14	14	14	14	14	14	14	10	10	10	10	10
	13	13	13	13	11	11	11	11	11	11	11	11	11	17	17	17	17	17	17	17	3	3		
	17	14	13	18	3	4	9	0	15	7	10	5	5	5	0	2	2	2	2	6	6	6	6	6

Principal | Coste geometria | Coste colores | Enum | Hoja3

Fig. 30 Interfaz con la salida de datos de la metaheurística diseñada.

A partir de aquí es cuando entra la heurística de búsqueda local.

f) En esta primera parte del código se hace la perturbación de tipo uno, que como se ha mencionado anteriormente, es la permutación de dos productos. Tras esto, se genera otra vez con el código anterior la matriz solución, y se repite de nuevo el código para calcular los cambios y los costes por cambio de color y geometría.

```
Do While marcador1 <= 50
    marcador2 = 0
    marcador3 = 0
    Do While marcador2 <= 5
        aleatorio1 = WorksheetFunction.RandBetween(1, N_Modelos)
        aleatorio2 = WorksheetFunction.RandBetween(1, N_Modelos)
            PosicionAux = Solucion2(aleatorio1)
            Solucion2(aleatorio1) = Solucion2(aleatorio2)
            Solucion2(aleatorio2) = PosicionAux
            marcador1 = marcador1 + 1

                                a = 1
                                aux = 0
                                For a = 1 To N_Modelos
                                    b = 1
                                    For b = 1 To Instancia(3, Solucion2(a))
                                        aux = aux + 1
                                        solucion(aux) = Instancia(4, Solucion2(a))
                                        'Escribe la solución partiendo del vector Solucion2
                                        Cells(13, 1 + aux) = solucion(aux)
                                    Next
                                Next
```

Fig. 31 Código de la metaheurística diseñada (VI).

g) Tras alcanzar una nueva solución, esta parte del código compara el nuevo coste con el óptimo. Si se ha mejorado (es decir, reducido) el coste total, se guarda este como nuevo mejor coste y se prosigue con esta ordenación. Si no se ha mejorado este mejor coste, se revierten los cambios en la ordenación y se vuelve a ejecutar otra perturbación a partir de la solución anterior, que será la del mejor coste.

```
If Cells(11, 6) < Costemin Then
    Costemin = Cells(11, 6)
    Cells(11, 7) = Costemin
    condicion = True
    marcador2 = 0
Else
    PosicionAux = Solucion2(aleatorio1)
    Solucion2(aleatorio1) = Solucion2(aleatorio2)
    Solucion2(aleatorio2) = PosicionAux
    marcador2 = marcador2 + 1
                a = 1
                aux = 0
                For a = 1 To N_Modelos
                    b = 1
                    For b = 1 To Instancia(3, Solucion2(a))
                        aux = aux + 1
                        solucion(aux) = Instancia(4, Solucion2(a))
                        'Escribe la solución partiendo del vector Solucion2
                        Cells(13, 1 + aux) = solucion(aux)
                    Next
                Next
End If
```

Fig. 32 Código de la metaheurística diseñada (VII).

h) En el caso de haber alcanzado las 5 iteraciones con perturbaciones de tipo 1 sin haber mejorado la solución, se para a realizar perturbaciones del tipo 2, y tras estas perturbaciones, se volvería otra vez a realizar el proceso de generar la matriz solución nueva y calcular los costes.

```
Loop
Do While marcador3 <= 5
    aleatorio1 = WorksheetFunction.RandBetween(1, N_Modelos)
    aleatorio2 = WorksheetFunction.RandBetween(1, N_Modelos)
    aleatorio3 = WorksheetFunction.RandBetween(1, N_Modelos)
    PosicionAux = Solucion2(aleatorio1)
    Solucion2(aleatorio1) = Solucion2(aleatorio2)
    Solucion2(aleatorio2) = Solucion2(aleatorio3)
    Solucion2(aleatorio3) = PosicionAux
    marcador1 = marcador1 + 1
                        a = 1
                        aux = 0
                        For a = 1 To N_Modelos
                            b = 1
                            For b = 1 To Instancia(3, Solucion2(a))
                                aux = aux + 1
                                solucion(aux) = Instancia(4, Solucion2(a))
                                'Escribe la solución partiendo del vector Solucion2
                                Cells(13, 1 + aux) = solucion(aux)
                            Next
                        Next
```

