1. OBJETO

El objeto de este proyecto es implementar, mediante un programa informático, un determinado algoritmo para resolver el problema que se describe a continuación, y recoger y analizar los resultados obtenidos mediante ese programa.

Se tienen un conjunto de piezas y un conjunto de máquinas para fabricarlas. El problema mencionado consiste en determinar, por un lado, con qué máquinas fabricar cada una de esas piezas, y por otro, cómo agrupar todas las máquinas. Cada una de esas piezas puede ser fabricada por un número conocido de planes de proceso, y a su vez cada plan consta de otro número también conocido de operaciones. No existen restricciones acerca de la secuencia en la que han de efectuarse las operaciones, y cada operación puede ser ejecutada por un determinado subconjunto de máquinas. Existen restricciones de capacidad de las máquinas. Toda esta información es conocida, así como los costes y los tiempos de realización de las operaciones, y los costes de transporte de las piezas.

El algoritmo que se va a emplear se puede dividir en tres fases. En la primera de ellas, el objetivo es conseguir una matriz de adecuación máquina-pieza. Para ello emplea la información sobre los costes de realización de las operaciones que forman los planes de las piezas. Esa información, que relaciona piezas, planes, operaciones y máquinas, es agregada (en esta fase, puesto que en la tercera volverá a emplearse en su totalidad) como información sobre piezas y máquinas, en la forma de la matriz de adecuación citada.

La segunda fase tiene como punto de partida la matriz de adecuación máquina-pieza obtenida en la fase anterior. El objetivo de la segunda fase es conseguir agrupar las piezas en familias (a cada una de las cuales corresponderá una célula de máquinas). Se empleará una matriz de pertenencia de las piezas a las familias que será “borrosa”, es decir, no binaria. El método que se emplea para obtener esta última es el Fuzzy G-Means Modificado. La matriz de pertenencia borrosa obtenida es transformada en binaria al final de la segunda fase.

La tercera fase tiene tres variantes. La primera variante se caracteriza por dos restricciones: que cada pieza sólo puede ser fabricada mediante un plan de proceso, y que cada operación de los distintos planes elegidos sólo puede ser realizada por una sola

máquina. La segunda variante parte de una relajación de la variante anterior, a saber: se va a permitir que el número total de operaciones pueda repartirse para su realización entre varias de las máquinas que pueden efectuar esa operación. Y respecto a la tercera variante, no sólo admite que una operación pueda ser realizada por varias máquinas, sino que también permite que la demanda de cada pieza pueda ser repartida entre los distintos planes de proceso de esa pieza. En todas las variantes se emplea una estrategia de resolución en la que se combina la metaheurística Búsqueda Tabú con la programación lineal.

En las pruebas realizadas con el programa-algoritmo, los mejores resultados en los costes de producción de las piezas se produjeron en la tercera variante, seguidos por los conseguidos en la variante segunda. Esto concuerda con lo esperado, puesto que la segunda variante permite un mejor aprovechamiento de la capacidad de las máquinas con respecto a la primera variante, y lo mismo cabe decir de la tercera variante con respecto a las otras dos.

2. FABRICACIÓN CELULAR Y BÚSQUEDA TABÚ

En este capítulo se van a exponer definiciones y características de, por un lado, una técnica (conocida como Fabricación Celular) de organizar las máquinas y las piezas sobre las que operan aquéllas. Y por otro lado, de una metaheurística (Búsqueda Tabú), que se emplea para obtener soluciones de un problema combinatorio, debido a que ambas técnicas aparecerán en este proyecto.

2.1 Tecnología de Grupos

La Tecnología de Grupos (GT) puede definirse como el conjunto de técnicas para organizar la fabricación de piezas consistentes en la agrupación organizada de conceptos, principios, problemas y tareas comunes, para incrementar la productividad. Pese a sus muchas ventajas, entre las que se encuentran rutas de fabricación más sencillas, menores tiempos de set-up y de transporte, stocks más pequeños, etc., diversas razones (fundamentalmente basadas en algún tipo de desconocimiento sobre estos métodos) hacen que la GT se encuentre poco difundida en la industria occidental. La aplicación más avanzada de la GT es la Fabricación Celular.

2.2 Fabricación Celular: Definición

El sistema tradicional, o job shop, forma grupos de máquinas atendiendo a la función que realizan. La fabricación celular (FC) consiste en agrupar las distintas máquinas en lo que es conocido como “células de producción”, de forma que cada célula se encargue de producir –o lo haga en gran medida- una familia de piezas. Una familia de piezas es un grupo de éstas que precisan para su fabricación de algún o algunos recursos similares: maquinaria, equipos, operarios, etc. Lo ideal es que las piezas de una familia sean elaboradas por completo dentro de una sola célula. Pero, como en el caso que nos ocupa, esto no siempre es posible.

2.3 Ventajas y desventajas de la fabricación celular

Las ventajas de la FC sobre el sistema tradicional son considerables, y son la causa para su instalación. Esas ventajas son las siguientes:

- Disminuye el inventario de proceso, puesto que cada trabajo se realiza en una pequeña célula.

- Disminuyen los desplazamientos de material. Prácticamente consisten en el transporte de la materia prima desde el almacén a las células, y de éstas a otro almacén de las piezas terminadas. Por tanto los tiempos de transporte, comparados con los del sistema job shop, serán menores.

- Se aminoran los tiempos de set-up, puesto que, al tratarse de las mismas operaciones, no hay que efectuar cambios en las máquinas. También se reducen al ser parecidas, como se ha dicho en el párrafo anterior, las herramientas de las máquinas. Y por último, porque de esa forma los set-up son los mismos, o parecidos, o son pocos, y por tanto son mejor conocidos por los operarios.

- Se mejoran las relaciones humanas. En las células suele haber de dos a quince operarios, que se sienten como un equipo, y pueden competir en la realización de las piezas con otros equipos. En general, un operario en un equipo produce más que ese mismo operario en solitario.

- Al ser las piezas similares, los operarios de una célula se hacen más expertos en ese tipo de piezas.

- Debido a que las piezas que se fabrican en una célula poseen similares tamaño, forma y composición, se reduce la complejidad de la gestión de las herramientas de las máquinas.

Pero la FC también tiene desventajas:

- Se incrementa la inversión de capital, puesto que cada célula ha de contener todo lo necesario (maquinaria, instalaciones, etc.) para producir sus piezas, puesto que si la pieza sale de la célula porque una máquina es compartida por varias células, muchas de las ventajas de la FC se pierden. Aun así, muchas veces las piezas salen de su célula.

- Debido a lo anterior, al incremento del número de máquinas, la utilización de éstas disminuye en términos generales respecto al sistema funcional job

shop. Además, al estar las células muy especializadas en fabricar un determinado tipo de piezas, puede ocurrir que la demanda de algunos de esos tipos sufra bruscos cambios, produciéndose, por ejemplo, un agudo aumento en la carga de trabajo de las máquinas de algunas células al tiempo que las de otras células están infrautilizadas. Trasladando este problema a los operarios de las distintas células, pueden coincidir operarios saturados de tareas en algunas células con operarios ociosos en otras, lo cual puede generar conflictos entre ellos. Si se trasladan operarios de unas células a otras, se pierde la ventaja de la FC que se deriva de la especialización de los operarios en fabricar una determinada clase de piezas.

2.4 Supuestos básicos

El diseño, el control subsiguiente, y el funcionamiento de la FC se consiguen mediante un grupo de supuestos no estrictos. Acercándose a ellos las ventajas de la FC se maximizan, al tiempo que se minimizan las desventajas. Esos supuestos, que son comúnmente aceptados, son los que siguen:

1. Las piezas son agrupadas según su forma específica y las operaciones que requieren.

2. Si es posible, las máquinas son agrupadas de forma que todas las operaciones de una familia de piezas sean realizadas en una sola célula.

3. Las operaciones requeridas en un trabajo no deben ser realizadas en distintas células.

4. Las células pueden compartir maquinaria, pero el número de máquinas compartidas debe ser minimizado.

5. Cada célula se diseña como un flow shop modificado.

6. Las máquinas no agrupadas en células especializadas, deben agruparse en una célula residual.

7. Algunas máquinas, de hecho, no podrán ser agrupadas, debido a su toxicidad, requerimientos de energía, u otros.

8. Para cualquier trabajo, existe al menos una célula donde todas sus operaciones pueden ser terminadas.

9. Hay más de una de esas posibles células para cualquier trabajo.

10. Si hay una célula especializada a la que un trabajo puede ser encomendado, el trabajo debe ser asignado a la célula especializada en vez de a la célula residual.

