DESARROLLO DE UN SOFTWARE ACADÉMICO PARA PROGRAMAR LA PRODUCCIÓN EN LOS SISTEMAS DE MANUFACTURA FLEXIBLE

KATHERINE FLÓREZ RODRIGUEZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD INGENIERÍA INDUSTRIAL
PEREIRA
2013

DESARROLLO DE UN SOFTWARE ACADÉMICO PARA PROGRAMAR LA PRODUCCIÓN EN LOS SISTEMAS DE MANUFACTURA FLEXIBLE

KATHERINE FLÓREZ RODRIGUEZ

Trabajo de grado

Profesor Pedro Daniel Medina Varela
Director del proyecto

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD INGENIERÍA INDUSTRIAL
PEREIRA
2013

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

Firma del jurado

Pereira, 15 de Febrero del 2013

DEDICATORIA

A mis padres que me apoyaron económica y moralmente durante todo mi proceso de aprendizaje, por su amor, paciencia, dedicación y entrega.

A Johan Andrés Idarraga Villa que durante todo este camino sentí su apoyo incondicional.

Y sobre todo gracias a Dios que me fortaleció día a día y fue mi refugio en los momentos de mayor presión.

AGRADECIMIENTOS

A Oscar Andrés Granada porque con su conocimiento sobre JAVA me hizo ver lo fácil que puede ser la vida.

A Juan David Gómez que su asesoría sobre JAVA fue muy útil en la última fase del proyecto.

Al Ingeniero Pedro Daniel Medina Varela que me apoyo con los conocimientos necesarios de los sistemas de manufactura flexible y los ejemplos para desarrollar el presente trabajo.

GLOSARIO

ESTACIÓN CUELLO DE BOTELLA: Estación de servicio que determina la velocidad del flujo en el sistema.

PORTAHERRAMIENTAS: Un portaherramientas es un dispositivo de sujeción de la herramienta de una máquina herramienta. Hay muchas herramientas diferentes en cuanto a forma y tamaño. El tipo de portaherramientas debe ser elegido en función de la máquina y de la herramienta a utilizar. En las máquinas modernas de control numérico por computadora (CNC), la elección de un portaherramientas adecuado es importante para asegurar un mecanizado preciso con productividad.

REGLA LPT (LARGEST PROCESSING TIME): Se le da prioridad a las tareas de mayor tiempo de procesamiento y se asigna a la máquina que quede con mayor tiempo disponible después que la operación ha sido asignada.

CLUSTER DE OPERACIONES: Es una seudo-operación, está formada por varias operaciones de una misma pieza que es programada o manejada como una sola operación. Se forma con operaciones de una misma pieza, se usa para minimizar el intercambio de material entre máquinas.

GRUPO DE MÁQUINAS: Se forma con máquinas redundantes, es decir con máquinas que en cierto momento está en capacidad de asumir la producción de otra.

RESUMEN

La producción de un sistema de manufactura flexible se debe planear de acuerdo a los criterios adecuados, además se debe balancear la carga tanto en los lotes como en cada máquina disponible para la producción, para que de este modo garantice el ahorro esperado durante la producción, y también se debe certificar la flexibilidad con la rapidez con que se programe la producción y se responda a imprevistos en los cambios de número de piezas, máquinas, herramientas entre otros. Para esto se espera que el software resultado de este proyecto permita resolver la producción en menor tiempo que calculando las heurísticas a mano. Se ha tomado la heurística de Greddy que resuelve el problema de selección de partes, la heurística de formación de lotes y la heurística de Carga para programar un software académico que permite que la producción se defina rápidamente con fines académicos bien sea para enseñar los principios básicos o para investigaciones más avanzadas. Durante el transcurso de este trabajo se explica el proceso de desarrollo de un software que se ha llamado APROTI (Asistente de Producción y Tiempos), el cual se ha creado para resolver las heurísticas principales que resuelven los problemas de la planeación de la producción en un sistema de manufactura flexible, es un software académico que se basa inicialmente en ingresar los datos necesarios para resolver las heurísticas y por ultimo indica cual es la mejor respuesta para el desarrollo del sistema según los datos ingresados.

Palabras claves: sistema de manufactura flexible (FMS), APROTI, software académico, producción flexible, heurística de Greddy que resuelve el problema de selección de parte, la heurística de formación de lotes y la heurística de Carga.

CONTENIDO

TABLA DE ILUSTRACIONES

INTRODUCCIÓN

Hoy en día la tecnología y la comunicación han avanzado mucho, permitiendo que todas las actividades del ser humano evolucionen a grandes pasos, los computadores son grandes aliados en las empresas para realizar todo tipo de actividades, por lo tanto se desea desarrollar un aplicativo en la plataforma de java que le permita a la Facultad de Ingeniería Industrial de la Universidad Tecnológica de Pereira, facilitar la enseñanza de los algoritmos usados para programar la producción en un sistema de manufactura flexible y de este modo contrastarlo con la celda de manufactura flexible que posee dicha facultad en sus instalaciones para investigaciones o con objetivos pedagógicos. A menudo los estudiantes que van a realizar alguna actividad en la celda de FMS[1] controlan la producción a mano o en Excel pero con un software que haga estos cálculos por ellos podrán llevar a cabo sus estudio a mayor velocidad, con más precisión y se concentraran en el objetivo de la tarea más que en la solución de las herramientas de control de producción. Además los estudiantes que cursen la asignatura de FMS aprenderán las heurísticas y algoritmos en teoría e igualmente los verán en funcionamiento tanto con el software como con la celda.

MARCO TEÓRICO

La manufactura es una actividad humana practicada desde el comienzo de la civilización, inicialmente se ejerció como una acción individual y muy artesanal, luego surgió una primera revolución industrial con la máquina de vapor que permitió producir en grande cantidades; después floreció una segunda revolución

[1] FMS: flexible manufacturing system, sistema de manufactura flexible

industrial con la aparición del computador y diversos aparatos electrónicos que beneficiaron la producción en las empresas permitiendo producir a menores costos, mayor eficiencia, control y administración.

"El sistema de manufactura implica la fabricación de productos que satisfagan a los clientes, en las fechas y términos estipulados con la calidad requerida y bajo principios de racionalización, de minimización de costos y maximización de utilidades.

En la administración de manufactura debemos prever la demanda de productos y factores de producción, ajustar la programación del trabajo, determinar los mecanismos de control, llevar a cabo el análisis y administración de las adquisiciones y del control de inventarios, determinar la localización de la planta, llevar a cabo métodos de trabajo y determinar los medios de medición, así como llevar a cabo el análisis y el control de costos."[2]

Los sistemas de manufactura son sistemas que se estructuran a través de actividades y procesos relacionados entre sí para producir un bien o un servicio buscando por medio de métodos eficientes dar valor agregado al cliente. Además, hoy en día la demanda de productos o servicios obedece a cambios frecuentes y en ocasiones drásticos, por ello un sistema de manufactura además de producir debe responder rápidamente a dicha demanda sin olvidar la calidad que el cliente espera recibir al obtener el bien o servicio; debido a esto se vio la necesidad de mejorar el sistema de producción para que respondiera a estas necesidades y ser capaz de producir varios productos, en diversos volúmenes ajustados a la demanda, sin aumentar costos ni disminuir la calidad. En respuesta a lo anterior surgió el sistema de manufactura flexible.

---

[2] Tomado de http://www.joseacontreras.net/manuf/page.htm, 08 de octubre del año 2011

“Antes de empezar a hablar de manufactura flexible debemos conocer algunos conceptos de automatización que pueden ser desconocidos para algunos, estos conceptos son automatización fija programable; para luego abordar el tema de manufactura flexible.

La automatización fija se caracteriza por la secuencia única de operaciones de procesamiento y ensamble. Sus operaciones son simples pero su integración en las diferentes estaciones de trabajos dan lugar a sistemas complejos y costos aplicados a la producción masiva pero cuando se cambia de un producto a otro, es necesario la puesta a punto manual de todo el equipo implicando otras tareas, cambio de herramientas y utilage.

En la automatización programable la secuencia de operaciones es controlada por un programa y puede cambiar para diferentes configuraciones del producto, Este tipo de automatización es apropiado para la producción por lotes de tamaño bajo o medio, la inversiones equipo es alta, y las velocidades son inferiores a las características de la producción fija y el tiempo de preparación de los equipos para cada lote es considerable. (Ej. El control numérico).

En cambio la automatización flexible es una extensión de la programable que se ha desarrollado durante las últimas décadas a la par de los computadores y de la tecnología de la automatización. Además de la capacidad para trabajar diferentes secuencias de operaciones en forma automática permitiendo la fabricación continúa de mezclas variables de productos con tiempos de preparación y cambio de herramientas virtualmente nulos, al pasar de un producto a otro. Esta requiere alta inversión en equipo adaptado a las necesites del cliente y está orientada a la manufactura de partes afines en lotes de tamaño bajo y medio bajo a una velocidad media de producción.

La automatización flexible ha hecho factible los sistemas de manufactura flexible y la manufactura integrada por computador."[3]

SISTEMA DE MANUFACTURA FLEXIBLE[4]

Un sistema de manufactura flexible utiliza un conjunto de máquinas-herramientas de control numérico computarizado (CNC) y estaciones de soporte interconectados a través de un sistema automático de manejo de materiales y coordinado por un computador central.

Componentes:

- Máquinas-herramientas: se utiliza para crear partes o piezas, deben ser genéricas para asumir cambios además de poder realizar tareas distintas, son fácilmente programables y automáticas así se disminuye tiempo de alistamiento.
- Control numérico computarizado (CNC)[5]: es un sistema que permite controlar en todo momento la posición de un elemento físico, normalmente una herramienta que está montada en una máquina. Esto quiere decir que mediante un software y un conjunto de órdenes, controlaremos las coordenadas de posición de un punto (la herramienta) respecto a un origen (0,0,0 de máquina), o sea, una especie de GPS pero aplicado a la mecanización, y muchísimo más preciso.
- Estaciones de soporte: permiten una capacidad de trabajo integral, pueden ser: de almacenamiento (de materia prima, producto terminado o ambos),

---

[3] Tomado de http://ingenieria.udea.edu.co/CURSOS/DOCUM/manofacturaflexible(opcional).doc, 08 de octubre del año 2011.
[4] Tomado de http://www.enotes.com/management-encyclopedia/flexible-manufacturing, 08 de octubre del año 2011.
[5] Ver mayor información en http://cadcamcae.wordpress.com/2007/06/14/el-control-numerico-por-computadora-el-cnc/

de control automático de calidad (se usa un patrón comparativo) o de manejo de materiales automáticos (cargar o descargar en las estaciones de trabajo; brazo robot; y otros para transporte de material entre estaciones; bandas de transporte o vehículos guiados -AGV-).

- Computador central: sistema de información encargado de controlar los movimientos y los procesos; de este modo se puede programar el trabajo con poca supervisión por largos periodos de tiempo.

Objetivo:

Acercarse a la eficiencia y economía del sistema de manufactura en masa pero con tal flexibilidad que permita producir en las cantidades demandadas por el mercado y en el tiempo esperado.

Beneficios:

- Menos residuos.
- Menos estaciones de trabajo.
- Cambios rápidos de herramientas, moldes y maquinaria.
- Menor tiempo de inactividad u ocio.
- Mejor control de calidad.
- Uso más eficiente de la maquinaria.
- Reducción de inventarios del producto en proceso.
- Aumento de la capacidad de producción.
- Flexibilidad en las cantidades a producir.

Limitaciones:

- Debe programarse la producción por familias de piezas con características similares, es necesario clasificar las piezas adecuadamente según sus características.
- Se tiene menor tiempo para programar la producción y se debe hacer a más largo plazo.
- Mayor complejidad y costos.
- A veces la utilización de los equipos no es tan alta como se podría esperar.
- Falta de conocimientos técnicos, la incompetencia de gestión, y la deficiente aplicación del proceso del sistema de manufactura flexible.
- Rápido cambio de la tecnología y la reducción de los ciclos de vida del producto puede causar que los bienes del capital se conviertan rápidamente en obsoletas.
- Los costos de la capacitación de personal puede llegar a ser relativamente altos o bien se necesita mano de obras especializada.

Problemas a resolver para optimizar la producción:

- Controlar la producción.
- Carga en las máquinas.
- Ruta de las partes.
- Agrupación de partes.
- Administración de las herramientas.
- Ubicación de dispositivos de almacén.

Para resolver dichos problemas se han establecido algoritmos de heurística, como las heurísticas de Greddy, de formación de lotes y de Carga, que permiten obtener una buena solución para el control de la producción; las empresas pueden utilizar

cualquier lenguaje de programación, para hallar una solución rápida y eficaz. Cabe notar que existe un programa llamado Robocell que permite simular la utilización de los robots en un sistema de manufactura flexible.

Clases de sistemas:[6]

- Sistemas de manufactura de espectro reducido: producen un número limitado de partes con pequeñas diferencias en la geometría de su diseño.
- Sistema de manufactura flexible de alto espectro: Producen familias de partes (agrupación de partes con características similares) numerosas con variaciones sustanciales en la configuración de las partes y en la secuencia de operaciones
- Módulo de manufactura flexible: Unidad compuesta por una sola máquina con capacidad para cambio de herramientas equipo para manejo de materiales y almacenamiento temporal de partes
- Celda de manufactura flexible: grupo de modos que comparten el mismo sistema de materiales.
- Sistema de manufactura flexible de máquinas múltiples: conjunto de módulos conectados por medio de un sistema de manejo de materiales capaz de visitar dos o más máquinas al tiempo.

[6] Tomado de http://ingenieria.udea.edu.co/CURSOS/DOCUM/manofacturaflexible(opcional).doc, 08 de octubre del año 2011.

OBJETIVOS DEL PROYECTO

OBJETIVO GENERAL

Desarrollar un software para la Facultad de Ingeniería Industrial de la Universidad Tecnológica de Pereira que facilite la enseñanza del control de la producción en un sistema de manufactura flexible.

OBJETIVOS ESPECIFICOS

- Recopilar todos los algoritmos que se van a resolver con el software académico para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira.
- Diseñar el software académico para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira, describiendo datos de entrada y de salida.
- Programar el software académico para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira.
- Presentar el software académico desarrollado para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira, dando un manual y ejemplos para demostrar su aplicabilidad.

ALCANCE

El software es académico destinado a resolver la heurística de Greddy, la heurística de formación de lotes y la heurística de carga, algoritmos que solucionan los problemas de la producción en un sistema de manufactura flexible, como seleccionar cuales partes se van a producir en el sistema o continuar produciéndolas de la forma tradicional, también orienta en la formación de lotes de producción según las fechas de entrega de las piezas ordenadas para el FMS, y

por ultimo también indica como producir en un FMS equilibrando las cargas primero en los tipos de máquinas y luego en cada máquina individualmente.

LIMITACIONES

El software al ser netamente académico tiene previstos muy pocos errores, por tanto se debe tener mayor cuidado en el ingreso de datos, de que si correspondan al tipo de dato necesario, si es numero o letras, además como el programa es desarrollado para ingresar los datos y resolver la heurística mas no está diseñado para mostrar paso a paso la solución de los problemas, es decir solo muestra el resultado de la heurística.

METODOLOGÍA EMPLEADA

Para desarrollar este trabajo se siguió la siguiente metodología:

Tabla 1. Metodología

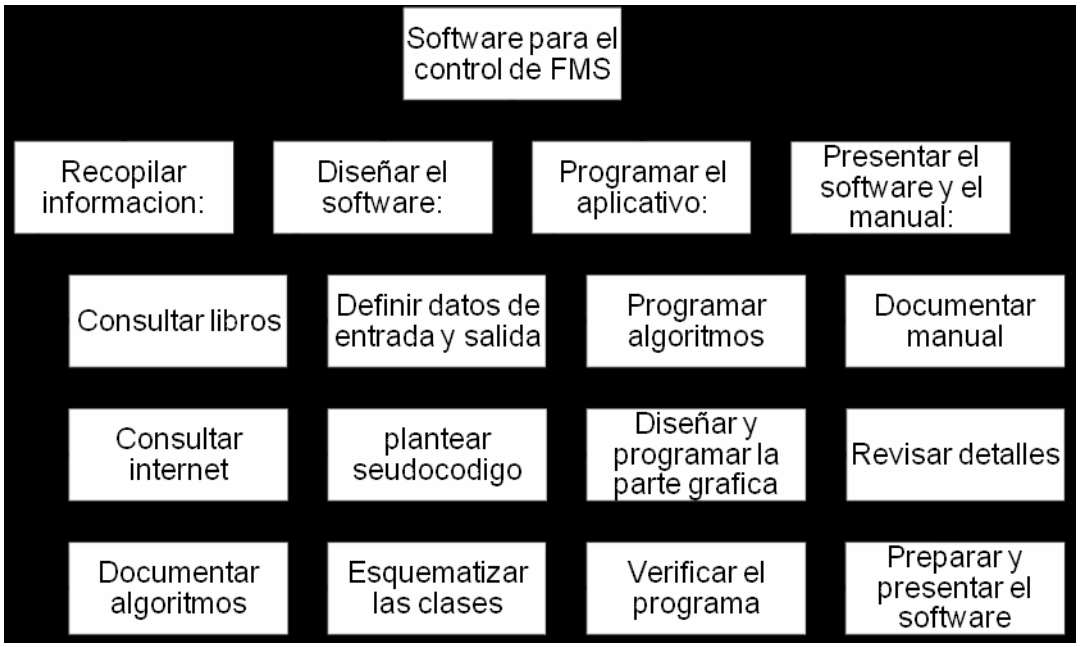
<table>
<tr><th>Objetivo general</th><th>Objetivos específicos</th><th>Actividades</th><th>Recursos</th></tr>
<tr><td rowspan="3">Desarrollar un software para la Facultad de Ingeniería Industrial de la Universidad Tecnológica de Pereira que facilite la enseñanza del control de la producción en un sistema de manufactura flexible.</td><td rowspan="3">Recopilar todos los algoritmos que se van a resolver con el software académico para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira.</td><td>Se consultó libros de producción buscando los algoritmos que se utiliza para controlar un FMS.</td><td>Transporte</td></tr>
<tr><td rowspan="2">Se consultó en páginas web los algoritmos para programar un FMS.</td><td>Computador</td></tr>
<tr><td>internet</td></tr>
</table>

		Se documentó los algoritmos utilizados para programar un FMS.	Computador
	Diseñar el software académico para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira, describiendo datos de entrada y de salida.	Se definió los datos de entrada y de salida.	Papel y lápiz
			Algoritmos recopilados
		Se planteó un seudocódigo de los algoritmos a resolver.	Papel y lápiz
			Algoritmos recopilados
		Se hizo un esquema de las clases del programa.	Algoritmos recopilados
			Papel y lápiz
	Programar el software académico para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira.	Se programo los algoritmos del FMS en JAVA.	Computador
			Programa de soporte para programar en java
		Se diseñó y programo la parte gráfica del programa planteado.	Computador
			Programa de soporte para programar en java
		Se realizó pruebas que verifiquen la correcta programación del software.	Computador
			Programa de soporte para programar en java
			Programa diseñado
			Ejemplos aplicativos de un FMS

	Presentar el software académico desarrollado para la Facultad Ingeniería Industrial de la Universidad Tecnológica de Pereira, dando un manual con ejemplos para demostrar su aplicabilidad.	Se documento el trabajo realizado en forma de manual.	Computador
		Se reviso la forma del documento para la presentación.	Computador
		Se preparó la presentación de los resultados del trabajo.	Computador
			Impresora
		Se presento el software y ejemplos que muestren su funcionamiento.	Computador

Fuente: Responsable del Proyecto

Ilustración 1. Resumen de la metodología.



Fuente: Responsable del Proyecto

APLICACIÓN DEL PLOYECTO

El software desarrollado con este proyecto servirá a la facultad de Ingeniería Industrial de la Universidad Tecnológica de Pereira para complementar los estudios realizados en la Celda de Manufactura Flexible por docentes y estudiantes además también soportara técnicamente la electiva sobre este tema del pensum de la carrera de la facultad, permitiendo ver aplicado el tema visto durante el transcurso del semestre en el cual se vea dicha materia. Además se puede plantear la posibilidad de tener un grupo de investigación alrededor del tema.

1. RECOPILACIÓN DE INFORMACIÓN

1.1. HEURÍSTICA DE GREDDY

Esta heurística se utiliza en los casos donde se está comprando diferentes piezas pero se dispone de la posibilidad de producirlas en un sistema de manufactura flexible, entonces se evalúa el beneficio (Ahorro) y se seleccionan las piezas a producir teniendo como limitante el tiempo disponible para la producción. Para determinar que piezas se deben producir y cuales se deben seguir comprando, se debe tener la siguiente información inicial:

Datos iniciales para resolver el problema de selección de partes:

- Precio de compra unitario (PCU_i)
- Costo de material por unidad (CMU_i)
- Demanda por periodo (D_i)
- Tiempo de proceso por unidad (TPU_i)
- Costo de operación por unidad (COU_i)
- Tiempo total disponible para la producción en el cuello de botella (P)

Pasos para resolver el problema de la selección de partes:

- Calcular el ahorro por unidad (AU_i).

$$AU_i = PCU_i - (CMU_i + COU_i \times TPU_i)$$

- Calcular el ahorro por periodo por cada pieza (S_i).

$$S_i = AU_i \times D_i$$

- Calcular tiempo de proceso por periodo para cada pieza (P_i).

$$P_i = TPU_i \times D_i$$

- Calcular la relación $\frac{S_i}{P_i}$ para cada pieza.
- Ordenar las piezas en forma descendente según la relación $\frac{S_i}{P_i}$, sin tener en cuenta las relaciones negativas.
- Asignar piezas al sistema de acuerdo al orden anterior garantizando no violar la restricción del tiempo disponible en la estación cuello de botella (P).

Para la solución se indica cuales son las piezas a producir en el FMS y cuál es el ahorro total obtenido (sumatoria de todos los ahorros por periodo). Esta solución se podría utilizar como parte de los datos iniciales de la heurística de formación de lotes, es decir el usuario podrá utilizar las piezas seleccionadas como las piezas a construir y las cuales requieren definir los lotes de producción. Estas piezas son cargadas en los datos iniciales de la heurística de lotes con días restantes para la entrega 0, y tamaño de la orden la demanda establecida en los datos iniciales de la presente heurística, y como si se pudiera fraccionar el lote.

1.2. HEURÍSTICA DE FORMACIÓN DE LOTES

Se usa cuando se sabe que piezas se van a producir y el número de portaherramientas disponible es menor que las herramientas necesarias para fabricarlas, por tanto se hace necesario producir por lotes buscando reducir los

tiempos de alistamiento de modo que los alistamientos sean entre lotes y no interlotes. Los lotes están formados por pedidos con fechas de entrega similares, en su conjunto todas las piezas del lote garantizan el uso de todas las máquinas y el número de herramientas distintas para fabricar las piezas, en el lote son menores o iguales al número de portaherramientas del sistema.

Datos iniciales para resolver el problema de formación de lotes:

- Nombre de la pieza pedida.
- Tamaño de la orden por pieza.
- Fecha de entrega en días faltantes para la entrega de la orden (d_i).
- Nombre del tipo de máquina.
- Numero de máquinas por cada tipo de máquina (m_j).
- Tiempo disponible para la producción en cada tipo de máquina (P_j).
- Numero de portaherramientas disponible por cada tipo de máquina (K_j).
- Máquinas necesarias para la producción de cada pieza.
- Tiempo de proceso necesario para la producción de cada pieza en su respectiva máquina.
- Herramientas necesarias para la producción de cada pieza en su respectiva máquina.
- Se debe indicar la posibilidad de fraccionar los pedidos por si no se puede programar todo un mismo pedido en el mismo lote.