Fig. 33 Código de la metaheurística diseñada (VIII).

i) Como ocurría antes, tras alcanzar una nueva solución, esta parte del código compara el nuevo coste con el óptimo. Si se ha mejorado (es decir, reducido) el coste total, se guarda este como nuevo mejor coste y se prosigue con esta ordenación. Si no se ha

mejorado este coste almacenado como mejor coste, se revierten los cambios en la ordenación y se vuelve a ejecutar otra perturbación a partir de la solución anterior, que será la del mejor coste.

```
If Cells(11, 6) < Costemin Then
    Costemin = Cells(11, 6)
    Cells(11, 7) = Costemin
    condicion = True
    marcador3 = 5
Else
    PosicionAux = Solucion2(aleatorio3)
    Solucion2(aleatorio3) = Solucion2(aleatorio2)
    Solucion2(aleatorio2) = Solucion2(aleatorio1)
    Solucion2(aleatorio1) = PosicionAux
    marcador3 = marcador3 + 1
                a = 1
                aux = 0
                For a = 1 To N_Modelos
                    b = 1
                    For b = 1 To Instancia(3, Solucion2(a))
                        aux = aux + 1
                        solucion(aux) = Instancia(4, Solucion2(a))
                        'Escribe la solución partiendo del vector Solucion2
                        Cells(13, 1 + aux) = solucion(aux)
                    Next
                Next
End If
```

Fig. 34 Código de la metaheurística diseñada (IX).

j) Una vez se alcanzan las 50 iteraciones, se finaliza la heurística de búsqueda local y con ello el GRASP, dejando reflejado el mejor coste obtenido durante todas las iteraciones realizadas en la interfaz y la matriz solución gráficamente mostrada.

6.4. ESTUDIO EXPERIMENTAL.

Una vez diseñadas las 6 metaheurísticas, se procede al estudio experimental.

En el caso del GRASP, los tiempos de computación son de máximo 37 segundos en el caso de 1200 de demanda, que es el más crítico. Se ha fijado el límite de iteraciones en 50, por lo que se obtienen las soluciones de forma muy rápida.

Se identifica que en las instancias a partir de N=300, es decir, 300 de demanda, los resultados ofrecidos por el GRASP siguen mejorando los resultados del algoritmo de Laguna. Sin embargo, en este caso, ambas instancias de 300 están incluidas en las que es preferible utilizar el GRASP. Incluso en una de las instancias de N=100, hay una heurística que mejora los resultados del algoritmo de Laguna y en mucho menos tiempo.

						(MS-Dem)	(MS-Col)	(MS-Geo)	(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	(N)	(PPL)	(G)	(C)	FO (Laguna)	FO'	FO'	FO'	FO'	FO'	FO'
5	**30**	**10**	3	5	**18**	**72**	**75**	**72**	**62**	**66**	**62**
5	**30**	**5**	5	20	**36**	**36**	**36**	**36**	**47**	**39**	**39**
5	**50**	**25**	3	20	**15**	**49**	**49**	**43**	**48**	**48**	**43**
5	**50**	**25**	10	20	**19**	**80**	**80**	**80**	**60**	**60**	**60**
5	**50**	**25**	20	20	**29**	**116**	**108**	**108**	**101**	**105**	**134**
10	**60**	**5**	10	10	**84**	**84**	**77**	**86**	**92**	**96**	**81**
10	**100**	**50**	10	10	**76**	**183**	**130**	**150**	**137**	**72**	**147**
50	**300**	**25**	10	5	**1144**	**1139**	**1048**	**886**	**1036**	**969**	**931**
50	**300**	**5**	3	20	**2134**	**531**	**550**	**248**	**630**	**671**	**413**
100	**600**	**100**	20	5	**5103**	**2222**	**2007**	**2268**	**2353**	**1816**	**2157**
100	**600**	**50**	20	20	**6022**	**2819**	**2526**	**1926**	**2941**	**2470**	**2368**
200	**1200**	**100**	20	10	**12039**	**5765**	**5086**	**5035**	**5613**	**5369**	**4359**

Tabla 14 Resultados del estudio experimental de las metaheurísticas diseñadas.

