Universitat Autònoma de Barcelona

2232: OPTIMIZADOR DE OPERARIOS INDUSTRIALES

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Raúl Suárez Dabó
i dirigit per
Toni Laserna García
Bellaterra, 9 de setembre de 2010

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Toni Laserna García.

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Raúl Suárez Dabó

I per tal que consti firma la present.

Signat: Toni Laserna García.

Bellaterra, 9 de setembre de 2010

Índice

Resumen ejecutivo del proyecto final de carrera.

Este es un proyecto orientado hacia la optimización de actividades productivas en plantas industriales de empresas *pymes* (pequeñas y medianas empresas) mediante entornos de programación. Más concretamente la idea es realizar una aplicación que permita realizar planificaciones **eficientes** de cómo distribuir el trabajo a los operarios de producción dentro de una planta industrial convencional. Dichas plantas disponen de una serie de recursos, máquinas, que junto a su plantilla de trabajadores deben realizar las diferentes órdenes de fabricación. Estas tareas se han de poder realizar dentro de un marco de tiempo aceptable y a la vez adecuado al servicio que desea dar la planta a sus clientes.

Por tal de abarcar de forma correcta este problema se distribuye el proyecto final de carrera en tres bloques principales:

El primero de ellos es buscar un algoritmo/s que permita encontrar una solución aceptable, una buena solución.

En segundo lugar realizar un estudio de los datos necesarios para poder llevar a cabo dicho algoritmo y diseñar una base de datos que disponga de la información adecuada para la organización de los recursos.

Seguidamente, el programa que ejecutará la planificación del conjunto de órdenes de fabricación de la planta y su evaluación del nivel de eficiencia de la solución propuesta. El programa será realizado en el lenguaje de programación *Java*.

Y finalmente, la presentación de los resultados y la bondad de la heurística desarrollada.

1 Introducción.

1.1 Estado del arte.

Actualmente las pymes son organizaciones con serias limitaciones de recursos (financieros, tecnológicos, etc), actuando generalmente como proveedoras de empresas más grandes (Sistema de Valor, Porter).

Las pymes no suelen disponer de ningún tipo de software específico que les permita gestionar la carga de trabajo de los operarios de la planta de manera eficiente. Debido a ello se realiza dicha distribución de carga de manera **manual**, confiando en la experiencia del responsable de producción, encargado de turno, etc. Quedando registrado a lo sumo, en ficheros de *Excel* cumplimentados a criterio del responsable.

Aunque actualmente en el mercado hay herramientas *MRP* (Planificación de los requerimientos de material) ya consolidadas, suelen ser de carácter general y poco orientadas exclusivamente a la planificación eficiente de tareas a los operarios. Se orientan a la jerarquización de preferencias para asegurar materiales a la producción, mantener el nivel de inventario y planear las actividades de manufactura. Este último como trazo complementario a los dos primeros. En ningún caso se contempla como una herramienta de gestión de personal productivo como único concepto.

Otras herramientas que a priori podrían satisfacer las necesidades de asignación de operarios, son los *FCS* (*Finite Capacity Scheduling*). Estas soluciones están orientadas a la planificación simultánea de recursos (materiales, máquinas, personal y herramientas) y el cálculo de las fechas de producción para satisfacer la demanda y calidad de servicio.

Los métodos de cálculo del tiempo suelen ser **hacia atrás** de la orden de fabricación, teniendo en cuenta el tiempo de producción de la materia prima y dependientes de ésta. Todo esto prefijando un tiempo de entrega estándar, es decir, con un tiempo indicado para cualquier entrega.

De la misma manera, otro método es hacia delante a partir de la disposición de la materia prima.

Una de las soluciones mas implantadas a nivel internacional es *Preactor*, una aplicación poco orientada a pymes, sobretodo pequeñas empresas, por su nivel de inversión (adquisición de licencias, coste de la implementación y mantenimiento anual, del orden de 50Keur/lic) y complejidad e impacto en su proceso de implementación y uso; p.e. se exige un nivel de información (datos de la planta) generalmente no disponibles.

1.2 Justificación del proyecto.

Las plantas de producción suelen estar distribuidas en diferentes secciones. En donde hay un responsable y éste distribuye manualmente el trabajo a su criterio según el estado de carga de trabajo actual y futuro a c/p. Como consecuencia, es frecuente realizar movilidad de trabajadores a diferentes secciones, horas extras o subcontrata por *ETT* del personal necesario por tal de satisfacer la demanda.

Por ello, se justifica la necesidad de disponer de una aplicación capaz de ofrecer las siguientes funcionalidades:

- Facilitar la planificación y programación de las tareas a demanda de las peticiones recibidas.
- Centralizar una planificación que permita distribuir el trabajo en conjunto de toda la planta.
- Utilizar una base de datos que disponga de toda la información necesaria y actualizada.
- Reducir el nivel de recursos y tiempo en la actividad de planificación, ya que actualmente se estima que trabajadores altamente cualificados designan gran parte de su jornada laboral en realizar tareas de planificación.
- Minimizar los costes directos de mano de obra (MOD) gracias a una planificación más eficiente.

- Disponer de una herramienta que permita realizar una planificación de manera rápida (a nivel de cálculo computacional) de los recursos ante una incidencia (avería de una máquina, baja de un trabajador, etc.).

El componente informático donde recae toda la justificación del proyecto es el algoritmo de planificación, el cual consiste en desarrollar un sistema inteligente que planifique con un horizonte temporal (planificación táctica-operativa-estratégica) y programe eficientemente a corto-plazo (dentro del turno de trabajo) las actividades/operaciones de cada uno de los operarios.

La dificultad radica en la complejidad de la optimización de las actividades en centros productivos debido a su gran número, restricciones y combinatoria posible, y por el grado de eficiencia que requiere el método-técnica de optimización a nivel de tiempo de cálculo computacional.

A nivel de investigación, nos encontramos con un Problema de Reparto y Asignación combinado con un Problema de Secuenciación:

Def. Problema de Reparto y Asignación

A partir de un conjunto de recursos, se han de hacer un conjunto de actividades. El objetivo es determinar como asignar los recursos a las actividades de manera que la utilidad o rendimiento se maximice.

Def. Problema de Secuenciación:

Se dispone de un conjunto de actividades a realizar y que están relacionadas entre si con ciertas restricciones. El objetivo es determinar el momento en que se debe iniciar y finalizar cada una de las actividades, minimizando el tiempo total de hacer todas las actividades relacionadas.

El gran volumen de actividades, sus interrelaciones, recursos, restricciones temporales, etc. que tiene la propia naturaleza de un centro productivo, hacen que **este problema de optimización combinatoria sea Np-Hard.**

Un problema Np-Hard (o Np-Complejo o Np-Dificil), en síntesis, es un problema de difícil solución en tiempo computacional razonable si utilizamos las técnicas clásicas de optimización (procedimientos exactos-PE).

Los PE son procedimientos analíticos que en caso que haya una solución, la encuentran (óptimo global), generalmente son técnicas matemáticas en donde el óptimo se calcula mediante la definición de la función de coste objetivo, realizando su primera derivada e igualándola a valor cero. Un ejemplo de PE es la Programación Lineal.

A nivel académico hay multitud de problemas resolubles mediante PE. Cuando se pretenden aplicar a casos reales de cierta complejidad sin posibilidad de simplificación, ya no es viable la aplicación de los procedimientos exactos. Por ello, se desarrollaron los Métodos Heurísticos (MH), los cuales en tales situaciones son capaces de encontrar una buena solución (solución de óptimo local).

Los Métodos Heurísticos generan soluciones mediante método. Existen multitud de métodos como los procedimientos de exploración dirigida (lógicas de Ramificación y Poda "*Branch&Bound*") de amplio conocimiento en el ámbito científico.

No obstante, frente a la complejidad que nos encontramos en la optimización de las actividades en centros productivos, los métodos conocidos son de escasa eficiencia.

Por ello **se toma la decisión de realizar un procedimiento metaheurístico propio con el objetivo de realizar la distribución, no necesariamente optima pero si aceptable**. Dentro del marco de restricciones que disponemos y los escasos datos matemáticos que perfilan el flujo de trabajo dentro de la pyme.

1.3 Objetivo

El principal objetivo de este proyecto final de carrera es el diseño, programación y su verificación en casos de estudio, de una aplicación que realice la planificación del trabajo de los operarios de un determinado tiempo.

Con la idea de automatizar y mejorar cualitativamente el proceso de planificación manual en la gran mayoría de pymes.

Debido al carácter innovador de éste, su primer objetivo siendo el más relevante es desarrollar un nuevo método basado en heurísticas para planificar y distribuir la carga de trabajo a los operarios de producción. Por tanto, no es simplemente el diseño y programación de una aplicación informática, sino el estudio y análisis de cuál sería la metodología más adecuada en términos generales (no en un caso en concreto de planta industrial).

1.4 Viabilidad del proyecto.

Se considera que la futura aplicación no requiere de ninguna tecnología informática de vanguardia que implique altos riesgos en su utilización. Se trata de desarrollar un *software* que permita utilizar una serie de recursos como son una base de datos *MySQL* y el uso de un lenguaje de programación *Java*. Las cuales permiten realizar el desarrollo del *software* de una manera rápida y a la vez da la portabilidad a cualquier tipo de plataforma. Además éste lenguaje dispone de un amplio abanico de librerías para realizar las consultas a la base de datos y generar el diagrama de *Gantt*.

Por tanto, **la viabilidad tecnológica** del proyecto está garantizada, al no existir riesgo tecnológico alguno que sea el factor critico para no alcanzar los objetivos del proyecto.

La viabilidad técnica del proyecto consiste en encontrar una buena o muy buena solución factible al problema expuesto por medios métodos heurísticos. Por ello, se centrarán gran parte de los esfuerzos en tiempo del proyecto final de carrera, en encontrar dicho nuevo método; minimizando el riesgo al fracaso del proyecto.

2 Descripción de la solución.

Para la solución del problema planteado se ha optado por la utilización de una metaheurística según se justifica al final del aptdo.1.2.

La metaheurística a desarrollar permitirá buscar una solución al problema pero no necesariamente será la más óptima (optimización combinatoria), por lo que en el momento que encuentre una solución que converja con los parámetros condicionales propuestos, la aceptará como definitiva.

Estos condicionales son la base de poder encontrar una solución, ya que según las restricciones podremos obtener una mejor o peor solución. Aquí radica la viabilidad técnica del proyecto. ¿Qué orden y secuencia de restricciones podemos imponer para encontrar una solución viable?

Es obvio que dependiendo del orden y que criterio de condiciones introduzcamos dentro del algoritmo obtendremos un resultado u otro muy diferente. **Este hecho me ha llevado a planificar seis algoritmos que se pueden agrupar en tres grupos.**

A continuación detallo como he llegado a tal resultado:

El hecho es que si analizamos detenidamente la situación de una empresa que fabrica cierto producto, nos encontramos que las órdenes de fabricación no son una restricción, sino el “trigger” del sistema de producción. Por esto, toda la algorítmica ha desarrollar en este proyecto se basa en la idea de que sin una orden de fabricación no hay procedimiento de planificación alguno. Gracias a ello, reducimos las restricciones de nuestra optimización a dos, la maquinaria y los trabajadores que producen.

La explicación de las tres agrupaciones se fundamenta en el hecho de haberme basado en dos conceptos: la utilización de los recursos y el factor tiempo. El **primer grupo** se basa en utilizar los recursos en un determinado tiempo y una vez exentos de estos, avanzar en él. El **segundo** se basa en asignar un recurso y adjudicar durante un periodo de tiempo cierta carga de trabajo, una vez hecho el intervalo de tiempo indicado pasaríamos al siguiente recurso iterativamente hasta planificarlos todos. Y el último y **tercer grupo** se

basa en buscar las fechas de entrega de las diferentes órdenes de fabricación y retroceder en el tiempo para indicar cuándo deben de empezar por tal de acabar en la fecha de entrega. Acercándonos a la política *JIT* (*Just In Time*).

Respecto los seis algoritmos que realizarán la organización, su numero es debido al factor dependiente que existe entre una máquina y un trabajador. Es el hecho de que si fijamos como primer condicionante una máquina, nos podemos encontrar con la inexistencia de un trabajador disponible para ésta. Ídem si primero fijamos el trabajador y luego la máquina (ausencia de máquina pero disposición de operario). Por todo ello, se obtienen seis algoritmos con tres agrupaciones al tratar las mismas bases de recursos, tiempo y conmutación de los condicionantes para ver sus resultados:

KronosParallelOMW *KronosParallelOWM*

KronosLinearMOW *KronosLinearWOM*

KronosReverseOMW *KronosReverseOWM*

En el siguiente apartado, detallamos los criterios de filtraje de máquinas, trabajadores y órdenes de fabricación aplicados en todos los procedimientos por tal de hacer una explicación más comprensible.

2.1 Criterios de Filtraje.

En este proyecto se ha establecido un criterio de preferencia de las órdenes de fabricación, las máquinas y trabajadores que conviene aclarar antes de explicar los algoritmos ya que será más claro su objetivo final.

- Órdenes de fabricación: éstas deben ser organizadas de forma adecuada, por lo que hay que contextualizarlas con el entorno de producción. Una *pyme* descrita en los apartados anteriores, lo más probable es que no se designe toda su producción a un único cliente, sino todo lo contrario, que sea un proveedor de varios clientes. Los cuales le realizarán pedidos a medida que los necesitan. Por lo que sus clientes tampoco realizarán grandes pedidos de producción al intentar minimizar el *stock* del almacén para evitar costes añadidos. Todo esto lleva a presuponer que dicha *pyme* dispondrá de un gran

número de peticiones de producción pero por el contrario serán de pocas unidades por tal de favorecer a su clientela (lotes de fabricación pequeños).

Por ello las órdenes de fabricación serán organizadas de dos maneras al tratar la línea de tiempo de dos maneras:

- En las heurísticas *KronosParallelOMW*, *KronosParalellOWM*, *KronosLinearMOW* y *KronosLinearWOM* las organizaremos por fecha de entrega más corta en el tiempo.
- Los algoritmos *KronosReverseOMW y KronosReverseOWM* organizarán las órdenes al inverso. Primero las menos recientes en el tiempo y, progresivamente, las más recientes.

- Máquinas: El criterio de preferencia utilizado en este caso es bastante sencillo de comprender. Si tenemos una máquina que dispone de la capacidad de realizar cierto conjunto de piezas (el cual puede abarcar desde uno a *n*). Y basándonos en el hecho de que no sabemos cuál será la siguiente orden de fabricación que podemos tratar, tenemos que maximizar la capacidad de atención de nuestra maquinaria para así realizar una **premisa** muy importante, **atender todas las peticiones posibles**.

  Por lo que usaremos un criterio de filtraje basado en la polivalencia de la maquinaria, las menos polivalentes y más eficientes (en cuestión de tiempo de ciclo) serán las primeras para ir seguidas de las más polivalentes y menos eficientes. Con lo cual, aumentamos la capacidad de reacción al beneficiar la posibilidad de producir al mantener la flexibilidad de producción frente a situaciones inesperadas.

- Trabajadores: Nos encontramos en una situación muy semejante al apartado de maquinaria. Los trabajadores al ser un condicionante situado al mismo nivel de importancia que la maquinaria tenemos que seguir beneficiando la producción de los diferentes productos. Y a la vez disponer de los trabajadores más capaces para así poder disponer de ellos ante situaciones inesperadas (fallos de línea, etc.) por lo que nos

sigue interesando un criterio de filtraje basado en la polivalencia de los trabajadores, el menos polivalente y el más eficiente. Y así mantener la capacidad de producción y disponer recursos a cambios inesperados.

Con este conjunto de criterios lo que hacemos es beneficiar la capacidad de reacción de la planta al dar flexibilidad de atender múltiples peticiones sin limitar su abanico de recursos y mantener la capacidad de respuesta ante situaciones inesperadas.