11. La eficiencia de una célula y/o de las máquinas incluidas en ella para realizar las operaciones de un trabajo está parcialmente correlacionada con las características del trabajo.

12. Muchas de las máquinas de una célula tienen la flexibilidad de realizar múltiples operaciones.

2.5 Planificación y Control de la Producción

En una encuesta realizada sobre 57 empresas de Estados Unidos que habían implementado la FC, se observaron los siguientes resultados respecto a los métodos de Planificación y Control de la Producción (PPC):

- Mientras que el 81.8% de las empresas mantuvieron un solo método de PPC antes de implementar la FC (74.6% de los cuales eran MRP), las empresas confiaron menos en un único sistema de PPC tras la implementación, usando híbridos de ellos.

- Antes de la instalación de las células, la mayoría de las empresas usaban el MRP antes que ningún otro método. Tras la instalación, ese porcentaje no varió apenas.

- La proporción de empresas que usaban el kanban creció espectacularmente: desde un 7.3% antes de la instalación, se llegó a un 56.4% tras la misma.

- Una vez realizada la instalación, MRP y kanban fue el sistema híbrido más usado. Como el porcentaje de las que usaban MRP no varió apenas antes y después de la instalación, se conluyó que la mayoría de las empresas añadieron el kanban al MRP existente antes de instalar las células.

- Mientras que el 30.9% de las empresas mantuvieron el MRP como el único sistema de PPC después de la instalación de la FC, sólo el 12.7% hizo lo mismo con el kanban.

- La proporción de empresas que usaban otros métodos de PPC (ROP, OPT, y otros), permaneció baja tras la llegada de las células, en comparación con MRP, kanban y el híbrido de ambos.

- La probabilidad de la introducción del kanban (tanto como único método de PPC como componente de un híbrido) no depende del grado de celularización de la planta ni del número de células en la misma, pero sí, y en buena medida, del número de líneas de producto. Cuanto mayor es ese número (no el número de productos terminados), mayor probabilidad hay de que la planta adopte el kanban.

- También se observaron los siguientes cambios: a) los pedidos son realizados más frecuentemente en un 89.5% de los casos, debido a la reducción del tamaño de los lotes; b) los trabajos son planificados dentro de las células en un 89.5% de las empresas, puesto que más trabajos son asignados a las células en vez de a máquinas individuales; c) los trabajos son registrados en un 78.9% en las células, en vez de en las máquinas individuales. Los cambios b) y c) significan pasar, como unidades de registro y fabricación, de dichas máquinas a las células.

Una clara lectura que podría hacerse de los primeros seis resultados es que las empresas que implemetaron la FC no se deshicieron totalmente de los métodos de PPC que tenían antes de la instalación de las células. Esto es comprensible por el hecho de que la mayoría de las empresas no adaptaron el total de su producción a la FC. El método híbrido más usado es el MRP/kanban, o, como es conocido en la literatura habitual, el método push/pull, o también MRP/JIT.

MRP y kanban se complementan. La ventaja básica del MRP es su capacidad para los requerimientos de planificación (de material o de capacidad) para grandes números de productos terminados, y también su capacidad para manejar demandas muy fragmentadas en caso de demanda variable. El MRP es pobre en el control y ejecución de planta. El kanban, por otro lado, tiene un superior sistema de control que encaja en la

fabricación repetitiva, pero falla donde el MRP es bueno: en la anticipación o planificación. Así pues, MRP/kanban puede ser usado de forma que el MRP se encargue de la planificación de la materia prima y componentes, mientras que el montaje, ajustado al plan del mismo, se controla mediante el kanban. El hecho es que sólo el 12.7% de las empresas encuestadas usaba el kanban como único método de PPC, debido a su debilidad en la planificación.

Los resultados de la encuesta confirman también lo obvio: que la introducción de los sistemas kanban (tras instalar la FC) no sería aplazada hasta un punto en el que se alcance una mayor proporción de la producción llevada a cabo por las células, o en el que se consiga una mayor instalación de células, sino que las empresas querrían empezar a usar el kanban inmediatamente.

Se suele afirmar que el kanban no es apropiado para grandes números de productos distintos. La conclusión obtenida en este estudio de que es más probable que las empresas que producen un gran número de líneas de producto introduzcan el kanban tras la implementación de la FC no confirma ni niega esa creencia general, ya que líneas de producto y producto son dos cosas distintas. Puesto que muchas familias de piezas se organizan en base a líneas de producto, es probable que el kanban traiga orden al control de proceso en planta del mayor número de líneas de producto que se fabrican en las células.

2.6 Formación de células

Las cuestiones a resolver en la implementación de la FC son complejas y de diversos tipos, siendo la fundamental de ellas la formación de las células, que consiste en la identificación de las familias de piezas y el grupo de máquinas en las que aquéllas serán procesadas. Tras la formación de las células, se evalúan sus propiedades, lo que es conocido como evaluación celular.

En la misma encuesta anterior, los métodos de formación celular fueron clasificados de la siguiente manera:

1. CLUS: Métodos de agrupación y de coeficiente de similitud, basados en las similitudes de máquinas o piezas.

2. CODE: Las familias de piezas se obtienen codificando y clasificando atributos de las piezas.

3. JVE: Las familias de piezas se forman sin tener en cuenta las rutas de fabricación, de manera informal, basándose en la familiaridad con las piezas.

4. KEYM: Una máquina se coloca en el centro de la célula, a la que se van añadiendo las máquinas que necesitan las piezas que usan la primera.

5. PFA: Análisis del flujo de producción. Las rutas de fabricación de las piezas son examinadas para encontrar piezas que son procesadas por el mismo grupo de máquinas, o grupos de máquinas que procesan el mismo grupo de piezas.

6. PLF: Basado en submontajes o componentes que la empresa produce.

La conclusión que se obtuvo de esta parte de la encuesta fue que la formación celular todavía depende más en las empresas de la opinión, de la familiaridad y de la experiencia de los encargados de la planta sobre el espectro de máquinas y piezas, que de complejos algoritmos, programas y heurísticas, y puesto que éstos provienen del mundo académico, también se concluyó que existe una separación entre éste y las empresas.

2.7 Experiencias en la implementación y mejoras en la actividad de la FC

En otra encuesta efectuada sobre 46 empresas de E.E.U.U. que disponían de una parte de sus plantas de producción organizadas mediante FC, se recogieron en la siguiente tabla la media de las respuestas que dieron sobre los motivos por los que implementaron FC, calificándolas con un valor comprendido entre “1 = Sin importancia”, y “5 = Muy importante”.

Ranking	Motivo	Media
1	Reducir tiempos de proceso	4,51
2	Reducir inventarios WIP (de proceso)	4,33
3	Mejorar la calidad del producto	4,22
4	Reducir el tiempo de respuesta ante pedidos de clientes	4,22
5	Reducir distancias y tiempos de desplazamientos	4,14
6	Incrementar la flexibilidad de fabricación	3,81
7	Reducir el coste unitario	3,80
8	Simplificar el control y la planificación de la producción	3,62
9	Facilitar la implicación o complicidad de los empleados	3,57
10	Reducir tiempos de set-up	3,43
11	Reducir el inventario de productos terminados	3,41

Las cinco razones o motivos más importantes de búsqueda de mejora mediante FC recibieron una media superior a 4 en la escala citada. Esas cinco razones están relacionadas entre sí y con el tiempo, por ejemplo: reducir la distancia y el tiempo de los desplazamientos, reduce el tiempo de proceso, y un menor tiempo de proceso reduce el inventario de WIP ("work in progress"), lo que provoca una reducción del tiempo de respuesta ante los pedidos de clientes. Y finalmente, una mejora en la calidad de fabricación del producto, disminuye el tiempo de proceso, y, por ende, menores tiempos de proceso y de respuesta.