Tras el análisis general, se va a hacer un análisis específico para cada una de las instancias en las que los resultados de todas las metaheurísticas mejoran los resultados del algoritmo de Laguna. De esta forma, podemos saber si el comportamiento de alguna de las metaheurísticas es predecible analizando la instancia previamente y de este modo elegir la metaheurística en función de la instancia.

Instancia 8:

						(MS-Dem)	(MS-Col)	(MS-Geo)	(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	(N)	(PPL)	(G)	(C)	FO (Laguna)	FO'	FO'	FO'	FO'	FO'	FO'
50	**300**	**25**	10	5	**1144**	**1139**	**1048**	**886**	**1036**	**969**	**931**

Tabla 15 Resultados del estudio experimental de las metaheurísticas diseñadas para la instancia 8.

La matriz solución tendría 12 filas y 25 columnas. No destaca especialmente por tener muchas más filas que columnas o viceversa. Por tanto, difícilmente podríamos predecir, previamente al estudio experimental, cuál podría ser más efectiva.

Una vez realizado el estudio se observa que la heurística más efectiva es la de ordenación por demanda de geometría monosentido por poca diferencia con la bisentido.

Instancia 9:

						(MS-Dem)	(MS-Col)	(MS-Geo)	(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**
50	**300**	**5**	3	20	**2134**	**531**	**550**	**248**	**630**	**671**	**413**

Tabla 16 Resultados del estudio experimental de las metaheurísticas diseñadas para la instancia 9.

La matriz solución sería una matriz de 60 filas por tan sólo 5 columnas. Analizando esta instancia antes de haber ejecutado la metaheurística, podríamos haber llegado a la conclusión de que, al tratarse de loops muy cortos, los lotes de productos de mismas geometrías y distintos colores, si fuesen consecutivos podrían minimizar en gran medida los costes por cambio de geometría. Y al ejecutar la metaheurística podemos comprobar experimentalmente que así es, ya que la mejor metaheurística en este caso es la de ordenación de mayor a menor de productos por demanda de geometría (MS-Geo y BS-Geo).

Sin embargo, a priori habría sido difícil haber deducido entre la monosentido o la bisentido cuál iba a ser más efectiva. Experimentalmente se comprueba que la monosentido, ya que la bisentido, al ser loops muy cortos, produce mucho corte de lotes del mismo color al pasar de una fila a la siguiente, y no lo compensa reduciendo tanto coste por cambio de geometría como podría hacer en otras instancias.

Instancia 10:

						(MS-Dem)	(MS-Col)	(MS-Geo)	(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**
100	**600**	**100**	20	5	**5103**	**2222**	**2007**	**2268**	**2353**	**1816**	**2157**

Tabla 17 Resultados del estudio experimental de las metaheurísticas diseñadas para la instancia 10.

La matriz solución sería una matriz de 100 columnas por tan sólo 6 filas. En este caso, analizando esta instancia antes de ejecutar la metaheurística, podríamos observar que los loops son muy

largos, es decir, que la mayoría de cambios serán en horizontal, por colores. Experimentalmente, se comprueba esto mismo, ya que las metaheurísticas cuyos criterios de ordenación son por productos de mismos colores, de mayor a menor demanda, es la que ofrece mejores resultados. En este caso, ofrece mejores resultados la heurística bisentido dentro de las de ordenación por color, pero previamente habría sido difícil haberlo podido predecir.

Instancia 11:

						(MS-Dem)	(MS-Col)	(MS-Geo)	(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**
100	**600**	**50**	20	20	**6022**	**2819**	**2526**	**1926**	**2941**	**2470**	**2368**

Tabla 18 Resultados del estudio experimental de las metaheurísticas diseñadas para la instancia 11.