2.2 Descripción de la Meta heurística.

Se puede apreciar que el nombre de las metaheurísticas viene relacionado con el dios griego del tiempo, *Chronos*. Al trabajar constantemente con el tiempo decidí utilizarla como terminología base para todos modificando el nombre del dios griego a *Kronos*.

Las palabras “parallel” y “linear” hacen hincapié en la organización en el diagrama de Gantt. Siendo la primera, que va colocando los recursos en paralelo y una vez ocupados avanza en el tiempo. En caso de “linear” realiza lo opuesto, selecciona un recurso y avanza hasta el final para seguir al siguiente recurso.

Por otro lado tenemos “reverse” que al utilizar la línea temporal a la inversa.

Para finalizar las letras M O W hacen referencia a *machine*, orden de fabricación y *worker*, respectivamente. Con lo que indicamos el orden de preferencia en el filtraje de recursos pero vamos a explicar más concretamente cada uno de los algoritmos para que sea más claro.

Antes de entrar en la explicación de cada uno de los seis métodos, detallo la definición de estos y comprender la complejidad inicial que he intencionado darle a los diferentes algoritmos. Por ello **cada orden de fabricación se debe de terminar por la misma máquina y trabajador** una vez asignada. Como **consecuencia** trataremos de un **único turno al ser totalmente transparente** la transición de maquina o trabajador. Cada **orden de fabricación referencia a un tipo de pieza** y no agrupará ningún conjunto de estas.

2.2.1 KronosParallelOMW.

La iniciativa de esta metaheurística es el planteamiento inicial de ir atendiendo las órdenes de fabricación por orden de entrega, de las más recientes a las más antiguas. Para ello se selecciona la orden organizada por margen de entrega más corto. Se selecciona la primera de la lista y a partir de esta decide que máquinas pueden ejecutar dicha orden utilizando el concepto de las menos polivalentes a las que más. Se obtiene un listado de maquinaria con la organización condicionada a lo expuesto antes y se decide iterativamente si la máquina indicada está disponible. Si no lo está, se selecciona la siguiente y si está se fija como máquina asignada. En el caso de haber ido buscando una máquina y no está disponible ninguna en el tiempo indicado, se busca la máquina que disponga de un tiempo más corto al actual tiempo y se avanza hasta él, una vez fijada se busca un trabajador disponible para esta. A continuación se selecciona al trabajador utilizando la máquina y orden de fabricación ya seleccionadas, por el hecho de tener que saber que pieza se produce. Y igual que ocurre con la máquina, se organiza de menos polivalente a más y de más efectivo a menos. Seleccionando al primero de la lista e ir iterando hasta encontrar si está disponible, no ocupado realizando otra tarea. Si está libre se asigna la nueva operación en la base de datos con toda la información relatada. Si no está libre se descarta la orden de fabricación y se pasa a la siguiente.

Dado el flujo de ejecución aquí expuesto se tiene en cuenta como un condicionante flexible al trabajador. **Por ello en este algoritmo se supone un mayor número de trabajadores que de maquinaria.** Al tratarse de este último despreciable a nivel conceptual.

A continuación muestro el diagrama de flujo.

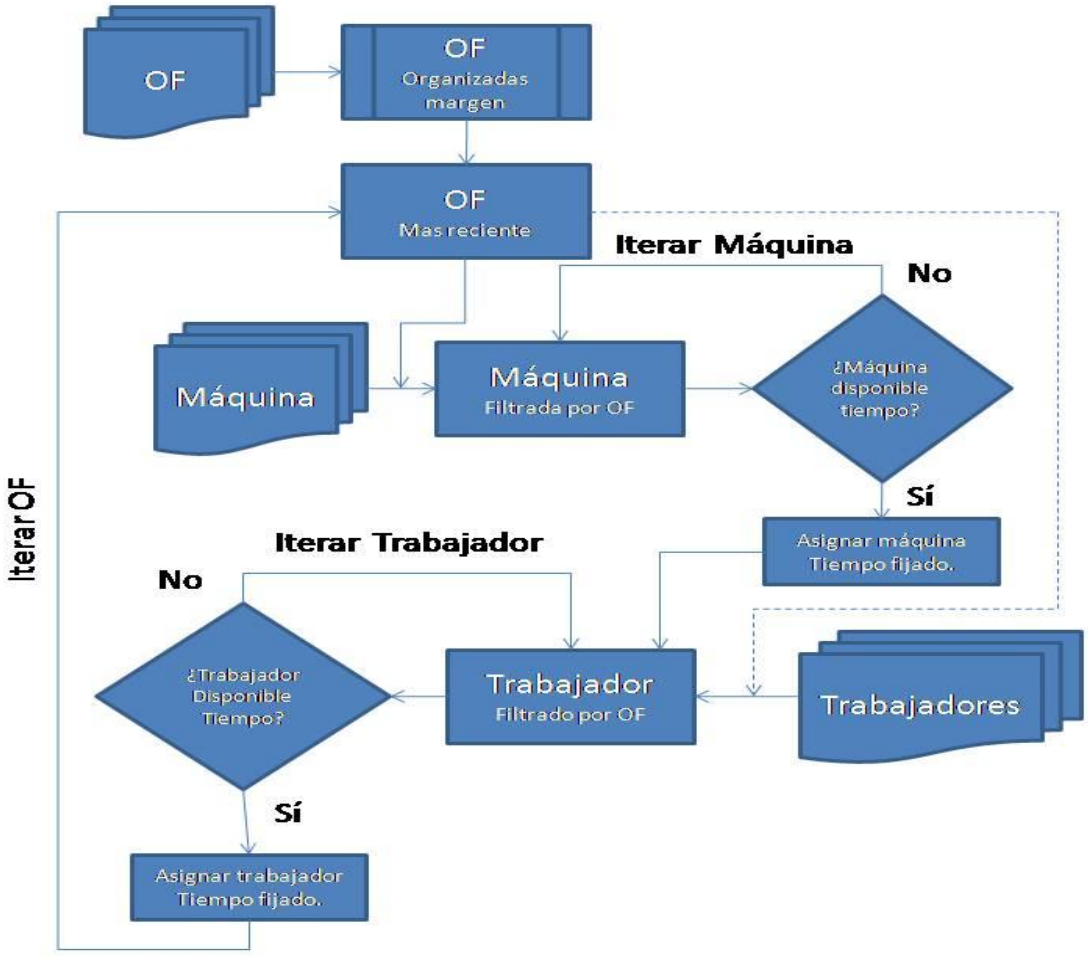


Diagrama de flujo del *KronosParallelOMW*.

Es fácil prever que esta heurística permite la utilización de los recursos máquina desde el principio de la planificación (hablando desde un punto de vista de tiempo). Por lo que **las empresas que deseen utilizar toda la maquinaria que disponen dentro de todo el turno** (al tener un factor más determinante a nivel económico que la mano de obra) **se verán beneficiadas** al colocar los tiempos de trabajo ajustados a éstas.

2.2.2 KronosParallelOWM.

El siguiente algoritmo responde a la conmutación de intercambiar la preferencia de máquina y trabajador. Al no saber si es mejor dimensionar la carga de trabajo sobre la maquinaria y luego sobre el trabajador, he decidido realizar la

permutación y ver que tal resultaría. Realizar la organización de las órdenes de trabajo por margen, como en el caso anterior, y a partir de ahí realizar la asignación de trabajadores filtrados por las órdenes. Fijarlo y buscar una máquina que atienda la petición para, finalmente, insertar la operación en la base de datos.

Como asimetría al algoritmo anterior, en este caso suponemos que esta metaheurística considera como algo flexible la disposición de la maquinaria y algo más fijo los operarios. Por lo que **disponemos de maquinas disponibles para su uso a comparativa de operarios.**

Por tanto, de entrada este algoritmo favorece el nivel ocupacional de todos los trabajadores, creando unos cuadros de horarios basados en ellos y su productividad, dejando a segundo plano el recurso de la maquinaria. Por lo que en un principio se conseguirá una distribución de la tarea bastante homogénea del primero de los condicionales (el trabajador en este caso).

Escoge la primera orden de fabricación organizadas por margen más corto, y a partir de ésta decide que trabajador la ejecuta con el filtraje ya explicado. Se itera sobre el trabajador hasta encontrar uno disponible, en caso de no encontrarlo se busca al trabajador disponible más corto en distancia temporal, sino la orden se descarta. En caso de encontrar uno se fija y se pasa a buscar una máquina en el tiempo actual. Realizando los mismos pasos de filtraje e iteración hasta encontrar una. Si no se encuentra la orden se descarta y se pasa a la siguiente.

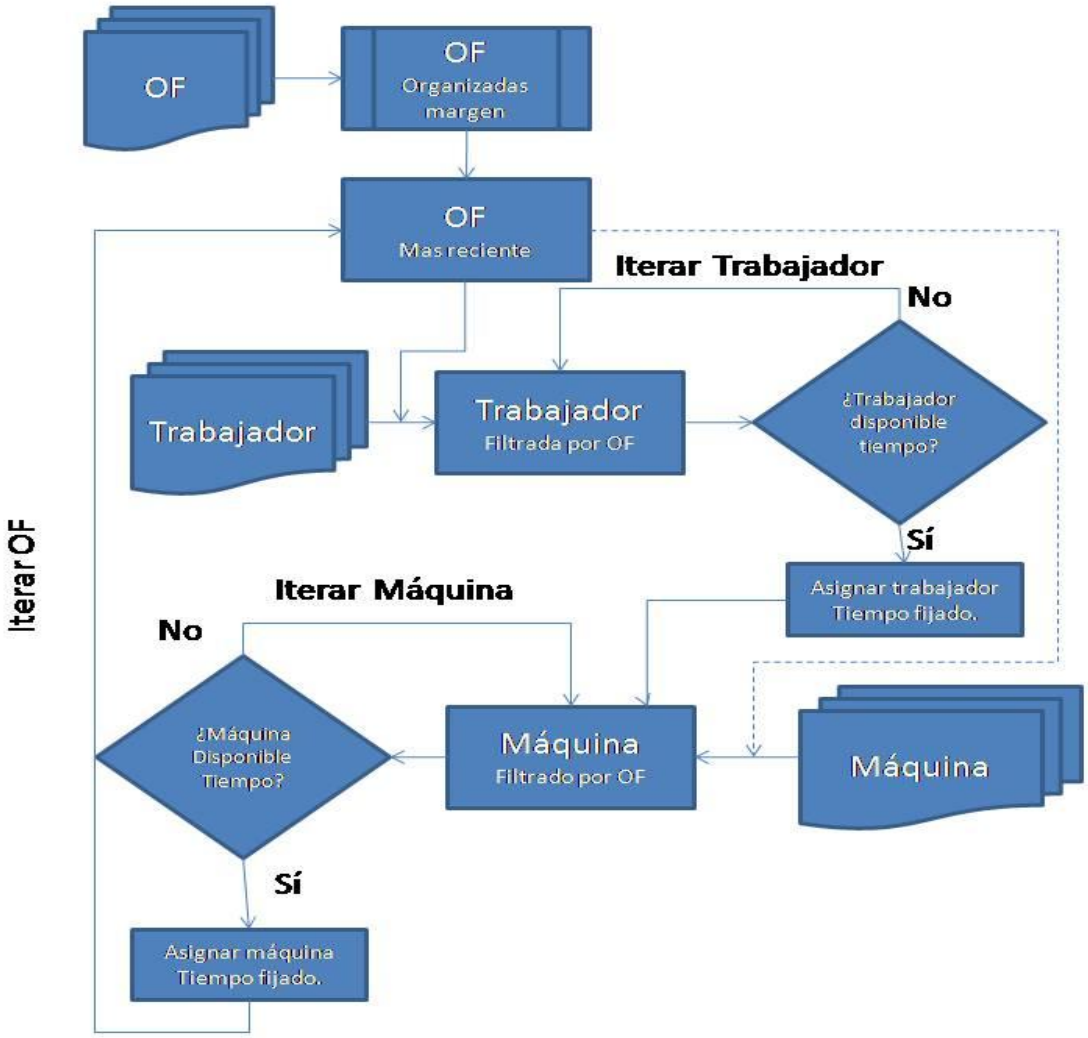


Diagrama de flujo del *KronosParallelOWM*.

2.2.3 KronosLinearMOW.

Este algoritmo tiene como punto de partida fijar una máquina y partir de ahí asignarle trabajo. Pero por tal de favorecer la preferencia temporal, las órdenes de fabricación son organizadas por margen. Se escoge la primera orden y con ella se escoge la máquina para fijarla, mirando si está disponible. Una vez fijada se busca un trabajador que atienda esta petición de fábrica con esa máquina (teniendo que estar libre de otras operaciones, en caso de no estarlo se descarta la petición y se itera a la siguiente). Se almacena la información en

la base de datos y si la máquina no ha superado el tiempo de planificación especificado al incrementar su situación temporal (su turno) se busca la siguiente orden que puede producir esta máquina con margen más corto. Si no tiene la capacidad de atender más peticiones se busca la siguiente máquina para asignar producción en ella aplicando el listado de órdenes de fabricación pendientes y realizando el indexado sobre máquina ya explicado desde un tiempo inicial.

En este caso el proceso **permite realizar la puesta en marcha de la maquinaria marcando un único arranque y una única parada** de ésta. Pasando a ser un factor secundario la mano de obra que girará alrededor de ésta. También se aprecia que lo más seguro es que dispongamos de una planificación desequilibrada (**mínimo balanceo**), asignando trabajo a la maquinaria menos polivalente y posiblemente con muy poca o ninguna carga a las más polivalentes. Siendo muy destacable la situación de encontrarnos con maquinaria parada y trabajadores sin realizar tareas productivas.

Para finalizar la explicación de este algoritmo destaco el hecho de suponer **mayor número de trabajadores que de maquinaria**, por tal de favorecer el flujo de ejecución de esta última.

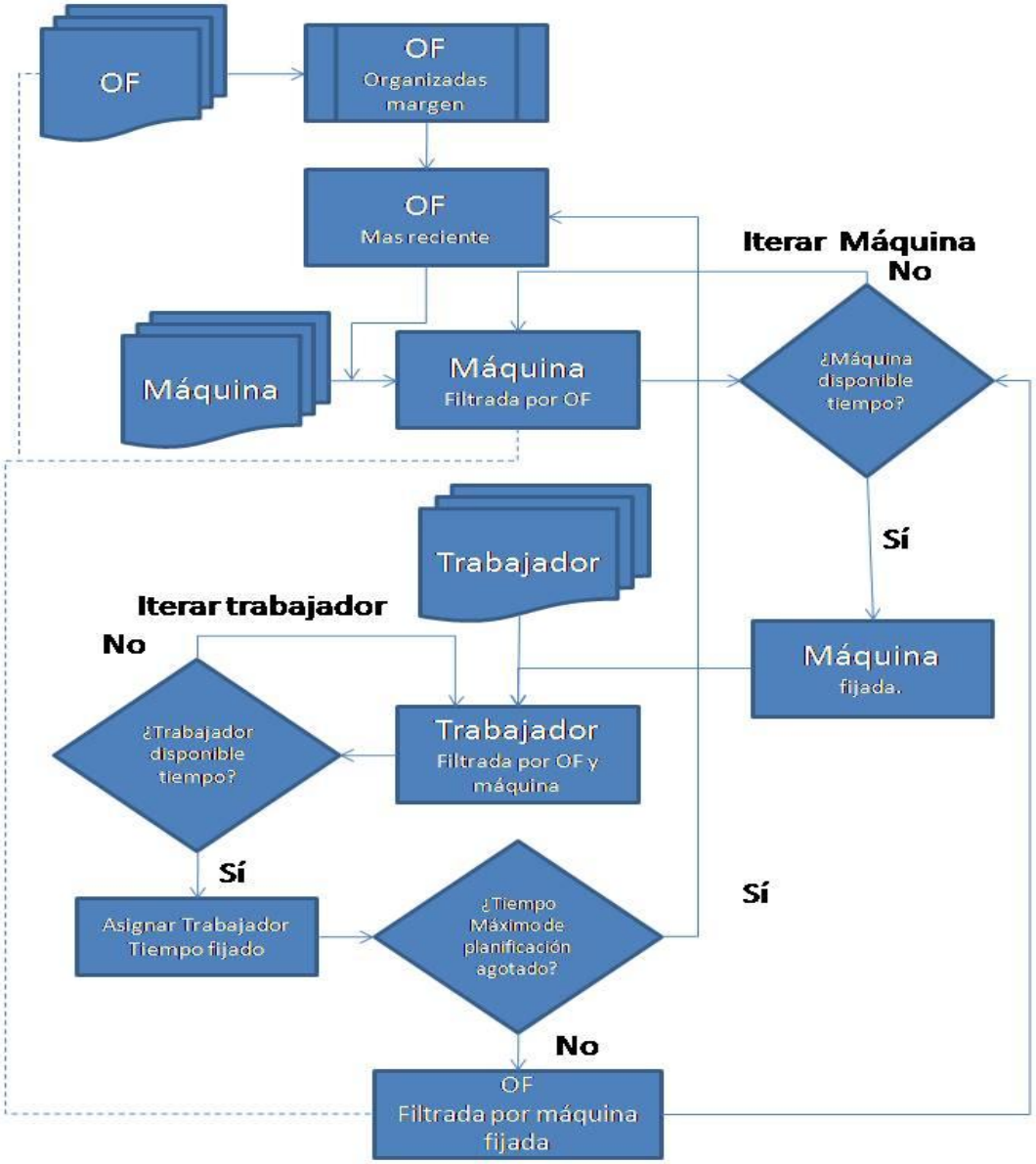


Diagrama de flujo del *KronosLinearMOW*.