Pasos para resolver la heurística de formación de lotes:

- Ordenar de manera ascendente de acuerdo a la fecha de entrega d_i del pedido. Cuando las fechas de entrega son iguales, se da prioridad al

pedido con mayor tiempo total de producción requerido para así maximizar la utilización de las máquinas.

- Se asignan los pedidos al lote de manera secuencial siguiendo la lista anteriormente obtenida, teniendo en cuenta no sobre pasar el tiempo total disponible en las máquinas de cada tipo por periodo, ni superar el numero de portaherramientas total disponible en la máquina de cada tipo por periodo, además se debe respetar el hecho de que en un mismo lote todas las piezas deben utilizar todas las máquinas asignadas al lote.

  Si el pedido no se puede programar completo en un lote por requerir más tiempo disponible el pedido se podrá dividir y asignar al lote actual la mayor cantidad dejando el resto para el siguiente lote.

Para la solución se debe indicar el pedido, a que lote se asigno, el tiempo de utilización de las máquinas, las herramientas asignadas y el porcentaje de utilización de cada máquina tanto en tiempo como en portaherramientas.

1.3. HEURÍSTICA DE CARGA

El objetivo de esta heurística es asignar operaciones y herramientas a máquinas, de modo que se balanceen las cargas de trabajo entre máquinas, se minimice el intercambio de material entre máquinas, se maximice la flexibilidad de flujo de material y se minimice la inversión en herramientas (esta última es opcional por que puede afectar la flexibilidad de flujos). Para su solución la heurística cuenta con dos fases:

1.3.1. Fase I

En esta fase se asignan las operaciones a tipos de máquinas, solo se aplica si hay operaciones que pueden ser asignadas a más de un tipo de máquina, el objetivo principal de esta fase es balancear las cargas de trabajo entre tipos de máquinas.

Datos iniciales para la fase I de la heurística de Carga:

- Numero de la pieza.
- Nombre de la operación.
- Tamaño de la orden
- Nombre de la herramienta para realizar la operación.
- Nombre del tipo de máquina.
- Numero de máquinas de cada tipo (m_j).
- Tiempo disponible de fabricación continua en cada tipo de máquina (P_j).
- Numero de portaherramientas por cada tipo de máquina (K_j).
- Tiempo de procesamiento de cada operación en cada tipo de máquina de posible asignación.

Pasos para resolver la fase I de la heurística de Carga:

- Se inicializan $Y_j = P_j$ y $k_j = K_j$.

- Se determina para cada operación i a cuantas máquinas diferentes se puede asignar. De modo que se cumpla $Y_j \geq P_{ij}/m_j$, donde P_{ij} es el tiempo disponible de fabricación si la operación i se asigna a la máquina tipo j; y $k_j \geq \Delta k_{ij}/m_j$, donde Δk_{ij} es el numero de portaherramientas que se tienen que ocupar si operación i se asigna a la máquina tipo j. En otras palabras sí si se cumplen las anteriores condiciones se identifica si es viable o no la producción de la operación i en la máquina tipo j.
- De aquellas operaciones con menor número de posibilidades de asignación se selecciona la que requiera mayor tiempo total de operación (P_{ij}) y se asigna al tipo de máquina con más tiempo disponible después de la asignación. Es decir se elige: La operación donde $i^* = max_i(\min P_{ij})$ y la máquina donde $j^* = max_j(Y_{ij} - P_{ij}/m_j)$

En la solución se debe indicar para cada operación i a que máquina j fue asignada y que herramienta utilizara. Esta solución puede ser utilizada para resolver la fase II o no, dicha decisión queda en las manos del usuario.

1.3.2. Fase II

Se asignan las operaciones y las herramientas a máquinas dentro de cada tipo, para garantizar el balanceo de cargas de trabajo entre grupos de máquinas. Esta heurística se aplica tantas veces como tipos de máquina halla es decir para cada tipo de máquina.

Datos iniciales para resolver la fase II de la heurística de carga:

- Numero de la pieza a fabricar.
- Nombre de la operación a realizar.
- Tiempo de procesamiento de la operación.
- Herramienta utilizada para la fabricación de la operación.
- Nombre del tipo de máquina asignada.
- Cantidad de máquinas de cada tipo.
- Tiempo disponible para la producción continua en cada tipo de máquina.
- Numero de portaherramientas disponible por cada tipo de máquina.

Pasos para resolver la fase II de la heurística de carga:

- Formar clúster: agregando de forma secuencial operaciones al clúster hasta la carga de trabajo que permita el tiempo disponible del tipo de máquina. Teniendo en cuenta que las operaciones sean de una misma pieza; se asigna un nuevo clúster cuando una nueva operación viola la restricción de la máxima carga de trabajo en tiempo que se puede asignar, cuando se requieran más herramientas de las que permite la máquina o cuando se requiera una nueva máquina.
- Formar grupos de máquinas: Se halla g_j que es el número de grupos en los que se debe dividir las máquinas; g_j es igual al menor entero mayor o igual que σ_j/k_j es decir, $g_j = \left\lceil \frac{\sigma_j}{k_j} \right\rceil$ donde σ_j es el numero de tipos de herramientas distintas que se deben programar en máquinas tipo j, en otras palabras es el numero de portaherramientas a ocupar en máquinas tipo j; y k_j es el numero de portaherramientas disponibles en máquinas tipo j.

- Asignar clúster a grupo de máquina: Las máquinas deben de ser asignadas a los grupos de tal manera que el tamaño de ellos sean aproximadamente igual.

  Se asigna aplicando la regla LPT[7] entre grupos:

  - ✓Ordenar clúster de forma descendente de acuerdo a sus tiempos totales de operación.
  - ✓Dar prioridad a los clúster con mayor tiempo total de operación requerido y se asigna al grupo de máquinas con el mayor tiempo disponible después de asignar el clúster.

Si el tiempo total de operación requerido es igual se da prioridad al clúster que tenga mayor número de operaciones, y si tienen el mismo número de operaciones, se asigna el clúster que necesite menor número de herramientas.

Para la solución se debe indicar el tipo de máquina (j), el número de máquinas del tipo j asignado, el grupo de máquinas respectivo, las operaciones, las herramientas asignadas y el tiempo de ocupación del grupo.

1.4. SOBRE JAVA

[7] Largest Processing Time: Se le da prioridad a las tareas de mayor tiempo de procesamiento y se asigna a la máquina que quede con mayor tiempo disponible después que la operación ha sido asignada.

Para el desarrollo del software objeto del presente trabajo se ha elegido el lenguaje JAVA para la programación de este, debido a sus gráficos y a su compatibilidad con todos los sistemas operativos.

Breve reseña histórica[8]:

Sun Microsystems desarrolló, en 1991, el lenguaje de programación orientado a objetos que se conoce como Java. El objetivo era utilizarlo en un set-top box, un tipo de dispositivo que encarga de la recepción y la decodificación de la señal televisiva. El primer nombre del lenguaje fue Oak, luego se conoció como Green y finamente adoptó la denominación de Java.

La intención de Sun era crear un lenguaje con una estructura y una sintaxis similar a C y C++, aunque con un modelo de objetos más simple y eliminando las herramientas de bajo nivel.

Los pilares en los que se sustenta Java son cinco: la programación orientada a objetos, la posibilidad de ejecutar un mismo programa en diversos sistemas operativos, la inclusión por defecto de soporte para trabajo en red, la opción de ejecutar del código en sistemas remotos de manera segura y la facilidad de uso.

Lo habitual es que las aplicaciones Java se encuentren compiladas en un bytecode (un fichero binario que tiene un programa ejecutable), aunque también pueden estar compiladas en código máquina nativo.

---

[8] Tomado de Definición de Java - Qué es, Significado y Concepto: http://definicion.de/java/#ixzz2A2cT4Csv, 22 de octubre del año 2012.

Sun controla las especificaciones y el desarrollo del lenguaje, los compiladores, las máquinas virtuales y las bibliotecas de clases a través del Java Community Process. En los últimos años, la empresa (que fue adquirida por Oracle) ha liberado gran parte de las tecnologías Java bajo la licencia GNU GPL.

La aplicación de Java es muy amplia. El lenguaje se utiliza en una gran variedad de dispositivos móviles, como teléfonos y pequeños electrodomésticos. En los navegadores web, Java permite desarrollar pequeñas aplicaciones conocidas como applets que se incrustan en el código HTML de las páginas. El navegador debe contar con un plug-in que permita ejecutar las aplicaciones Java.

Para el desarrollo del lenguaje se utilizó el entorno de desarrollo integrado (EDI) llamado NetBeans:

“NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

Al día de hoy hay disponibles dos productos: el NetBeans IDE y NetBeans Platform:

- NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.
- También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se

integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. Ambos productos son de código abierto y gratuito para uso tanto comercial como no comercial."[9]

[9] Tomado de http://netbeans.org/index_es.html, 22 de octubre del 2012

2. DISEÑO DEL SOFTWARE

Se brindara un resumen de lo que se espera que haga los algoritmos en el programa para mostrar los requisitos planteados:

Al abrir el programa APROTI (Asistente de Producción y Tiempos) se tendrá la opción de resolver cualquier heurística para el desarrollo del sistema de manufactura flexible, además se podrán conectar en el siguiente orden heurística de Greddy, formación de lotes y heurística de carga para resolver un problema tan completo como sea necesario según las necesidades.

2.1. HEURÍSTICA DE GREDDY

Se ingresa los datos de cada pieza:

- Número de la pieza.
- Demanda en el periodo.
- Costo de operación por hora.
- Precio de compra por unidad.
- Costo de material pro unidad.
- Tiempo de proceso por unidad.

Cada pieza se debe poder modificar o eliminar para efectos de la flexibilidad del software, además se puede guardar el problema para un análisis posterior.

Y también se ingresa el tiempo disponible en el cuello de botella por periodo, el cual limitara la selección de piezas. Luego de tener estos datos se debe calcular la relación ahorro/tiempo de fabricación por periodo, para poder ordenar las piezas descendentemente según este cálculo y asignar las piezas hasta que se cope el tiempo en el cuello de botella, estos pasos se verán más detallados a continuación:

- Se calcula el ahorro (S_i) por periodo si la pieza i se fabrica en el sistema:

  $$S_i = \left(PCU_i - (CMU_i + POU_i \times TPU_i)\right) \times D_i$$

  Donde:

  $D_i =$ Demanda en el periodo.

  $POU_i =$ Costo de operación por hora.

  $PCU_i =$ Precio de compra por unidad.

  $CMU_i =$ Costo de material por unidad.

  $TPU_i =$ Tiempo de proceso por unidad.

- Calcular el tiempo (P_i) necesario de fabricación por periodo si la pieza i se fabrica en el sistema:

  $$P_i = TPU_i \times D_i$$

  Donde:

D_i = Demanda en el periodo.

TPU_i = Tiempo de proceso por unidad.

Con los anteriores cálculos se obtiene el valor de la relación ahorro/tiempo S_i/P_i

- Ordenar cada pieza con ahorro positivo en forma descendente con respecto a la relación ahorro/tiempo, así:

Si *ahorro* de la pieza *i* es positivo;

Entonces comparar la relación *ahorro/tiempo* de la pieza *i* con las relaciones de las otras piezas y ubicarla de tal modo que:

$$\frac{S_1}{P_1} > \cdots > \frac{S_i}{P_i} > \frac{S_{i+1}}{P_{i+1}} > \cdots > \frac{S_n}{P_n}$$

Donde n = número de piezas a evaluar

Es decir que la pieza con el mayor valor en la relación este en primer lugar y la de menor valor de última posición.

La anterior lista de piezas delimita el orden de asignación, que me definirá las piezas a producir según el beneficio costo-tiempo. Se asigna teniendo en cuenta el tiempo del cuello de botella por periodo y el tiempo de operación necesario para producir la pieza i en el periodo, teniendo en cuenta que la suma de todos los

tiempos necesarios para producir las piezas asignadas no supere el tiempo del cuello de botella:

Generar un contador que se iguale al valor del cuello de botella por periodo;
Generar una variable donde se sumen todos los ahorros por periodo de las piezas;
Iniciar una lista donde se va a agregar las piezas asignadas;
Tomar la lista generada de las piezas según las relaciones ahorro/tiempo:
Para cada pieza i en el orden de la lista, evaluar:
Si el ahorro de la pieza i es positivo y el tiempo disponible en el cuello de botella (el contador) menos la demanda en el periodo por el tiempo de proceso es mayor o igual a cero

Si (Ahorro > 0 && cuello - (Tiempo* Demanda) >= 0)

Entonces asignar la pieza i a la lista de piezas a producir, restar al contador del tiempo disponible en el cuello de botella el producto del tiempo de proceso de la pieza por la demanda por periodo de la pieza, y sumar a la variable que guarda el ahorro total el ahorro en el periodo de la pieza asignada:

A npiezas asignar la pieza

Cuello -= Tiempo * Demanda;

ahorroTotal += Ahorro;

Por último se muestra la respuesta de la heurística: lo cual se resume en mostrar la lista de las piezas asignadas y el ahorro total generado. Se da la posibilidad de que la respuesta se utilice como datos iniciales de las piezas para la heurística de formación de lotes, guardando el resultado y luego importándolo en heurística de formación de lotes.

2.2. HEURÍSTICA DE FORMACIÓN DE LOTES

Al iniciar la heurística de formación de lotes se muestra la ventana para el ingreso de datos, la heurística requiere información de piezas, de máquinas y de herramientas, además de definir que máquinas y que herramientas requiere cada pieza para ser producida mas el tiempo requerido para la producción de dicha pieza.

El primer paso que se debe hacer es definir las piezas a producir y sus características, bien sea ingresadas a mano o importar una solución anteriormente guardada en heurística de Greddy:

- El número de la parte a producir.
- Tamaño de la orden de la pieza, si la parte viene de los datos de la heurística de formación de lotes es igual a la demanda del periodo.
- Los días restantes para la entrega de la orden, si la parte es tomada de la heurística de carga los días restantes es cero.

Se debe confirmar si la orden de la parte a producir se puede fraccionar o no, si la parte es tomada de la heurística de Greddy se inicia la parte como si se pudiera fraccionar el pedido. Las partes se pueden agregar, modificar y eliminar, del listado de partes que intervienen en la solución de la heurística.

Después se requiere la información de las herramientas que intervienen en el problema, se tendrá en cuenta que diferentes piezas pueden necesitar la misma herramienta.

Como siguiente paso es ingresa los datos de cada máquina que se requiere en la producción de las piezas:

- Nombre de tipo de máquina.
- Numero de máquinas de este tipo.
- Tiempo disponible para la producción continua.
- Numero de portaherramientas en el tipo de máquina.

  Cada máquina se puede ingresar, eliminar o modificar.

Para poder calcular la solución de la heurística se debe definir para cada pieza:

- La máquina
- La herramienta
- El tiempo de procesamiento que necesita la parte para ser producida en el sistema. Dichas asociaciones se pueden ver para modificarse, eliminarse o tan solo para estar informados de cuáles son las relaciones parte-máquina-herramienta-tiempo.

El problema se puede guardar para ser calculado en otro momento o para probar su sensibilidad a los cambio.

Por último para solucionar la heurística se cumplen los siguientes pasos:

- Se debe ordenar ascendentemente las partes según sus fechas de entrega, según los días que faltan para la entrega de la orden, de modo que la parte que le falten menos días para la fecha de entrega se programe primero.
- Ordenar las partes de tal modo que los días restantes para la entrega estén ordenados así:

$$d_1 < \cdots < d_i < d_{i+1} < \cdots < d_n$$

Donde d_i son los días restantes para la fecha de entrega de la parte i y n es el número de piezas del problema.

Entonces se debe seguir un algoritmo semejante al siguiente:

Si $d_i == d_{i+1}$

Entonces

Si tiempo *procesamiento* * *tamaño de la orden* de la pieza i es mayor que el producto de la pieza i+1

Entonces

La pieza i va de primero

De lo contrario

La pieza i+1 va de primero

De lo contrario

Si $d_i < d_{i+1}$

Entonces

La pieza i va de primero

De lo contrario

La pieza i+1 va de primero

- Con la lista anterior se procede a formar los lotes de producción:

Se crean dos contadores uno igual al tiempo total disponible en las máquinas tipo j y otro con el número de portaherramienta disponible en la máquina j para evaluarse:

Para todas las piezas:

Si contador del tiempo total disponible en la máquina j – tiempo de producción de la pieza i en la máquina j >= 0 y además que el contador de portaherramienta disponible en la máquina j – portaherramienta necesario para producir la pieza i en la máquina j >= 0

Entonces

Asignar al lote l la pieza i;
Contador tiempo total disponible en la máquina j – = tiempo de producción de la pieza i en la máquina j;
Contador de portaherramienta disponible en la máquina j – = portaherramienta necesario para producir la pieza i en la máquina j;

De lo contario

Si fraccionar el lote == verdadero

Entonces

Asignar al lote l, el número de piezas i == mayor entero menor o igual que [(tiempo total disponible en la máquina j – contador del tiempo total

disponible en la máquina j)/tiempo de procesamiento de la pieza i];

Asignar al siguiente lote l+1 la pieza i el tamaño de la orden restante: Tamaño de orden – (mayor entero menor igual que [(tiempo total disponible en la máquina j – contador del tiempo total disponible en la máquina j)/tiempo de procesamiento de la pieza i]);

De lo contrario

Asignar la pieza i al siguiente lote l+1;

Para la asignación en el siguiente lote se debe de evaluar la condición de: Si el contador del tiempo total disponible en la máquina j – tiempo de producción de la pieza i en la máquina j >= 0 y que el contador de portaherramienta disponible en la máquina j – portaherramienta necesario para producir la pieza i en la máquina j >= 0, pero con los contadores reiniciados para el lote de asignación actual.

Ya con los lotes formados es necesario mostrar la solución la cual sería mostrar cada lote como está formado, que piezas se van a producir y las herramientas que se van a utilizar. Además se tendrá la opción de guardar la solución de los lotes para ser utilizada como datos iniciales para resolver la heurística de carga, pero se debe tener en cuenta que si una o más piezas tienen en las operaciones ingresadas más de una máquina o herramienta. Al momento de importar en la heurística de carga no importara las asociaciones y a las operaciones se les asignará solo la herramienta que se asocio a la primera máquina.

2.3. HEURÍSTICA DE CARGA

2.3.1. Fase I

Primero se deben introducir los datos iniciales para resolver el problema de carga en su primera fase, bien sea ingresadas manualmente o se importe desde heurística de formación de lotes:

Datos sobre las operaciones de las piezas:

- Numero de la pieza.
- Nombre de la operación.
- Tamaño de la orden, cuantas veces se debe realizar la operación en el periodo.
- Herramienta necesaria para realizar la operación.

Las operaciones se pueden agregar, modificar y eliminar.

Datos sobre las máquinas que intervienen en la realización de las operaciones de las piezas:

- Nombre del tipo de máquina.
- Numero de máquinas de este tipo.
- Tiempo disponible para producción por periodo en el tipo de máquina.
- Numero de portaherramientas en el tipo de máquina.

Las máquinas se pueden agregar, modificar y eliminar.

Luego se debe asociar cada operación, con la máquina en la que se puede producir y el tiempo que tarda el procesamiento de esta. Es opcional ingresar costos de funcionamiento de las maquinas, que se podrán asociar a las operaciones, en este caso se seleccionara la maquina que tenga un menor costo de funcionamiento al producir la operación seleccionada. Los datos iniciales se pueden guardar para cargarlos en otro momento y seguir trabajando sobre estos; al terminar de ingresar los datos ya se podrá calcular la heurística de carga en la fase 1.

Para el cálculo se deben seguir los siguientes pasos:

- Crear las siguientes variables:

  Y_j = tiempo disponible de fabricación *sin asignación* por máquina tipo j por periodo; y se inicializa con el valor que se introdujo del tiempo disponible para producción por periodo en el tipo de máquina.

  k_j = Numero de portaherramientas disponible *sin asignación* por máquina tipo j por periodo; y su valor inicial es el que se introdujo en el número de portaherramientas en el tipo de máquina.

- Se hacen los siguientes cálculos para cada operación i en cada máquina tipo j donde se podría producir:

  ✓ $\frac{P_{ij}}{m_j}$

  Donde P_{ij} es el tiempo requerido de fabricación si la operación i se asigna a la máquina tipo j, y m_j es la cantidad de máquinas tipo j.

- ✓ $\frac{\Delta K_{ij}}{m_j}$

  Donde ΔK_{ij} es el numero de portaherramientas que se tiene que ocupar si la operación i se asigna a la máquina tipo j, y m_j es la cantidad de máquinas tipo j.

- ✓ Numero de factibilidades de cada operación, es decir, en cuantas máquinas diferentes se puede asignar cada operación, lo cual se calcula así:

  Para todas las operaciones:

  Se crea un contador de número de factibles;

  Para cada máquina:

  Si $\frac{P_{ij}}{m_j} <= Y_j$ y $\frac{\Delta K_{ij}}{m_j} <= k_j$

  Entonces

  Numero de factibles se incrementa en 1

- Después se selecciona la operación que se va a asignar, para saber qué operación se asigna se ordenan las operaciones ascendentemente de acuerdo con el numero de factibles de cada operación, y si son iguales va primero la operación que requiera un mayor P_{ij} (tiempo requerido de fabricación si la operación i se asigna a la máquina tipo j), a continuación se describe más a fondo este paso:

Crear una lista vacía donde se pondrán las operaciones en el orden correcto.

Para todas las operaciones se evalúa:

Si numero de factibles de la operación i y la de i+1 son iguales

Entonces

Si P_{ij} de una operación > P_{ij} de la siguiente operación

Entonces

Guardar la primera operación en la lista y luego la otra

De lo contrario

Guardar en la lista la segunda operación y luego la operación que se evaluó primero

De lo contrario

Si número de factibles de la operación i es menor que el número de factibles de la operación i+1

Entonces

Guardar la primera operación en la lista y luego la otra

De lo contrario

Guardar en la lista la segunda operación y luego la operación que se evaluó primero

✓ De este modo tendremos una lista donde la primera operación cumple con los requisitos necesarios para asignarse, ahora se procede a definir en qué máquina se va a producir la operación, dicha decisión se toma teniendo en cuenta el tiempo disponible después de la asignación, el cual se calcula con la siguiente operación: $Y_j - \frac{P_{ij}}{m_j}$ donde Y_j es tiempo disponible de fabricación *sin asignación* por máquina tipo j por periodo, P_{ij} es el tiempo requerido de fabricación si la operación i se asigna a la máquina tipo j, y m_j es la cantidad de máquinas tipo j. La máquina que

tenga el mayor $Y_j - \frac{P_{ij}}{m_j}$ para la operación que se selecciono en el paso anterior, será la elegida para la fabricación de la operación seleccionada. Este paso se puede plantear de la siguiente manera:

La máquina tipo j seleccionada = max ($Y_j - \frac{P_{ij}}{m_j}$), evaluado en todas las máquinas factibles.

Con las operaciones y las máquinas seleccionadas, se muestra el resultado el cual indicara la operación, herramienta a utilizar y en la máquina que se va a fabricar. Este resultado se utilizara como datos iniciales para la fase II cuando se requiera solucionar también esta fase.

2.3.2. Fase II

Para resolver esta fase se requiere saber la operación de cada pieza que herramienta y que máquina se requiere, además las limitantes de tiempo y portaherramientas de las máquinas, dichos datos se pueden obtener de una solución de la fase I anterior o bien del ingreso de datos manual, este último consta de las siguientes partes:

Datos de las operaciones:

- ✓ Numero de la pieza.
- ✓ Nombre de la operación.