Adicionalmente, nueve empresas adujeron sus propios motivos para implementar la FC, que calificaron con un 4 o un 5, y que pueden ser agrupados como:

- Mejoras asociadas a la utilización de recursos (“reducción del espacio de fabricación”, “mejor uso de las capacidades de los empleados”, “inversión en la rentabilidad de la generalidad de los recursos de la empresa”).

- Mejoras asociadas a la organización y el control (“evitar quebraderos de cabeza en la dirección”, “mejorar la organización en la planta”, “mejorar el control del producto dentro del área de responsabilidad de los empleados”).

- Mejoras asociadas a los recursos humanos (“incrementar la satisfacción por el trabajo de los empleados”).

En cuanto a las mejoras registradas, se solicitó a las empresas el tipo y la cuantía de éstas. Una gran mayoría de las empresas registraron mejoras en cada una de las categorías sobre las que se le preguntó, de las cuales las mayores fueron en los tiempos y distancias de desplazamiento, en los tiempos de proceso, en los tiempos de respuesta a los clientes, en los inventarios de WIP, y en los tiempos de set-up, es decir, en cuatro de las cinco razones más importantes para implantar FC. Además, las empresas, en su mayoría, presentaron mejoras respecto a la planificación y control de la producción, en el compromiso o implicación de los empleados, y en la flexibilidad en la fabricación. Hay que decir que sólo el 29% de las empresas implantaron las células sin inversión adicional en nuevos equipos, a los que hay que atribuir un influjo no cuantificado en las mejoras. Pero el 87% de las empresas respondió que estaban planeando implantar más células, signo de que su coste está justificado para aquéllas.

El estudio encontró también que los grandes beneficios de la FC aparecen en torno a la variable “tiempo”, y es la elección razonable para las empresas que compiten en base a esta variable . Así, las empresas que compiten en base al precio no encontrarían una forma de medir una reducción de costes en el área de fabricación (aunque probablemente la encontrarían fuera de ella).

También recogió la encuesta la impresión (necesitada de más estudios de investigación) de que el análisis previo para identificar las familias supone después un ahorro de tiempo a la hora de hacer operar y producir a las células como se planeó.

Como conclusiones del estudio se extrajeron:

- La FC, pese a haber sido introducida hace muchos años, sigue teniendo muy poca difusión: incluso entre las empresas en las que es implantada, suele serlo en un porcentaje pequeño.

- Hay una gran variación entre las magnitudes de las mejoras conseguidas gracias a las células, debido a la gran variación también en la forma de realizar los análisis y pasos para implantarlas por las empresas, y de los cuales el resultado de la implantación depende en buena medida.

- Los problemas más frecuentes en la implantación se pueden agrupar así: el diseño de las células, el proceso de implementación, y problemas humanos.

- La adopción de la FC es un problema no sólo técnico o ingenieril, sino un cambio general en todo el proceso, y donde lo humano es el asunto dominante.

- Son necesarios más estudios empíricos en áreas como: cuándo es apropiada la FC, cómo y por qué deciden las empresas optar por la FC, qué factores técnicos y humanos hacen que los resultados de las células sean mejores o peores.

2.8 Problemas combinatorios y Métodos de resolución

Los problemas combinatorios son aquellos que poseen un número de soluciones finito. Otra característica importante de estos problemas es que este número de soluciones suele ser muy elevado.

Hay varios grupos de métodos de resolución de métodos combinatorios:

- Exactos: Consiguen la solución óptima del problema (o una de ellas, si no es única). Por el elevado número de soluciones de estos problemas, el tiempo de computación que requieren puede ser enorme, razón por la cual surgen los métodos aproximados.

- Aproximados: No garantizan la solución óptima. Estos métodos se subdividen a su vez en:

1. Métodos Exactos Podados: Aunque no está garantizado el logro de la solución óptima, se conoce una cota del error existente entre el valor obtenido y el valor óptimo.

2. Heurísticas: Se basan en la lógica, el sentido común, o la experiencia. Suelen ser rápidas, sencillas y específicas de cada problema. Debido a su sencillez, no aspiran a obtener soluciones muy buenas. No tienen carácter exhaustivo.

3. Metaheurísticas: Son generales, adaptables a cada problema. Exploran el espacio de soluciones con exhaustividad.

Entre las Metaheurísticas se encuentra, como se ha mencionado más arriba, la Búsqueda Tabú.

2.9 Procedimientos y técnicas de la Búsqueda Tabú

El método consta, primeramente, de un procedimiento para perturbar una solución, y generar otra que es, así, "vecina" de la solución anterior. Todas las soluciones vecinas de una solución forman la "vecindad" de esa solución. El tamaño de la vecindad depende de la perturbación: por ejemplo, si la perturbación consiste en variar todos los elementos que forman la solución, la vecindad asociada a esta solución es todo el espacio de soluciones.

Si el método operase solamente explorando la vecindad de una determinada solución (llamada "actual" o "incumbent"), y al encontrar una solución mejor que la anterior, aquélla pasase a ser la actual, y así sucesivamente, el método obtendría efectivamente un mínimo (considerando, sin pérdida de generalidad, que estamos ante un problema de minimización), pero, salvo casualidad, sería un mínimo local, no absoluto. Al ser toda la vecindad del mínimo peores soluciones que éste, el método se detendría en esa solución, y los mínimos locales no suelen ser buenas soluciones, por lo que hay que obtener una considerable cantidad de ellos. Por tanto, el método, además de operar como acaba de decirse, prohíbe explorar zonas ya exploradas del espacio de soluciones. Así, el método avanza en un primer momento espontáneamente hacia un mínimo local, y de forma determinista (porque siempre es alcanzado). Una vez logrado,

las prohibiciones sobre lo ya explorado “obligan” (porque tiene que admitir peores soluciones) al algoritmo a buscar en otra zona (llamada “valle”) con otro mínimo local.

La forma de prohibir es almacenar características (los “atributos”) de las soluciones exploradas durante un número de iteraciones, durante las cuales, las soluciones que posean los atributos prohibidos no son exploradas. Hay que prohibir grandes zonas del valle para que el algoritmo no pueda permanecer en él recorriendo algunas trayectorias de soluciones que se hayan permitido.

Una perturbación efectuada en una solución es un “movimiento”. Hay atributos de las soluciones de partida de los movimientos (atributos “from”), e igualmente de las soluciones de llegada (atributos “to”). Para que se acepte un movimiento, se comprueba si los to de este movimiento coinciden con from anotados como prohibidos. O al revés: se comprueba si los from coinciden con to prohibidos. Es decir, se prohíben movimientos que devuelvan la exploración a zonas ya visitadas. Los atributos se marcan como prohibidos en una “lista tabú” o “Tabu-end”.

“Tabu Tenure” es el número de iteraciones que un atributo debe permanecer (prohibido) en Tabu-end. En la lista aparece la iteración en la que el atributo dejará de estar prohibido. En la Búsqueda Tabú, suelen calcularse, aunque no es el caso de este proyecto, todos los incrementos de la función objetivo asociados a los movimientos posibles desde la solución actual, sobre la que está evaluada la función objetivo. La lista tabú y los incrementos citados necesitan estructuras de memoria particulares para cada problema.

El concepto “Aspiration Criterium”, o “Criterios de Aspiración” se refiere a permitir movimientos prohibidos, bien porque porque se haya llegado a una situación sin salida (“Por Defecto”, o “Default”), bien porque se estén perdiendo soluciones mejores (“Por Función Objetivo”). Cuando existen criterios de aspiración, hay que evaluar incrementos de la función objetivo asociados a movimientos prohibidos, lo que no ocurre si no existen dichos criterios. Suele habilitarse entonces una “Candidate List” de soluciones candidatas a ser evaluadas, para no hacerlo para toda la vecindad.

Conceptos de “Intensificación” y “Diversificación”:

- Intensificación: Consiste en probar soluciones con atributos que no hayan sido tratados.

- Diversificación: Puede significar dos cosas: una, la exploración de soluciones con atributos que aparecen con frecuencia en las mejores soluciones, y dos, ampliar los movimientos, hacerlos mayores.

Para estas estrategias de intensificación y diversificación, existen dos tipos de memoria: “recent-based memory” (sobre los atributos de soluciones recientes) y “frequency-based memory” (número de veces que se ha aceptado un atributo en una solución).