2.2.4 KronosLinearWOM.

Como su nombre indica cambiamos el orden de pasos por tal de ver si responden de forma correcta. En esta heurística utilizamos como inicio la orden de fabricación más corta en margen y fijamos un trabajador. Asignamos el cronograma de órdenes de fabricación y máquinas que puede atender dicho trabajador teniendo en cuenta el margen de entrega. En caso de indisposición de la maquinaria la orden de fabricación quedará descartada. Cuando el tiempo del trabajador es agotado, turno agotado, iteramos a la siguiente orden de

fabricación no atendida, asignamos un nuevo trabajador y hacemos lo mismo que lo explicado anteriormente.

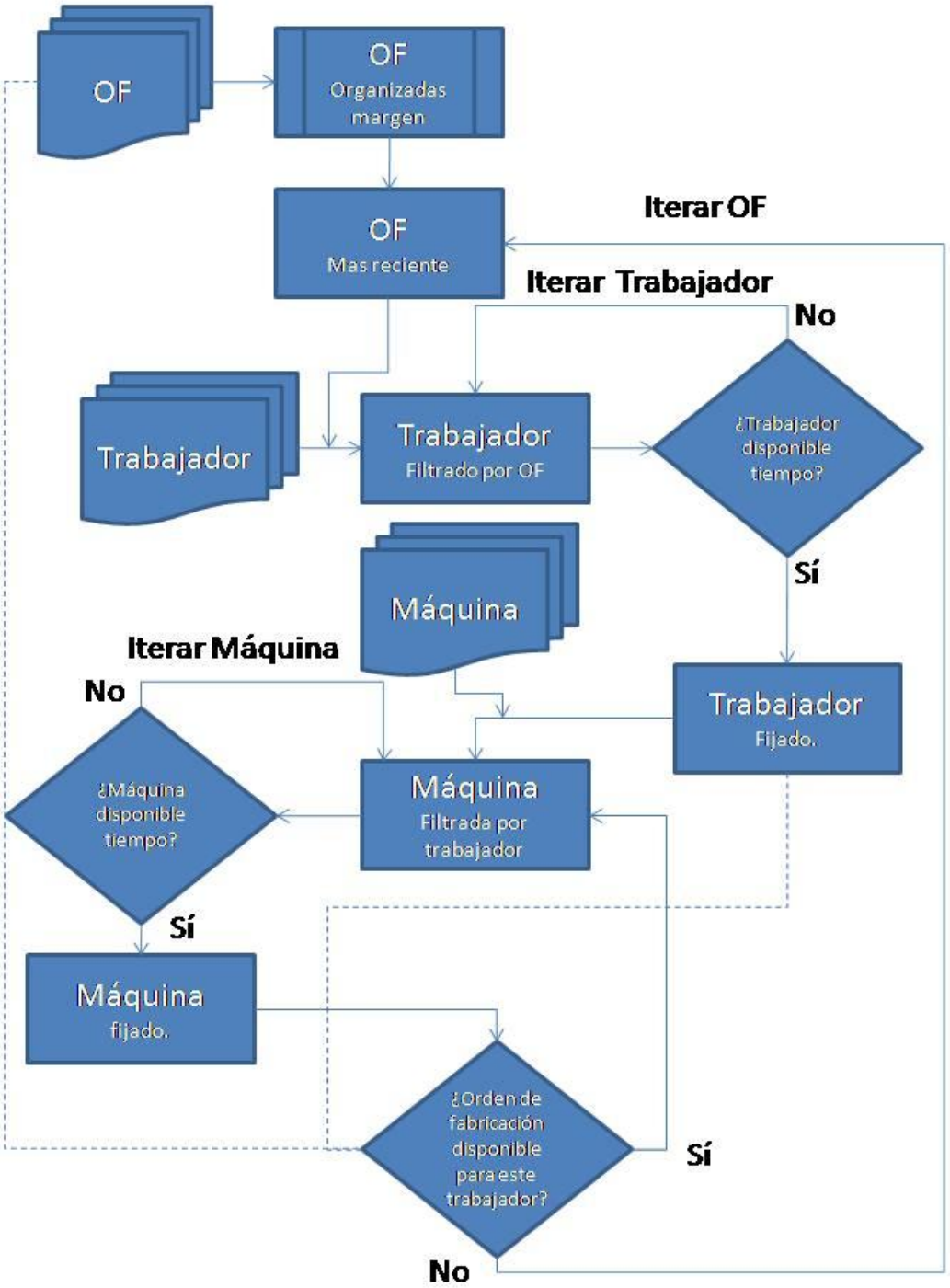


Diagrama de flujo del *KronosLinearWOM*.

Este proceso está concebido para potenciar al trabajador, nos permite realizar una **planificación temporal de su jornada laboral teniendo en cuenta su**

nivel ocupacional. Por el contrario **lo más seguro es la carencia de balanceo de carga que dispondrá**.

Como el proceso anterior se da preferencia al trabajador, disponiéndole mayor número de maquinaria y así aumentar la capacidad de producción de éste.

2.2.5 Contextualización de KronosReverseOMW y KronosReverseOWM.

Para poder explicar el algoritmo primero hay que explicar la motivación de esta heurística. Contextualizando, muchas empresas multinacionales, como en el sector del automóvil, minimizan el uso de almacenes de material o *stock*. Para asegurar el correcto funcionamiento de la cadena de montaje y eliminar las paradas por falta de material necesitan un stock de seguridad.

Estas heurísticas están orientadas a la minimización del tiempo de producto acabado en *stock* y así evitar sobredimensionar el espacio necesario para almacenar el material antes de su entrega a cliente.

2.2.6 *KronosReverseOMW*.

KronosReverseOMW tiene como objetivo básicamente organizar las órdenes de fabricación de la más lejana en entrega hasta la más cercana. Seleccionando la más distante, asigna que máquina puede aplicar dicha orden, bajo los criterios explicados anteriormente, y finalmente que trabajador. En caso de no encontrar una máquina disponible en tiempo busca la máquina disponible en tiempo negativo más cercana (asignando un tiempo con más margen del necesario) y la asigna para buscar un trabajador. Si no encuentra un trabajador disponible descarta la orden de fabricación. Una vez calculado el tiempo que necesita para atender la petición realiza una resta en tiempo. Del tiempo límite (de entrega) que hemos fijado para calcular la planificación calcula cuanto tiempo antes necesita para entregarla en la fecha indicada. Una vez asignada pasaremos a la siguiente orden de fabricación.

Tal y como en los algoritmos anteriores pasamos a realizar un mayor número de trabajadores que de maquinaria para facilitar la aceptación de las peticiones de producción.

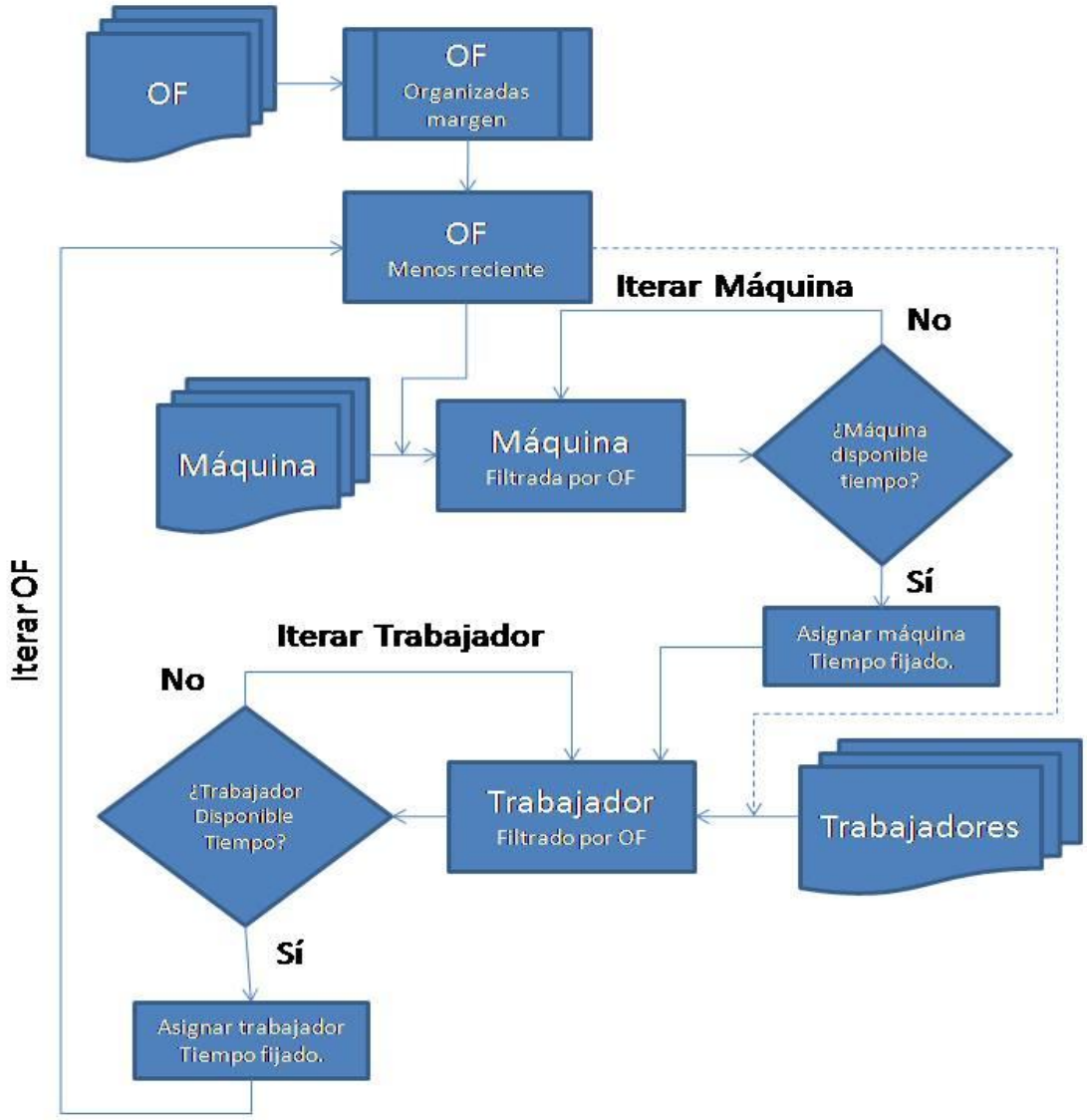


Diagrama de flujo del *KronosReverseOMW*.

Este algoritmo lo más probable es que las primeras ordenes de fabricación realice la planificación correctamente, pero las últimas se verán afectadas a nivel temporal siendo un **factor clave el margen de tiempo** que indiquemos para la planificación.

2.2.7 KronosReverseOWM.

Una vez explicado el anterior flujo, éste es una conmutación del anterior. Bajo la misma premisa realizamos la minimización del uso del almacén. Esta vez con la iniciativa de dimensionar primero sobre los trabajadores y luego sobre la maquinaria.

Realizamos la organización de las órdenes de fabricación por margen. Seleccionamos la primera de la lista y buscamos un trabajador que pueda atenderla. En caso de no encontrarla retrocede en el tiempo buscando el trabajador más cercano al tiempo actual. Si no está disponible descarta la orden, en caso afirmativo fija al trabajador y pasa a buscar una máquina. Si la encuentra la asigna y itera a la siguiente orden, si no la descarta y pasa a la siguiente orden de fabricación. Tal y como se muestra en el siguiente esquema.

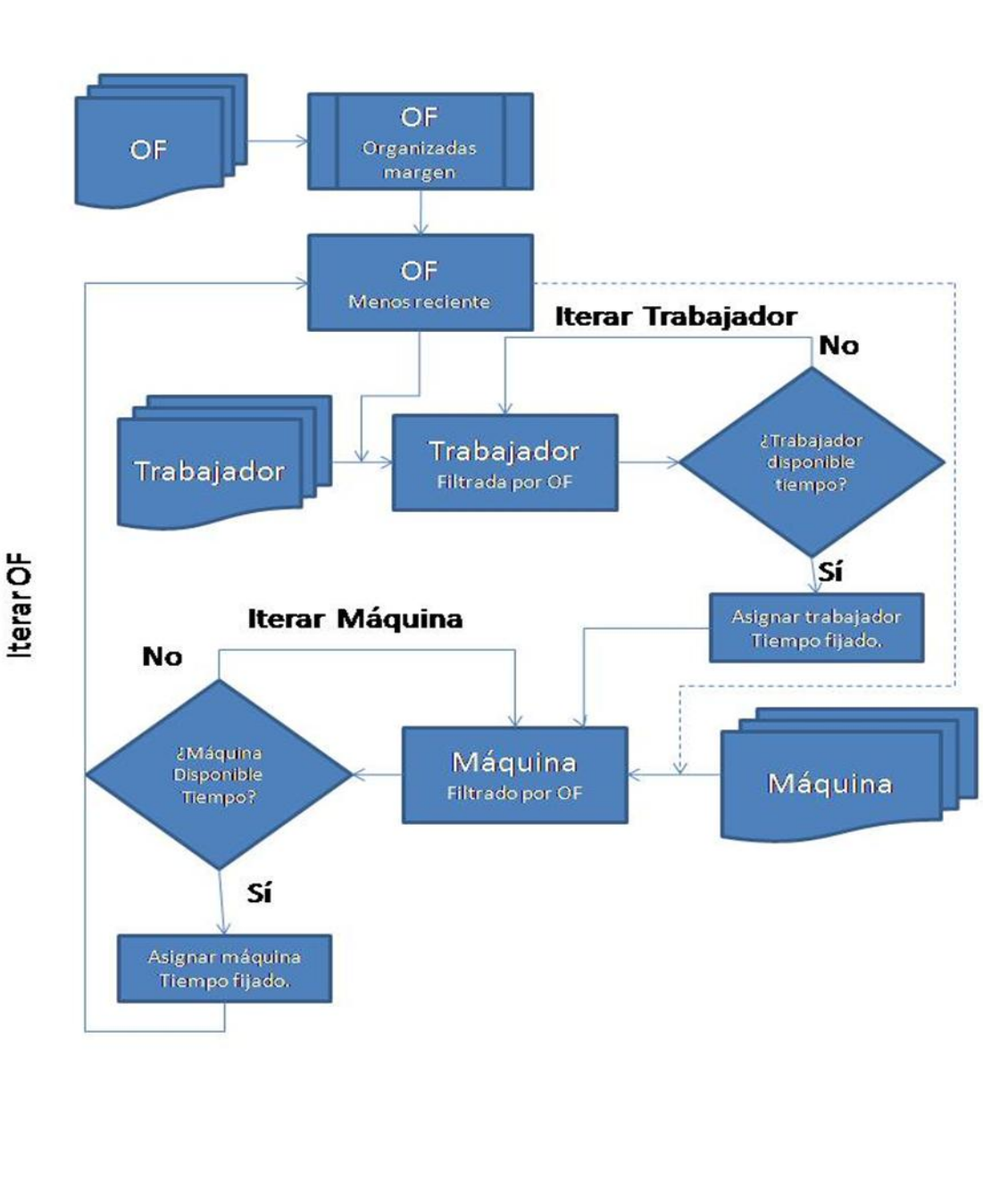


Diagrama de flujo del *KronosReverseOWM.*

Finalmente aquí tenemos mayor número de máquinas que de trabajadores para aumentar la posibilidad de realizar operaciones de producción por disposición de éstas.