- ✓ Tiempo de proceso total de la operación, si los datos se importaron desde heurística de lotes este valor será la sumatoria de los tiempos asociados durante la asociación de maquinas.
- ✓ Herramienta necesaria para la producción de la operación.

Las operaciones cada pieza se puede agregar, modificar y eliminar.

Datos de las máquinas:

- Nombre del tipo de máquina.
- Numero de máquinas de este tipo.
- Tiempo disponible para el tipo de máquina en el periodo.
- Numero de portaherramientas en el tipo de máquina.

Los tipos de máquina se pueden agregar, modificar y eliminar.

Después se debe asociar cada operación con el tipo de máquina en la cual se va a producir. Los datos de los problemas se pueden guardar y luego cargar para continuar con el cálculo de la heurística de carga en su fase II.

Para resolver esta fase se debe realizar los siguientes pasos que se repetirán tantas veces como máquinas de diferentes tipos hayan:

- Crear clúster de operación con operaciones de la misma pieza que se realicen en la misma máquina y sin exceder ni tiempo ni portaherramientas, lo cual se logra de la siguiente manera:

  Para cada máquina:

Para cada pieza que utilice en su fabricación el tipo de máquina j en análisis, se hace lo siguiente:

Crear un clúster donde se agregaran las operaciones que tenga inicializada una variable igual al valor del tiempo disponible para el tipo de máquina en el periodo (τ) y otra igual al número de portaherramientas en el tipo de máquina (k_j).

Se calcula para cada operación i que se fabrique en el tipo de máquina j en análisis y haga parte de la pieza en consideración:

Si τ – Tiempo de proceso total de la operación i >= 0 y $k_j - 1 >= 0$

Entonces

Agregar la operación i al último clúster creado;

$\tau -=$ Tiempo de proceso total de la operación i;

$k_j -= 1$

De lo contrario

Crear un nuevo clúster inicializando de nuevo τ y k_j

Asignar la operación i a este último.

- Después de obtienen los clúster sobre los cuales recae la solución de esta fase. Se procede a definir cuantos grupos de máquinas tipo j se van a crear, de la siguiente manera:

  Se calcula σ_j que es igual al número de tipos de herramientas distintas que se deben programar en las máquinas tipo j y con el número de portaherramientas disponibles en el tipo de máquina j (k_j) se obtiene g_j el número de grupos en los que se debe dividir las máquinas tipo j, entonces g_j es igual al número menor entero mayor o igual al cociente entre σ_j y k_j, así: $g_j = \left\lceil \frac{\sigma_j}{k_j} \right\rceil$

  Y ahora se define cuantas máquinas de tipo j estarán en cada grupo, la idea es que se asignen de tal manera que el tamaño de los grupos sean aproximadamente iguales, entonces:

  Se define: a == numero de máquinas tipo j;

  b == número de máquinas asignadas;

  g == número de grupos.

  Se crean la cantidad de grupos definidos g.

  Mientras a > b

  Para el contador i ==1, desde que sea menor que la cantidad de grupos creados, y que incremente de uno en uno, entones:

  Se incrementa la cantidad de maquinas en el grupo en 1;

  b Incrementa en 1.

- Después es preciso asignar los clúster a los grupos de máquina de tal manera que se asigne primero los que tengan mayor tiempo de operación y se asigne al grupo de máquinas con el mayor tiempo de disponible después de asignar el clúster, entonces:

  Primero se debe ordenar los clúster descendentemente de acuerdo a sus tiempos totales de operación y en ese orden ir asignándolos, luego:

  Inicializar variables iguales al tiempo disponible de operación en cada grupo de máquinas; para después:

  Evaluar para cada clúster los siguientes pasos:

    Calcular para cada grupo de máquinas:

      Una lista con los tiempos disponible después de asignados cada uno de los clúster,

    El mayor valor de la anterior lista indica al grupo al que se asigna el clúster;

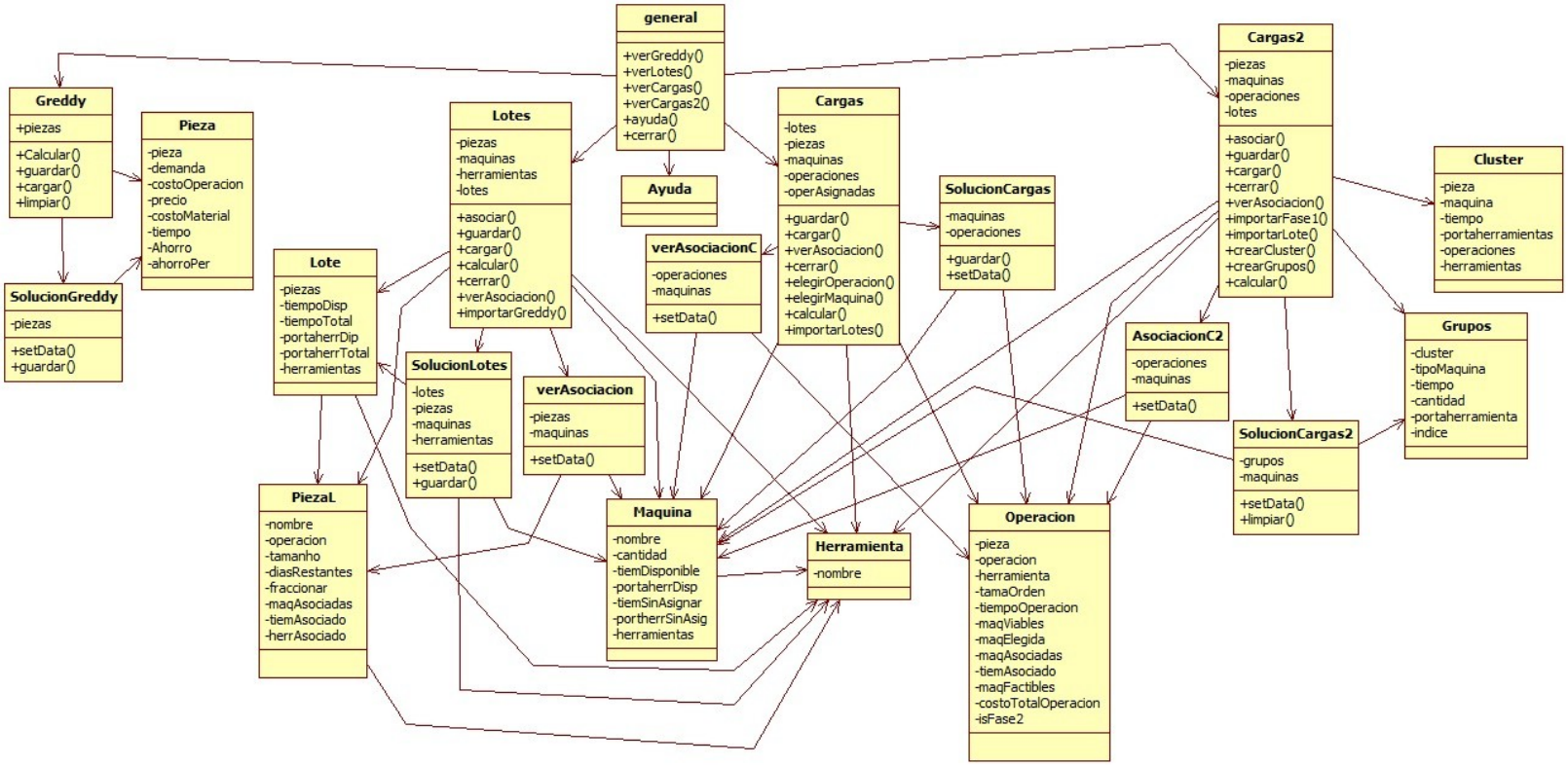
    Luego se le resta a la variable que se inicio al principio con el tiempo disponible de operación en el grupo de máquinas el tiempo de producción del clúster asignado.

Por último se muestra la solución de la heurística la cual contiene para cada tipo de máquina, los grupos formados: con el numero de máquinas que contiene, los clúster que se asignaron, con las operaciones que recoge, las herramientas que se utilizan, el numero de portaherramientas disponible, el porcentaje de utilización del portaherramientas, tiempo de operación asignado, el tiempo de operación disponible, y el porcentaje de utilización del grupo en tiempo.

2.4. ESQUEMA DE CLASES

El software presenta el siguiente esquema entre las clases desarrolladas en el programa (Ver Anexo 1 Resumen Código Fuente):

Ilustración 2. Esquma de clases



Fuente: Responsable del proyecto

En el anterior diagrama se pueden ver cada clase que se diseño y un resumen de las variables y métodos, que se programaron para llevar a cabo las tareas planteadas. Cada recuadro contiene tres secciones la primera casilla contiene el nombre de la clase, en la casilla central se escriben la variables principales de dicha clase y en la parte inferior se detallan el nombre de los métodos que se llevan a cabo para realizar las tareas en cada clase. Y por último se tienen las flechas indican que una clase llama a otra para hacer instancias de este tipo, por ejemplo la clase Greddy llama la clase Pieza para generar objetos de tipo pieza. La clase Maquina es la más utilizada, debido a que la llaman de múltiples clases, esto se llama reutilización de código, y la clase ayuda es la menos utilizada solo la llama la clase general, que indica la pagina principal, y no requiere ningún objeto de las demás clases creadas.

3. VERIFICACION DEL SOFTWARE

Se presentará el desarrollo de unos ejemplos y sus resueltos en el software desarrollado para cada una de las heurísticas:

3.1. HEURÍSTICA DE GREDDY

Ejemplo 1:

Se está considerando la fabricación de 8 partes en una celda automatizada de fabricación estas partes actualmente son compradas a un proveedor se estima que la celda estará disponible en total de 250 horas por periodo, se calcula que el costo de operación de la celda será de 50 $/hora. Determine cuales piezas deberían ser fabricadas en el sistema.

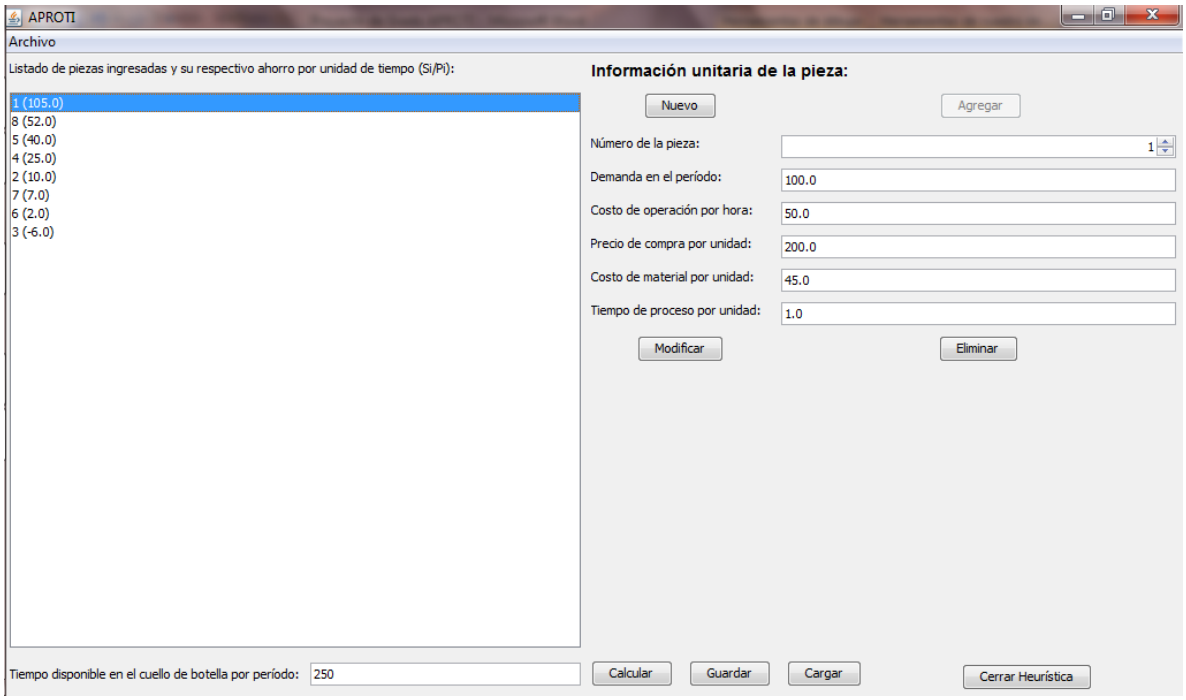
Tabla 2. Datos iniciales ejemplo 1 heurística de Greddy.

Tipo de parte	1	2	3	4	5	6	7	8
Precio de compra unitario	200	155	300	125	300	86	93	165
Costo de material por unidad	45	35	124	50	120	34	36	114
Demanda por periodo	100	50	50	75	60	30	50	600
Tiempo de proceso por unidad	1	2	4	1	2	1	1	0,5
Costo de operación	50							
Tiempo en Cuello de botella	250							

Fuente: Ing. Pedro Daniel Medina

Se ingresa en APROTI todos los datos de las partes y el cuello de botella respectivo:

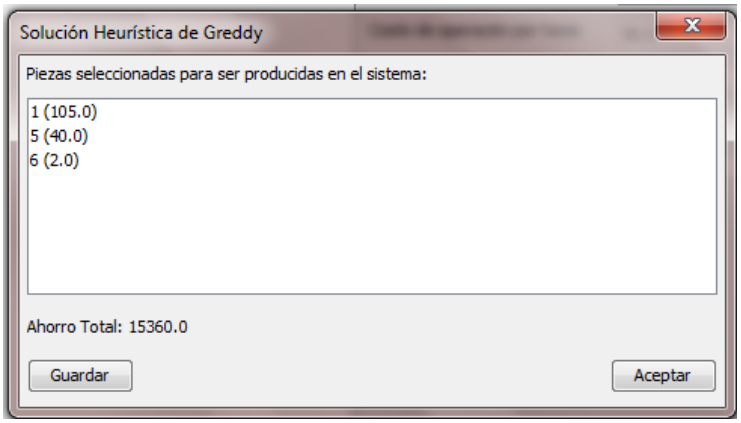
Ilustración 3. Ingreso de datos del ejemplo 1 heurística de Greddy.



Fuente: Responsable del proyecto

Se da clic en el botón calcular y se obtiene el resultado indicando que de las 8 piezas se deben escoger las piezas 1, 5 y 6, y estas permiten un ahorro total de $15.360.

Ilustración 4. Solución del ejemplo 1 heurística de Greddy.



Fuente: Responsable del proyecto

Ejemplo 2:

Se está planeando el montaje de un FMS. El sistema incluye un torno y dos fresas CNC, no se espera que el torno sea tan utilizado como el centro de fresado; se planea que el sistema estará en funcionamiento 16 horas por día, 6 días por semana, y se espera que las máquinas estén disponibles un 90% de este tiempo. Las máquinas operan a un costo de $45 por hora. Usando los datos de la siguiente tabla determine el conjunto de familias de partes a ser producidas en el FMS.

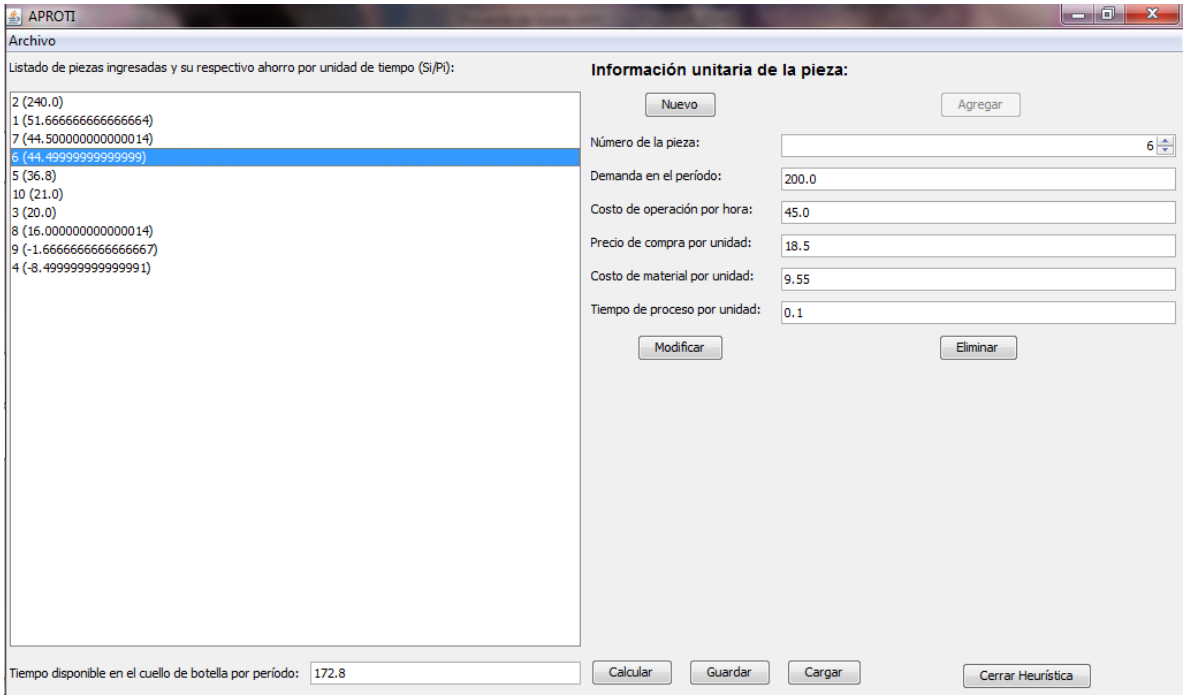
Tabla 3. Datos ejemplo 2, heurística de Greddy.

Tipo de parte	1	2	3	4	5	6	7	8	9	10
Costo unitario si la parte es subcontratada	25,00	20,00	35,00	29,50	62,35	18,50	24,35	10,40	21,95	19,95
Costo de material por unidad	10,50	5,75	12,25	18,55	21,45	9,55	6,45	7,35	15,45	3,45
Demanda semanal	400	800	400	400	800	200	400	400	800	200
Tiempo de proceso por unidad en la fresa CNC	0,15	0,05	0,35	0,3	0,5	0,1	0,2	0,05	0,15	0,25
Tiempo de proceso por unidad en el torno CNC	0,00	0,05	0,00	0,00	0,01	0,02	0,00	0,02	0,00	0,00
Costo de operación	45									
Tiempo en Cuello de botella	86,4									

Fuente: Ing. Pedro Daniel Medina

Se evalúa solo con el tiempo de proceso en la fresa debido a que es el cuello de botella, dicha conclusión se obtiene del enunciado y además en la tabla de los datos también se puede ver que para el torno no se necesita mucho tiempo de operación en el. Además el tiempo del cuello de botella se calcula multiplicando las 16 horas por día * 6 días por semana * 90%, este último es el tiempo que se espera que las máquinas estén disponibles y por último se multiplica por 2 que es la cantidad de fresas que hay disponibles. Teniendo esto en cuenta se ingresan los datos de las piezas y del cuello de botella al software.

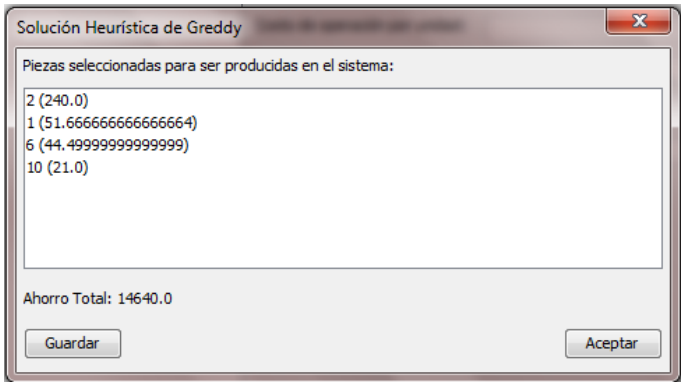
Ilustración 5. Ingreso de datos ejemplo 2, heurística de Greddy.



Fuente: Responsable del proyecto

Se da clic en el botón calcular y sale es siguiente resultado:

Ilustración 6. Solución ejemplo 2, heurística de Greddy.



Fuente: Responsable del proyecto

Lo cual indica que se seleccionan las piezas 1, 2, 6 y 10 para producirse en el sistema y generan un ahorro total de $12.640.

3.2. HEURÍSTICA DE FORMACIÓN DE LOTES

Ejemplo 1

Se deben de programar seis pedidos que cubren cinco tipos de partes diferentes para ser producidos en un sistema de manufactura flexible que cuenta con dos tipos de máquinas distintos. El sistema está compuesto por tres máquinas de tipo A y una de tipo B. Las máquinas son alistadas una vez al día y deben de estar disponibles para una producción de 12 horas continuas por día. Ambos tipos de máquinas poseen dos portaherramientas, por lo tanto son capaces de almacenar dos herramientas cada una. La letra inicial en el nombre de la herramienta indica la máquina requerida. El objetivo es programar los lotes que se producirán en los siguientes días. Tenga en cuenta la información de la tabla 4.

Se ingresan los datos de las piezas, las máquinas y las herramientas según específica en el problema planteado, se habilita fraccionar el pedido ya que no hay nada en el problema que indique lo contrario, ver ilustración 6. Y se asocian según la tabla 4, ver ilustración 7.

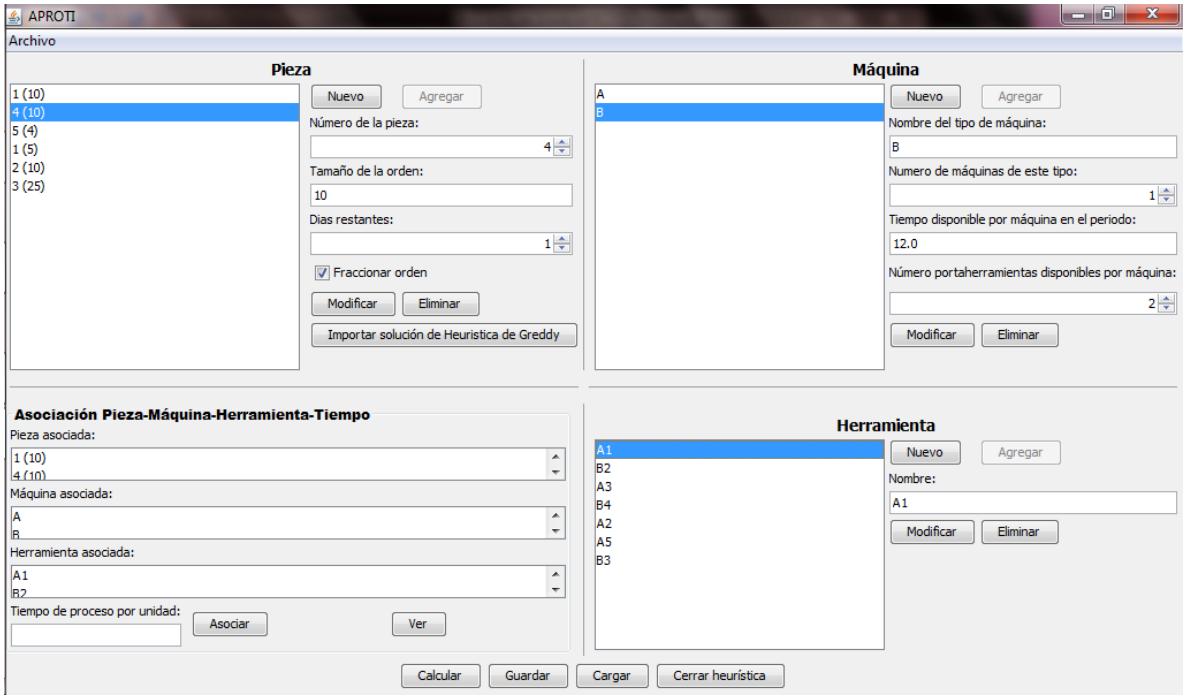
Y se resuelve dado clic en el botón Calcular, ver ilustración 8.