“Strategic Oscillation”: En esta estrategia, hay dos tipos de movimientos: los que construyen una solución, y los que la deshacen. Se usan en primer lugar movimientos sólo en la dirección de construir, hasta que se alcanza un margen (controlado) de inadmisibilidad, con el que comienza una fase destructiva, de nuevo hasta que se alcanza un cierto margen para construir.

“Tabu Thresholding”: Tiene una fase de mejora, y cuando llega al mínimo local del valle, inicia un proceso para provocar que el método vaya a otro valle y vuelva a iniciar la mejora. Una estrategia similar a ésta será usada en este trabajo.

3. METODOLOGÍA EMPLEADA

En este capítulo se hace una primera descripción del método o algoritmo utilizado para conseguir las soluciones del problema. Dicho método puede dividirse en tres fases: agregación de costes, identificación de familias de piezas, y elección de planes, operaciones y células para cada máquina.

3.1 Primera Fase: Agregación de costes

Como se ha dicho, hay múltiples tipos de piezas, y cada uno de esos tipos puede ser fabricado por un número conocido de planes de proceso. Un número también conocido de operaciones forma cada uno de esos planes de proceso, y cada una de aquéllas, a su vez, puede ser realizada por determinadas máquinas. Es decir: para cada pieza, están determinados los planes de proceso con los que podría fabricarse, las operaciones que configuran cada plan de proceso, y las máquinas que podrían realizar cada una de las operaciones. Son conocidos, por tanto, los conjuntos de máquinas a las que podría asignarse para su realización una operación cualquiera perteneciente a un plan de proceso determinado de una pieza fijada. Y, asimismo, se conocen los conjuntos de todas las operaciones (pertenecientes a cualquier plan de proceso de una pieza cualquiera) que pueden ser efectuadas por una determinada máquina.

También son datos del problema, si una máquina puede realizar una determinada operación (caracterizada por el plan, la pieza a la que pertenece, y su índice dentro del plan), el tiempo que tardaría en efectuarla y el coste de su realización.

Caracterizado así el problema, se impone como primera etapa o fase transformar (agregándola, y sin perderla, puesto que la información completa se necesitará en la tercera fase) esa información de forma que sólo aparezcan relaciones entre máquinas y piezas. El proceso de agregación constará de tres pasos.

En el primero de ellos, se calcula el valor de un índice normalizado que mida cómo de adecuada es una máquina a una determinada operación. El índice toma el valor cero para una máquina que no pueda realizar esa operación. Si puede realizarla, el índice se calcula como la exponencial del coste de efectuarla afectado con el signo negativo, partido por la suma de las exponenciales de los costes de realización (afectados por el signo menos) correspondientes a todas las máquinas que pueden

efectuar la operación citada. Como resultado, la adecuación o conveniencia de que una máquina realice una operación será mayor cuanto menor sea el coste asociado a su realización, y estará comprendida entre cero y uno. Por definición, la suma de las adecuaciones de todas las máquinas que pueden llevar a cabo una operación vale uno. Y en el caso de que sólo una máquina pueda realizar una operación, entonces, valga lo que valga el coste correspondiente, el cálculo de la adecuación tendrá que tomar el valor de la unidad.

El segundo paso ya forma parte de la agregación en sí, cuyo objeto es conseguir una matriz agregada de adecuación máquina-pieza, aunque en un principio no sea posible saber ni el plan de proceso con el que se realizará cada pieza ni las máquinas a las que le serán asignadas las operaciones que lo forman. El segundo y el tercer paso de la agregación usan el mismo criterio. En este segundo paso, se va a considerar que una máquina es conveniente para fabricar una pieza si lo es para realizar cualquier operación de cualquier plan de proceso de esa pieza. Así, pasamos del índice definido en el primer paso (de adecuación de las máquinas a las operaciones) a un índice de adecuación máquina-plan de proceso, y ello lo hacemos tomando el máximo, para cada par formado por un plan y una máquina, de las adecuaciones máquina-operación.

Asimismo, en el tercer paso conseguimos el índice máquina-pieza tomando, para cada par máquina-pieza, el máximo de las adecuaciones de esa máquina a los planes de proceso de esa pieza. El resultado es la matriz de adecuación máquina-pieza buscada.

3.2 Segunda Fase: Identificación de las familias de piezas

La segunda fase recoge, como punto de partida, la matriz agregada máquina-pieza conseguida en la fase anterior. Las componentes de dicha matriz están comprendidas entre cero y uno, como consecuencia del proceso de su cálculo, y representan cómo de adecuada es cada máquina para cada pieza, según los criterios de adecuación indicados en el apartado anterior. Las filas de la matriz representan a las máquinas, y las columnas a las piezas. Se va a emplear esa matriz, y éste es el objetivo de esta fase, para obtener familias de piezas similares, y la mayor o menor similitud entre las piezas vendrá dada por la mayor o menor coincidencia entre las máquinas que tienen mayor índice de adecuación para esas piezas. Puesto que se sabe que los métodos

borrosos procesan muy bien matrices no binarias, y puesto que tal es el caso de nuestra matriz máquina-pieza, escogemos uno de ellos para identificar las familias de piezas.

El método borroso elegido es Fuzzy C-Means Modificado, y las causas de su idoneidad se deben a que posee dos características relacionadas con nuestro problema: no emplea información acerca de la secuencia de operaciones, y toma como dato una matriz A de incidencia máquina pieza (lo que significa que es apropiado cuando en el problema existen múltiples planes de proceso alternativos).

Para los métodos más comunes de formación de familias, se parte de una matriz de agrupamiento binaria, con las columnas representando a las piezas, y las filas a las familias, o viceversa. Sus elementos valdrán cero si la pieza pertenece a la familia, o cero en caso contrario.

Esto no sucede en los métodos de agrupamiento borroso, que utilizan matrices no binarias (borrosas), y en las que sus elementos indican el grado relativo (no absoluto, como ocurre en las matrices binarias) de pertenencia de la pieza a la familia. Estos elementos tienen que cumplir unas restricciones de admisibilidad, a saber: todos ellos han de tomar valores comprendidos entre cero y uno, la suma de los elementos correspondientes a una pieza ha de valer uno, y la suma de las componentes correspondientes a una familia o célula ha de estar comprendida entre cero y el número total de piezas.

Fuzzy C-Means (Bezdek, 1981), el método borroso elegido, tiene, como función objetivo, minimizar la suma de todas las componentes de la matriz de pertenencia borrosa (variables del problema) elevadas a un “parámetro de borrosidad” comprendido entre los valores enteros uno y cuatro, y multiplicadas por la distancia entre dos vectores: el vector columna de la matriz de incidencia o adecuación máquina-pieza correspondiente a la pieza, y el vector de referencia (también llamado centroide o prototipo) correspondiente a la familia o célula, medida esa distancia como la norma euclídea elevada al cuadrado de la diferencia de ambos vectores.

Las iteraciones del algoritmo consisten en modificar los centroides (que dependen de las componentes de la matriz de pertenencia obtenidas en la iteración anterior), y en modificar las componentes de la matriz de pertenencia, que a su vez dependen de los centroides inmediatamente anteriores a la iteración actual.

La convergencia se alcanza cuando la diferencia entre cada componente de la matriz de pertenencia de una iteración y esa misma componente en una iteración anterior (y para todas las componentes), es menor, en valor absoluto, que un determinado valor.

Las pertenencias de las piezas a las familias se hallan tomando el máximo de las componentes del vector correspondiente a cada pieza de la matriz final de pertenencias.

3.3 Tercera Fase: Elección de planes, operaciones y células para cada máquina

En la tercera fase, con las familias de piezas ya formadas, son elegidos los planes de proceso que satisfarán la demanda de las piezas, las máquinas que realizarán las operaciones de los mismos, y las células en las que se incluirán cada una de esas máquinas. Esta fase tiene tres variantes.