En este caso, no se trata de una matriz que destaque especialmente por tener muchas más filas que columnas o viceversa, y que una de ellas sea muy reducida, ya que tendría 12 filas y 50 columnas. Por tanto, difícilmente podríamos predecir previamente al estudio experimental cuál podría ser más efectiva.

Una vez realizado el estudio, se observa que la heurística más efectiva es la de ordenación por demanda de geometría monosentido.

Instancia 12:

						(MS-Dem)	(MS-Col)	(MS-Geo)	(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	**(N)**	**(PPL)**	**(G)**	**(C)**	**FO (Laguna)**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**	**FO'**
200	**1200**	**100**	20	10	**12039**	**5765**	**5086**	**5035**	**5613**	**5369**	**4359**

Tabla 19 Resultados del estudio experimental de las metaheurísticas diseñadas para la instancia 12.

En este caso, tampoco se trata de una matriz que destaque especialmente por tener muchas más filas que columnas o viceversa, y que una de ellas sea muy reducida, ya que tendría 100 filas y 12 columnas. Por tanto, difícilmente podríamos predecir previamente al estudio experimental cuál podría ser más efectiva.

Una vez realizado el estudio, se observa que la heurística más efectiva es la de ordenación por demanda de geometría bisentido.

6.4. CONCLUSIONES.

Se parte de la base de que todas las metaheurísticas utilizadas, igualan o mejoran los resultados de las heurísticas anteriores, ya que parten de la mejor solución de la heurística constructiva, por tanto, siempre van a ofrecer mejores soluciones que las heurísticas sencillas. El tiempo de computación es más elevado, pasando de los 5 segundos de máximo, a los 37 segundos de la instancia más compleja. Tampoco es un aumento excesivo y se considera asumible en el caso estudiado, por lo que se puede concluir que las metaheurísticas diseñadas, para los rangos de instancias del problema, nos aportan mejores soluciones que el algoritmo de Laguna a partir de demanda 300 y mejores soluciones globalmente que las heurísticas sencillas.

CAPÍTULO 7. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN.

Se marca la mejor solución de entre las metaheurísticas y la solución ofrecida por el algoritmo de Laguna dentro del *time limit*.

Una vez realizado todos los estudios experimentales con las instancias aleatorias generadas, hay que proponer unos criterios a seguir en el caso real para facilitar la comprensión de la conclusión.

						(MS-Dem)	(MS-Col)	(MS-Geo)	(BS-Dem)	(BS-Col)	(BS-Geo)
(S)	(N)	(PPL)	(G)	(C)	FO (Laguna)	FO'	FO'	FO'	FO'	FO'	FO'
5	**30**	**10**	3	5	**18**	**72**	**75**	**72**	**62**	**66**	**62**
5	**30**	**5**	5	20	**36**	**36**	**36**	**36**	**47**	**39**	**39**
5	**50**	**25**	3	20	**15**	**49**	**49**	**43**	**48**	**48**	**43**
5	**50**	**25**	10	20	**19**	**80**	**80**	**80**	**60**	**60**	**60**
5	**50**	**25**	20	20	**29**	**116**	**108**	**108**	**101**	**105**	**134**
10	**60**	**5**	10	10	**84**	**84**	**77**	**86**	**92**	**96**	**81**
10	**100**	**50**	10	10	**76**	**183**	**130**	**150**	**137**	**72**	**147**
50	**300**	**25**	10	5	**1144**	**1139**	**1048**	**886**	**1036**	**969**	**931**
50	**300**	**5**	3	20	**2134**	**531**	**550**	**248**	**630**	**671**	**413**
100	**600**	**100**	20	5	**5103**	**2222**	**2007**	**2268**	**2353**	**1816**	**2157**
100	**600**	**50**	20	20	**6022**	**2819**	**2526**	**1926**	**2941**	**2470**	**2368**
200	**1200**	**100**	20	10	**12039**	**5765**	**5086**	**5035**	**5613**	**5369**	**4359**

Tabla 20 Resultados del estudio experimental de las metaheurísticas diseñadas.