3 Diseño.

3.1 La Base de Datos.

3.1.1 Requerimientos de la base de datos.

Tras tratar los diferentes algoritmos que vamos a utilizar en la aplicación podemos ver la información que podemos necesitar para llevarlos a cabo.

En el caso de los trabajadores podemos ver que nos hace falta:

- Nombre
- DNI o NIF
- Domicilio
- Teléfono/s
- Turno de trabajo.
- Catalogo de maquinaria que saben utilizar.
- Calendario laboral.
- Órdenes de fabricación que tienen asignadas.

En la maquinaria:

- Modelo de la máquina
- Conjunto de piezas que puede producir
- Tiempos de *setup* para poder producir dicha pieza.
- Tiempo de ciclo para cada una de las diferentes piezas.
- Fecha de revisión.
- Órdenes de fabricación que tiene vinculadas.

En las órdenes de fabricación.

- Propietario de la petición.

- Fecha de entrega.
- Pieza solicitada[1].
- Número de unidades.

3.1.2 Esquema Entidad-Relación de la base de datos.

Con esta reflexión vemos en el diagrama E-R que **el nexo de unión de todo es la entidad operaciones**. Ésta dispone del tiempo de inicio, tiempo de fin de la operación, pieza a producir, maquina que la realiza, trabajador que manipula y propietario de dicha petición.

El trabajador mantiene una relación con la operación antes comentada y a la vez utiliza la relación *Asignado_a* para la entidad Turno, con la que sabemos que turno tiene en una fecha determinada y que horario dispone en ese turno, en definitiva su disponibilidad laboral. Obviamente un turno puede pertenecer a varios trabajadores, del mismo modo que un trabajador puede disponer de un turno rotativo. Cosa que contempla la base de datos, al igual que acepta la flexibilidad laboral de cada uno.

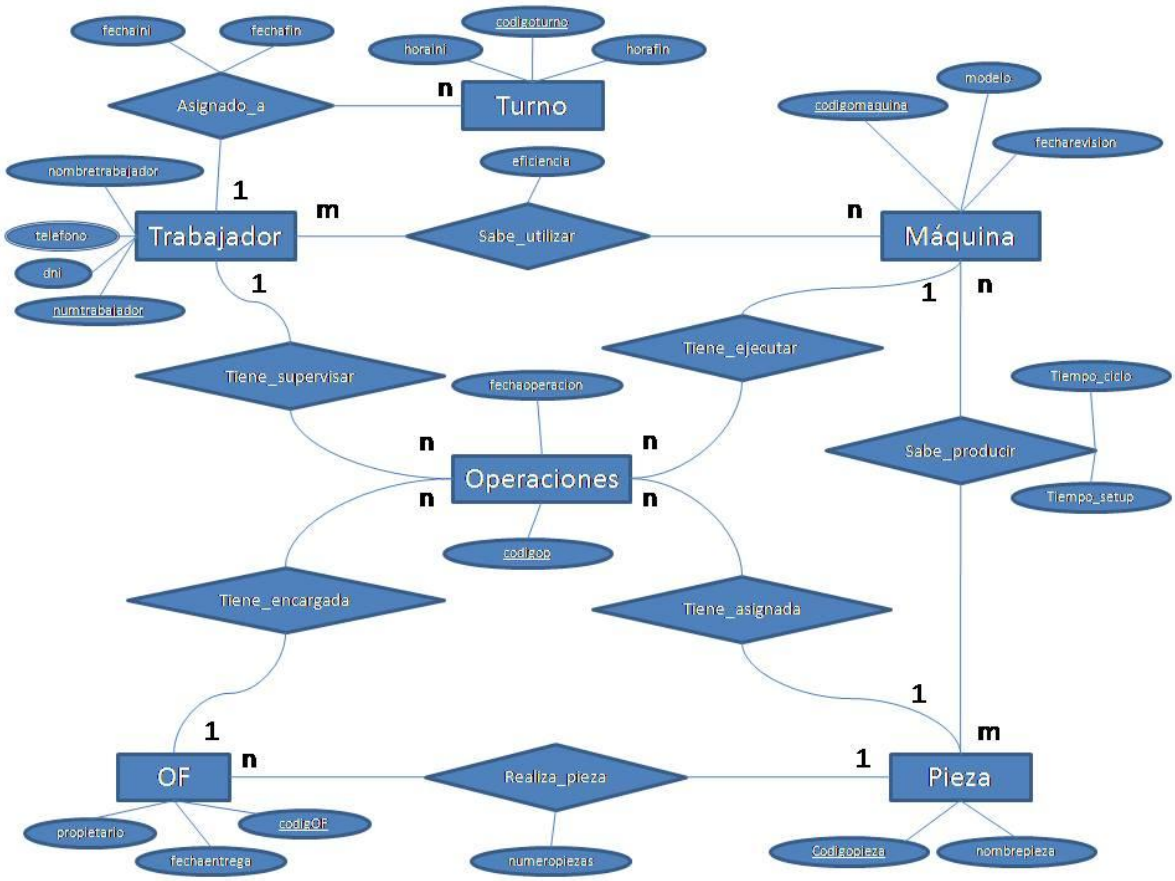
El trabajador dispone de una relación con *Sabe_utilizar*, la cual permite vincular que trabajador y como de bien sabe utilizar esa máquina/s. Ya que un trabajador puede saber utilizar diferentes tipos de máquinas.

La entidad máquina dispone de la información del modelo y fecha de revisión. Y se relaciona con la pieza utilizando la relación *Sabe_producir* para vincular el tiempo de ciclo y *setup* a cada tipo de pieza que sabe producir esa máquina.

Pieza como entidad dispone del nombre de la pieza a producir pero está vinculada por medio de *Sabe_producir* por el hecho de que una o unas máquinas pueden producir un tipo de pieza. Dependiendo del tipo de maquinaria que dispongan se puede dar el caso.

OF es la entidad que dispone de las fechas de entrega de las diferentes órdenes de fabricación y por medio de la entidad *realiza_pieza* podemos vincular que pieza se fabrica en una orden y que número de unidades solicitan.

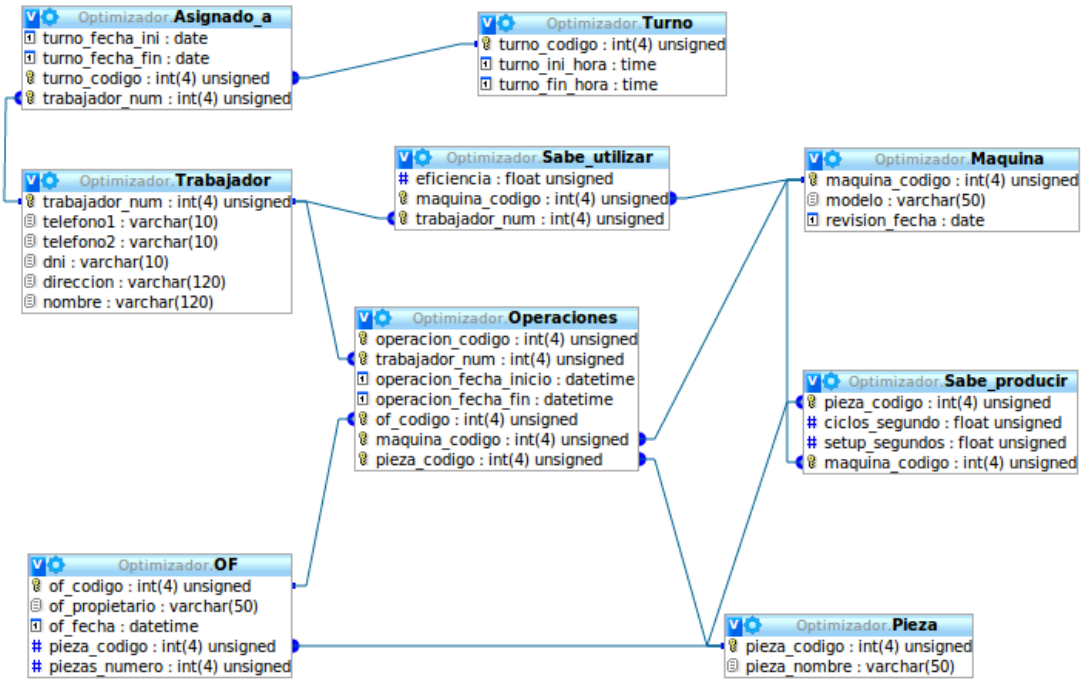
[1] Como aclaración destaco el hecho de por tal de no complicar inicialmente el problema se vincula una orden de fabricación a una única pieza.



Esquema E-R de la base de datos.

3.1.3 Modelo relacional de la base de datos.

Por tal de satisfacer las necesidades expuestas se ha aplicado reducciones al modelo entidad-relación y se ha obtenido:



Modelo relacional de la base de datos.

Las relaciones *tiene_supervisar*, *tiene_ejecutar*, *tiene_encargada*, *tiene_asignada* han sido eliminadas y llevado las claves primarias a la entidad Operaciones como clave foránea. Al igual que la relación *realiza_pieza*, donde su atributo *numeropiezas* ha pasado a la entidad OF. Y la clave primaria de Pieza ha sido agregada como clave externa a OF.

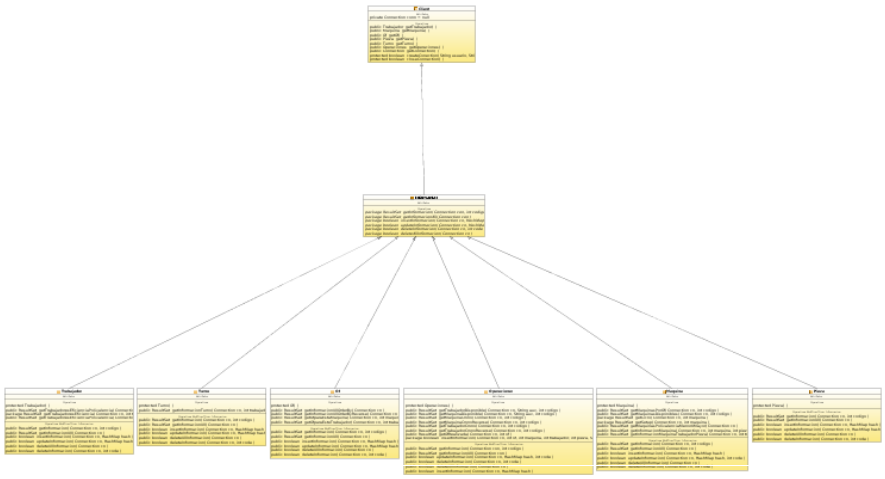
Se aprecia que el **objetivo de la base de datos era interrelacionar lo más óptimamente las operaciones con cada uno de los miembros** que llevan a cabo dicha tarea.

3.2 Aplicación.

La idea es realizar una aplicación que sea capaz de conectar con la base de datos y que permita realizar las consultas a ésta y a su vez sea lo más sencilla, al tratarse de una versión prototipo, y a la vez lo más escalable posible. Ya que los algoritmos anteriormente explicados pueden modificarse o agregar nuevos que ofrezcan mejores resultados.

3.2.1 Diagrama de clases.

Ésta dispondrá de la siguiente estructura: Dos *templates* donde dispondrán en uno el acceso a las diferentes tablas de la base de datos y el otro será el que organice las diferentes heurísticas. Estos dos unidos por un *Facade* que dispondrá de los recursos del segundo. Es decir, la idea es de disponer la instanciación del objeto *Cliente*, el cual dispondrá de los parámetros de conexión a la base de datos y a partir de él por herencia y por medio de la clase abstracta *Informacion* accedemos a las tablas de la base de datos. Ahí es donde tenemos una clase que accederá a cada una de las diferentes tablas. De este modo, por ejemplo, las consultas que principalmente trabaja con la tabla *Operaciones* serán contemplada s por la clase de su mismo nombre. Evitando así la aglomeración de consultas desorganizadas.



Por otro lado la clase *Facade* responde a la necesidad de utilizar una interface unificada a los diferentes algoritmos. Así lo que conseguimos es pasar el objeto *Cliente* con toda la indexación a la información necesaria para así sólo preocuparse de la secuencia de pasos a aplicar.

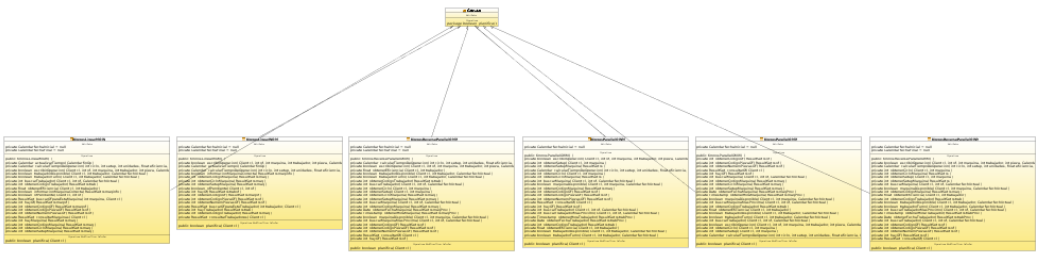


Diagrama parcial de clases del template *Calcula*

De esta manera accede a la clase *Calcula*, abstracta, que permite utilizar el método abstracto *planifica* y que instancia a alguna de las clases herederas.

Ahí la motivación de utilizar otro *Template* y unificar a nivel externo el acceso, aunque a nivel interno no realice los mismos pasos. Conseguimos una modulación de la aplicación que deja utilizar los seis métodos y contemplar modificación o incluir unas nuevas.

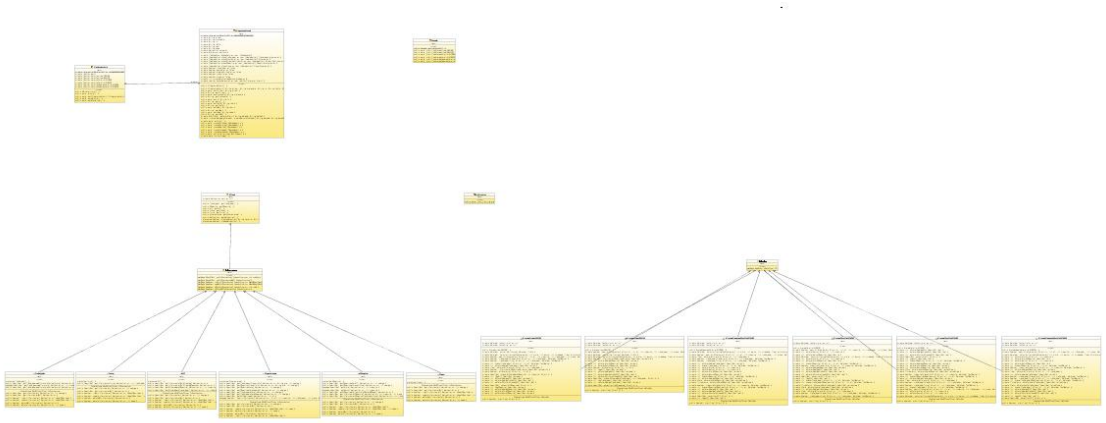
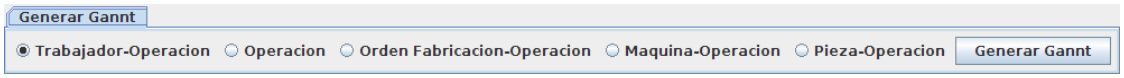


Diagrama de clases de toda la aplicación.