Tabla 4. Datos del ejemplo 1, heurística de formación de lotes.

			Tiempo de procesamiento por unidad (hrs)		
Tipo de parte	Tamaño de la orden	Fecha de entrega	Máquina tipo A	Máquina tipo B	Herramientas
a	10	4	0,3	0,2	A1, B1
d	10	1	0,1	0,2	A1, B1
e	4	2	0,3	0,2	A5, B3
a	5	0	0,1	0,3	A1, B1
b	10	1	1,2	0,0	A2
c	25	1	0,7	0,4	A3, B4

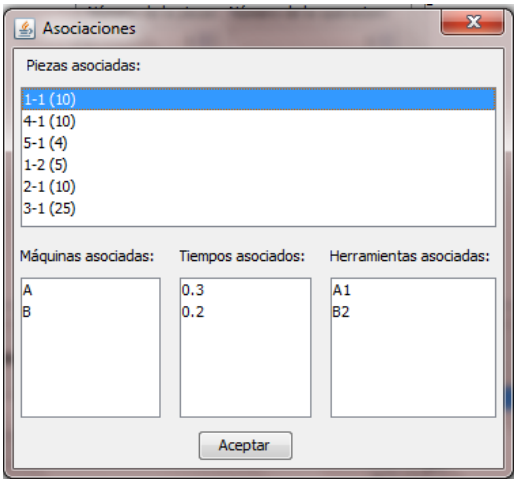
Fuente: El hombre y la Máquina No. 32, Enero-junio de 2009

Ilustración 7. Ingreso de datos ejemplo 1, heurística de formación de lotes.



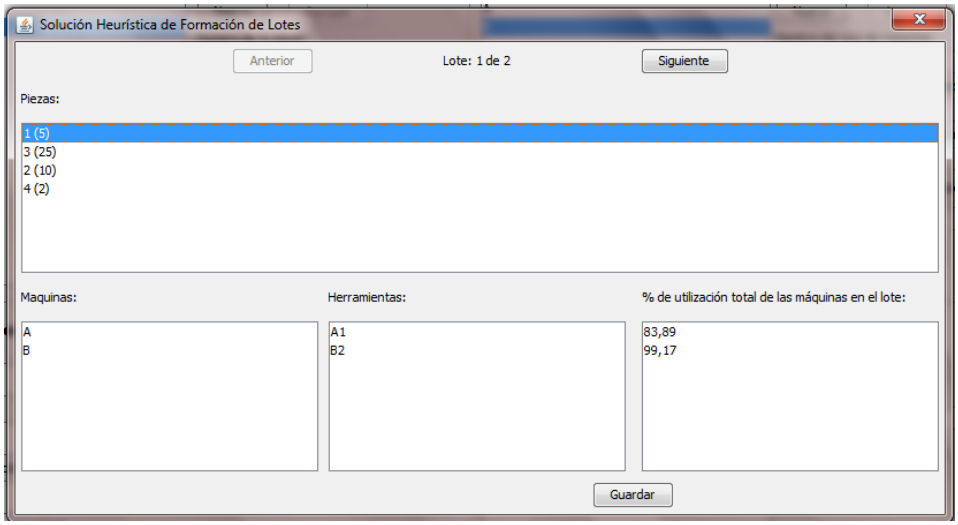
Fuente: Responsable del proyecto

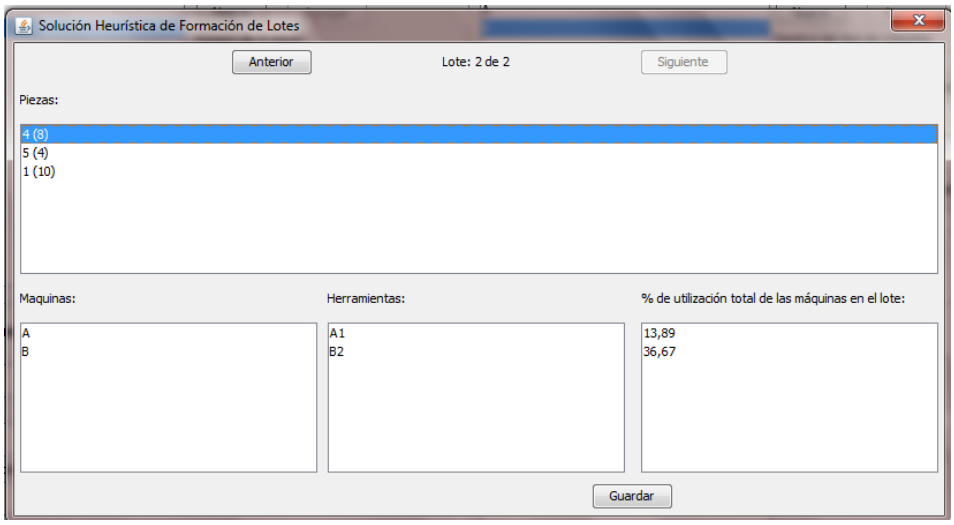
Ilustración 8. Asociaciones ingresadas ejemplo 1, heurística de formación de lotes.



Fuente: Responsable del proyecto

Ilustración 9. Lotes formados al calcular la heurística.





Fuente: Responsable del proyecto

Al resolver la heurística se obtiene que en el lote 1 se producirán las piezas a (1, 5 unidades), c (3, 25 unidades), b (2, 10 unidades), y d (4, 2 unidades). Se puede notar que para este lote se partió el pedido de la pieza d y que para el siguiente lote se producirán de primero las 2 unidades restantes del pedido. En el lote 2 además de las 8 unidades de la pieza d también se producirán las piezas e (5, 4 unidades) y a (1, 10 unidades).

Ejemplo 2

Un sistema de manufactura tiene operaciones de corte, ensamble y manufactura. Hay tres máquinas de corte una estación de ensamble automática y una operación de inspección. Cada estación de trabajo tiene 7.5 horas de trabajo disponible al día. La siguiente tabla tiene el conjunto de trabajos listos para ser producidos. Programe los lotes a producir, tenga en cuenta que una exigencia de la gerencia es que cualquier trabajo deberá ser terminado el día en que se inicia.

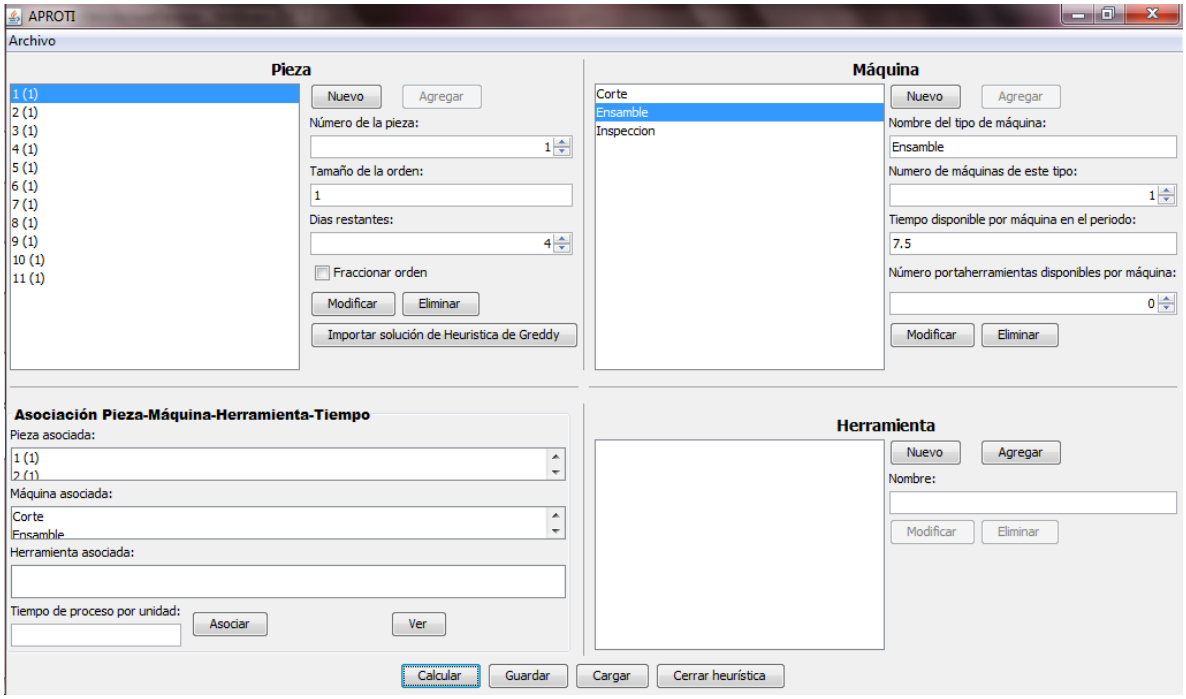
Tabla 5. Datos del ejemplo 2, heurística de formación de lotes.

		Tiempo de procesamiento (hrs estándar)		
Trabajo	Fecha de entrega	Corte	Ensamble	Inspección
a	4	10,4	2,4	0,1
b	1	2,3	0,3	0,4
c	2	4,5	1,2	1,5
d	3	4,1	0,6	3,5
e	4	18,9	5,4	4,5
f	1	5,7	0,6	0,6
g	2	23,9	4,2	3,5
h	3	12,4	4,6	7,2
i	1	10,5	2,4	0,5
j	3	2,3	1,1	0,8
k	1	3,5	1,0	0,6

Fuente: Ing. Pedro Daniel Medina

Se ingresan los datos en el software teniendo en cuenta que cada trabajo tiene un tamaño de orden 1 y además no existen para este problema herramientas necesarias ni portaherramientas disponible.

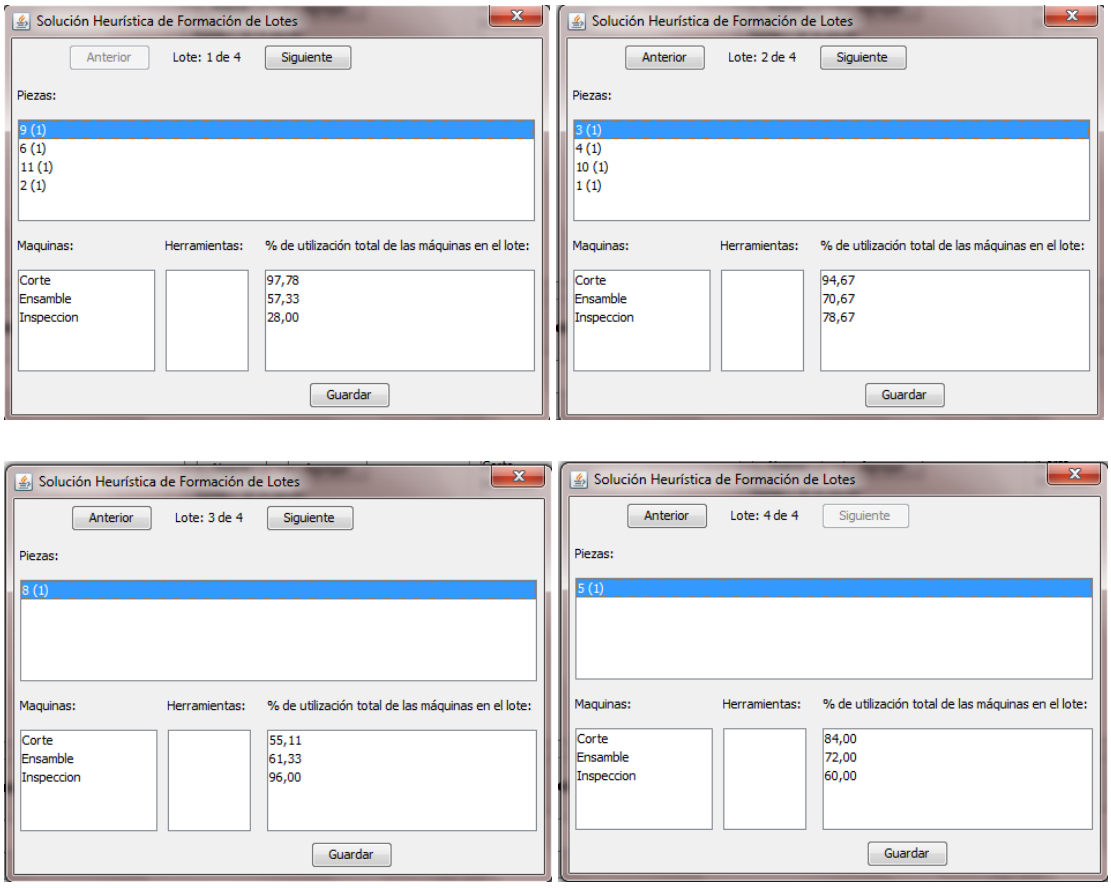
Ilustración 10. Ingreso de datos ejemplo 2, heurística de formación de lotes.



Fuente: Responsable del proyecto

Se da clic en el botón Calcular y se obtienen 4 lotes ver ilustración 10. Cuando se selecciona cualquiera de las piezas solo se pueden observar las maquinas en las que se produce ya que no existen herramientas en este ejemplo. Las piezas se enumeraron de 1 a 11 en representación de los trabajos de a – k planeados inicialmente, además se propuso que fuera cada trabajo la operación 1 de la pieza correspondiente.

Ilustración 11. Solución ejemplo 2, heurística de formación de lotes.



Fuente: Responsable del proyecto

3.3. HEURÍSTICA DE CARGA

3.3.1. Fase I

Ejemplo 1

Un grupo de máquinas va a ser cargado para producir tres tipos de partes. Los datos relevantes son mostrados en la tabla siguiente. Ciertas operaciones pueden ser llevadas a cabo por más de un tipo de máquina. Solo un tipo de máquina será

elegido para cada operación. El grupo de máquinas tiene dos máquinas de tipo A, dos de tipo B y una de tipo C. Las máquinas de tipo A, B y C pueden almacenar tres, una y cuatro herramientas, respectivamente. Se espera que cada máquina esté disponible durante 800 minutos en este periodo. Es necesario desarrollar un programa de asignación de herramientas y operaciones a máquinas.

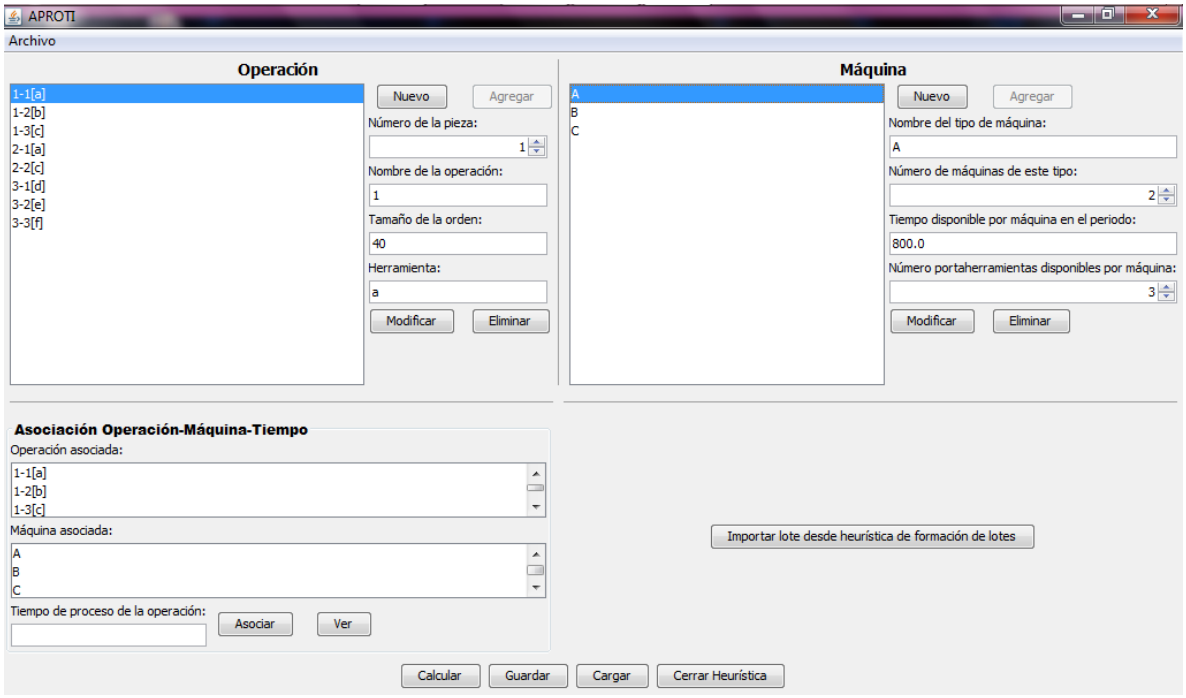
Tabla 6. Datos ejemplo 1, heurística de carga, fase 1.

Tipo de parte	Demanda	Operación	Tiempo de procesamiento por unidad			Herramienta
			A	B	C	
1	40	1	12	11	10	a
		2	13	15	0	b
		3	14	14	0	c
2	100	1	2	4	0	a
		2	2	6	6	c
3	100	1	4	0	0	d
		2	5	0	8	e
		3	0	0	4	f

Fuente: El hombre y la Máquina No. 30, Enero-junio de 2008

Se ingresan los datos en el software:

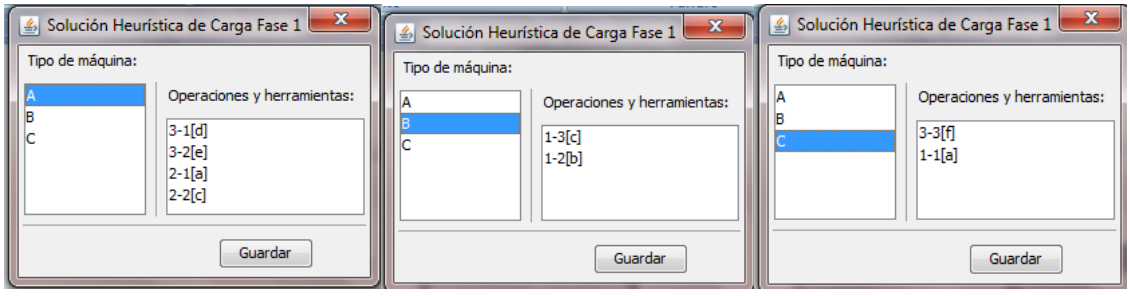
Ilustración 12. Ingreso de datos ejemplo 1, heurística de carga fase 1.



Fuente: Responsable del proyecto

Se da clic en calcular y se obtiene:

Ilustración 13. Solución ejemplo 1, heurística de carga fase 1.



Fuente: Responsable del proyecto

Ejemplo 2

Una celda de manufactura debe producir cuatro partes en el próximo turno. El costo de funcionamiento de la máquina A es de $50/hora, mientras las máquinas B y C poseen un costo de operación de $30/hora. Cualquier máquina puede ser usada para producir cualquier parte; sin embargo, la máquina A tiene más precisión, lo cual es muy importante para el tipo de piezas b. Cada máquina posee tres portaherramientas y tiene 60 horas de operación disponible. Los datos importantes se dan en la siguiente tabla:

Tabla 7. Datos ejemplo 2, heurística de carga fase 1.

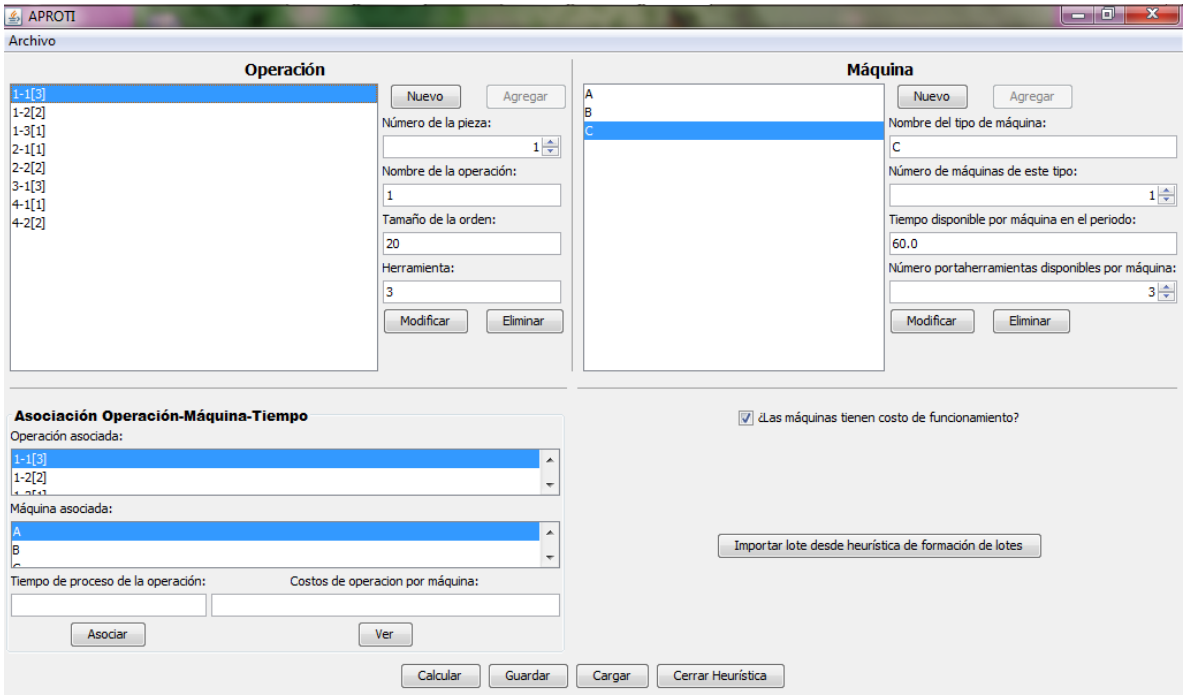
Tipo de parte	Demanda	Operación	Tiempo unitario de procesamiento (hrs)			Herramienta
			Máquina A	Máquina B	Máquina C	
a	20	1	0,2	0,3	0,3	3
		2	0,4	0,6	0,7	2
		3	0,1	0,1	0,1	1
b	40	1	0,5	0,6	0,7	1
		2	0,3	0,5	0,5	2
c	40	1	0,2	0,2	0,3	3
d	30	1	0,3	0,4	0,4	1
		2	0,1	0,2	0,2	2

Fuente: Ing. Pedro Daniel Medina

Obtenga una asignación de herramientas y operaciones a tipos de máquinas.

Se ingresan los datos del problema planteado teniendo en cuenta habilitar el costo de operación por maquina y de asociar a la pieza d solo la maquina A para darle prioridad tal y como se planteo en el enunciado.

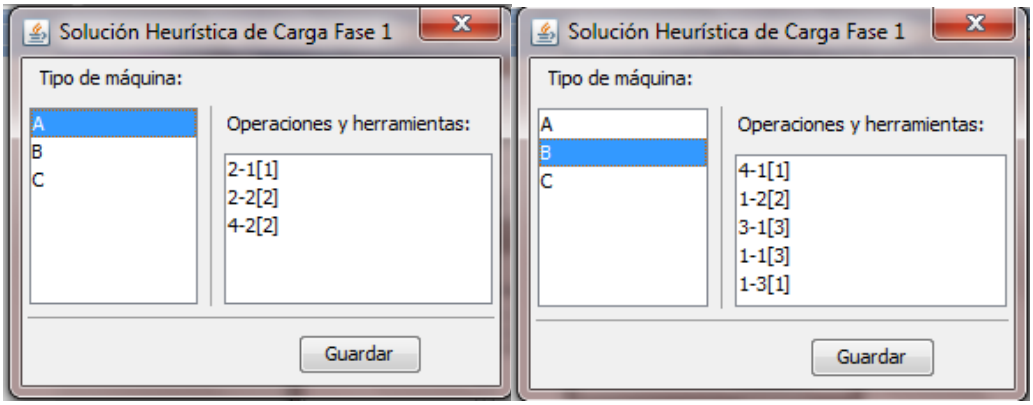
Ilustración 14. Ingreso de datos ejemplo 2, heurística de carga fase 1.