En la primera variante, el último paso es la asignación de cada máquina a su célula. Para conseguirlo, previamente se necesita saber cuáles serán los planes mediante los que se realizarán las piezas, y a qué máquinas se les encomendará la realización de las operaciones que forman esos planes. Intuitivamente, se podría llegar a la conclusión de que la solución es que cada operación se asigne a la máquina que la realiza con el coste mínimo. Pero obtenidas así las asignaciones de las máquinas a las operaciones, podrían muy bien saturarse las capacidades de las máquinas más eficientes, resultando otras ociosas, y quedando demasiado descompensadas las cargas de trabajo: una situación, en resumen, que no es admisible. Por todo ello, resulta más lógico establecer restricciones para que no se exceda la capacidad de las máquinas, y que las asignaciones de las operaciones a las máquinas sean precisamente las variables del problema. En la función objetivo propuesta, se incluyen, no sólo la suma de todos los costes de realización de las operaciones por parte de las máquinas, sino también los que se derivan de transportar las piezas de una máquina a otra cuando dichas máquinas se encuentran en distintas células. Estos últimos costes se recogen o modelan mediante el producto de un valor fijo para cada pieza, por el número de operaciones que realizan sobre ella máquinas ajenas a la célula de la pieza en cuestión. Se pretende, como es natural, minimizar esa función objetivo.

Las variables de decisión del modelo son binarias y enteras, e indican:

- Mediante qué plan de proceso se realiza cada pieza (binaria).
- Qué máquina efectúa cada operación de los planes de proceso elegidos (binaria).
- El número de operaciones que una máquina realiza sobre una pieza (entera).
- A qué célula se asigna cada máquina (binaria).

Con la función objetivo descrita, estas variables, y las restricciones que se van a enumerar, resulta un modelo matemático de programación entera. Las restricciones son:

- Toda máquina ha de pertenecer a una y sólo una célula.
- A toda pieza debe asignársele un y sólo un proceso.
- Cada operación de un plan de proceso elegido para fabricar una pieza, ha de ser asignada a una máquina (y ésta debe estar entre el conjunto de las que pueden realizarla). Si el plan de proceso no es el seleccionado, entonces no pueden asignársele sus operaciones a ninguna máquina.
- El nº de operaciones que una determinada máquina realiza sobre una pieza fijada, ha de concordar con la suma de las operaciones de los planes de proceso de esa pieza asignados a aquella máquina.
- El tiempo de trabajo de una máquina, expresado por la suma de las operaciones que realiza multiplicadas por el tiempo de su realización y por la demanda de la pieza a la que corresponde la operación, no debe sobrepasar la capacidad de la máquina.
- La cantidad de máquinas en una célula debe estar entre un tope mínimo y otro máximo.

Todas las restricciones son lineales, pero no así la función objetivo (sí lo es la expresión de los costes de fabricación, pero no la de los costes de transporte, que resulta cuadrática). Se va a considerar como estrategia de solución que el problema tiene dos niveles de programación. En el primer nivel, se asignarán valores a las variables de

selección de planes de proceso y de asignación de operaciones a máquinas. En un segundo nivel, se decidiría a qué célula pertenecerá cada máquina. Lo que se pretende es que, una vez resuelto el primer nivel donde asignamos los planes de proceso y elegimos las máquinas que realizarán las operaciones, la resolución del problema del destino de las máquinas a las células (es decir: el segundo nivel) sea tratado como un modelo lineal de minimización del coste del flujo en una red, problema para el que se puede hallar su solución óptima. Así, se podría usar una metaheurística que explorase el espacio de soluciones admisibles correspondientes al problema del primer nivel (selección de planes de proceso y asignación de operaciones a máquinas), y calcular el valor de la función objetivo tras resolver el problema de segundo nivel asociado al del primer nivel.

La metaheurística que vamos a usar es la Búsqueda Tabú. Ésta podría ser una codificación de las soluciones:

- Se disponen en un vector fijo de dimensión igual al número de piezas, los números enteros que indican para cada pieza cuál de sus planes de proceso será el empleado.

- Además, en una matriz con cada fila representando una pieza, y con tantas columnas como el número máximo de operaciones por plan, se disponen los valores enteros que señalan qué máquina realizará, por orden, cada operación del plan elegido. Puesto que no todos los planes tienen por qué tener el número máximo de operaciones, los posibles huecos de la matriz se rellenan con ceros.

Esta codificación de las soluciones coincide en significado con dos de los tipos de variables del modelo: las que señalaban qué plan era el seleccionado para cada pieza, y las que indicaban qué máquina hacía cada operación de dicho plan.

Respecto a los movimientos para explorar el espacio de soluciones, se realizan dos tipos:

- De corto alcance, en los que se cambia la asignación de una máquina a una operación.

- De largo alcance, en los que se modifica para una de las piezas el plan de proceso asignado. Para obtener las nuevas componentes de la matriz que indica qué máquinas efectúan las operaciones, se propone una búsqueda aleatoria entre las máquinas que pueden realizar cada operación del plan de proceso elegido, lo que confiere al algoritmo de resolución una componente estocástica. Esto puede verse como método de exploración parcial de la vecindad, debido a la alta cardinalidad de ésta.

En lo que respecta a los atributos de los dos tipos de movimientos para gestionar las respectivas listas tabú, podríamos tener:

- Respecto a los de corto alcance: la pieza, el plan de proceso y la operación cuya máquina es cambiada, e igualmente las máquinas elegidas antes y después del movimiento.

- Respecto a los de largo alcance: la pieza para la que cambiamos el plan de proceso, y los planes de proceso asignados a esa pieza antes y después del movimiento.

La propuesta que se hace es una exploración a dos niveles de la vecindad. En el primer nivel, se exploran exhaustivamente los movimientos de corto alcance. Una vez estancada dicha exploración (y esto es detectado mediante un número de iteraciones que es superado sin que se obtengan mejoras en la función objetivo), se procede a explorar los movimientos de largo alcance, los cuales proporcionan una estrategia de diversificación a la exploración. Efectuado un movimiento de largo alcance, se exploran de nuevo los de corto alcance hasta que se dé otra vez el estancamiento. Se continúa de este modo hasta que se exploren un número máximo de movimientos de largo alcance, o un número máximo de éstos sin que la función objetivo mejore.

Por lo que se refiere a las limitaciones de capacidad de las máquinas, la propuesta es que se consideren en la propia definición de la vecindad, para que una solución que no cumpla alguna restricción de las máquinas nunca sea explorada. Si se opera así, queda garantizado que ninguna solución “caerá” fuera de la región de admisibilidad.

La segunda variante resulta de relajar condiciones del modelo de la variante anterior. Esta relajación consiste en que se va a permitir que, aunque el plan de proceso asignado a una pieza sea único, cada una de sus operaciones pueda ser procesada por más de una máquina. Con ello se consiguen una mejor utilización de la capacidad de las máquinas, y reducciones en los costes de realización de las operaciones y del transporte de piezas entre células. Como desventaja, la gestión del sistema aumenta en complejidad, toda vez que podrán existir diferentes rutas de fabricación para fracciones de la demanda de un mismo tipo de pieza. Para conseguir el modelo de esta variante, simplemente hay que realizar unas pequeñas modificaciones en el modelo de la primera variante, concretamente en la función objetivo, en la definición de algunas variables, y en la formulación de ciertas restricciones.

Y la tercera variante que consideramos para esta fase surge de relajar (a partir del modelo de la segunda variante, o sea: considerando que distintas unidades de un mismo tipo de pieza pueden seguir distintas rutas de fabricación) la condición de que todas las piezas de un mismo tipo hayan de ser fabricadas según un mismo plan de proceso. Se puede decir lo mismo que para la variante anterior, que se mejora de nuevo tanto el uso de la capacidad de las máquinas como los costes de operación y de transporte entre células, pero que vuelve a aumentar la complejidad de la gestión del sistema. Esta vez las modificaciones consisten únicamente en retocar las definiciones de algunas variables y la formulación de las restricciones afectadas, quedando intacta la función objetivo.

4. GENERACIÓN DE PROBLEMAS

En este capítulo se describe la forma en que se generan y plantean los problemas para los que el algoritmo buscará una solución. Dicha generación consiste en la asignación de valores por medio del usuario a los parámetros del problema. También consiste en la creación, a partir de éstos, de unos vectores y matrices a cuyas componentes, debido a que por el gran número de éstas sería enormemente laborioso para el usuario introducir sus valores, el programa debe dar valores aleatorios. Estos vectores y matrices, junto con los parámetros citados, configuran el problema, y son los datos de partida para el algoritmo. La exposición se ilustra para problemas sencillos de representar, sensiblemente más pequeños que los que se usarán para obtener resultados, pero idénticos conceptualmente.