3.2.2 Interface Gráfica.

La interface es como versión prototipo, limitándonos a una relación de ventanas que ofrecen los seis tipos de cálculo diferente y un botón que accede a visualizar una nueva ventana para el diagrama de Gantt. Tal y como muestra la imagen tenemos una cabecera con el título y debajo los botones de algoritmos y visualización de diagrama.

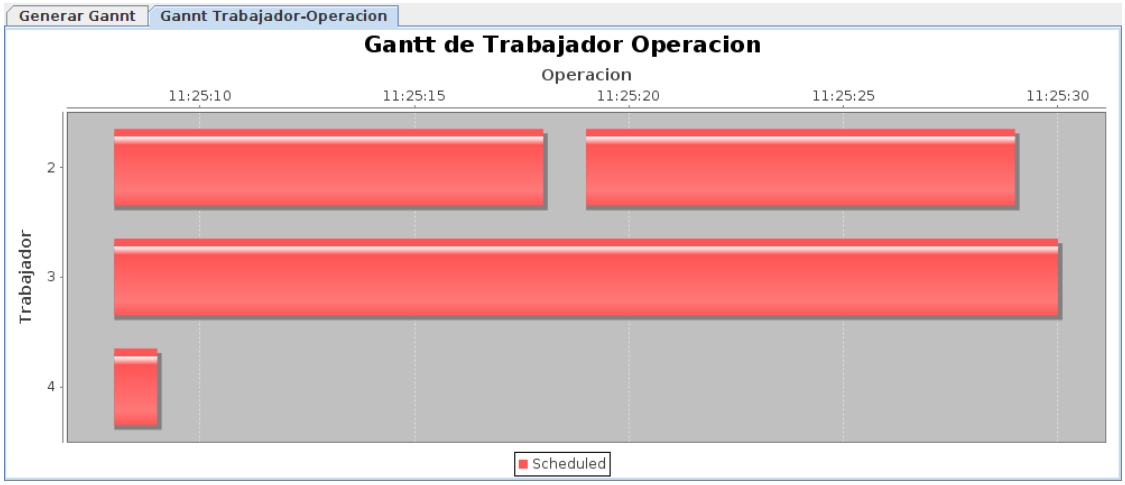
En el caso de pulsar cualquiera de los seis primeros botones no se aprecia ningún mensaje, en cambio, si se pulsa “Visualizar Gantt” vemos que aparece la siguiente ventana.



Ventana secundaria que accede a las diferentes combinaciones para visualizar el Gantt.

En ella podemos seleccionar entre las diferentes opciones de visualización. Para la generación del diagrama de Gantt he utilizado la librería *iText* que permite generar diagramas fácilmente. Las opciones es ver trabajador en ordenadas al igual que en Orden de fabricación, maquina, pieza y operación, y ver la línea de tiempo en las coordenadas.

Seleccionando uno de las opciones y pulsando el botón “Generar Gantt” obtenemos una nueva pestaña con los cálculos realizados antes de entrar en este menú contextual.



Ejemplo de diagrama de Gantt.

4 Caso de estudio.

4.1 Introducción.

Este caso de estudio está planteado como un ejemplo de funcionamiento y analizar resultados, pros y contras de cada una de las diferentes metaheurísticas. Destaco la imposibilidad de haber podido obtener datos de una empresa autentica y llevar a cabo una simulación basada en un caso de una planta real. Por ello, el caso de estudio se basa en una relación aleatoria entre la maquinaria con las piezas, la maquinaria y los trabajadores y la organización de las peticiones de producción.

Entendemos que habrá heurísticas que darán mejor respuesta, al converger con los valores de entrada y otras por el contrario, peor respuesta al tener condicionales que impiden todo su potencial[2].

También destaco la situación de este caso de estudio el **tener el mismo número de maquinas que de trabajadores**. He tomado esta decisión por tal de poner más dificultades a los algoritmos y así ver su comportamiento sin darle facilidades.

4.1.1 Comportamiento en caso de estudio del KronosParallelOMW.

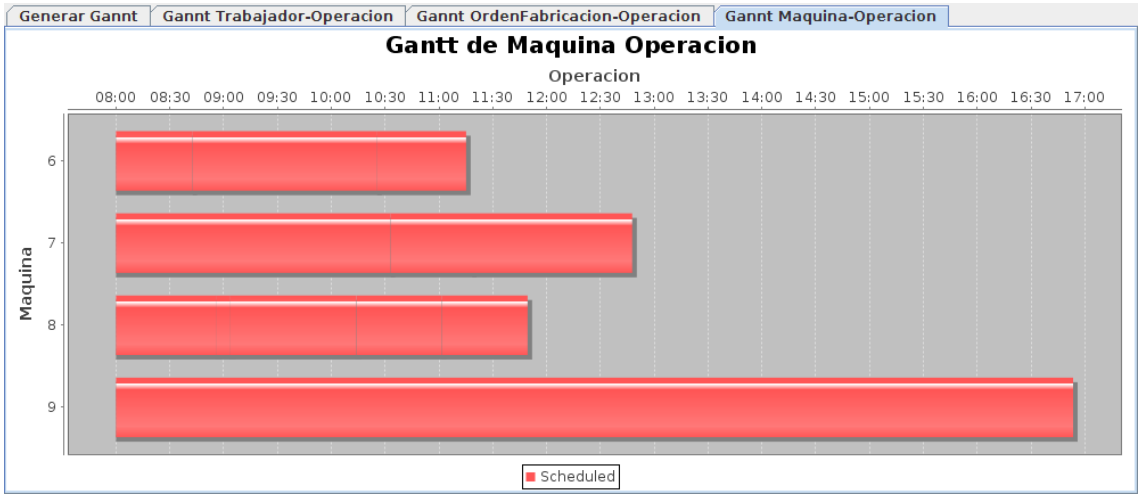


Diagrama del caso de estudio del KronosParallelOMW, máquina.

[2] En el *Anexo I: Contenido caso de estudio* está disponible la información de las tablas de la base de datos del caso de estudio.

La simulación[3] la hemos calculado con un turno de trabajo que empieza a las ocho de la mañana y termina a las cuatro horas (para facilitar el funcionamiento de las heurísticas). Se adjunta el diagrama de Gantt de las órdenes de trabajo para comprender mejor el nivel de efectividad.

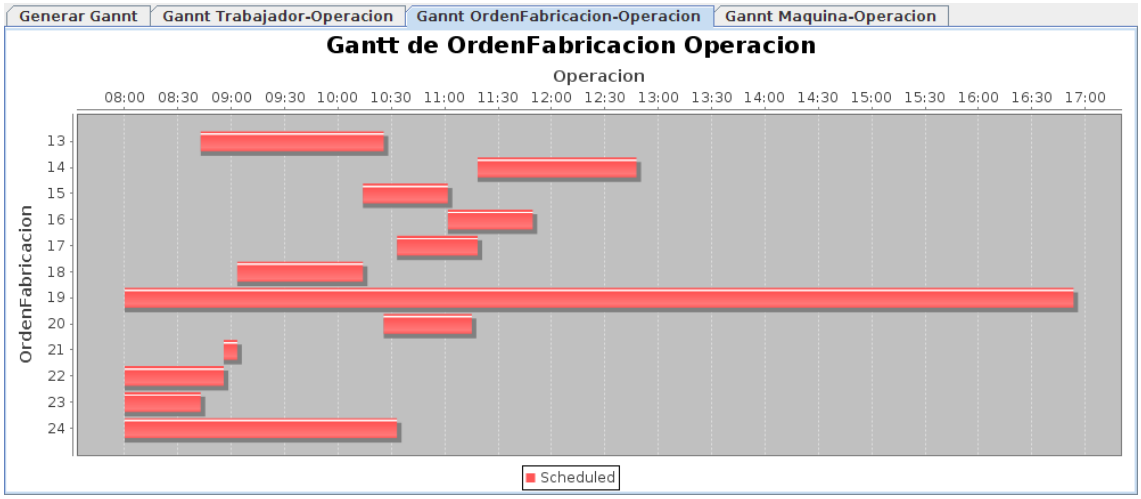


Diagrama del caso de estudio del KronosParallelOMW, orden de fabricación.

Es muy destacable que ha realizado una distribución aceptable desde el punto de vista de que cumple la **ejecución de todas las peticiones**. Concretamente atiende las doce, producidas dentro del turno de trabajo, claro está sin tener en cuenta el hecho de que la diecinueve pasa del rango de turno y su máquina y trabajador deben terminarla fuera de su horario.

Esta heurística nos permite confirmar la hipótesis que estimábamos sobre balanceo del trabajo. Nos encontramos ante una heurística que nos permite realizar una **distribución a nivel horizontal que aprovecha la capacidad disponible de toda la maquinaria**. Es ideal para el caso de pymes que necesiten priorizar la ocupación de su parque de máquinas frente a la MOD-mano de obra.

También consideramos que a pesar de balancear la carga también hemos encontrado que el filtraje de la maquinaria más trabajadores ha funcionado al 100%. Lo que nos permite ver un resultado muy satisfactorio.

[3] En el *Anexo I-a: Caso de estudio del KronosParallelOMW* podemos ver los detalles de la tabla Operaciones.

4.1.2 Comportamiento en caso de estudio del KronosParallelOWM.

En la metaheurística que preferencia el trabajador[4] sobre la máquina nos ha dado un resultado esperado. Si observamos la imagen que sigue a continuación podemos ver que continuamos con una distribución homogénea del trabajo.

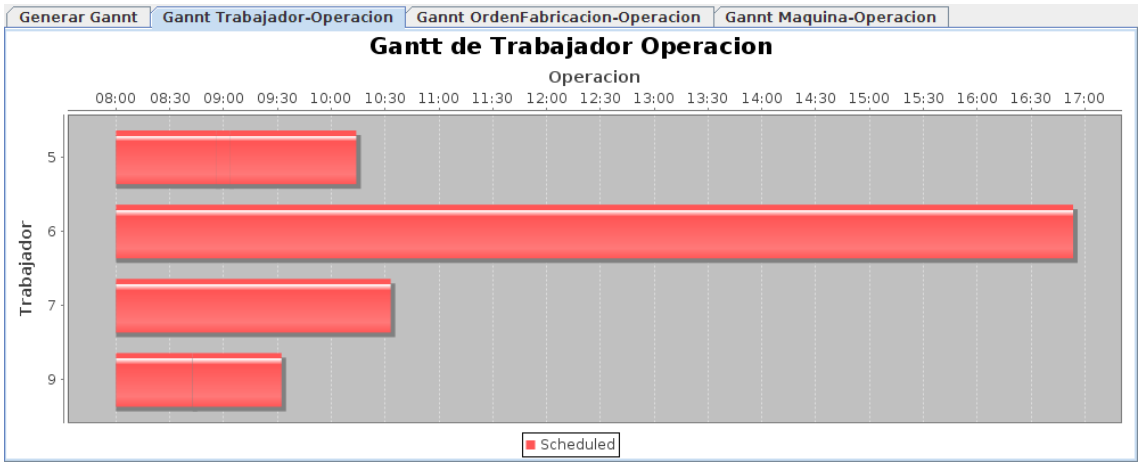


Diagrama del caso de estudio del KronosParallelOWM, trabajador.

Por el contrario el criterio de filtraje no ha sido favorecedor. Las órdenes de fabricación no han sido ejecutadas al completo. Más concretamente las ordenes trece a diecisiete han sido descartadas.

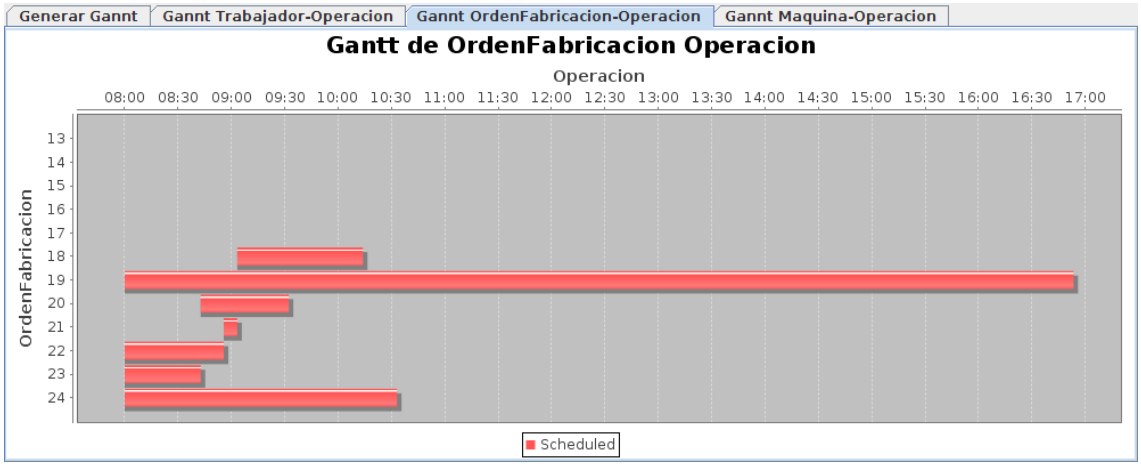


Diagrama del caso de estudio del KronosParallelOWM, orden de fabricación.

El motivo es muy sencillo, al ser las últimas en ejecutarse (organización por fecha) han sido penalizadas con los diferentes criterios de filtraje. Por lo que al

[4] En el *Anexo I-b: Caso de estudio del KronosParallelOWM* podemos ver los detalles de la tabla Operaciones.

avanzar en tiempo para encontrar un trabajador se ha encontrado pero una máquina no ha sido posible. En este caso los criterios de filtraje al ser intercambiados han dado una efectividad del 58´33%. Por el contrario y a pesar de la suposición del algoritmo anterior sigue habiendo homogeneidad en la distribución de la maquinaria, por lo que podemos deducir que si hay una distribución horizontal satisfactoria en un primer nivel de filtraje, lo habrá en un segundo nivel y que únicamente queda condicionado a la disponibilidad del segundo. Por lo que al trabajar de presente a futuro podemos considerar como **hipótesis de mejora la posibilidad de avanzar en tiempo si el segundo condicionante no está disponible ya que el primero seguirá disponible.**

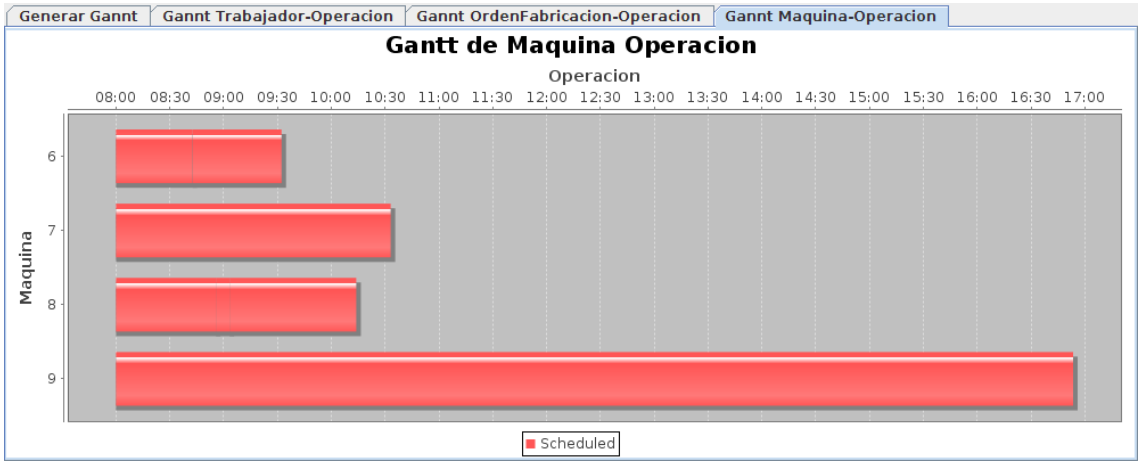


Diagrama del caso de estudio del KronosParallelOWM, máquina.