Fuente: Responsable del proyecto

Se calcula el resultado y se obtiene una agrupación para solo dos máquinas debido a que el análisis de asignación se realizo por medio del costo de funcionamiento más que por el tiempo disponible en la máquina después de asignada la operación.

Ilustración 15. Solución ejemplo 2, heurística de carga fase 1



Fuente: Responsable del proyecto

3.3.2. Fase II

Ejemplo 1

Se va a tomar el ejemplo 1 de la fase 1en donde se tienen 3 tipos de maquinas A, B y C; de las cuales hay disponibles 2, 2 y 1 respectivamente, todas tienen 800 minutos disponibles en el periodo y pueden almacenar 3, 1 y 4 herramientas respectivamente y se obtuvo el siguiente resultado:

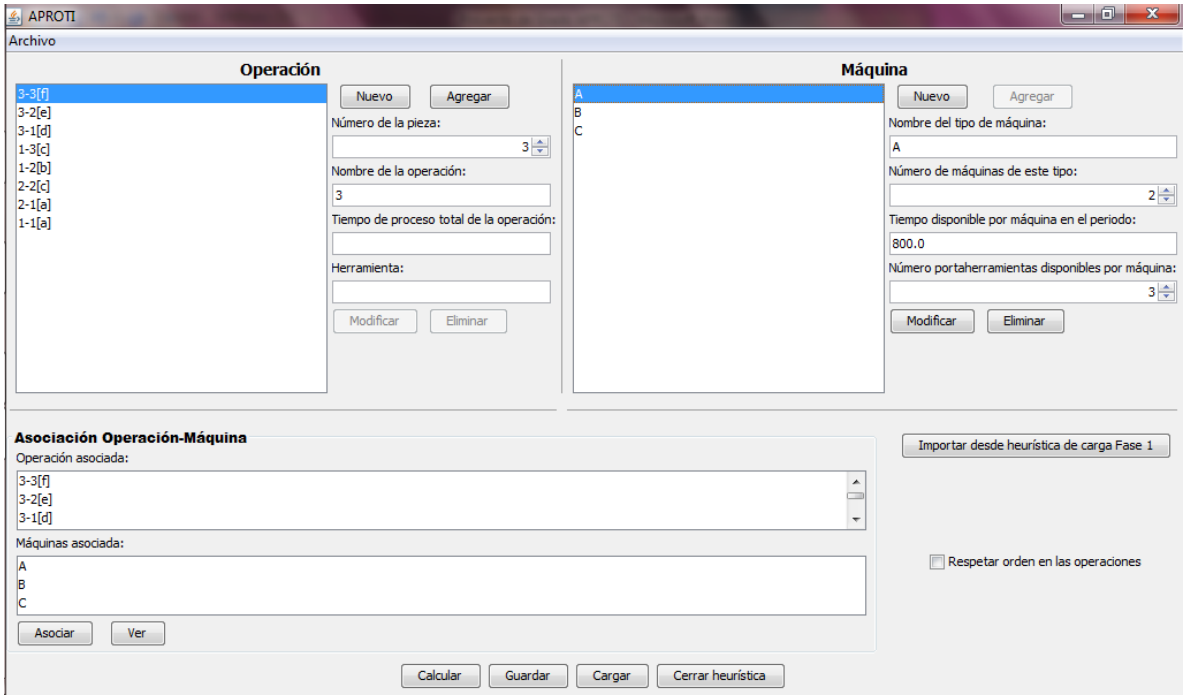
Tabla 8. Datos iniciales para ejemplo 1, la heurística de carga fase 2.

Operación	Tipo de máquina asignado	Herramienta asignada
11	C	a
12	B	b
13	B	c
21	A	a
22	A	c
31	A	d
32	A	e
33	C	f

Fuente: Scientia et Technica año XIV, No 38, Junio de 2008. Universidad Tecnológica de Pereira.

Se ingresan los datos en la heurística de carga fase 2, importándolos de la solución en la fase 1:

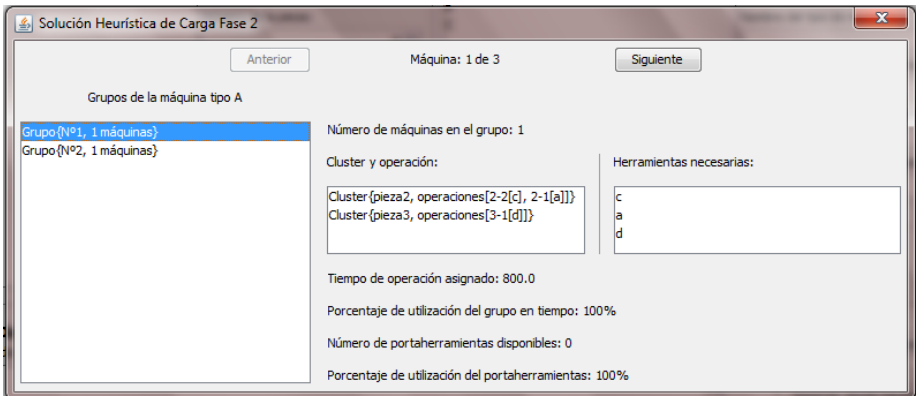
Tabla 9. Datos iniciales ejemplo 1, heurística de carga fase 2.

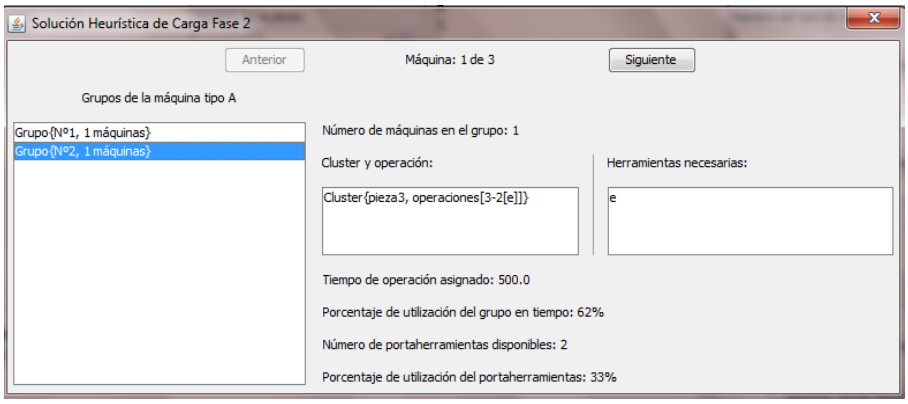


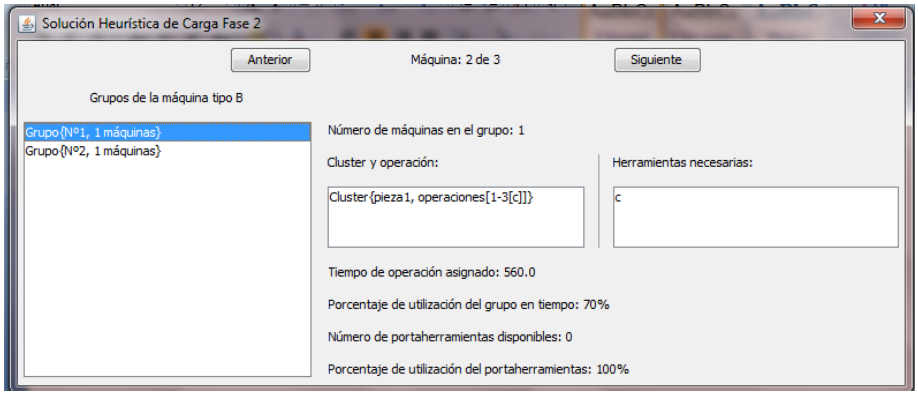
Fuente: Responsable del proyecto

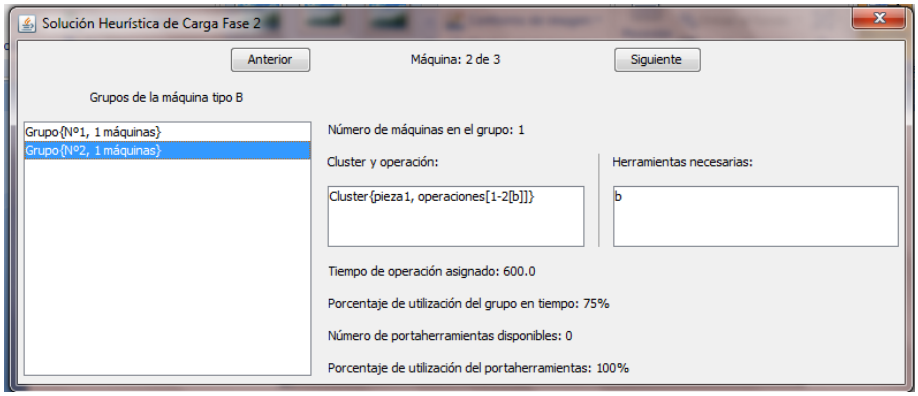
Y se calcula el resultado:

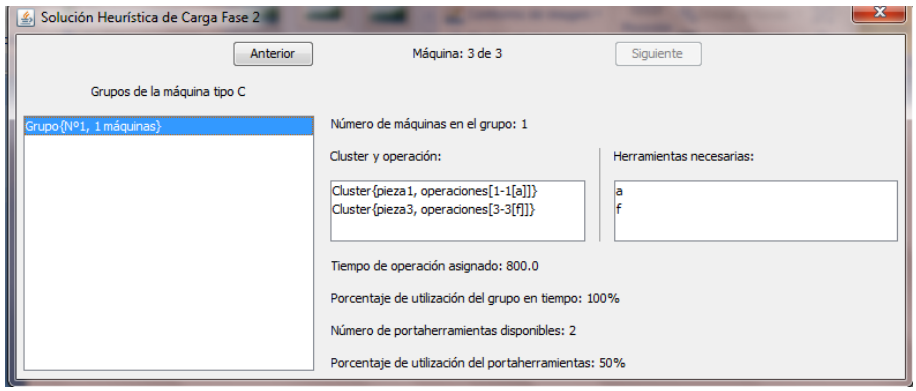
Ilustración 16. Resultado ejemplo 1 heurística de carga fase 2.











Fuente: Responsable del proyecto

En resumen el resultado se muestra en la siguiente tabla:

Tabla 10. Resultado ejemplo 1, heurística de carga fase 2.

Máquina	Operaciones Asignadas	Herramientas Asignadas	Herramientas Disponibles	% de portaherramientas usado	Tiempo de operación asignado	Tiempo de operación total disponible	% de tiempo usado
A1	3-2	a	2	33%	500	800	62%
A2	2-1, 2-2, 3-1	a,c, d	0	100%	800	800	100%
B1	1-2	b	0	100%	600	800	75%
B2	1-3	c	0	100%	560	800	70%
C	1-1, 3-3	a, f	2	50%	800	800	100%

Fuente: Responsable del proyecto

Ejemplo 2

La siguiente tabla contiene las partes que se fabricaran el día de hoy, el sistema contiene tres máquinas de tipo A y seis de tipo B. Todas las máquinas poseen tres portaherramientas y están disponibles por 7 horas. Los tiempos tabulados representan los agregados pro todo el lote y los trabajos sobre las partes pueden ser subdivididos debido a que cada trabajo consiste de múltiples pallets de material. Determine un programa de carga de operaciones y herramientas en máquinas.

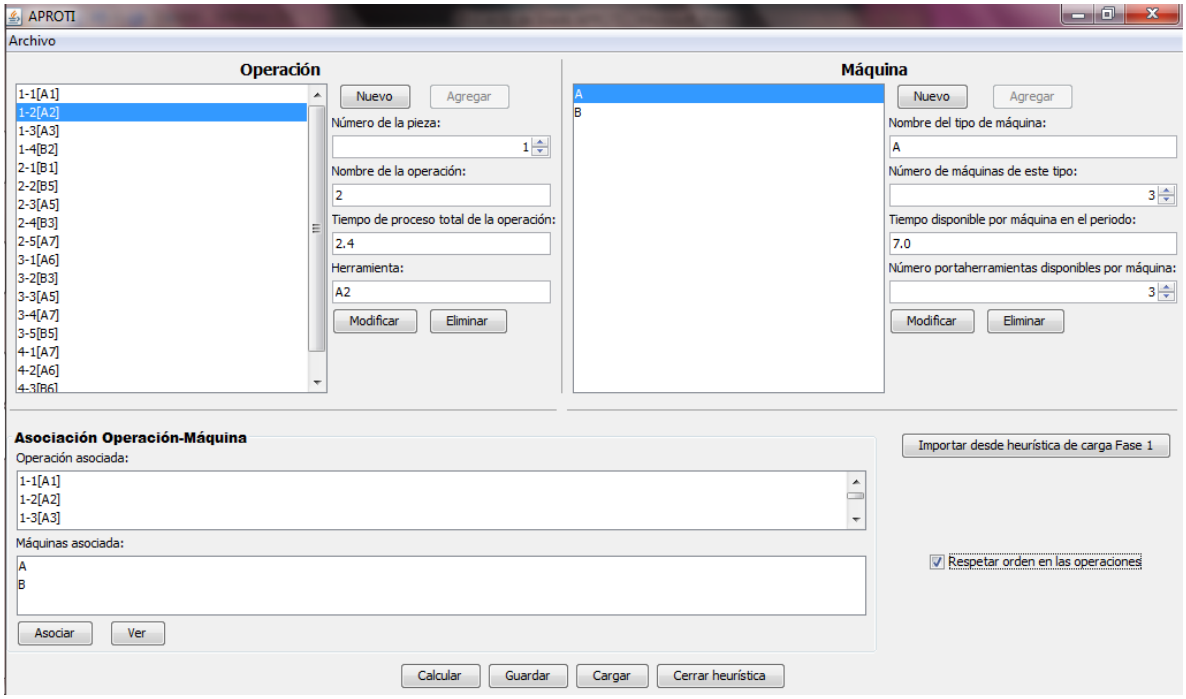
Tabla 11. Datos ejemplo 2, heurística de cargas fase 2.

Parte	Operación	Tipo de máquina	Horas Totales en máquina	Herramienta
3IE244	1	A	1,5	A1
	2	A	2,4	A2
	3	A	1,2	A3
	4	B	12,6	B2
20E139	1	B	7,1	B1
	2	B	1,3	B5
	3	A	1,6	A5
	4	B	4,5	B3
	5	A	2,5	A7
10F865	1	A	1,4	A6
	2	B	3,9	B3
	3	A	2,4	A5
	4	A	1,6	A7
	5	B	2,8	B5
24F621	1	A	2,4	A7
	2	A	1,5	A6
	3	B	4,8	B6
	4	B	3,3	B9

Fuente: Ing. Pedro Daniel Medina

Se ingresan los datos en el software:

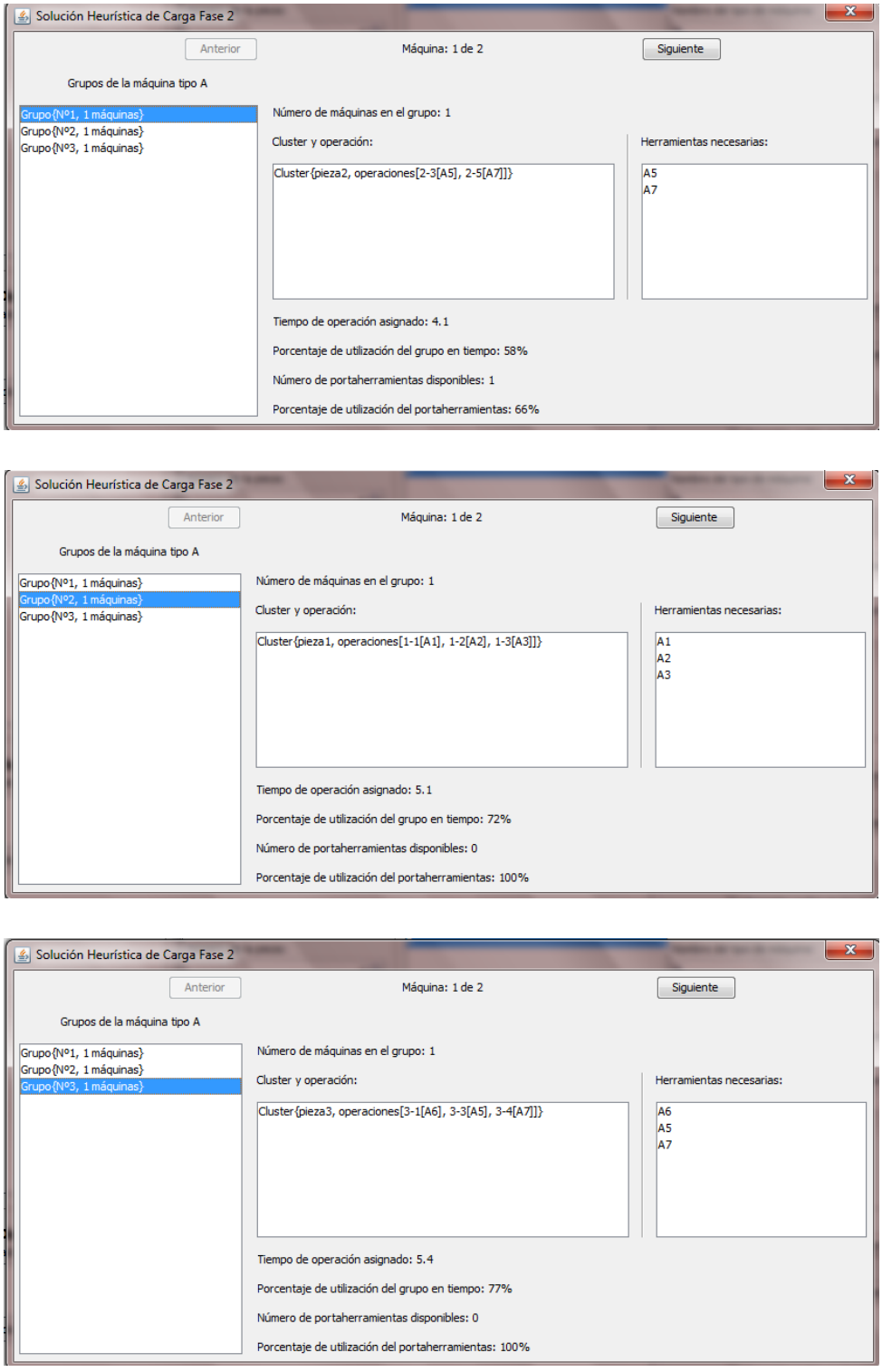
Ilustración 17. Datos ingresados ejemplo 2, heurística de carga fase 2.

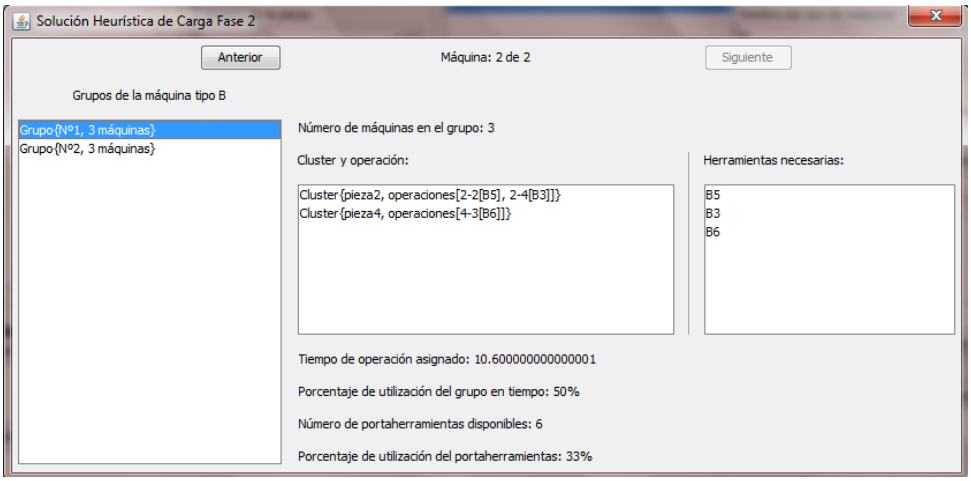


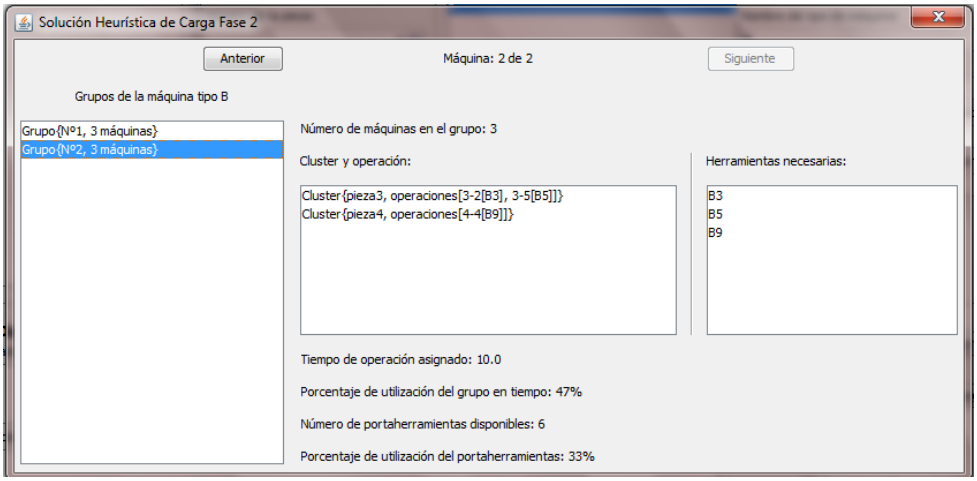
Fuente: Responsable del proyecto

Al resolver el problema se advierte que la operación 4 de la primera pieza no se puede asignar por que este trabajo excede el tiempo disponible de la máquina, al igual que la operación 1 de la segunda pieza, el clúster formado por las operaciones 1 y 2 de la cuarta pieza en consideración no se pudo asignar debido a que el tiempo libre en los grupos no alcanza para el tiempo de producción de dicho clúster. Los grupos formados de pueden observar en la ilustración siguiente:

Ilustración 18.Solución ejemplo 2, heurística de carga fase 2.

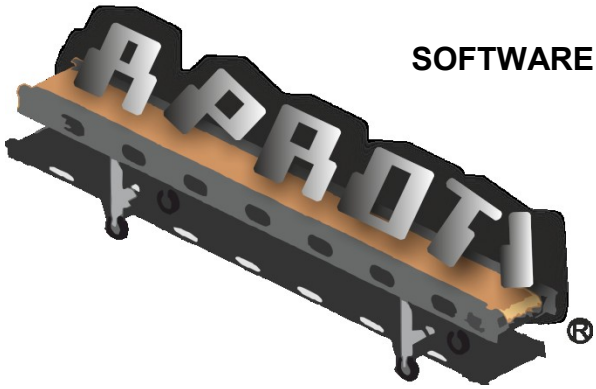






Fuente: Responsable del proyecto

4. PRESENTACIÓN DEL SOFTWARE



MANUAL DE USUARIO

INTRODUCCIÓN

Este manual está orientado a facilitarle al usuario el manejo del aplicativo APROTI, así como para introducirlo en las múltiples herramientas que para efectos prácticos incluye desde ingresar los datos iniciales hasta resolver las heurísticas necesarias para programar la producción en un Sistema de Manufactura Flexible (siglas en ingles FMS).