4.1 Parámetros del problema

En este apartado se enumeran (mediante el nombre que toman en el programa) y definen los parámetros del problema, cuyo valor debe ser introducido por el usuario:

P nº de piezas

M nº de máquinas

Ce nº de células

cost coste máximo de realización de una operación

cost2 ídem de transporte de una pieza

pla nº máximo de planes por pieza

oper ídem de operaciones por plan

maqs ídem de máquinas que pueden realizar una operación

d demanda máxima posible de una pieza

t tiempo máximo de realización de una operación

H capacidad, o tiempo máximo de ocupación, de las máquinas

Mmax nº máximo de máquinas por célula

Mmin ídem mínimo

f parámetro de borrosidad

E parámetro de precisión en la convergencia del método FCM

limmax nº máximo de soluciones exploradas

limsegur nº máximo de soluciones exploradas sin que mejore la función objetivo

limmax2 nº máximo de soluciones exploradas sin que mejore la función objetivo para una misma asignación de planes a piezas en la primera variante.

limest nº máximo de soluciones exploradas sin que mejore la función objetivo durante una iteración, en las variantes segunda y tercera.

T, T2 nº de iteraciones en los que los atributos de las listas tabú permanecen prohibidos

4.2 Vectores y Matrices de partida del algoritmo

A continuación, se enumeran, y se explica cómo se obtienen, los vectores y matrices que acompañan a los parámetros anteriores en la configuración del problema. Como ilustración se generarán tres problemas, que serán empleados los tres como ilustración de la primera y la segunda fase del algoritmo, pero con cada uno de ellos destinado a ilustrar sólo una de las tres variantes de la tercera fase, para que el tamaño del problema y sus resultados sean los adecuados para una representación clara de cada una de esas variantes. Los tres problemas se diferencian en los valores de los siguientes parámetros:

Para el primer problema (destinado a la 1ª Variante): **P** = 5, **M** = 4, **pla** = 3, **oper** = 4, **H** = 1000, **Mmax** = 3, **Mmin** = 1, **T** = 10, **T2** = 4.

Para el segundo problema (que ilustrará la 2ª Variante): **P** = 5, **M** = 4, **pla** = 3, **oper** = 3, **H** = 700, **Mmax** = 3, **Mmin** = 1, **T** = 2, **T2** = 3.

Y para el tercero (que se empleará en las representaciones de la 3ª Variante): **P** = 4, **M** = 3, **pla** = 2, **oper** = 3, **H** = 800, **Mmax** = 2, **Mmin** = 1, **T** = 2 (en la tercera variante no se usa **T2**).

El resto de parámetros (alguno de los cuales no se emplea en la generación de problemas) toma los mismos valores para los tres problemas: **cost** = 100, **cost2** = 5, **maqs** = 3, **Ce** = 2, **t** = 10, **d** = 100, **f** = 10, **E** = 0,01.

Los vectores y matrices que genera el programa son los que siguen:

El vector **planes**, con tantas componentes como piezas, en las que se almacena el nº de planes que posee la pieza a la que representa la componente. Cada nº de planes es un entero obtenido aleatoriamente, y comprendido entre 1 y el nº máximo de planes (**pla**). Los ejemplos para los tres problemas son:

Problema 1º:

$$\begin{vmatrix} 2 & 3 & 3 & 1 & 2 \end{vmatrix}$$

Problema 2º:

$$\begin{vmatrix} 2 & 3 & 3 & 1 & 2 \end{vmatrix}$$

Problema 3º:

$$\begin{vmatrix} 1 & 1 & 1 & 2 \end{vmatrix}$$

Otro vector, llamado **operaciones**, cuyo nº de componentes es la suma de todos los planes para todas las piezas (es decir, la suma de las componentes de **planes**). Cada una de las componentes corresponde a un plan, y en ellas se guardan números enteros aleatorios comprendidos entre 1 y el número máximo de operaciones por plan (**oper**). En **operaciones** está por tanto el nº de operaciones que posee cada plan. Continuando con los tres ejemplos:

Problema 1º:

3	1	1	4	3	2	4	1	1	1	2

Problema 2º:

1	3	1	3	2	1	1	2	1	1	1

Problema 3º:

2	1	1	2	1

Otro vector, **nm**, con tantas componentes como la suma total de operaciones, y en cada una de ellas el número total de máquinas que pueden efectuar cada operación. Dicho número es un entero comprendido entre 1 y **maqs** (nº máximo de máquinas que pueden realizar una operación). Ejemplos:

Problema 1º:

1	2	1	2	3	3	2	1	2	1	2	1	1	2	1	3	1	3	3	3	2	2	3

Problema 2º:

1	2	1	2	3	3	2	1	2	1	2	1	1	2	1	3	1

Problema 3º:

1	1	1	2	1	2	3

La matriz, llamada **cosoper1**, con los costes de realización de las operaciones mediante las máquinas. Para crearla, lo que es cometido de la función **creacosoper1**, se hace primero una elección aleatoria de las máquinas que pueden efectuar cada operación, hasta completar el nº de máquinas distintas que pueden efectuarla, nº que está guardado en **nm**. Cada vez que se elige una máquina, se le asigna aleatoriamente un coste entre 1 y **cost**. En los huecos de la matriz, es decir, en las componentes que representan una máquina y una operación que aquélla no puede llevar a cabo, se colocan ceros. Para los ejemplos anteriores:

Problema 1º:

100	0	0	0	0	12	0	86	0	0	21	0	0	78	85	72	0	0	0	28	0	0	97
0	0	42	0	43	11	21	0	0	53	0	0	0	0	0	0	0	32	17	73	18	6	92
0	57	0	55	98	35	0	0	91	0	0	0	0	0	0	46	0	69	70	14	0	0	100
0	95	0	86	41	0	76	0	45	0	74	24	8	100	0	83	55	70	93	0	39	87	0

Problema 2º:

10	0	100	0	0	0	0	0	12	0	86	0	0	0	0	21	0
0	29	0	0	42	43	11	55	0	21	0	0	0	53	0	84	0
0	30	0	57	55	98	35	0	23	0	0	0	91	0	0	0	0
0	0	0	95	86	41	0	0	0	0	76	45	0	74	24	24	41

Problema 3º:

0	0	0	0	0	73	17
11	0	0	100	0	95	95
0	96	88	40	87	0	42

Al mismo tiempo que **cosoper1** se crea **tiempos**, matriz que contiene los tiempos de realización de las operaciones mediante las máquinas, y que es análoga a **cosoper1**. El papel que en ésta juega **cost**, aquí lo realiza **t**. Para los tres ejemplos que estamos tratando:

Problema 1º:

7	0	0	0	0	7	0	8	0	0	8	0	0	8	1	10	0	0	0	5	0	0	9
0	0	5	0	8	3	8	0	0	7	0	0	0	0	0	0	0	7	5	8	3	1	1
0	9	0	4	10	6	0	0	4	0	0	0	0	0	0	7	0	4	9	2	0	0	7
0	10	0	1	7	0	9	0	7	0	2	6	9	9	0	6	9	7	4	0	3	9	0

Problema 2º:

5	0	7	0	0	0	0	0	6	0	5	0	0	0	0	2	0
0	5	0	0	2	3	10	5	0	6	0	0	0	2	0	8	0
0	10	0	9	9	5	10	0	2	0	0	0	7	0	0	0	0
0	0	0	8	8	3	0	0	0	0	1	8	0	7	4	5	3

Problema 3º:

0	0	0	0	0	5	7
4	0	0	2	0	4	6
0	9	5	7	8	0	1

La matriz **opermaq**, creada a partir de **cosoper1**, y que tiene el mismo número de columnas que ésta, y **maqs** (nº máximo de máquinas que pueden realizar una operación) como número de filas, presenta, por orden, las máquinas que pueden realizar cada una de las operaciones. Como en general no todas las operaciones podrán ser

efectuadas por el número máximo **maqs**, los huecos se rellenan con ceros. En los tres ejemplos:

Problema 1º:

1	3	2	3	2	1	2	1	3	2	1	4	4	1	1	1	4	2	2	1	2	2	1
0	4	0	4	3	2	4	0	4	0	4	0	0	4	0	3	0	3	3	2	4	4	2
0	0	0	0	4	3	0	0	0	0	0	0	0	0	0	4	0	4	4	3	0	0	3

Problema 2º:

1	2	1	3	2	2	2	2	1	2	1	4	3	2	4	1	4
0	3	0	4	3	3	3	0	3	0	4	0	0	4	0	2	0
0	0	0	0	4	4	0	0	0	0	0	0	0	0	0	4	0

Problema 3º:

2	3	3	2	3	1	1
0	0	0	3	0	2	2
0	0	0	0	0	0	3

El vector **D** para indicar la demanda de cada pieza, generada cada una como un valor entero aleatorio comprendido entre 1 y **d**.