4.1.3 Comportamiento en caso de estudio del KronosLinearMOW.

Dentro de las suposiciones que he estado presentando cuando explicaba el funcionamiento de éste era el hecho de que beneficiaba a la máquina[5] al entrar en estado de trabajo y no parar hasta su finalización de producción. Esta idea queda confirmada con la siguiente imagen.

[5] En el *Anexo I-c: Caso de estudio del KronosLinearMOW* podemos ver los detalles de la tabla Operaciones.

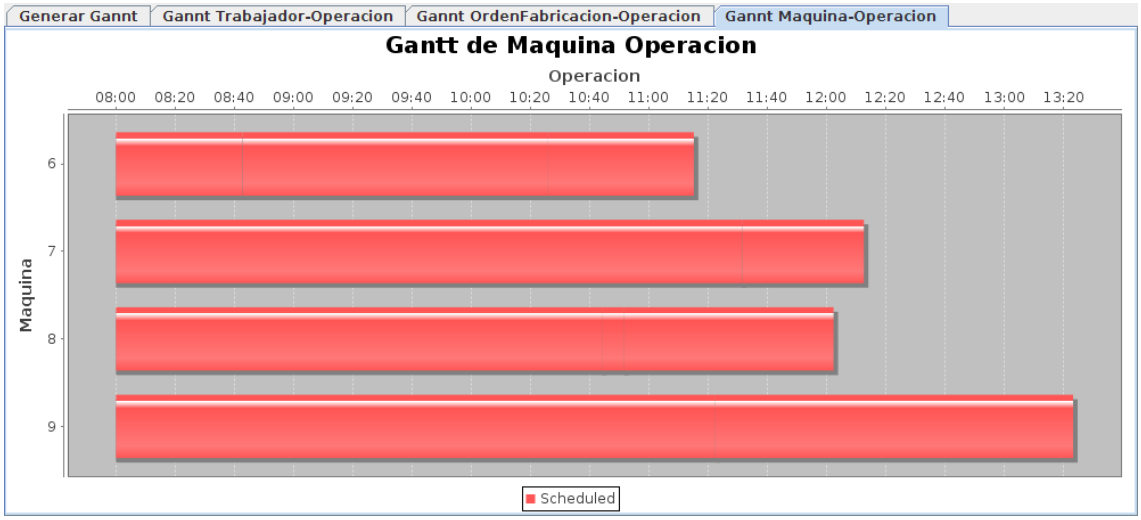


Diagrama del caso de estudio del KronosLinearMOW, máquina.

Disponemos en toda la planificación, pero esta situación es un tanto engañosa. Si observamos la distribución de órdenes de fabricación y trabajadores que están a continuación:

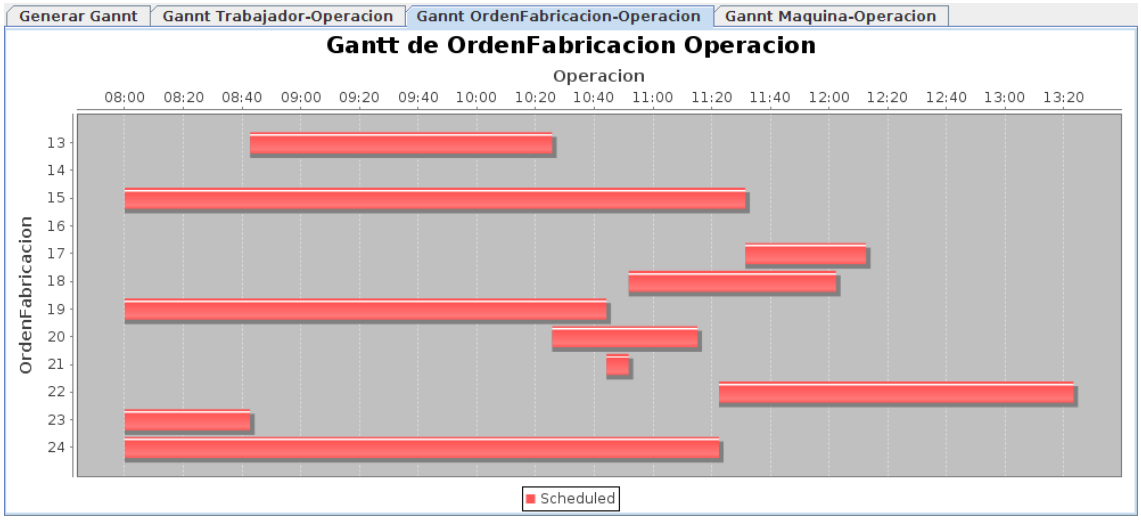


Diagrama del caso de estudio del KronosLinearMOW, orden de fabricación.

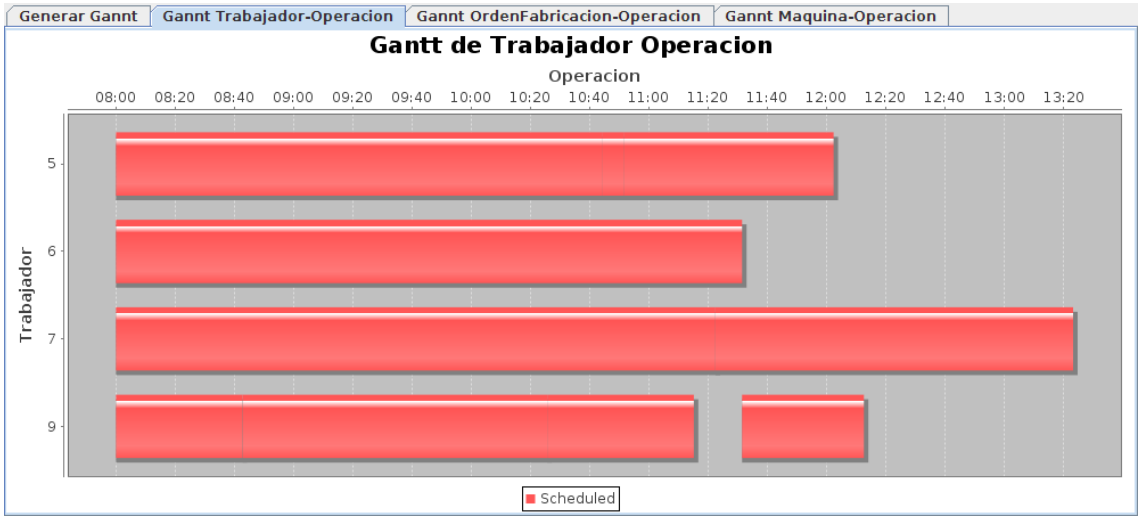


Diagrama del caso de estudio del KronosLinearMOW, trabajador.

Por un lado observamos que no cumple el porcentaje de éxito de atender todas las órdenes, reduciéndolo a un 83.33% de éxito en adjudicar.

Por el otro lado, si recordamos el turno indicado es de ocho de la mañana a doce del medio día. Por lo que la máquina que entra como primera en la selección es la seis, adjudicándose las órdenes veintitrés, trece y veinte. No sigue con adjudicación de trabajo al no poder atender más peticiones. Por lo que pasamos a la nueve, esta si ejemplifica el corto tiempo de turno al sobre pasar las doce del medio día. Por tanto los resultados aquí mostrados si verifican el hecho de que al entrar en ejecución una máquina empieza y no para hasta no finalizar todas sus tareas (véase como ejemplo la máquina seis) dentro del tiempo marcado. Pero por el contrario **muestra un balanceo de carga irreal al tratarse de un turno corto en tiempo**.

4.1.4 Comportamiento en caso de estudio del KronosLinearWOM.

En este caso podemos ilustrar los beneficios del criterio de preferencia de los trabajadores como la maquinaria[6].

[6] En el *Anexo I-d: Caso de estudio del KronosLinearWOM* podemos ver los detalles de la tabla Operaciones.

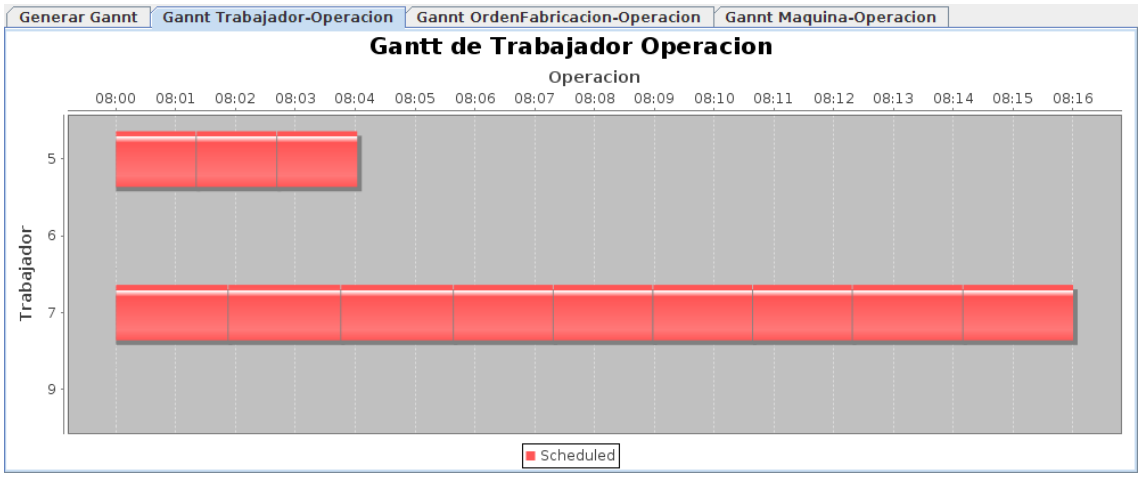


Diagrama del caso de estudio del KronosLinearWOM, trabajador.

El trabajador siete es el primero en entrar en juego al tratar la primera orden de fabricación. Éste juega constantemente con la máquina ocho, con la que tiene una gran eficiencia y junto a la máquina, realiza todo el trabajo que puede en un tiempo muy bajo. Una vez agotado itera sobre el trabajador cinco que realiza el trabajo restante.

Aquí se **ilustra el efecto de no balanceo de carga** y el hecho de que al trabajar con el menos polivalente pero más efectivo ha llevado, por casualidad, a unos tiempos inferiores del algoritmo “hermano”.

Evidentemente y al igual que en el caso del *parallel* existe la simetría en ambos niveles de condicional. Por ello nos encontramos con una distribución equivalente a nivel de maquinaria.

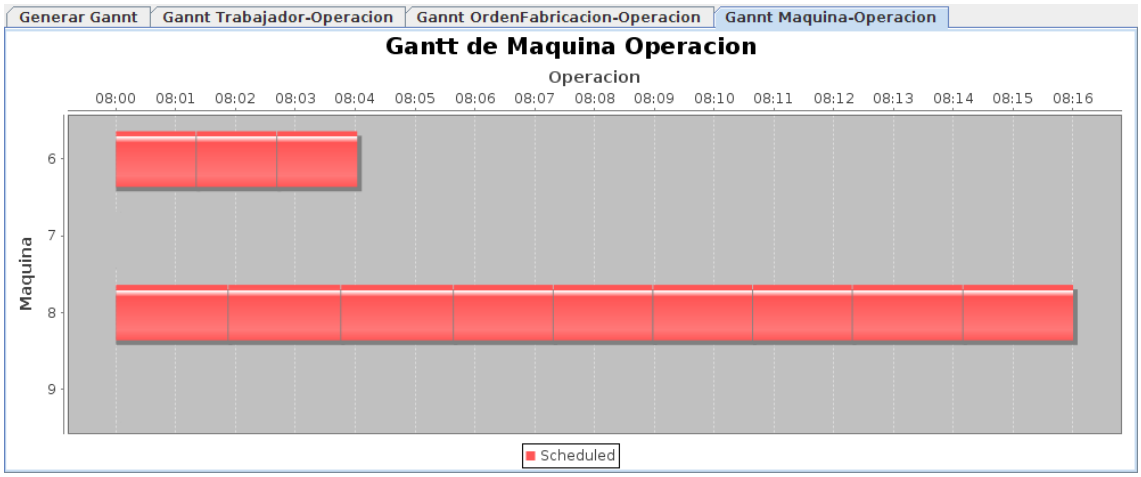


Diagrama del caso de estudio del KronosLinearWOM, máquina.

El nivel de eficiencia en asignación de órdenes de fabricación es de un 100% al atenderlas todas, cosa muy razonable al encontrarnos en un ámbito de recursos sin realizar ninguna tarea.

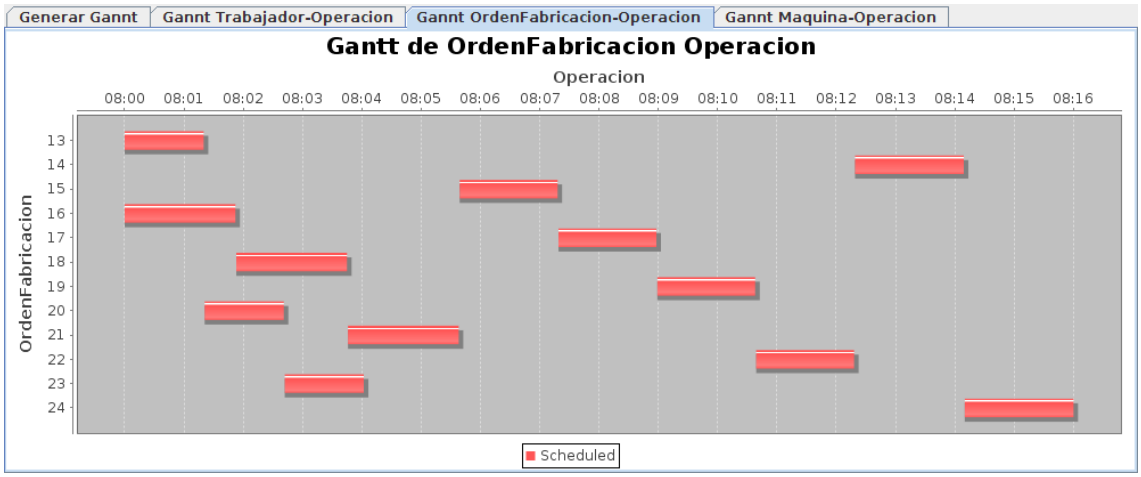


Diagrama del caso de estudio del KronosLinearWOM, orden de fabricación.

4.1.5 Comportamiento en caso de estudio del KronosReverseOMW.

Este caso es muy interesante al tratar de minimizar utilizando la filosofía de *Just In Time*. Toda ella radica en el factor tiempo, para que quede más claro pasemos al caso de estudio[7].

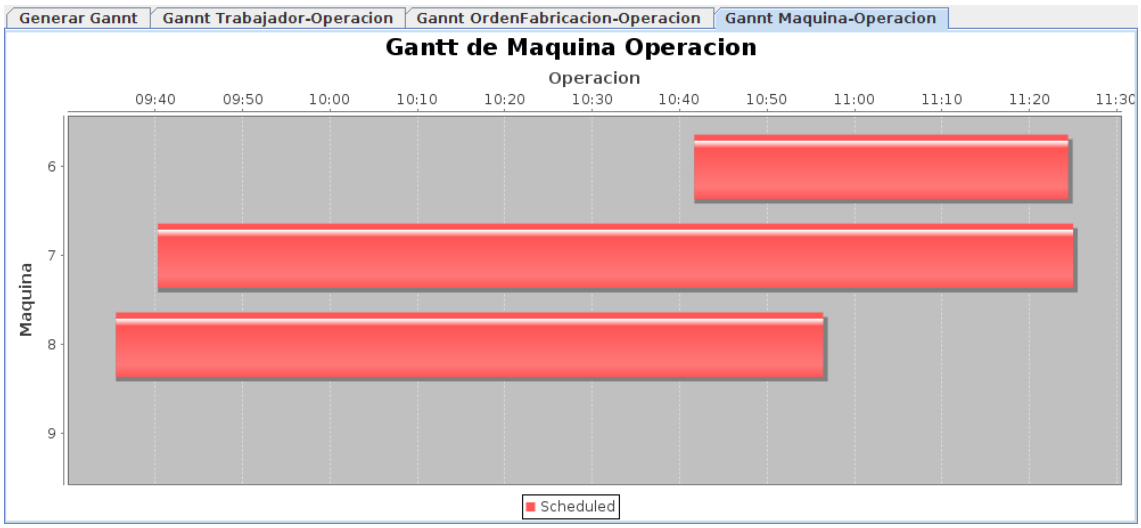


Diagrama del caso de estudio del KronosReverseOMW, máquina.