En el caso real, se va a tener un tamaño de loop fijo, y entrarán instancias o pedidos en los que habrá S productos, N de demanda, y C colores y G geometrías. Y una vez se reciba esta información, se deberá utilizar la herramienta adecuada.

Se puede deducir que el valor crítico a la hora de aumentar la función objetivo y el tiempo de computación es la demanda N. Por tanto, en instancias de hasta 300 de demanda, se utilizaría el algoritmo de Laguna que como se ha comprobado, ofrece mejores soluciones en instancias pequeñas.

A partir de 300 de demanda, se utilizará una de las metaheurísticas disponibles, o varias de ellas, pero siempre teniendo que tomar la decisión antes de tener los datos experimentales.

En tres de los cinco casos en los que se utilizaría el GRASP, el mejor resultado lo aporta la metaheurística monosentido de ordenación por geometría, y en otro de los casos, esta metaheurística aporta la segunda mejor solución. Sólo hay uno de los cinco casos en los que esta metaheurística no nos daría una de las 2 mejores soluciones, y está marcada en azul en la tabla. En ese caso, haciendo un análisis previo, se podría deducir que una metaheurística basada en el ordenamiento por colores sería más eficiente.

Por tanto, el criterio sería este:

$N \leq 100 \rightarrow$ Algoritmo de Laguna

$N > 100 \rightarrow$ GRASP (Monosentido – Ordenación por color) salvo que se prevea claramente que teniendo un loop muy largo y dando muy pocas vueltas, el GRASP (Bisentido – Ordenación por colores) pueda ofrecer mejores soluciones.

Y de este modo, se consigue el objetivo inicial: Obtener una herramienta y un criterio que, siguiéndolo, se consigan buenas soluciones con poco coste computacional para cualquier tipo de instancia dentro de los rangos en los que se trabaja en esa línea de producción.

7.1. FUTURAS LINEAS DE INVESTIGACIÓN.

Una vez se ha concluido el trabajo, se han advertido posibles mejoras en el mismo. Partiendo de lo experimentado durante el desarrollo de este trabajo, como futuras líneas de investigación se definen las siguientes:

En primer lugar, la mejora de las heurísticas de búsqueda local para alcanzar los resultados del algoritmo de Laguna con instancias simples. En el caso actual, la herramienta mejora los resultados del algoritmo de Laguna a partir de una demanda determinada, pero en instancias más simples, el algoritmo de Laguna alcanza el óptimo y la herramienta diseñada no. Por tanto, este sería el principal objetivo de las futuras líneas de investigación, que la herramienta alcance el óptimo global para instancias sencillas.

Por otra parte, el desarrollo de otros algoritmos para compararlos con los estudiados ayudaría a clarificar la eficacia de la herramienta diseñada.

Otra de las futuras líneas de investigación sería la de encarar problemas con huecos por cambios de setup. En el caso estudiado no había huecos por cambio de setup, pero en caso reales esto si que sucede y resultaría interesante encarar un problema con este tipo de característica.

Y por último, la mejor forma de averiguar la robustez y eficacia de la herramienta diseñada sería implementarla en una empresa del ámbito industrial donde se producto este tipo de problema, como la citada en la introducción.

CAPÍTULO 8. BIBLIOGRAFÍA.

Orlando De Antonio Suárez (2010). Una aproximación a heurísticas y metaheurísticas.

Subhamoy Ganguly & Manuel Laguna (2014). Modeling and solving a closed-loop scheduling problema with two types of setups.

Paola Festa & Mauricio G. C. Resende (2007). An annotated bibliography of GRASP-Part II: Applications.

Thomas A. Feo & Mauricio G.C. Resende (1994). Greedy Randomized Adaptive Search Procedures.

Mauricio G.C. Resende (1998). Greedy Randomized Adaptive Search Procedures (GRASP).

Ana Virginia Ruescas Nicolau (2015). Aplicación de técnicas metaheurísticas para la asignación de turnos de trabajo.

Mauricio G.C. Resende & Celso C. Ribeiro (2010) Greedy Randomized Adaptative Search Procedures: Advances, Hybridization, and Applications.