Problema 1º:

64	87	85	73	1

Problema 2º:

$$\begin{vmatrix} 4 & 88 & 20 & 40 & 33 \end{vmatrix}$$

Problema 3º:

$$\begin{vmatrix} 40 & 41 & 21 & 67 \end{vmatrix}$$

Ídem **hache** para el coste de transporte de cada pieza. El papel que ejerce **d** en el caso anterior, aquí lo realiza **h**.

Problema 1º:

$$\begin{vmatrix} 5 & 5 & 5 & 2 & 4 \end{vmatrix}$$

Problema 2º:

$$\begin{vmatrix} 5 & 4 & 2 & 5 & 3 \end{vmatrix}$$

Problema 3º:

$$\begin{vmatrix} 4 & 2 & 2 & 3 \end{vmatrix}$$

5. PRIMERA FASE

Bajo este epígrafe se recogen los pasos conceptuales que da el algoritmo para obtener la matriz agregada máquina-pieza, a partir de los costes de realización de cada operación en las máquinas que pueden llevarla a cabo. También se ilustran estos procedimientos con resultados obtenidos por el programa para los tres problemas sencillos de representar que se mostraron en el capítulo anterior. En la última sección del capítulo se explica con mayor detalle cómo opera el programa en esta fase.

5.1 Datos y Notación

N_j es el número de planes de proceso con los que se supone que puede ser fabricada cada pieza j. Éste número se guarda en el vector **planes**, de los que tomamos los tres ejemplos vistos en el capítulo anterior:

Problema 1º:

$$\begin{vmatrix} 2 & 3 & 3 & 1 & 2 \end{vmatrix}$$

Problema 2º:

$$\begin{vmatrix} 2 & 3 & 3 & 1 & 2 \end{vmatrix}$$

Problema 3º:

$$\begin{vmatrix} 1 & 1 & 1 & 2 \end{vmatrix}$$

Hay o_{jp} operaciones en cada plan p correspondiente a cada pieza j, que se almacenan, como ya se vio, en el vector **operaciones**, cuyos tres ejemplos, correspondientes a los respectivos **planes**, eran:

Problema 1º:

3	1	1	4	3	2	4	1	1	1	2

Problema 2º:

1	3	1	3	2	1	1	2	1	1	1

Problema 3º:

2	1	1	2	1

y cualquiera de esas operaciones puede ser realizada en diferentes máquinas. Las máquinas que pueden llevar a cabo la operación l del plan p de la pieza j, forman los conjuntos I(j, p, l), que son conocidos. Se suponen asimismo conocidos los O(i), conjuntos en los que se hallan, para la máquina i, las operaciones de cualquier plan de proceso de cualquier pieza que puede realizar dicha máquina.

Es decir, si a_{ijpl} es una matriz binaria que indica , en las componentes que valen 1, que la máquina i puede efectuar la operación l del plan p de la pieza j, y el caso contrario en las que valen 0, se tiene:

$$I(j,p,l) = \{i:\ a_{ijpl} = 1\}$$

$$O(i) = \{(j,p,l) : a_{ijpl} = 1\}$$

Los conjuntos I(j, p, l) y O(i), así como la matriz a_{ijpl} pueden obtenerse de la información contenida en **planes**, **operaciones**, y la matriz **opermaq**, cuyo significado se explicó en el capítulo anterior, y de la que sus muestras para los tres problemas de ejemplo eran:

Problema 1º:

1	3	2	3	2	1	2	1	3	2	1	4	4	1	1	1	4	2	2	1	2	2	1
0	4	0	4	3	2	4	0	4	0	4	0	0	4	0	3	0	3	3	2	4	4	2
0	0	0	0	4	3	0	0	0	0	0	0	0	0	0	4	0	4	4	3	0	0	3

Problema 2º:

1	2	1	3	2	2	2	2	1	2	1	4	3	2	4	1	4
0	3	0	4	3	3	3	0	3	0	4	0	0	4	0	2	0
0	0	0	0	4	4	0	0	0	0	0	0	0	0	0	4	0

Problema 3º:

2	3	3	2	3	1	1
0	0	0	3	0	2	2
0	0	0	0	0	0	3

El tiempo de proceso y el coste resultantes de que la máquina i realice la operación l del plan p de la pieza j (t_{ijpl} y c_{ijpl} respectivamente) son también datos, que quedan recogidos en las matrices **tiempos** y **cosoper1**, respectivamente. Recordamos los tres casos de **tiempos**:

Problema 1º:

7	0	0	0	0	7	0	8	0	0	8	0	0	8	1	10	0	0	0	5	0	0	9
0	0	5	0	8	3	8	0	0	7	0	0	0	0	0	0	0	7	5	8	3	1	1
0	9	0	4	10	6	0	0	4	0	0	0	0	0	0	7	0	4	9	2	0	0	7
0	10	0	1	7	0	9	0	7	0	2	6	9	9	0	6	9	7	4	0	3	9	0

Problema 2º:

5	0	7	0	0	0	0	0	6	0	5	0	0	0	0	2	0
0	5	0	0	2	3	10	5	0	6	0	0	0	2	0	8	0
0	10	0	9	9	5	10	0	2	0	0	0	7	0	0	0	0
0	0	0	8	8	3	0	0	0	0	1	8	0	7	4	5	3

Problema 3º:

0	0	0	0	0	5	7
4	0	0	2	0	4	6
0	9	5	7	8	0	1

Y los tres de **cosoper1**:

Problema 1º:

100	0	0	0	0	12	0	86	0	0	21	0	0	78	85	72	0	0	0	28	0	0	97
0	0	42	0	43	11	21	0	0	53	0	0	0	0	0	0	0	32	17	73	18	6	92
0	57	0	55	98	35	0	0	91	0	0	0	0	0	0	46	0	69	70	14	0	0	100
0	95	0	86	41	0	76	0	45	0	74	24	8	100	0	83	55	70	93	0	39	87	0

Problema 2º:

10	0	100	0	0	0	0	0	12	0	86	0	0	0	0	21	0
0	29	0	0	42	43	11	55	0	21	0	0	0	53	0	84	0
0	30	0	57	55	98	35	0	23	0	0	0	91	0	0	0	0
0	0	0	95	86	41	0	0	0	0	76	45	0	74	24	24	41

Problema 3º:

0	0	0	0	0	73	17
11	0	0	100	0	95	95
0	96	88	40	87	0	42

5.2 Etapas para la agregación

Hay tres pasos en el proceso de agregación de la información de los costes. En el primero, se define un índice comprendido en el intervalo [0, 1] que proporcione la medida de lo adecuada que es la máquina i para realizar la operación (j, p, l). Dicho índice tiene que valer 0 si la máquina no puede realizar esa operación (a_{ijpl}=0). En el caso (a_{ijpl}=1) de que sí sea capaz, el índice propuesto es:

$$a_{ijpl} = \frac{e^{-c_{ijpl}}}{\sum_{i' \in I(j,p,l)} e^{-c_{i'jpl}}}$$

que crece cuando decrece el coste. De esta definición se deduce que:

$$\sum_{i \in I(j,p,l)} a_{ijpl} = 1$$

y que, en aquel caso en el que sólo una máquina puede realizar una determinada operación, entonces este índice de adecuación valdrá 1, sea cual sea el valor que tome el coste de realizarla.