Como vemos hay una asimetría total en la asignación de maquinaria que se extiende a trabajadores.

[7] En el *Anexo I-e: Caso de estudio del KronosReverseOMW* podemos ver los detalles de la tabla Operaciones.

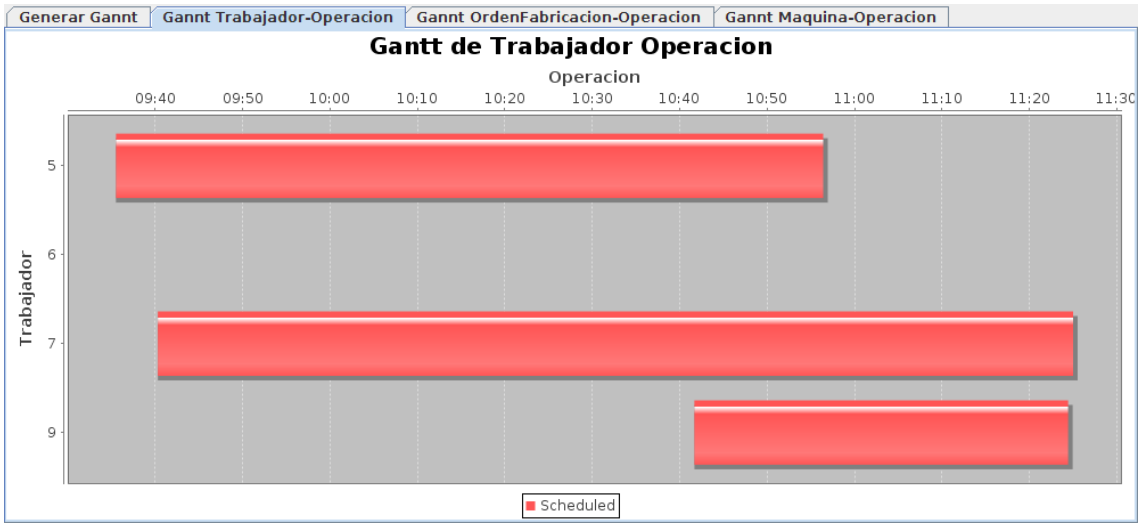


Diagrama del caso de estudio del KronosReverseOMW, trabajador.

Pero lo mas negativo son las ordenes de fabricación, el índice de éxito es muy bajo (hablamos de un 25% de éxito). Con lo que vemos que hay algún error en la planificación. La asimetría que disponemos al **basarse en las fechas de entrega condiciona al máximo el margen de planificación** y si recordamos el hecho de disponer de cuatro horas para planificar condiciona mucho dentro de este marco.

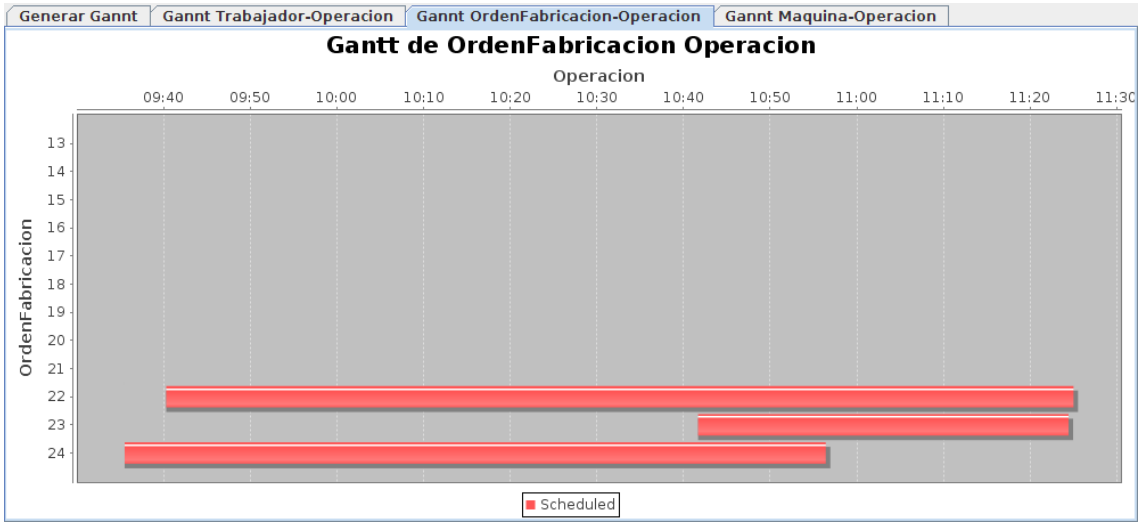


Diagrama del caso de estudio del KronosReverseOMW, orden de fabricación.

4.1.6 Comportamiento en caso de estudio del KronosReverseOWM.

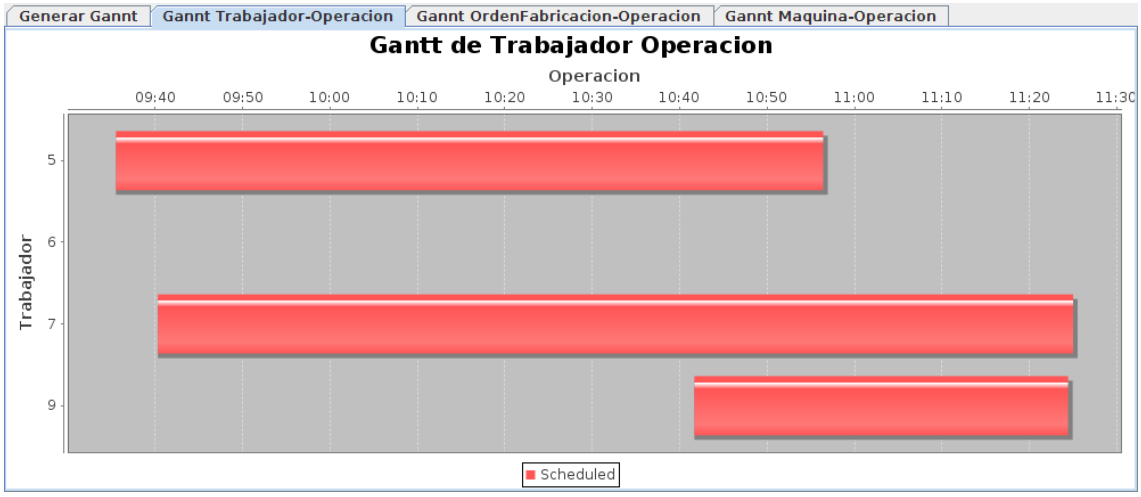


Diagrama del caso de estudio del KronosReverseOWM, trabajador.

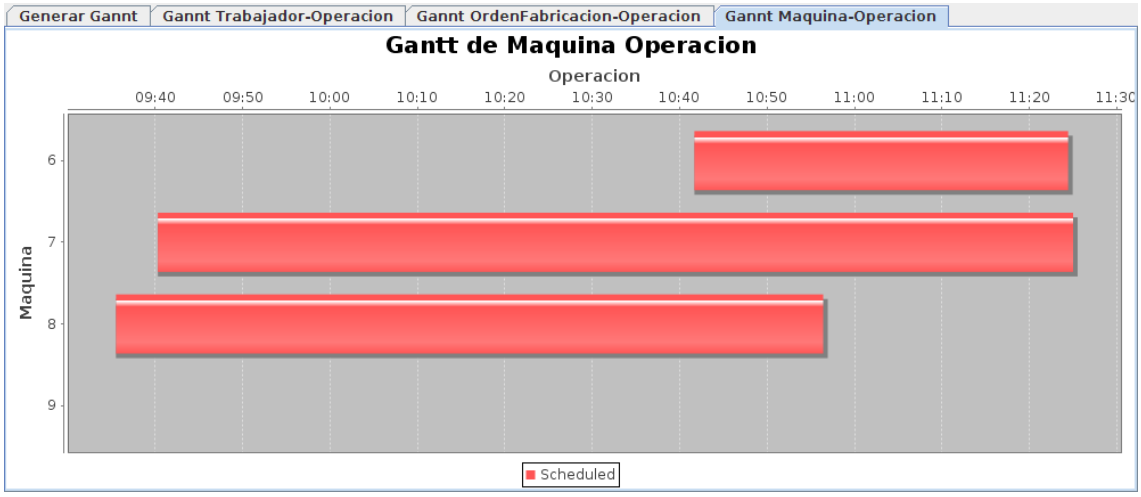


Diagrama del caso de estudio del KronosReverseOWM, maquina.

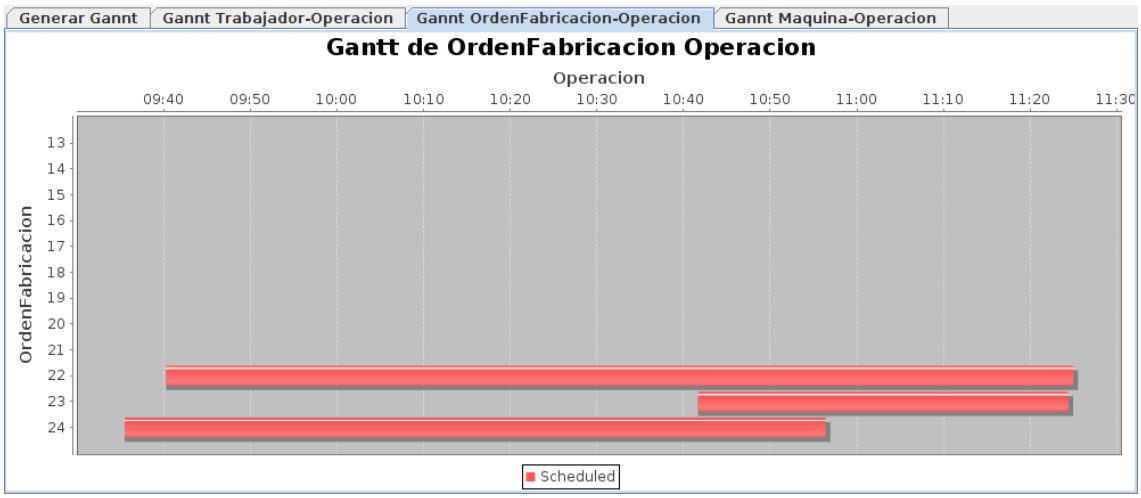


Diagrama del caso de estudio del KronosReverseOWM, orden de fabricación.

[8]Nos encontramos ante una situación idéntica a la anterior. Ahora sí que es el momento de empezar a cuestionarse del por qué de la ineficiencia. A modo de ilustración modificaremos el turno de trabajo a ocho horas. Para ver qué resultados nos da y así confirmar la teoría de que esta heurística está condicionada por el turno.

4.1.6.1 *Caso de estudio del KronosReverseOWM con turno de ocho horas.*

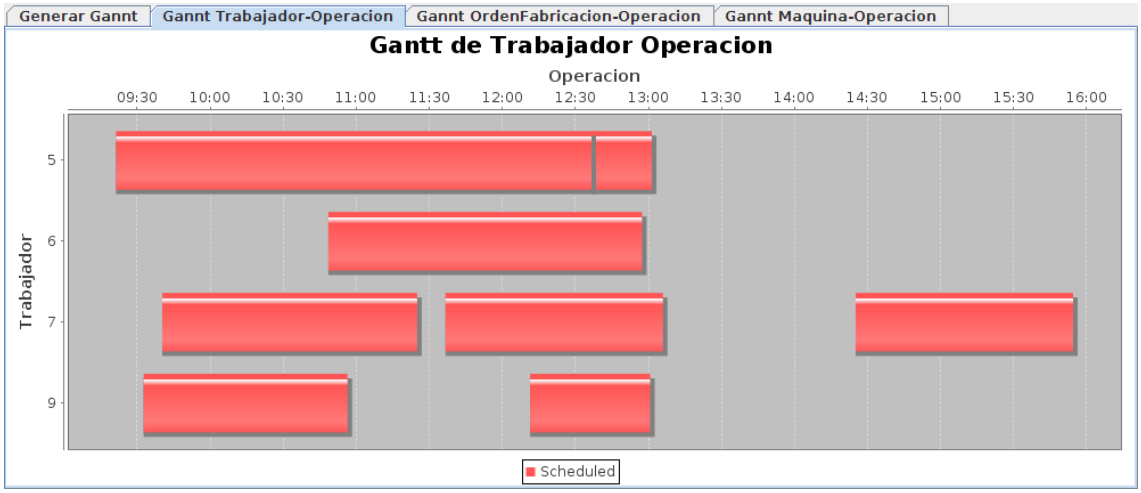


Diagrama del caso de estudio del KronosReverseOWM, trabajador con ampliación de turno.

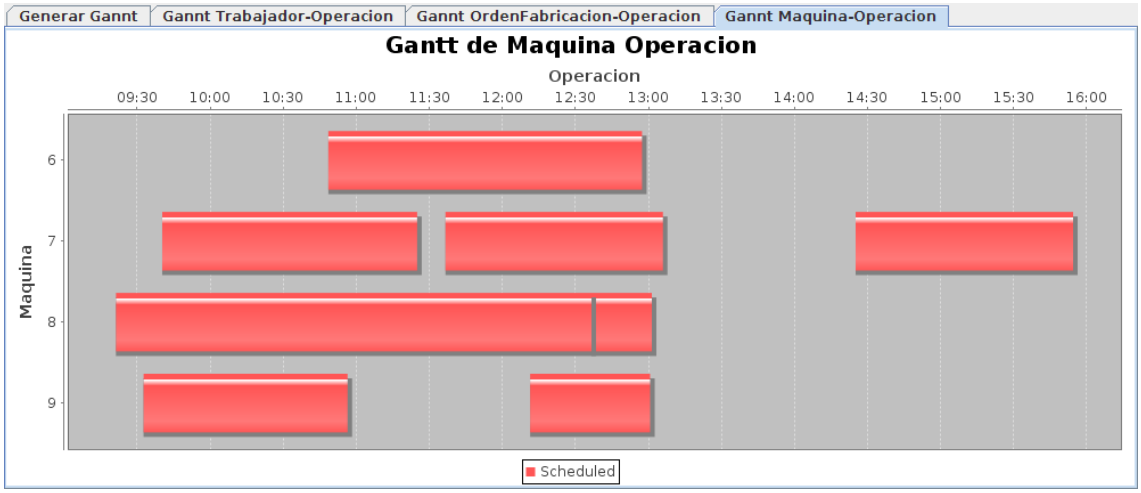


Diagrama del caso de estudio del KronosReverseOWM, máquina con ampliación de turno.

[8] En el *Anexo I-f: Caso de estudio del KronosReverseOWM* podemos ver los detalles de la tabla Operaciones.