APROTI está diseñado para resolver tres clases de heurísticas para tres diferentes problemas que se presentan en el momento de programar la producción en un FMS, las cuales son:

- ✓ Heurística de Greddy: Resuelve el problema de selección de partes para definir que cuales conviene dejar de producir de una forma tradicional para pasarlas a producir en un FMS.
- ✓ Heurística de Formación de lotes: Se emplea para definir los lotes de producción adecuados cuando el portaherramientas disponible en las máquinas es menor a las herramientas necesarias para la fabricación buscando reducir los tiempos de alistamiento.

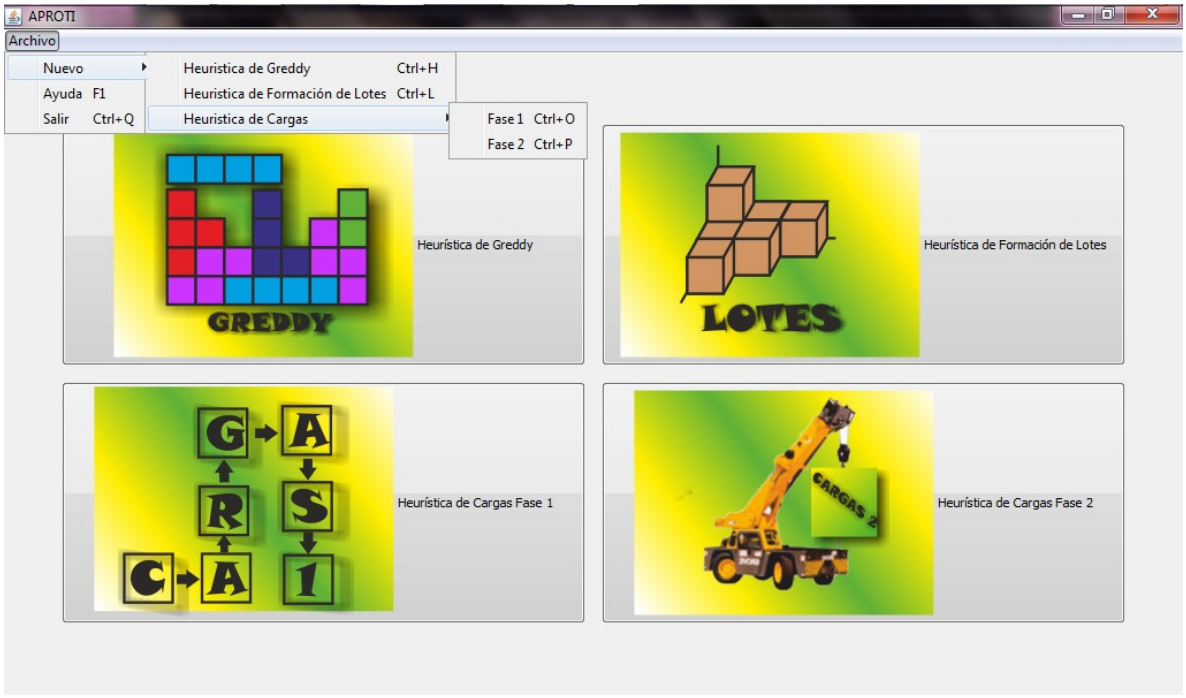
- ✓ Heurística de Carga: En esta heurística se asignan las operaciones y herramientas a máquinas buscando balancear las cargas de trabajo entre máquinas, minimizar el intercambio de material y maximizar la flexibilidad de los flujos de trabajo. Consta de dos fases para la solución completa de la heurística:
    - o Fase 1: Lleva a la asignación de operaciones a tipos de máquinas, solo se requiere resolver si hay operaciones que pueden ser asignadas a más de un tipo de máquina. Busca balancear la carga de trabajo entre tipos de máquinas.
    - o Fase 2: Permite asignar operaciones y herramientas a las máquinas, garantizando el balanceo de cargas de trabajo entre grupos de máquinas del mismo tipo.

APROTI está diseñado para el uso académico para que se estudien diferentes problemas de esta índole o de investigación, para la cual se ofrece la posibilidad de guardar los problemas para posteriormente cargarlos de nuevo y seguir con su análisis, además realizar cambios en las características de las piezas, máquinas y herramientas para análisis de sensibilidad.

Iniciando APROTI

Al momento de iniciar el aplicativo se podrá seleccionar el icono referente a la heurística requerida, o bien elegirlo desde el menú => nuevo y se selecciona la opción que se necesita:

Ilustración 19. Primera ventana al ingresar a APROTI



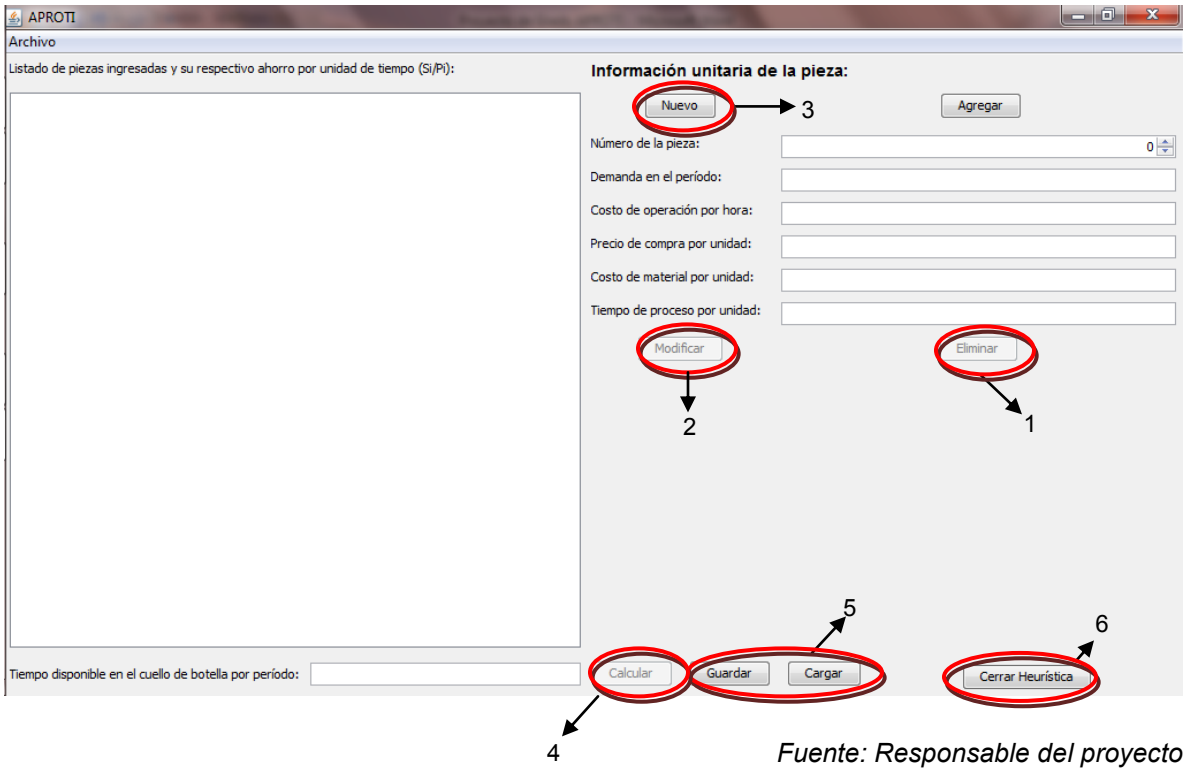
Fuente: Responsable del proyecto

En la ilustración 19 se puede observar que no solo se tienen los botones para ingresar a cualquier heurística sino que también está la opción de elegirlas desde el menú, además se tiene la opción de ayuda y el botón salir.

A continuación veremos cómo se comportan cada una de las opciones ofrecidas en APROTI:

1. *Heurística de Greddy:*

Ilustración 20. Ingreso de datos heurística de Greddy



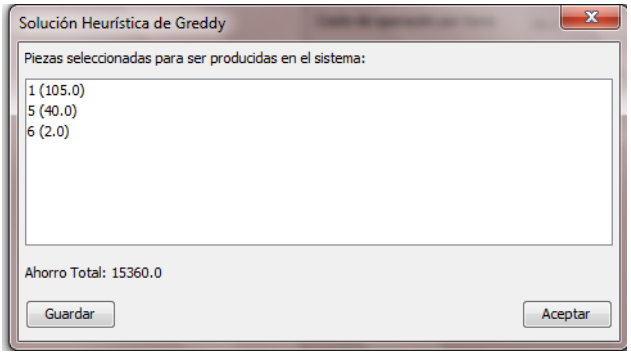
Fuente: Responsable del proyecto

Para el ingreso de los datos se debe ir ingresando la información de cada pieza una a una e irlas agregando después de terminar los datos de cada una de estas, cuando se ingrese un dato con decimal se debe ingresar con '.' Punto y no con coma ','. Si se desea eliminar alguna pieza ingresada solo se debe dar clic en la pieza que se desea suprimir y dar clic en el botón Eliminar (1), o bien si se desea modificar solo alguno de los datos ingresados de una de las piezas después de dar clic en la pieza a editar, luego se cambia el dato y se oprime el botón Modificar (2). Mientras este seleccionada una pieza de la lista el botón Agregar estará deshabilitado y para volverlo a habilitar tan solo es necesario oprimir el botón Nuevo (3), así se borraran de los campos los datos de la pieza que estaba

seleccionada y se podrá digitar la nueva pieza que se desea adicionar. En el caso de que se desee guardar el problema digitado y luego abrirlo para completar su ingreso se puede usar los botones Guardar y Cargar (5) respectivamente para dicho fin, para cerrar la heurística se tiene el botón Cerrar Heurística (6), el cual borra todos los datos que se han ingresado y se cerrara dejando en la pantalla la pagina principal. Es importante notar que se debe guardar antes de cerrar o se perderá toda la información ingresada.

Cuando se termine de ingresar los datos de las piezas, se debe digitar el tiempo disponible en el cuello de botella para habilitar el botón Calcular (4). Al dar clic en este se obtendrá en una ventana los datos de las piezas que se deben producir en el sistema además del ahorro total obtenido si estas piezas se producen. Si no se selecciona ninguna pieza saldrá un mensaje anunciando esto.

Ilustración 21. Ver resultado heurística de Greddy



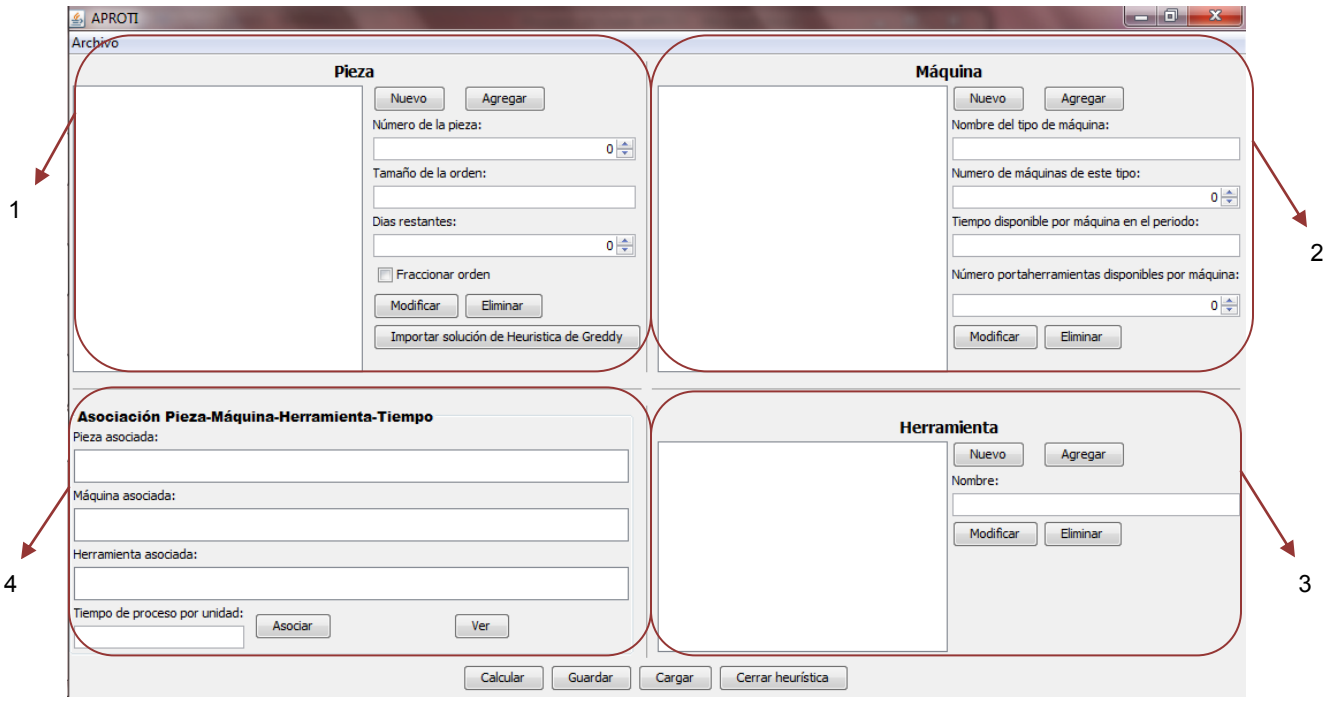
Fuente: Responsable del proyecto

El botón Guardar en la ventada de la solución de la heurística de Greddy permite almacenar las piezas seleccionada en un archivo que podrá ser abierto desde heurística de lotes para inicializar los datos de las piezas, allí se cargaran el

numero de la pieza, la demanda del periodo, la fecha de entrega en cero (0) y la opción de fraccionar orden habilitada. Allá se podrá modificar o eliminar dichas piezas si el usuario lo desea.

2. Heurística de Formación de lotes:

Ilustración 22. Ingreso de datos heurística de formación de lotes



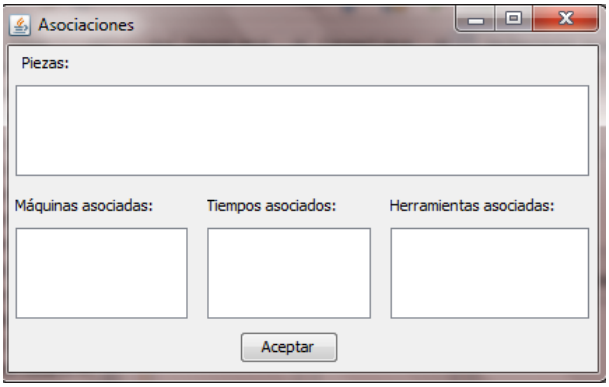
Fuente: Responsable del proyecto

Para ingresar los datos en la heurística de formación de lotes se debe tener en cuenta que la pantalla tiene 4 secciones que permitirán plantear el problema completamente: en la primera parte ira toda la información referente a las piezas, Es importante conocer que se requiere que una pieza solo tenga asociada una máquina para poder cargar óptimamente en la heurística de carga el resultado, de

lo contrario, la importación en dicha heurística será limitada. Las piezas pueden ser modificadas o eliminadas después de ser ingresadas, además se puede importar un resultado anteriormente guardado en la solución de la heurística de Greddy. En la sección 2 se ingresa la información de las máquinas, dicha información también puede ser cambiada o eliminada según se requiera, en la parte 3 se ingresan los datos de las herramientas que intervienen en el problema, es posible modificarlas o eliminarlas. En la cuarta sección se hacen las asociaciones, se selecciona la pieza, la máquinas, la herramienta y se digita el tiempo de proceso requerido para producir esa pieza en esa máquinas con esa herramienta, y se da clic en el botón Asociar, si se desea ver las asociaciones realizadas se oprime el botón Ver y en la ventana emergente se encontrara esta información.

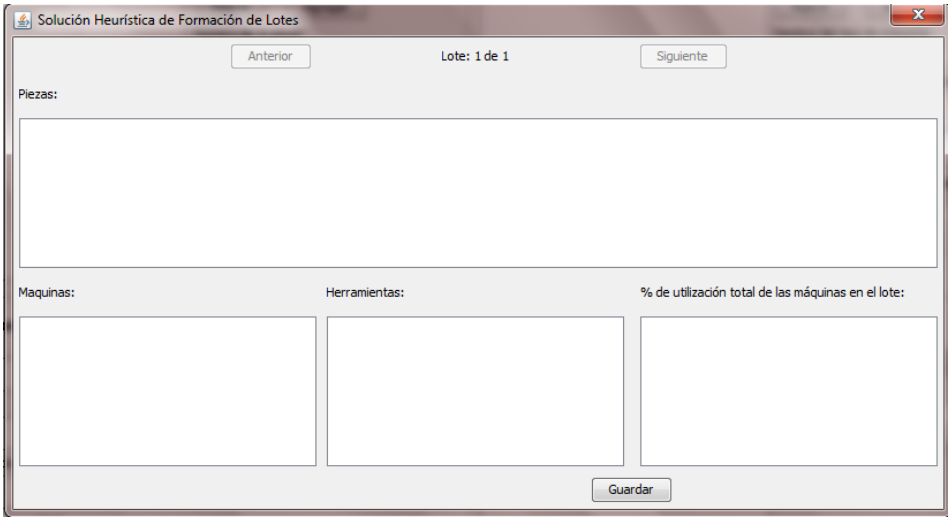
Por último encontramos los botones calcular, guardar, cargar y cerrar heurística, con el primero se resuelve la heurística, mostrando los datos de los lotes formados, los botones guardar y cargar permiten trabajar en un problema en diferentes momentos, según la disponibilidad del usuario, un problema se puede guardar y luego volverlo a retomar. Y el botón Cerrar heurística, borra toda la información que se haya ingresado (si está guardada se recarga cuando quiera) y cierra la heurística regresando a la página principal.

Ilustración 23. Ver asociación heurística de formación de lotes



Fuente: Responsable del proyecto

Ilustración 24. Ver resultado heurística de formación de lotes



Fuente: Responsable del proyecto

Tanto para la ventana Asociación como para la de Solución, se maneja la misma dinámica para visualizar la información se encuentra un listado de piezas que al seleccionar una se puede ver la información de ésta en la parte inferior. En la última, adicionalmente se tienen en la parte superior dos botones (anterior y siguiente) para visualizar lote por lote las piezas que pertenecen a cada uno de

estos. Y por ultimo cabe mencionar que el resultado de la heurística de formación de lotes se puede guardar para posteriormente importarlos desde la heurística de carga.

3. Heurística de Carga:

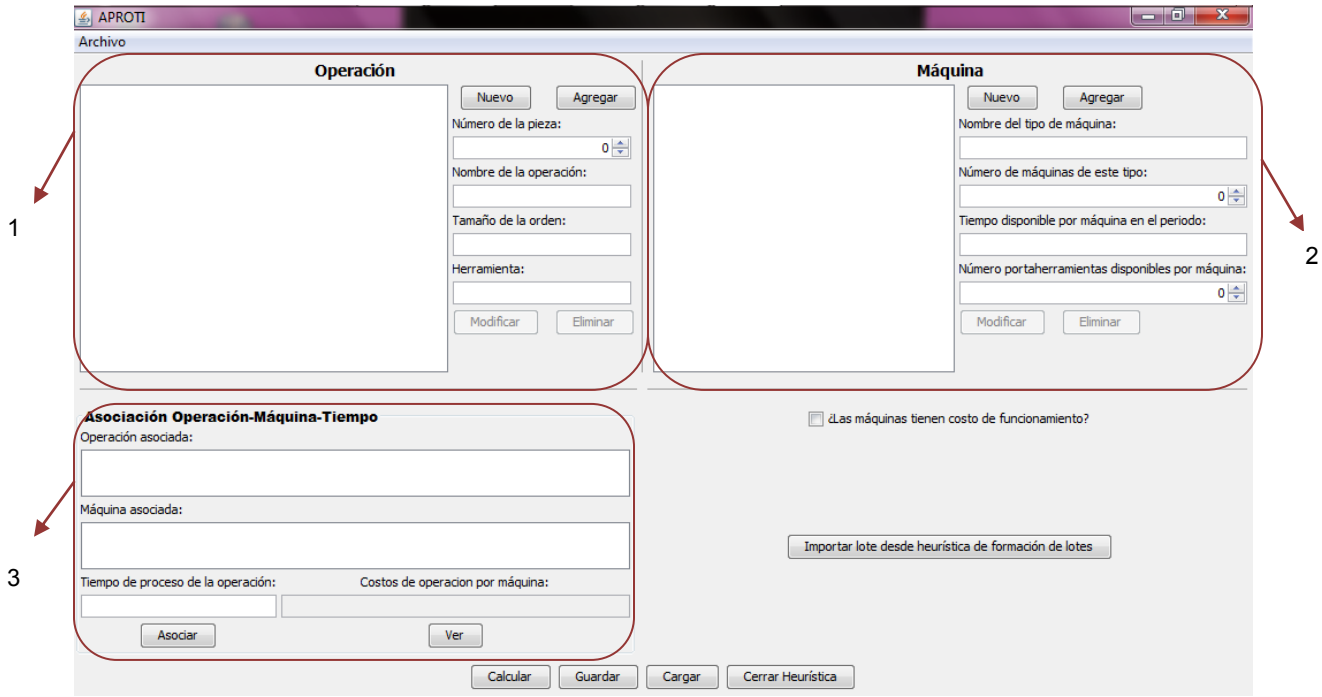
Esta heurística contiene 2 fases las cuales según las necesidades del usuario se pueden resolver ambas o solo una de ellas.

a. Fase 1:

El ingreso de los datos de esta fase se asemeja al ingreso de datos de la heurística de formación de lotes, en la sección 1 se ingresa los datos de las operaciones a producir, se introduce el numero de la pieza y el nombre de la operación para identificarla, además se ingresa el tamaño de la orden de la operación, por lo regular las operaciones de una sola pieza tienen el mismo tamaño, y por último se especifica con que herramienta se va a producir dicha operación. En la sección 2 encontramos los campos para caracterizar todas las máquinas en las que se podría producir las operaciones. En la tercera sección encontramos el espacio para ingresar las asociaciones: se da clic en la operación que se quiere asociar, en la máquina en la que se podría producir y se digita el tiempo necesario para el procesamiento de dicha operación en la máquina seleccionada, se debe ingresar este tiempo en las mismas unidades en las que se ingreso el tiempo disponible por máquina de cada tipo. Por último es opcional el ingreso del costo de operación para producir la pieza en la máquina, se habilita la opción ¿Las máquinas tienen costo de funcionamiento?, para poder digitar el

costo de funcionamiento de la máquina y que se asocie a la operación. Y para ver las asociaciones ya creadas se puede dar clic en el botón Ver.