Jesús Sáez Aguado (2016). Algoritmos heurísticos y metaheurísticos basados en búsqueda local aplicados a problemas de rutas de vehículos.

PRESUPUESTO

ÍNDICE DEL PRESUPUESTO

CAPÍTULO 1. INTRODUCCIÓN.

En este documento se presenta el presupuesto del trabajo. Al tratarse de un trabajo cuyo objetivo es crear una herramienta informática, los costes principales representan las horas dedicadas por los ingenieros que intervienen en la realización del trabajo.

Se tienen en cuenta las siguientes consideraciones para la elaboración del presupuesto:

Se establecen como costes complementarios los costes cuya cuantía es pequeña en comparación al resto de gastos, y que son difícilmente identificables. Se establecen como un 1% del total de costes. Esta asignación aumenta hasta el 5% si la actividad realizada conlleva gastos de transporte, y hasta el 10% si conlleva costes asociados a material de oficina.

Se estima que el beneficio industrial es del 6%. Esta partida representa el beneficio que obtendría una empresa si realizase este trabajo.

CAPÍTULO 2. CUADRO DE PRECIOS.

N.º Orden	N.º Precio	Ud.	Descripción	Rendimiento	Precio	Importe
01			**ESTUDIO PRELIMINAR DEL CASO**			
01.01			**Reunión del equipo de trabajo para analizar el problema y recopilar la primera información**			
	MOII1	h	Ingeniero industrial 1	3,00	60,00	180,00
	MOII2	h	Ingeniero industrial 2	3,00	30,00	90,00
		%	Gastos complementarios	0,03	270,00	8,10
					Subtotal	278,10
01.02			**Búsqueda bibliográfica**			
	MOII1	h	Ingeniero industrial 1	2,00	60,00	120,00
	MOII2	h	Ingeniero industrial 2	3,00	30,00	90,00
		%	Gastos complementarios	0,03	210,00	6,30
					Subtotal	216,30
01.03			**Estudio de la información recopilada**			
	MOII2	h	Ingeniero industrial 2	3,00	30,00	90,00
		%	Gastos complementarios	0,01	90,00	0,90
					Subtotal	90,90

N.º Orden	N.º Precio	Ud.	Descripción	Rendimiento	Precio	Importe
02			**ESTUDIO DE LAS TÉCNICAS DE RESOLUCIÓN DEL PROBLEMA**			
02.01			**Búsqueda de información**			
	MOII2	h	Ingeniero industrial 2	12,00	30,00	360,00
		%	Gastos complementarios	0,01	360,00	3,60
					Subtotal	363,60
02.02			**Estudio de la información recopilada**			
	MOII2	h	Ingeniero industrial 2	15,00	30,00	450,00
		%	Gastos complementarios	0,01	450,00	4,50
					Subtotal	454,50
02.03			**Redacción de informes sobre las técnicas de resolución a utilizar**			
	MOII2	h	Ingeniero industrial 2	6,00	30,00	180,00
		%	Gastos complementarios	0,01	180,00	1,80

					Subtotal	181,80

N.º Orden	N.º Precio	Ud.	Descripción	Rendimiento	Precio	Importe
03			**IMPLEMENTACIÓN DE HEURÍSTICAS Y METAHEURÍSTICAS**			
03.01			**Reunión del equipo de trabajo para fijar las técnicas de resolución definitivas**			
	MOII1	h	Ingeniero industrial 1	3,00	60,00	180,00
	MOII2	h	Ingeniero industrial 2	3,00	30,00	90,00
		%	Gastos complementarios	0,01	270,00	2,70
					Subtotal	272,70
03.02			**Diseño y programación de las heurísticas y metaheurísticas**			
	MOII2	h	Ingeniero industrial 2	80,00	30,00	2400,00
		%	Gastos complementarios	0,01	2400,00	24,00
					Subtotal	2424,00
03.03			**Simulación experimental de las instancias seleccionadas y recopilación de resultados**			
	MOII2	h	Ingeniero industrial 2	20,00	30,00	600,00
		%	Gastos complementarios	0,01	600,00	6,00
					Subtotal	606,00
03.04			**Redacción de informes sobre el funcionamiento de cada una de las heurísticas y metaheurísticas y análisis de resultados**			
	MOII2	h	Ingeniero industrial 2	30,00	30,00	900,00
		%	Gastos complementarios	0,01	900,00	9,00
					Subtotal	909,00