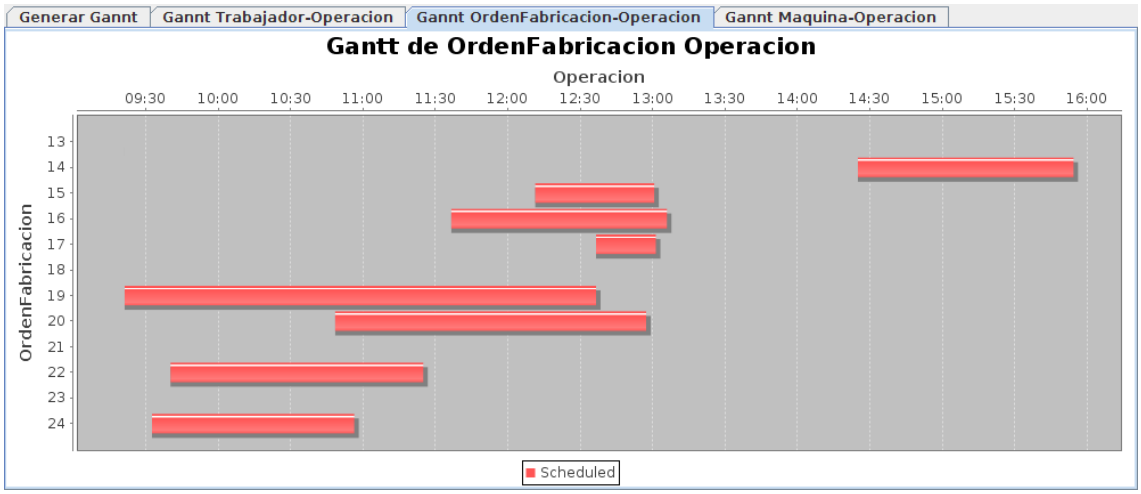


Diagrama del caso de estudio del KronosReverseOWM, orden de fabricación con ampliación de turno.

Las imágenes demuestran que los razonamientos eran correctos, en este caso el límite condicionante era el marco de entrega inferior. Siendo órdenes de fabricación que se encontraban en su mayoría fuera del alcance del turno. Aunque si nos encontramos en el caso contrario el limite sería el superior. Por ello el **factor clave de los algoritmos *KronosReverse* es la amplitud del turno con el que trabajan para poder llevar a cabo la planificación**[9].

[9] En el *Anexo I-g: Caso de estudio del KronosReverseOMW* podemos ver los detalles de la tabla Operaciones.

5 Conclusiones.

Ya desde un óptica más realista y valorando todas las simulaciones aquí explicadas y antes de profundizar en las conclusiones, comento la idea de que **no existe** la mejor metaheurística. Según la tipología de producción y su naturaleza, es mas eficiente una u otra.

Si hablamos de ***KronosParallelOMW,*** es recomendable para una planta productiva con alta inversión en maquinaria y por tanto, que le interese tener el máximo de ocupación y minimizar las paradas (set-up, mantenimientos) para su mejor amortización y satisfacer picos de demanda mediante mano de obra.

Un ejemplo sería un centro de mecanizado, donde la inversión en maquinaria supera a la mano de obra.

KronosParallelOWM se enfoca en la mano de obra, de esta manera lo que ocurre es que nos encontramos ante un personal mucho más valioso, por salarios, nivel de formación frente a la maquinaria. Un ejemplo podría ser un taller de confección, con personal cualificado y sencillas máquinas de coser.

KronosLinealMOW es una heurística orientada a procesos industriales que requieran grandes periodos de tiempo de puesta a punto. Un ejemplo sería un horno industrial de una planta para la fundición. Nos interesa que éste este en marcha durante todo el periodo de trabajo ya que apagarlo para volverlo a arrancar tiene un alto coste energético y por tanto monetario (set-up).

KronosLinearWOM se orienta en los casos en que las tareas de producción de los trabajadores tienen agruparse lo máximo posible para que los tiempos “muertos” sean lo mas continuos posibles y poder asignarles otras tareas no productivas.

Por ejemplo, una planta de esterilización de material clínico. En el que los trabajadores tienen que minimizar las entradas y salidas de la zona de empaquetado y esterilización por lo que una vez dentro tienen que aprovechar su estancia. Y que frente a la inversión de la maquinaria le interesa más al responsable minimizar el flujo de entrada y salida de trabajadores en la zona de producción.

KronosReverseMOW este es el algoritmo que se aproxima más al concepto de proveedor de una multinacional. En el que le interesa minimizar el *stock* y a la vez realizar la producción en el tiempo más próximo a la entrega. Por ello un ejemplo sería una planta de corte de chapa para fabricante de coches. Donde el capital humano puede ser ampliado fácilmente frente a picos de la demanda y que es despreciable en comparativa al valor de la maquinaria.

Para finalizar ***KronosReverseWOM*** sería también idóneo en procesos productivos donde el capital humano es lo más importante

6 Futuras líneas de trabajo.

En todas las metaheurísticas nos hemos encontrado con que las cotas temporales es un problema que debe ser solucionado. Tanto en los *KronosParallel*, *KronosLinear* y *KronosReverse*, tenemos el margen del final de turno como condicionante de los dos primeros y el inicio del último.

También una solución que mejore el balanceo de carga de trabajo, un equilibrio de carga favoreciendo unas máquinas o trabajadores y penalizando a otras por diversos motivos.

En el caso del *KronosReverse* limitar como cota inferior el inicio del turno y cualquier orden de fabricación que requiera más tiempo se debería fraccionar en dos o más partes para asignar a otra máquina que pueda atender la petición y su respectivo operario, y permitiría distribuir mejor la tarea. Aunque evidentemente habría que realizar los cálculos de número de piezas producidas en los diferentes intervalos para cuadrar las unidades, todo ello condicionado a los tiempos de ciclo y eficiencia de la máquina y trabajador, respectivamente.

Otra futura línea de trabajo seria mejorar la optimización mediante la fragmentación de la carga de trabajo.

Un aspecto que también podría mejorarse sería perfeccionar el concepto de filtraje de las órdenes de trabajo. El criterio aquí utilizado es un tanto rudimentario. Se podría mejorar dicho criterio aplicando como idea el hecho de utilizar las fechas de entrega y el número de piezas a producir, favoreciendo aquellas que tuvieran mayor número de unidades.

Otro aspecto seria el introducir el concepto de descansos y turnos rotatorios de los trabajadores. Donde el primero de ellos ya estaría beneficiado por los criterios de selección de maquinaria y trabajador aquí expuestos.

7 Anexo I: Contenido caso de estudio.

A continuación se especifica el contenido en la base de datos para así poder tener una idea exacta de la información referida en cada una de las tablas. Así como las órdenes de fabricación indicadas.

- Turno.

turno_codigo	turno_ini_hora	turno_fin_hora
2	08:00:00	12:00:00

- Asignado_a.

turno_fecha_ini	turno_fecha_fin	turno_codigo	trabajador_num
2010-09-07	2010-09-07	2	5
2010-09-07	2010-09-07	2	6
2010-09-07	2010-09-07	2	7
2010-09-07	2010-09-07	2	9

- Trabajador

trabajador_num	telefono1	telefono2	dni	direccion	nombre
5	932324567	666090909	55325633R	C/ Guadiana n 12, 3,4	Mateo Sanchez
6	971332255	659327939	12326644U	Av. Juan n 321 6, 4	Marcos Ibañez
7	915457893	688110043	46359833Q	C/ Provença n 222, 3, 6	Lucas Suarez
9	9831358123	678912345	09876554A	C/ Mallorca n 123, 4, 5	Juan Jimenez

- Sabe_utilizar.

eficiencia	maquina_codigo	trabajador_num
0,55	6	5
0,9	6	6
0,5	7	5
0,87	7	6
0,34	7	9
0,96	8	6
0,66	8	7
0,56	8	9
0,78	9	6

- Máquina.

maquina_codigo	modelo	revision_fecha
6	Hace platos.	2010-08-24
7	Hace tenedores.	2010-08-24
8	Hace cuchillos.	2010-08-24
9	Hace cucharas	2010-08-24

- Sabe_producir.

pieza_codigo	ciclos_segundo	setup_segundos	maquina_codigo
6	34,8	80	6
6	40,5	78	7
7	30	95	7
7	25,3	112	8
7	42,3	101	9
8	70,9	87	7
8	21,3	99	8
8	46,3	111	9
9	35,4	92	7
9	37,2	110	8
9	53,4	81	9

- Pieza.

pieza_codigo	pieza_nombre
6	plato
7	tenedor
8	cuchara
9	cuchillo

- OF.

of_codigo	of_propietario	of_fecha	pieza_codigo	piezas_numero
13	Propietario A	2010-09-07 12:54:06	6	527
14	Propietario B	2010-09-07 15:54:24	9	200
15	Propietario C	2010-09-07 13:00:43	8	200
16	Propietario D	2010-09-07 13:05:57	7	200
17	Propietario F	2010-09-07 13:01:23	8	100
18	Propietario E	2010-09-07 12:59:50	7	300
19	Propietario G	2010-09-07 12:56:12	8	844
20	Propietario H	2010-09-07 12:57:23	6	250
21	Propietario I	2010-09-07 12:58:38	7	25
22	Propietario J	2010-09-07 11:25:00	8	235
23	Propietario K	2010-09-07 11:24:26	6	215
24	Propietario L	2010-09-07 10:56:25	9	345

7.1.1 Anexo I-a: Caso de estudio *KronosParallelOMW.*

operacion_codigo	trabajador_num	operacion_fecha_inicio	operacion_fecha_fin	of_codigo	maquina_codigo	pieza_codigo
1	7	2010-09-07 08:00:00	2010-09-07 10:33:06	24	7	9
2	9	2010-09-07 08:00:00	2010-09-07 08:42:45	23	6	6
3	5	2010-09-07 08:00:00	2010-09-07 08:55:43	22	8	8
4	9	2010-09-07 08:42:46	2010-09-07 10:25:38	13	6	6
5	6	2010-09-07 08:00:00	2010-09-07 16:53:24	19	9	8
6	9	2010-09-07 10:25:39	2010-09-07 11:15:09	20	6	6
7	5	2010-09-07 08:55:44	2010-09-07 09:03:19	21	8	7
8	5	2010-09-07 09:03:20	2010-09-07 10:13:57	18	8	7
9	5	2010-09-07 10:13:58	2010-09-07 11:01:40	15	8	8
10	7	2010-09-07 10:33:07	2010-09-07 11:18:25	17	7	8
11	5	2010-09-07 11:01:41	2010-09-07 11:49:23	16	8	7
12	7	2010-09-07 11:18:26	2010-09-07 12:47:44	14	7	9

7.1.2 Anexo I-b: Caso de estudio *KronosParallelOWM.*

operacion_codigo	trabajador_num	operacion_fecha_inicio	operacion_fecha_fin	of_codigo	maquina_codigo	pieza_codigo
1	7	2010-09-07 08:00:00	2010-09-07 10:33:06	24	7	9
2	9	2010-09-07 08:00:00	2010-09-07 08:42:45	23	6	6
3	5	2010-09-07 08:00:00	2010-09-07 08:55:43	22	8	8
4	6	2010-09-07 08:00:00	2010-09-07 16:53:24	19	9	8
5	9	2010-09-07 08:42:46	2010-09-07 09:32:16	20	6	6
6	5	2010-09-07 08:55:44	2010-09-07 09:03:19	21	8	7
7	5	2010-09-07 09:03:20	2010-09-07 10:13:57	18	8	7

7.1.3 Anexo I-c: Caso de estudio *KronosLinearMOW.*

operacion_codigo	trabajador_num	operacion_fecha_inicio	operacion_fecha_fin	of_codigo	maquina_codigo	pieza_codigo
1	9	2010-09-07 08:00:00	2010-09-07 08:42:45	23	6	6
2	9	2010-09-07 08:42:46	2010-09-07 10:25:38	13	6	6
3	9	2010-09-07 10:25:39	2010-09-07 11:15:09	20	6	6
4	7	2010-09-07 08:00:00	2010-09-07 11:22:29	24	9	9
5	7	2010-09-07 11:22:30	2010-09-07 13:23:15	22	9	8
6	5	2010-09-07 08:00:00	2010-09-07 10:44:07	19	8	8
7	5	2010-09-07 10:44:08	2010-09-07 10:51:43	21	8	7
8	5	2010-09-07 10:51:44	2010-09-07 12:02:21	18	8	7
9	6	2010-09-07 08:00:00	2010-09-07 11:31:27	15	7	8
10	9	2010-09-07 11:31:28	2010-09-07 12:12:35	17	7	8

7.1.4 Anexo I-d: Caso de estudio *KronosLinearWOM.*

operacion_codigo	trabajador_num	operacion_fecha_inicio	operacion_fecha_fin	of_codigo	maquina_codigo	pieza_codigo
1	7	2010-09-07 08:00:00	2010-09-07 08:01:52	16	8	7
2	7	2010-09-07 08:01:53	2010-09-07 08:03:45	18	8	7
3	7	2010-09-07 08:03:46	2010-09-07 08:05:38	21	8	7

4	7	2010-09-07 08:05:39	2010-09-07 08:07:18	15	8	8
5	7	2010-09-07 08:07:19	2010-09-07 08:08:58	17	8	8
6	7	2010-09-07 08:08:59	2010-09-07 08:10:38	19	8	8
7	7	2010-09-07 08:10:39	2010-09-07 08:12:18	22	8	8
8	7	2010-09-07 08:12:19	2010-09-07 08:14:09	14	8	9
9	7	2010-09-07 08:14:10	2010-09-07 08:16:00	24	8	9
10	5	2010-09-07 08:00:00	2010-09-07 08:01:20	13	6	6
11	5	2010-09-07 08:01:21	2010-09-07 08:02:41	20	6	6
12	5	2010-09-07 08:02:42	2010-09-07 08:04:02	23	6	6

7.1.5 Anexo I-e: Caso de estudio *KronosReverseMOW*.

operacion_codigo	trabajador_num	operacion_fecha_inicio	operacion_fecha_fin	of_codigo	maquina_codigo	pieza_codigo
1	7	2010-09-07 09:40:18	2010-09-07 11:25:00	22	7	8
2	9	2010-09-07 10:41:41	2010-09-07 11:24:26	23	6	6
3	5	2010-09-07 09:35:30	2010-09-07 10:56:25	24	8	9

7.1.6 Anexo I-f: Caso de estudio *KronosReverseWOM*.

operacion_codigo	trabajador_num	operacion_fecha_inicio	operacion_fecha_fin	of_codigo	maquina_codigo	pieza_codigo
1	7	2010-09-07 09:40:18	2010-09-07 11:25:00	22	7	8
2	9	2010-09-07 10:41:41	2010-09-07 11:24:26	23	6	6
3	5	2010-09-07 09:35:30	2010-09-07 10:56:25	24	8	9

7.1.7 Anexo I-g: Caso de estudio *KronosReverseWOM*

operacion_codigo	trabajador_num	operacion_fecha_inicio	operacion_fecha_fin	of_codigo	maquina_codigo	pieza_codigo
1	7	2010-09-07 14:25:06	2010-09-07 15:54:24	14	7	9
2	7	2010-09-07 11:36:39	2010-09-07 13:05:57	16	7	7
3	5	2010-09-07 12:36:36	2010-09-07 13:01:23	17	8	8
4	9	2010-09-07 12:11:26	2010-09-07 13:00:43	15	9	8
5	6	2010-09-07 10:48:33	2010-09-07 12:57:23	20	6	6
6	5	2010-09-07 09:21:18	2010-09-07 12:36:35	19	8	8
7	7	2010-09-07 09:40:18	2010-09-07 11:25:00	22	7	8
8	9	2010-09-07 09:32:38	2010-09-07 10:56:25	24	9	9

8 Bibliografia.

- http://es.wikipedia.org/wiki/Planificación_de_los_requerimientos_de_material
- http://de.wikipedia.org/wiki/Advanced_Planning_and_Scheduling#Finite_Capacity_Scheduling
- http://es.wikipedia.org/wiki/Algoritmo_evolutivo
- http://es.wikipedia.org/wiki/Optimización_(matemática)
- http://es.wikipedia.org/wiki/Optimización_combinatoria
- http://es.wikipedia.org/wiki/Heurística
- A. Silberschatz, H.F. Korth, S. Sudarshan, **Fundamentos de Bases de Datos**, 5a edición, *McGraw-Hill*, 2006.
- M. Celma, J.C. Casamayor, L. Mota, **Bases de Datos Relacionales**, *Pearson-Prentice Hall*, 2003.
- Roger S. Pressman, Ingeniería del software, un enfoque práctico, Mc Grah-Hill, 4a. edición, 1997.