Ilustración 25. Ingreso de datos heurística de carga fase 1



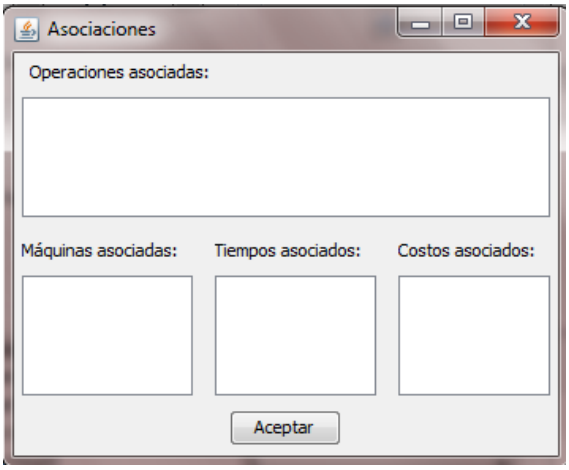
Fuente: Responsable del proyecto

Luego encontramos el botón Importar lote desde heurística de formación de lotes el cual permite cargar un lote de alguna solución de la heurística de lotes anteriormente guardada, se debe recordar que dicha importación tiene algunas limitaciones, debido a que las máquinas asociadas en esta fase son posibles máquinas utilizadas en la producción de la operación, y por otro lado las máquinas asociadas en la heurística de formación de lotes son todas necesarias para la producción de dichas operaciones, por tanto si en la asociación de las operaciones pertenecientes al lote que se quiere importar se asociaron más de un tipo de máquinas las asociaciones no se importaran solo se importaran las máquinas y las

operaciones del lote en consideración, además solo se importara la primera herramienta asociada en las operaciones de este lote.

Por último tenemos los botones Calcular, Guardar, Cargar y Cerrar heurística que ya se han explicado su funcionamiento en las anteriores heurísticas.

Ilustración 26. Ver asociación heurística de carga fase 1



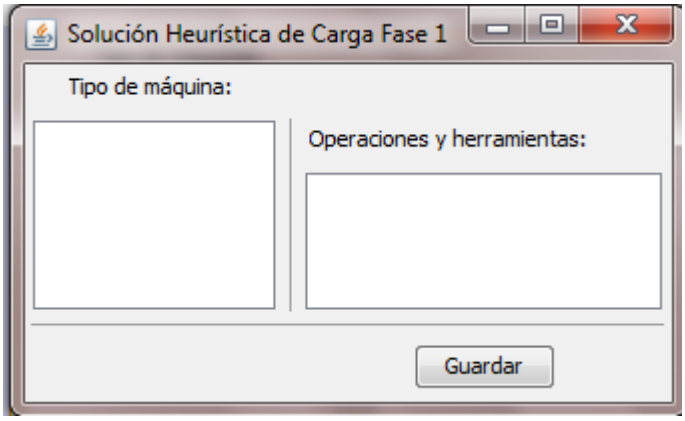
Fuente: Responsable del proyecto

Para ver las asociaciones se da clic en las operaciones y en la parte inferior aparecen las máquinas y sus correspondientes tiempos. Si se ingresaron los costos también se verán pero en esta ventana se verán los costos totales de funcionamiento es decir la multiplicación del tamaño de la orden de la operación por el tiempo asociado por el costo asociado.

Cuando se calcula el resultado aparecen los tipos de máquinas y al seleccionar uno de ellos se podrá apreciar las operaciones que se han asignado a dicho tipo

de máquina, además esta solución se puede guardar para luego ser importado desde la segunda fase de la heurística de carga.

Ilustración 27. Ver resultado de la heurística de carga fase 1

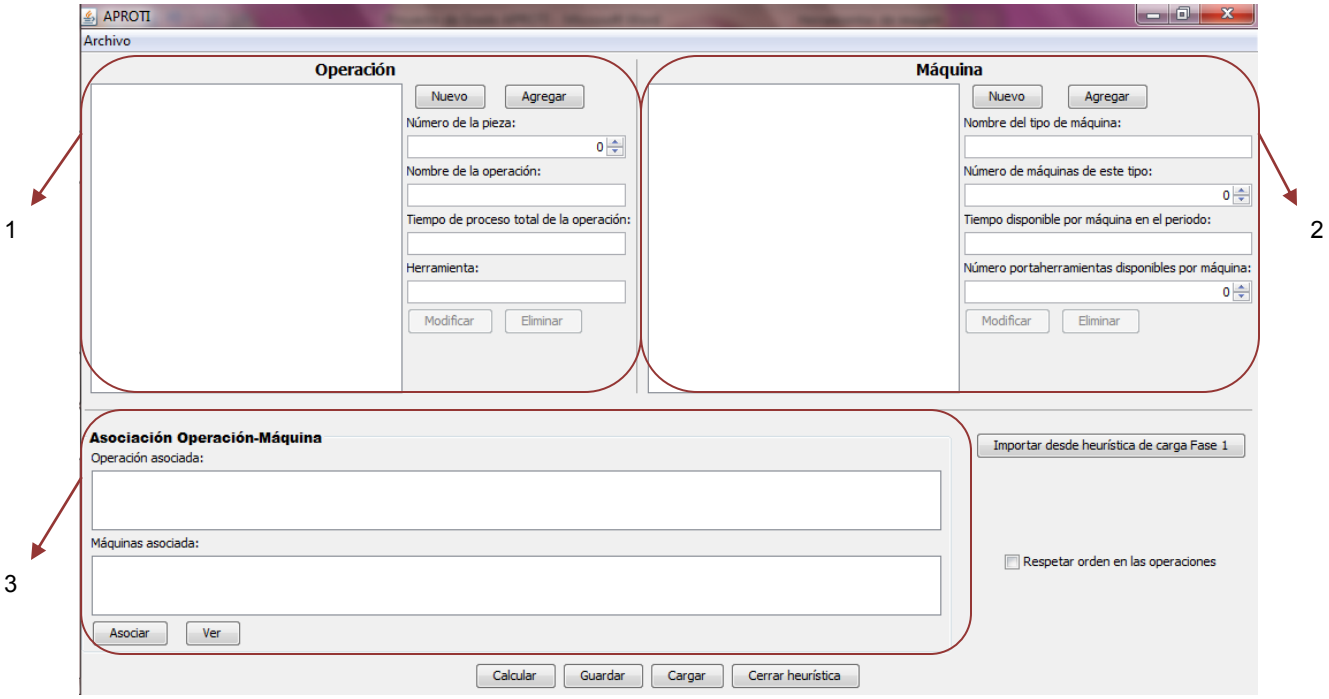


Fuente: Responsable del proyecto

b. Fase 2:

Para esta fase se pueden importar los datos iniciales desde la fase, utilizando el botón que está en la parte derecha de la ventana de esta fase; o bien digitar la información y hacer las asociaciones correspondientes, recuerde que solo se puede asociar una máquina por operación.

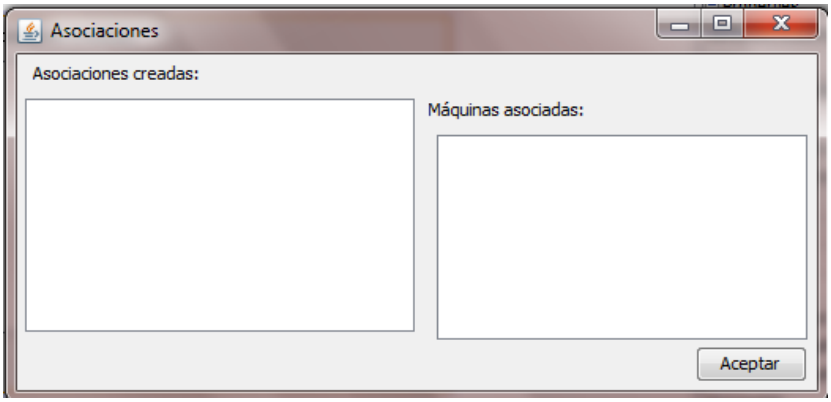
Ilustración 28. Ingreso de datos heurística de carga fase 2



Fuente: Responsable del proyecto

El ingreso de los datos manualmente se debe hacer primero integrando la información de las operaciones y de las máquinas para luego asociar a cada operación la máquina correspondiente. En la sección 1 donde se ingresa las operaciones se pueden modificar y eliminar las operaciones que se adhieran al problema, además es importante anotar que el tiempo que se ingrese sea el tiempo total necesario para producir la operación, en la sección 2 se ingresa la información de las máquinas tal y como se ha hecho hasta este momento, agregando, modificando o eliminando máquinas según sea necesario, después de tener esta información se asociara para cada operación el tipo de máquina en el que se va a producir, se pueden ver las asociaciones hechas al dar clic en el botón Ver

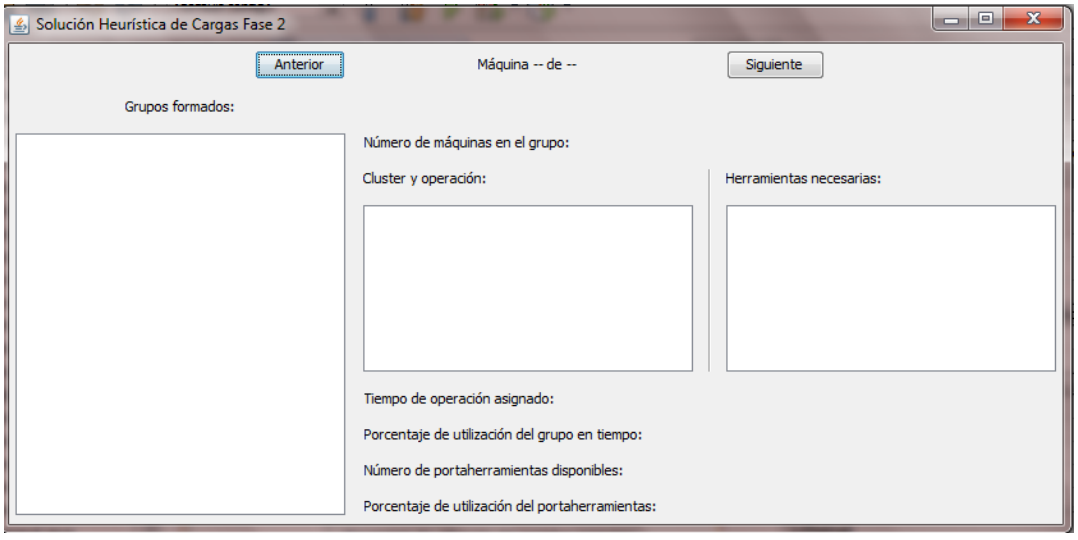
Ilustración 29. Ver asociación heurística de carga fase 2



Fuente: Responsable del proyecto

Antes de calcular, si el problema que se está solucionando requiere que las operaciones se realicen en orden se puede habilitar la opción en la parte derecha inferior para que se respete dicho orden, de lo contrario no se habilita la opción y se da clic en calcular:

Ilustración 30. Ver resultado heurística de carga fase 2



Fuente: Responsable del proyecto

En la ventana donde aparecen los resultados me especifica por tipo de máquina los grupos que se crearon, al dar clic sobre estos grupos en la parte derecha encontraremos un resumen de la información del lote, como cuantas máquinas de ese tipo se asignaron al grupo, los clúster y las operaciones que se realizaran por este grupo, y la utilización de herramientas, tiempo y portaherramientas. Para ver la información de todas las máquinas se utilizan los botones Anterior y Siguiente, para desplazarse de la información de un tipo de máquina a otro.

Preguntas frecuentes:

- ✓ ¿Cómo se ve la información de una pieza, operación, máquina o herramienta agregada?

  Se da clic en el registro que se desea verificar y en los campos correspondientes a la información seleccionada se vean los datos que se le han asignado al registro.

- ✓ ¿Cómo se modifica una pieza, operación, máquina o herramienta después de agregada?

  Se da clic en el registro que se desea modificar, en los campos de información se modifica las variables deseadas, y se le da clic en el botón Modificar, e inmediatamente se cambian el campo que se modifico. Cuando se modifica la información se puede causar inconsistencias con las asociaciones. En especial en las de las herramientas.

- ✓ ¿Cómo eliminar una pieza, operación, máquina o herramienta?

  Se debe seleccionar el registro que se quiere suprimir, se puede verificar que si es el elegido revisando la información de este en los campos donde se ingresaron anteriormente, y por último se da clic en eliminar. Tenga en cuenta que si había una asociación del dato removido se podría encontrar inconsistencias posteriormente.

- ✓ ¿Cómo se ve una asociación?

  Al dar clic en el botón Ver en la parte de asociaciones en la ventana emergente se pueden ver las piezas u operaciones ingresadas, y al dar clic sobre alguna de ellas se podrá observar en los campos de la parte inferior los datos asociados.

- ✓ ¿Cómo eliminar una asociación?

  Se debe identificar la pieza u operación a la cual se le asocio la máquinas, la herramienta o el tiempo erróneamente, se elimina dicha pieza y se vuelve a ingresar los datos de esta y las asociaciones correctas.

CONCLUSIONES Y RECOMENDACIONES

- Un software para ayudar al estudio de Sistemas de Manufactura flexible es importante para mejorar el modelaje y la programación de la producción. Además es un gran aporte para que una universidad como la Universidad Tecnológica de Pereira ofrezca a sus estudiantes diferentes maneras de incentivar su interés por la investigación y el uso de herramientas computacionales.
- El software fue desarrollado para el uso académico y como tal está basado en ejemplos de este tipo pero también fue pensado para soportar problemas más pesados, de mayores requisitos en número de piezas, máquinas y herramientas, de esta manera al resolver las heurísticas se obtendrán respuestas buenas, aunque tal vez no sea el óptimo.
- Leer el manual del usuario antes de utilizar el software. Se dispondrá del manual en el menú ayuda del software.
- Se deben ingresar primero los datos de las piezas, operaciones, máquinas o herramientas y estar seguros de su veracidad, y luego agregar las asociaciones, ya que si se elimina o modifica la información se puede causar inconsistencias con las asociaciones. En especial en las de las herramientas.
- Pasar de heurística de formación es de lotes para heurística de carga está limitado ya que las máquinas que se ingresan en lotes son necesarias para producir las operaciones mientras que las máquinas que se asocian en cargas son opcionales y solo se escoge una para producirse la operación.

BIBLIOGRAFÍA

- Página web consultada el 21 de octubre del año 2012 por medio de un enlace de Google: http://es.wikipedia.org/wiki/Portaherramientas

- Página web consultada el 06 de octubre del año 2011 por medio de un enlace de Google: http://usuarios.multimania.es/ramirovega/archivos/UNIDAD%201.pdf

- Página web consultada el 08 de octubre del año 2011 por medio de un enlace de Google: http://cadcamcae.wordpress.com/2007/06/14/el-control-numerico-por-computadora-el-cnc/

- Página web consultada el 08 de octubre del año 2011 por medio de un enlace de Google:www.intelmax.com/ensanluis/images/capitulo16.ppt

- Página web consultada el 08 de octubre del año 2011 por medio de un enlace de Google: http://ingenieria.udea.edu.co/CURSOS/DOCUM/manofacturaflexible(opcional).doc

- Página web consultada el 08 de octubre del año 2011 por medio de un enlace de Google: http://www.joseacontreras.net/manuf/page.htm

- Askin, Ronald G. y Standridge, Charles R. (1993). *Modeling and Analysis of Manufacturing System*. John Wiley & Sons, Inc.

- Página web consultada el 08 de octubre del año 2011 por medio de un enlace de Google: http://www.enotes.com/management-encyclopedia/flexible-manufacturing

- Tseng, Mei-Chiun. "Strategic Choice of Flexible Manufacturing Technologies." International Journal of Production Economics 91, no. 3 (2004)

- Chandra, Charu, Mark Everson, and Janis Grabis. "Evaluation of Enterprise-Level Benefits of Manufacturing Flexibility." Omega 33, no. 1 (2005)

- Pagina web consultada el 22 de octubre del año 2012 por medio de un enlace de Google: Definición de Java - Qué es, Significado y Concepto http://definicion.de/java/#ixzz2A2cT4Csv

- Joyanes Aguilar, Luis y Zahonero Martínes, Ignacio. Programación en JAVA, Algoritmos, programación orientada a objetos e interfaz grafica. McGRAW-HILL, 2011, primera edición.

- Drozdek, Adam. Estructura de datos y algoritmos con Java. Thomson editores, S.A. 2007, segunda edición.

- Videos tutoriales de JAVA para principiantes por illasaron en youtube: http://www.youtube.com/watch?v=4C1VZfvR0SM&feature=BFa&list=PL4D956E5314B9C253

Anexo 1. Resumen Código Fuente

Acciones que se hacen cuando se da clic en el botón agregar en la heurística de Greddy, lo cual se resume en que toma los datos digitados y se crea un objeto pieza nuevo con dichos datos, se agrega a la lista de piezas y esta se ordena según los criterios ya establecidos y se imprime dicha lista en la lista de las piezas. Sobre esta base se programan todos los botones de agregar en todas las demás heurísticas.

```
private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    Pieza p = new Pieza(((Integer) spnPieza.getValue()).intValue(), txtDemanda.getText(),
      txtCostoOperacion.getText(), txtPrecio.getText(), txtCostoMaterial.getText(), txtTiempo.getText());
    piezas.add(p);
    Collections.sort(piezas, new ComparadorPiezas());
    final ArrayList<Pieza> pzs = piezas;
    AbstractListModel mdl=getListModel(pzs);
    lstPiezas.setModel(mdl);
    limpiar();
    btnAgregar.setEnabled(true);
    btnModificar.setEnabled(false);
    btnEliminar.setEnabled(false);    }
```

Acciones que se hacen cuando se selecciona un elemento de la lista de piezas, el valor seleccionado se convierte en un objeto pieza y se imprimen los datos en los lugares correspondientes para que el usuario los vea. Y se desactiva el botón agregar para evitar que se ingresen dos objetos idénticos, además se habilitan los botones modificar y eliminar para cambiar los datos de la pieza si se desea. Sobre esta base se programan todas las listas en el software, no solo de las piezas sino de las máquinas, herramientas y resultados.

```
private void lstPiezasValueChanged(javax.swing.event.ListSelectionEvent evt) {
    Pieza p = (Pieza) lstPiezas.getSelectedValue();
    if (p != null) {
        spnPieza.setValue(p.getPieza());
        txtCostoMaterial.setText(p.getCostoMaterial().toString());
        txtCostoOperacion.setText(p.getCostoOperacion().toString());
        txtPrecio.setText(p.getPrecio().toString());
        txtDemanda.setText(p.getDemanda().toString());
        txtTiempo.setText(p.getTiempo().toString());
        btnAgregar.setEnabled(false);
        btnModificar.setEnabled(true);
        btnEliminar.setEnabled(true);
    }
}
```

Acciones que se hacen cuando se da clic en el botón Nuevo. Habilita el botón agregar y deshabilita los botones modificar y eliminar además se limpia todos los campos, en base a este se programan todos los botones nuevo en las demás heurísticas.

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    btnAgregar.setEnabled(true);
    btnModificar.setEnabled(false);
    btnEliminar.setEnabled(false);
    limpiar();
}
```

Acciones que se hacen cuando se da clic en el botón guardar. Se guarda en un vector los datos a guardar y se serializa en un archivo .java. Sobre este botón se basa la programación de los botones guardar de todo el software.

```
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
  try {
      int r = fc.showSaveDialog(this);
      if (r == JFileChooser.APPROVE_OPTION) {
         String a = txtCuello.getText();
          ObjectOutput out = new ObjectOutputStream(new FileOutputStream(fc.getSelectedFile().
            getAbsolutePath () ));
         Object[] arreglos = new Object[]{piezas, a};
         out.writeObject(arreglos);
         out.close();
         mostrarMensaje("Datos Guardados");
      }
   } catch (Exception w) {
      mostrarError(w.getMessage());
   }
}
```

Acciones que se hacen cuando se da clic en el botón cargar. Se carga un vector los datos que se había guardado y se extraen los datos de un archivo .java y se ubican los datos en las listas correspondientes. Sobre este botón se basa la programación de los botones cargar de todo el software.

```
private void btnCargarActionPerformed(java.awt.event.ActionEvent evt) {
   try {
      int r = fc.showOpenDialog(this);
      if (r == JFileChooser.APPROVE_OPTION) {
         ObjectInput in = new ObjectInputStream(new FileInputStream(fc.getSelectedFile().getAbsolutePath()
            ));
         Object[] arreglos = (Object[]) in.readObject();
         piezas = (ArrayList)arreglos[0];
         lstPiezas.setModel(getListModel(piezas));
         txtCuello.setText(String.valueOf(arreglos[1]));
         mostrarMensaje("Datos Cargados");
         if (txtCuello.getText().length() > 0 && Double.valueOf(txtCuello.getText()) > 0) {
            btnCalcular.setEnabled(true);
         } else {
            btnCalcular.setEnabled(false);
         }
      }
   } catch (Exception w) {
      mostrarError(w.getMessage());
   }
}
```

Método para ordenar descendentemente las piezas según el ahorro calculado según los pasos descritos en este trabajo para la selección de partes.

```
public static class ComparadorPiezas implements Comparator<Pieza> {

   @Override
   public int compare(Pieza o1, Pieza o2) {
      return o1.getAhorro() < o2.getAhorro() ? 1 : o1.getAhorro() == o2.getAhorro() ? 0 : -1;
   }
}
```

Indica que acciones se hacen cuando se da clic en el botón modificar. El cual agrega una nueva pieza y borra la que ya existía. Sobre esta base se programan todos los botones modificar del software.

```
private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
   try {
      piezas.remove(lstPiezas.getSelectedIndex());
```

```
            lstPiezas.remove(lstPiezas.getSelectedIndex());
            Pieza p = new Pieza(((Integer) spnPieza.getValue()).intValue(),
                    txtDemanda.getText(),
                    txtCostoOperacion.getText(),
                    txtPrecio.getText(),
                    txtCostoMaterial.getText(),
                    txtTiempo.getText());
            piezas.add(p);
            Collections.sort(piezas, new ComparadorPiezas());
            final ArrayList<Pieza> pzs = piezas;
            AbstractListModel mdl=getListModel(pzs);
            lstPiezas.setModel(mdl);
            limpiar();
            btnAgregar.setEnabled(true);
            btnEliminar.setEnabled(false);
            btnModificar.setEnabled(false);
        } catch (Exception w) {
            mostrarError(w.getMessage());
        }
    }
```

Indica que acciones se hacen cuando se da clic en el botón eliminar. Sobre este botón se basan todos los botones con este propósito en el software.

```
    private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            piezas.remove(lstPiezas.getSelectedIndex());
            lstPiezas.remove(lstPiezas.getSelectedIndex());
            final ArrayList<Pieza> pzs = piezas;
            AbstractListModel mdl=getListModel(pzs);
            lstPiezas.setModel(mdl);
            limpiar();
            btnAgregar.setEnabled(true);
            btnEliminar.setEnabled(false);
            btnModificar.setEnabled(false);
        } catch (Exception w) {
            mostrarError(w.getMessage());
        }
    }
```

Método para cerrar el algoritmo y volver a la página principal, sobre el cual se basan todos los botones cerrar heurística de todo el software. El cual pregunta si de verdad desea cerrar la heurística y según la respuesta cierra la heurística y borra todos los datos ingresados o no.

```
    private void btnCerrarActionPerformed(java.awt.event.ActionEvent evt) {
        Integer a = JOptionPane.showOptionDialog(null, "¿Desea cerrar la heuristica? Recuerde que se borraran
          todos los datos ingresados si no los ha guardado.", "Cerrar", JOptionPane.YES_NO_OPTION,
          JOptionPane.QUESTION_MESSAGE, null, null, 0);
        if (a==0){
            piezas.clear();
            lstPiezas.removeAll();
            txtCuello.removeAll();
            spnPieza.setValue(0);
            txtCostoMaterial.setText("");
            txtCostoOperacion.setText("");
            txtPrecio.setText("");
            txtDemanda.setText("");
            txtTiempo.setText("");
            this.setVisible(false);
        }
    }
```

Método que habilita el botón calcular después de ingresar un número en el cuello de botella.