N.º Orden	N.º Precio	Ud.	Descripción	Rendimiento	Precio	Importe
04			**EVALUACIÓN DE RESULTADOS**			
04.01			**Reunión del equipo de trabajo para fijar las técnicas de resolución definitivas**			
	MOII2	h	Ingeniero industrial 2	3,00	30,00	90,00
		%	Gastos complementarios	0,01	90,00	0,90
					Subtotal	90,90
04.02			**Reunión del equipo de trabajo para determinar la efectividad y eficiencia de la herramienta creada y corrección de errores**			
	MOII1	h	Ingeniero industrial 1	5,00	60,00	300,00

	MOII2	h	Ingeniero industrial 2	5,00	30,00	150,00
		%	Gastos complementarios	0,03	450,00	13,50
					Subtotal	463,50
04.03			**Redacción de informes finales y presentación de la solución propuesta**			
	MOII2	h	Ingeniero industrial 2	20,00	30,00	600,00
		%	Gastos complementarios	0,01	600,00	6,00
					Subtotal	606,00

N.º Orden	N.º Precio	Ud.	Descripción	Rendimiento	Precio	Importe
05			**MATERIAL INFORMÁTICO**			
05.01			**Hardware para la realización del trabajo**			
	MOS1	h	Microsoft Excel	40,00	2,00	80,00
	MOS2	h	Microsoft Word	60,00	2,00	120,00
		%	Gastos complementarios	0,10	200,00	20,00
					Subtotal	220,00
05.02			**Software para la realización del trabajo**			
	MOII1	h	Equipo informático	350,00	0,50	175,00
	MOII2	h	Consumibles	1,00	50,00	50,00
		%	Gastos complementarios	0,10	225,00	22,50
					Subtotal	247,50

2.1. Resumen partidas de presupuestos.

N.º Orden	N.º Precio	Ud.	Descripción	Importe
01			**ESTUDIO PRELIMINAR DEL CASO**	
01.01			Reunión del equipo de trabajo para analizar el problema y recopilar la primera información	278,10
01.02			Búsqueda bibliográfica	216,30
01.03			Estudio de la información recopilada	90,90
02			**ESTUDIO DE LAS TÉCNICAS DE RESOLUCIÓN DEL PROBLEMA**	
02.01			Búsqueda de información	363,60
02.02			Estudio de la información recopilada	454,50
02.03			Redacción de informes sobre las técnicas de resolución a utilizar	181,80
03			**IMPLEMENTACIÓN DE HEURÍSTICAS Y METAHEURÍSTICAS**	

03.01	Reunión del equipo de trabajo para fijar las técnicas de resolución definitivas	272,70
03.02	Diseño y programación de las heurísticas y metaheurísticas	2424,00
03.03	Simulación experimental de las instancias seleccionadas y recopilación de resultados	606,00
03.04	Redacción de informes sobre el funcionamiento de cada una de las heurísticas y metaheurísticas y análisis de resultados	909,00
04	**EVALUACIÓN DE RESULTADOS**	
04.01	Reunión del equipo de trabajo para fijar las técnicas de resolución definitivas	90,90
04.02	Reunión del equipo de trabajo para determinar la efectividad y eficiencia de la herramienta creada y corrección de errores	463,50
04.03	Redacción de informes finales y presentación de la solución propuesta	606,00
05	**MATERIAL INFORMÁTICO**	
05.01	Hardware para la realización del trabajo	220,00
05.02	Software para la realización del trabajo	247,50

Presupuesto Ejecución	7.424,80 €
Beneficio industrial 6%	445.49 €
Presupuesto General	**7.870,49 €**
IVA 21%	1.652,76 €
Presupuesto Total	**9.523,05 €**