```
private void txtCuelloKeyReleased(java.awt.event.KeyEvent evt) {
    if (txtCuello.getText().length() > 0 && Double.valueOf(txtCuello.getText()) > 0) {
        btnCalcular.setEnabled(true);
    } else {
        btnCalcular.setEnabled(false);
    }
}
```

Método que calcula la heurística de Greddy al dar clic en calcular.

```
private void btnCalcularActionPerformed(java.awt.event.ActionEvent evt) {
    ArrayList<Pieza> npiezas = new ArrayList<Pieza>();
    Double cuello = Double.valueOf(txtCuello.getText());
    Double ahorroTotal = 0.0;
    for (Pieza pz : this.piezas) {
        if (pz.getAhorroPer() > 0 && cuello - (pz.getTiempo() * pz.getDemanda()) >= 0) {
            npiezas.add(pz);
            cuello -= pz.getTiempo() * pz.getDemanda();
            ahorroTotal += pz.getAhorroPer();
        }
    }
    if (npiezas.size() > 0) {
        SolucionGreddy sg = new SolucionGreddy(null, true);
        sg.setLocationRelativeTo(null);
        sg.setData(npiezas, ahorroTotal);
        sg.setVisible(true);
    }
    else {
        mostrarMensaje("Ninguna pieza generaría ahorro si se produce en el sistema.");
    }

}
```

Acciones que se hacen cuando se da clic en el botón asociar. Es la base de las asociaciones en la heurística de carga. El fin de este método es agregar a la lista de máquinas asociadas y herramientas asociadas las selecciones al momento de pulsar dicho botón.

```
private void btnAsociarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int p = lstPartesMaquinas.getSelectedIndex();
        int m = lstMaqAsoPte.getSelectedIndex();
        int h = lstHerrAsoMaqu.getSelectedIndex();
        if (piezas.get(p).getMaqAsociadas().contains(m)){
            mostrarMensaje("La pieza "+piezas.get(p).toString()+" ya tiene asociada la máquina
             "+maquinas.get(m).getNombre()+".");
        } else {
            if (piezas.get(p).getHerrAsociada().contains(herramientas.get(h))){
                mostrarMensaje("La pieza "+piezas.get(p).toString()+" ya tiene asociada la herramienta
                "+herramientas.get(h).getNombre()+".");
            } else {
                piezas.get(p).maqAsociada(m);
                maquinas.get(m).herrAsignada(herramientas.get(h));
                piezas.get(p).addHerrAsociada(herramientas.get(h));
                piezas.get(p).tiempoAsociado(Double.valueOf(txtTiempoProcesamiento.getText()));
                mostrarMensaje("Asociación Agregada");
                mostrarMensaje("La pieza " + piezas.get(p) + " tiene asociadas " +
                  piezas.get(p).getMaqAsociadas().size() + " Maquinas.");
            }
        }
    } catch (Exception w) {
        mostrarError("Alguno de los datos es inadecuado, verifique");
```

```
    }
  }
```

Método para ordenar ascendentemente con respecto a los dias restantes para la entrega, si es igual las piezas se ordenan descendentemente según el tiempo de procesamiento total.

```
  public static class ComparadorPiezasL implements Comparator<PiezaL> {

    @Override
    public int compare(PiezaL o1, PiezaL o2) {
      return o1.getDiasRestantes() > o2.getDiasRestantes() ? 1 : o1.getDiasRestantes() ==
        o2.getDiasRestantes() ? o1.getTamanho()*o1.getTiempo() < o2.getTamanho()*o2.getTiempo() ? 1 :
        o1.getTamanho()*o1.getTiempo() == o2.getTamanho()*o2.getTiempo() ? 0 : -1 : -1;
    }
  }
```

Acciones realizadas al dar clic en el botón calcular en la heurística de formación de lotes, realiza los pasos descritos en el algoritmo formulado en este trabajo.

```
  private void btnCalcularLotesActionPerformed(java.awt.event.ActionEvent evt) {
    try {
      ArrayList<PiezaL> tem = new ArrayList<PiezaL>(piezas);
      ArrayList<Maquina> maquina= new ArrayList<Maquina>(maquinas);
      Collections.sort(tem, new ComparadorPiezasL());
      lotes = new ArrayList <Lote> ();
      Lote L1 = new Lote ();
      for (int j = 0; j < maquina.size(); j++){
        L1.addTiempo(maquina.get(j).getTiempoTotal());
        L1.addPortaHarr(maquina.get(j).getPortaHerrTotal());
        L1.addTiempoTotal(maquina.get(j).getTiempoTotal());
        L1.addPortaHarrTotal(maquina.get(j).getPortaHerrTotal());
      }
      lotes.add(L1);
      int l = 0;
      int nl =0;
      boolean asignar = false;
      for (int i = 0; i < tem.size(); i++){
        for (int lote=0; lote<lotes.size();lote++){
          int ev=0;
          int end=0;
          for (int j = 0; j < maquina.size(); j++){
            int her = 0;//interseccion entre las herramienas asociadas a maq y a pieza
            if (tem.get(i).getMaqAsociadas().contains(j)){
              int m = tem.get(i).getMaqAsociadas().indexOf(j);
              boolean he=true;
              boolean fin=false;
              for (int h=0;h<tem.get(i).getHerrAsociada().size();h++){
                while (fin==false){
                  for (int ha=0;ha<lotes.get(lote).getHerramientas().size();ha++){
                    if(he==true){
                      if (tem.get(i).getHerrAsociada().get(h)==lotes.get(lote).getHerramientas().get(ha)) {
                        he=false;
                      }
                    }
                  }
                  fin = true;
                }
                ArrayList<Herramienta> herpie=new ArrayList<Herramienta>();
                for (int hm=0;hm<maquina.get(j).getHerramientas().size();hm++){
                  if (maquina.get(j).getHerramientas().get(hm)==tem.get(i).getHerrAsociada().get(h)){
                    herpie.add(maquina.get(j).getHerramientas().get(hm));
                  }
```

```
                }
                if (herpie.size()>0 && he==true){
                    her++;

                }
            }
             Double nTiempoDisp = lotes.get(lote).getTiempoDisp().get(j)-(tem.get(i).getTiemAsociado().
               get(m)*tem.get(i).getTamanho());
            Integer nPortaHerDisp = lotes.get(lote).getPortaHerrDisp().get(j)-her;
            if (nTiempoDisp>=0 && nPortaHerDisp>=0 && ev==0){
                asignar = true;
                nl=lote;
                end+=1;
            } else {
                ev+=1;
                asignar = false;
                nl=l;
            }
        }
    }
    if (end==tem.get(i).getMaqAsociadas().size()){
        lote=lotes.size();
    }
}
if (asignar == true){
    lotes.get(nl).addPieza(tem.get(i));
    for (int j = 0; j < maquina.size(); j++){
        int her = 0;//interseccion entre las herramienas asociadas a maq y a pieza
        if (tem.get(i).getMaqAsociadas().contains(j)){
            int m = tem.get(i).getMaqAsociadas().indexOf(j);
            boolean he=true;
            boolean fin=false;
            for (int h=0;h<tem.get(i).getHerrAsociada().size();h++){
                while (fin==false){
                for (int ha=0;ha<lotes.get(nl).getHerramientas().size();ha++){
                    if(he==true){
                        if (tem.get(i).getHerrAsociada().get(h)==lotes.get(nl).getHerramientas().get(ha)) {
                            he=false;
                        }
                    }
                }
                fin = true;
            }
                ArrayList<Herramienta> herpie=new ArrayList<Herramienta>();
                for (int hm=0;hm<maquina.get(j).getHerramientas().size();hm++){
                  if (maquina.get(j).getHerramientas().get(hm)==tem.get(i).getHerrAsociada().get(h)){
                    herpie.add(maquina.get(j).getHerramientas().get(hm));
                  }
                }
                if (herpie.size()>0 && he==true){
                    her++;
                    lotes.get(nl).addHerramienta(herpie.get(0));
                }
            }
              Double nTiempoDisp = lotes.get(nl).getTiempoDisp().get(j)-(tem.get(i).getTiemAsociado().
               get(m)*tem.get(i).getTamanho());
            Integer nPortaHerDisp = lotes.get(nl).getPortaHerrDisp().get(j)-her;
            lotes.get(nl).getTiempoDisp().set(j, nTiempoDisp);
            lotes.get(nl).getPortaHerrDisp().set(j, nPortaHerDisp);
        }
```

```
        }
    }else {
        tem.get(i).setEvaluado(tem.get(i).getEvaluado()+1);
        if(tem.get(i).isFraccionar() == false && tem.get(i).getEvaluado()<2){
            Lote siguienteL = new Lote();
            for (int ja = 0; ja < maquina.size(); ja++){
                siguienteL.addTiempo(maquina.get(ja).getTiempoTotal());
                siguienteL.addPortaHarr(maquina.get(ja).getPortaHerrTotal());
                siguienteL.addTiempoTotal(maquina.get(ja).getTiempoTotal());
                siguienteL.addPortaHarrTotal(maquina.get(ja).getPortaHerrTotal());
            }
            lotes.add(siguienteL);
            l++;
            i--;
        } else {
            int pizsL1 = 0;
            for (int ji = 0; ji < maquina.size(); ji++){
                if (tem.get(i).getMaqAsociadas().contains(ji)){
                    int mq = tem.get(i).getMaqAsociadas().indexOf(ji);
                    int com = (int) (lotes.get(l).getTiempoDisp().get(ji)/tem.get(i).getTiemAsociado().get(mq));
                    if (pizsL1<=0){
                        pizsL1 = com;
                    }  else {
                        if (pizsL1 > com){
                            pizsL1 = com;
                        }
                    }
                }
            }
            if(tem.get(i).isFraccionar() == true && pizsL1>0){
                PiezaL pNueva;
                int cantNueva = tem.get(i).getTamanho()- pizsL1;
                tem.get(i).setEvaluado(0);
                  pNueva = new PiezaL (tem.get(i).getNombre(), tem.get(i).getOperacion(), pizsL1,
                    tem.get(i).getDiasRestantes(),tem.get(i).isFraccionar());
                pNueva.setMaqAsociadas(tem.get(i).getMaqAsociadas());
                pNueva.setHerrAsociada(tem.get(i).getHerrAsociada());
                pNueva.setTiemAsociado(tem.get(i).getTiemAsociado());
                tem.get(i).setTamanho(cantNueva);
                tem.add(i, pNueva);
                i--;
            } else {
                if (tem.get(i).isFraccionar() == true){
                    Lote siguienteL = new Lote();
                    for (int ja = 0; ja < maquina.size(); ja++){
                        siguienteL.addTiempo(maquina.get(ja).getTiempoTotal());
                        siguienteL.addPortaHarr(maquina.get(ja).getPortaHerrTotal());
                        siguienteL.addTiempoTotal(maquina.get(ja).getTiempoTotal());
                        siguienteL.addPortaHarrTotal(maquina.get(ja).getPortaHerrTotal());
                    }
                    lotes.add(siguienteL);
                    l++;
                    i--;
                }
            }
        }
    }
}
SolucionLotes sL= new SolucionLotes(null, true);
sL.setLocationRelativeTo(null);
```

```
            sL.setData(lotes, tem, maquina);
            sL.setVisible(true);
        } catch (Exception w) {
            mostrarError(w.getMessage());
        }
    }
Método para calcular la heurística de carga fase 1.
private void btnCalcularActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            opeAOr = new ArrayList<Operacion>(operaciones);
            opAsignadas.clear();
            maq=new ArrayList<Maquina>(maquinas);
            while (opeAOr.size()> 0){
                Operacion opeElegida = elegirOperacion();
                Integer mE = elegirMaquina(opeElegida);
                opeElegida.setMaqElegida(mE);
                opAsignadas.add(opeElegida);
            }
            SolucionCargas sC = new SolucionCargas (null, true);
            sC.setLocationRelativeTo(null);
            sC.setData(opAsignadas, maquinas);
            sC.setVisible(true);
        } catch (Exception w) {
            mostrarError(w.getMessage());
        }
    }

Método para elegir operación.
private Operacion elegirOperacion (){
        Operacion opElegida;
        for (int i=0;i<opeAOr.size();i++){
            int mV = 0;
            opeAOr.get(i).getMaqFactibles().clear();
            for (int j=0;j<maq.size();j++){
                Integer t =opeAOr.get(i).getMaqAsociadas().indexOf(j);
                if (t>=0){
                    Double y = maq.get(j).getTiempoSinAsignar();
                    Integer kj=maq.get(j).getPortaHerrSinAsignar();
                    Double pm = (opeAOr.get(i).getTiemAsociado().get(t) * opeAOr.get(i).getTamanhoOrden()) /
                      maq.get(j).getCantidad();
                    Integer km= 1/maq.get(j).getCantidad();
                    int herramien=maq.get(j).getHerramientas().indexOf(opeAOr.get(i).getHerramienta());
                    if(herramien>=0){
                        km=0;
                    }
                    if(pm <= y && km <=kj ){
                        mV+=1;
                        opeAOr.get(i).addMaqFactibles(j);
                    }
                }
            }
            opeAOr.get(i).setMaqViables(mV);
        }
        Collections.sort (opeAOr, new ComparadorOperacion()); Collections.sort(opeAOr, new
          ComparadorOperacion());
        Collections.sort(opeAOr, new ComparadorOperacion());
        opElegida= opeAOr.get(0);
        opeAOr.remove(0);
        return opElegida;
    }
```

Método para elegir maquina

```
private Integer elegirMaquina(Operacion opElegida){
    Integer jElegida = 0;
    Double tMayDisp = 0.0;
    if(opElegida.getCostoTO().isEmpty()){
        for (int j=0;j<maq.size();j++){
            Integer t =opElegida.getMaqAsociadas().indexOf(j);
            Integer f =opElegida.getMaqFactibles().indexOf(j);
            Double y = maq.get(j).getTiempoSinAsignar();
            if (t>=0&&f>=0&&y>0){
                Double p = opElegida.getTiemAsociado().get(t)*opElegida.getTamanhoOrden();
                Double pm= p/maq.get(j).getCantidad();
                if (tMayDisp <= y-pm){
                    tMayDisp = y-pm;
                    jElegida = j;
                }
            }
        }
    } else {
        Double min= opElegida.getCostoTO().get(0);
        for(int c=0; c<opElegida.getCostoTO().size();c++){
            if(min>opElegida.getCostoTO().get(c)){
                min=opElegida.getCostoTO().get(c);
                int jtem=opElegida.getCostoTO().indexOf(min);
                jElegida=opElegida.getMaqAsociadas().get(jtem);
            }
        }
        if(opElegida.getCostoTO().indexOf(min)!=opElegida.getCostoTO().lastIndexOf(min)){
            for (int j=0;j<maq.size();j++){
                Integer t =opElegida.getMaqAsociadas().indexOf(j);
                if(min==opElegida.getCostoTO().get(t)){
                    Integer f =opElegida.getMaqFactibles().indexOf(j);
                    Double y = maq.get(j).getTiempoSinAsignar();
                    if (t>=0&&f>=0&&y>0){
                        Double p = opElegida.getTiemAsociado().get(t)*opElegida.getTamanhoOrden();
                        Double pm= p/maq.get(j).getCantidad();
                        if (tMayDisp <= y-pm){
                            tMayDisp = y-pm;
                            jElegida = j;
                        }
                    }
                }
            }
        }else{
            Integer t =opElegida.getMaqAsociadas().indexOf(jElegida);
            Double y = maq.get(jElegida).getTiempoSinAsignar();
            Double p = opElegida.getTiemAsociado().get(t)*opElegida.getTamanhoOrden();
            Double pm= p/maq.get(jElegida).getCantidad();
            tMayDisp = y-pm;
        }
    }
    maq.get(jElegida).setTiempoSinAsignar(tMayDisp);
    int ne=maq.get(jElegida).getHerramientas().indexOf(opElegida.getHerramienta());
    if(ne<0){
        int pth = maq.get(jElegida).getPortaHerrSinAsignar()-1;
        maq.get(jElegida).setPortaHerrSinAsignar(pth);
        maq.get(jElegida).herrAsignada(opElegida.getHerramienta());
    }
```

```
        return jElegida;
        }
```

Metodo para formar cluster

```
    private ArrayList<Cluster> crearCluster (Integer maq, ArrayList<Operacion> o){
        ArrayList<Cluster> cluster = new ArrayList<Cluster>();
        Integer pieza = o.get(0).getPieza();
        Double t = maquinas.get(maq).getTiempoDisponible();
        Integer k = maquinas.get(maq).getTamPortaHerramientas();
        Cluster cl = new Cluster (pieza, maquinas.get(maq));
        cl.setTiempo(t);
        cl.setTiempoTotal(t);
        cl.setPortaHerrTotal(k);
        cl.setPortaHerr(k);
        cluster.add(cl);
        for (int ope =0; ope< o.size(); ope++){
            boolean asignado=false;
            for (int c =0; c<cluster.size(); c++){
                Cluster clEv= cluster.get(c);
                if (clEv.getTiempo()-o.get(ope).getTiempoOperacion() >= 0 && clEv.getPortaHerr()-1 >= 0){
                    Integer ci = cluster.indexOf(clEv);
                    clEv.addOperaciones(o.get(ope));
                    int hc=clEv.getHerramientas().indexOf(o.get(ope).getHerramienta());
                    if(hc<0){
                        clEv.addHerramientas(o.get(ope).getHerramienta());
                    }
                    asignado=true;
                    clEv.setTiempo(clEv.getTiempo()-(o.get(ope).getTiempoOperacion()));
                    clEv.setPortaHerr(clEv.getPortaHerr()-1);
                    cluster.set(ci, clEv);
                    c=cluster.size();
                }
            }
            if(asignado==false){
                if (t-((o.get(ope).getTiempoOperacion())) >= 0 && k-1 >= 0){
                    Cluster ncl = new Cluster (pieza, maquinas.get(maq));
                    ncl.setTiempo(t-o.get(ope).getTiempoOperacion());
                    ncl.setTiempoTotal(t);
                    ncl.setPortaHerrTotal(k);
                    ncl.setPortaHerr(k-1);
                    ncl.addOperaciones(o.get(ope));
                    int hc=ncl.getHerramientas().indexOf(o.get(ope).getHerramienta());
                    if(hc<0){
                        ncl.addHerramientas(o.get(ope).getHerramienta());
                    }
                    cluster.add(ncl);
                }else{
                    mostrarMensaje("La operacion "+o.get(ope)+" no se pudo asignar,\nya que no cupo en un cluster
                    sola.");
                }
            }
        }
        return cluster;
    }
```

Método para crear grupos

```
    private ArrayList<Grupos> crearGrupos (Maquina m, Integer g) {
```

```
        ArrayList<Grupos> gm = new ArrayList<Grupos>();
        Integer gj = g;
        Integer a = m.getCantidad();
        Integer b = 0;
        if(a==1){
            Grupos gr = new Grupos(m.getNombre(), m.getTiempoDisponible(), m.getTamPortaHerramientas());
            gr.setInd(1);
            gm.add(gr);

        }else{
            if(gj<a){
                for (int i = 0; i < gj; i++){
                    Grupos gr = new Grupos (m.getNombre(), m.getTiempoDisponible(), m.getTamPortaHerramientas
                        ());
                    gr.setInd(gm.size()+1);
                    gm.add(gr);
                }
            }else{
                for (int i = 0; i < a; i++){
                    Grupos gr = new Grupos (m.getNombre(), m.getTiempoDisponible(), m.getTamPortaHerramientas
                      ());
                    gr.setInd(gm.size()+1);
                    gm.add(gr);
                }
            }

        }
        while (a > b){
            for(int i = 0; i< gm.size(); i++){
                int cantAn = gm.get(i).getCantidad()+1;
                Double timNu = cantAn*m.getTiempoDisponible();
                int PortNu = cantAn*m.getTamPortaHerramientas();
                gm.get(i).setCantidad(cantAn);
                gm.get(i).setTiempoTotal(timNu);
                gm.get(i).setTiempoSinAsignar(timNu);
                gm.get(i).setPortaHerrTotal(PortNu);
                gm.get(i).setPortaHerrSinAsignar(PortNu);
                b++;
                if(b==a){
                    i=gm.size()+1;
                }
            }
        }
        return gm;
    }
```

Método para redondear hacia arriba.

```
    public Integer redondeo(Double n){
        Integer ent = n.intValue();
        if (ent/n<1){
            ent+=1;
        }
        return ent;
    }
```

Método para calcular la heurística de cargas fase 2 utilizando los tres métodos anteriores.

```
    private void btnCalcularActionPerformed(java.awt.event.ActionEvent evt) {
        ArrayList<Maquina> maqTem=new ArrayList<Maquina> (maquinas);
        grupos=new ArrayList<Grupos>();
        for (int m = 0; m < maquinas.size(); m++){
```

```
ArrayList<Grupos> gm;
piezas.clear();
Integer delta;
Integer k;
ArrayList<Herramienta> h = new ArrayList<Herramienta>();
ArrayList<Cluster> cl= new ArrayList<Cluster>();
for (int i = 0; i < operaciones.size(); i++){
    if(operaciones.get(i).getMaqAsociadas().contains(m)){
        Integer nombrep = operaciones.get(i).getPieza();
        int np= piezas.indexOf(nombrep);
        if(np<0){
            piezas.add(nombrep);
        }
    }
}
for (int p = 0; p < piezas.size(); p++){
    opetem.clear();
    for (int i = 0; i < operaciones.size(); i++){
  if (piezas.get(p) == operaciones.get(i).getPieza() && operaciones.get(i).getMaqAsociadas().get(0) ==
     m){
            opetem.add(operaciones.get(i));
        }
    }
    if(enOrden==true){
        Collections.sort(opetem, new ComparadorOperaciones());
    }
    ArrayList<Cluster> clfor = crearCluster(m, opetem);
    for (int clu = 0; clu < clfor.size(); clu++){
        cl.add(clfor.get(clu));
    }
}
for (int clu = 0; clu < cl.size(); clu++){
   for (int c = 0; c < cl.get(clu).getHerramientas().size(); c++){
        int her= h.indexOf(cl.get(clu).getHerramientas().get(c));
        if(her<0){
            h.add(cl.get(clu).getHerramientas().get(c));
        }
    }
}
delta = h.size();
k = maquinas.get(m).getTamPortaHerramientas();
Integer gj = redondeo(delta.doubleValue()/k);
gm = crearGrupos(maquinas.get(m), gj);
ArrayList<Cluster> cltem= cl;
Collections.sort(cltem, new ComparadorCluster());
for(int c=0; c<cltem.size();c++){
    boolean clAsig=false;
    int grupo = 0;
    Double tLibre = 0.0;
    Double tdasig;
    for(int i = 0; i< gm.size(); i++){
        tdasig=gm.get(i).getTiempoSinAsignar()-(cltem.get(c).getTiempoTotal()-cltem.get(c).getTiempo());
        if (tLibre <= tdasig && tdasig >= 0 && tLibre>=0 && gm.get(i).getTipoMaq().equals
        (maquinas.get(m).getNombre())){
            tLibre = tdasig;
            grupo = i;
            clAsig=true;
        }
    }
    if(clAsig==true){
```

```
                gm.get(grupo).addCluster(cltem.get(c));
                gm.get(grupo).setTiempoSinAsignar(tLibre);
              }else{
            mostrarMensaje("El cluster "+cltem.get(c)+" no se pudo asignar,\nya que no hay grupos disponibles
            con tiempo libre suficiente.");
              }
            }
            for(int i = 0; i< gm.size(); i++){
               grupos.add(gm.get(i));
            }
        }
        SolucionCargas2 sC = new SolucionCargas2 (null, true);
        sC.setLocationRelativeTo(null);
        sC.setData(grupos, maquinas);
        maquinas=maqTem;
        sC.setVisible(true);
    }
```