Ésta normalización, para los tres casos que estamos tratando, dio como resultado:

Problema 1º:

1	0	0	0	0	0,27	0	1	0	0	1	0	0	0,99	1	5,1*E-12	0	0	0	8,3*E-7	0	0	6,6*E-3
0	0	1	0	0,11	0,73	1	0	0	1	0	0	0	0	0	0	0	1	1	2,3*E-26	0,99	1	0,99
0	1	0	0,99	1,5*E-25	2,7*E-11	0	0	1,0*E-20	0	0	0	0	0	0	0,99	0	8,5*E-17	9,6*E-24	0,99	0	0	3,3*E-4
0	3,1*E-17	0	3,4*E-14	0,88	0	1,2*E-24	0	1	0	9,6*E-24	1	1	2,7*E-10	0	8,5*E-17	1	3,1*E-17	9,8*E-34	0	7,5*E-10	6,6*E-36	0

Problema 2º:

1	0	1	0	0	0	0	0	0,99	0	4,5*E-5	0	0	0	0	0,95	0							
0	0,73	0	0	0,99	0,11	0,99	1	0	1	0	0	0	0,99	0	4,1*E-28	0							
0	0,26	0	1	2,2*E-26	1,5*E-25	3,7*E-11	0	1,6*E-5	0	0	0	1	0	0	0	0							
0	0	0	3,1*E-17	7,7*E-20	0,88	0	0	0	0	0,99	1	0	7,5*E-10	1	0,04	1							

Problema 3º:

0	0	0	0	0	0,999	0,999
1	0	0	8,75*E-24	0	2,78*E-10	1,33*E-34
0	1	1	1	1	0	1,38*E-11

Concluido el primer paso, con el que se han obtenido, para cada máquina, los índices de adecuación a cada una de las operaciones que podría realizar esa máquina, en el segundo se puede decir que comienza la verdadera agregación, la cual tiene por objetivo que aparezca la información de costes o adecuaciones en una única matriz máquina-pieza, pasando por alto la circunstancia de que en principio no es posible conocer el plan de proceso que será asignado a cada pieza, ni, para las operaciones que lo componen, las máquinas que las realizarán. En este segundo paso, y el criterio en el tercero será análogo a éste, se va a considerar que una máquina se adecuará a la fabricación de una pieza si es adecuada para la realización de alguna operación de algún plan de proceso de la pieza en cuestión. Esto es lo mismo que decir que la adecuación de la máquina i al plan de proceso p de la pieza j (o sea, a_{ijp}) se tome como:

$$a_{ijp} = \underset{l=1,2,...,o_{jp}}{\text{máximo}} \; a_{ijpl}$$

Y en el tercer paso, siguiendo un criterio análogo al anterior, se consiguen las adecuaciones de cualquier máquina i a cualquier pieza j (esto es, a_{ij}) de la forma:

$$a_{ij} = \underset{p=1,2,...,N_j}{\text{máximo}} \; a_{ijp}$$

El resultado, para los tres problemas que estamos viendo, fue:

Problema 1º:

1	1	1	8,3*10^-7	0,0067
1	1	1	2,4*10^-26	1
1	2,8*10^-11	0,99	0,99	0,00033
3,4*10^-14	1	1	0	7,6*10^-10

Problema 2º:

1	0,99	4,5*E-5	0	0,95
0,73	1	0,99	0	4,1*E-28
1	1,6*E-5	1	0	0
3,1*E-17	0,88	1	1	1

Problema 3º:

0	0	0	0,999
1	0	8,75*E-24	2,78*E-10
1	1	1	1

5.3 Programación de la Primera Fase

A partir de **cosoper1**, la función **creacosoper** guarda en la matriz **cosoper**, de las mismas dimensiones que **cosoper1**, las exponenciales negativas de los costes si la máquina se encuentra entre las posibles para realizar la operación, o un cero si no lo está. Después, la función **nomcosoper** machaca los valores guardados en **cosoper** con las normalizaciones: divide cada componente por la suma de las componentes de la

columna (salvo que la columna esté llena de ceros, y en ese caso la columna se deja intacta) y ésos son los valores que quedan guardados en las componentes de **cosoper**.

La función **agregar** obtiene la matriz máquina-pieza (**maqpie**) a partir de **cosoper**. Para ello, opera de otra forma distinta a la descrita más arriba, obteniendo un resultado idéntico: calcula el máximo de todos los valores de **cosoper** para todas las operaciones de una pieza y una máquina determinadas: ése será el valor de la adecuación de esa máquina a esa pieza.

6. SEGUNDA FASE

En este apartado se describen los procedimientos teóricos de cálculo para la obtención de la matriz de pertenencia de las piezas a las familias o células, y se ilustra, al igual que en el apartado anterior, la exposición de esos procedimientos con los resultados obtenidos por el programa para problemas de pequeña magnitud que permitan una fácil representación. Dichos problemas son los que han sido empleados en los dos capítulos anteriores. En el último apartado del capítulo se detalla e ilustra cómo se ha programado esta fase del algoritmo.

6.1 Punto de partida, objetivo de la fase, y estrategias

La primera fase ha concluido con la obtención de una matriz a , con la información agregada de las adecuaciones entre las máquinas y las piezas, cuyas componentes a_{ijpl} pertenecen al intervalo [0, 1], y cuyas filas y columnas representan a cada una de las máquinas y a cada una de las piezas respectivamente. Para los ejemplos tomados del capítulo anterior las matrices a eran:

Problema 1º:

1	1	1	8,3*10^-7	0,0067
1	1	1	2,4*10^-26	1
1	2,8*10^-11	0,99	0,99	0,00033
3,4*10^-14	1	1	0	7,6*10^-10

Problema 2º:

1	0,99	4,5*E-5	0	0,95
0,73	1	0,99	0	4,1*E-28
1	1,6*E-5	1	0	0
3,1*E-17	0,88	1	1	1

Problema 3º:

0	0	0	0,999
1	0	8,75*E-24	2,78*E-10
1	1	1	1

La matriz a se puede emplear para obtener familias de piezas similares, siendo el criterio de similitud las máquinas que les son más adecuadas. Tratándose de matrices no binarias máquina-pieza, de las que se sabe que son apropiadamente procesadas por los métodos borrosos, se puede escoger uno de éstos para formar las familias.

El que aquí va a usarse es Fuzzy C-Means Modificado, que no emplea la información sobre la secuencia de operaciones, y que su aplicación parte de una matriz A de incidencia máquina-pieza no binaria (lo cual lo hace útil para los casos en que existen múltiples planes de proceso alternativos): dos características que se ajustan a este problema.

6.2 Algoritmo Fuzzy C-Means Modificado

Se realizan ahora una Introducción al método FCM Modificado y el detalle de los pasos del algoritmo.

6.2.1 Introducción

Los métodos de formación de familias convencionales tienen una matriz de agrupamiento binaria como variable. Esta matriz sería de la forma:

	pieza	1	2	3	...	P
	familia					
	1	y_{11}	y_{12}	y_{13}	...	y_{1P}
	2	y_{21}	y_{22}	y_{23}	...	y_{2P}
Y=	3	y_{31}	y_{32}	y_{33}	...	y_{3P}
	.	.	.	.	...	
	.	.	.	.	...	
	C	y_{C1}	y_{C2}	y_{C3}	...	y_{CP}

en la cual sus elementos y_{jk} valen 1 si la pieza j pertenece a la familia k, y cero en el caso contrario.

No ocurre así en los métodos de agrupamiento borroso, que toman matrices de pertenencia borrosas (no binarias), y para las que cada elemento y_{jk} tiene otro significado: la medida de la pertenencia de la pieza j a la familia k. Los elementos de las matrices de pertenencia borrosas deben cumplir las restricciones de admisibilidad siguientes:

$$0 \le y_{jk} \le 1, \quad j = 1,2,...,P; \quad k = 1,2,...,C$$

$$\sum_{k=1}^{C} y_{jk} = 1, \quad j = 1,2,...,P$$

$$0 < \sum_{j=1}^{P} y_{jk} < P, \quad k = 1,2,...,C$$

Por ejemplo, para cada uno de los casos de los capítulos anteriores, se obtuvieron estas matrices de pertenencia borrosa entre las iteraciones del método:

Problema 1º:

0,515...	0,484...
0,503...	0,496...
0,510...	0,489...
0,490...	0,509...
0,491...	0,508...

Problema 2º:

0,531...	0,468...
0,487...	0,512...
0,516...	0,483...
0,473...	0,526...
0,478...	0,521...

Problema 3º:

0,544...	0,455...
0,499...	0,500...
0,499...	0,500...
0,455...	0,544...

6.2.2 Algoritmo Fuzzy C-Means (FCM)

El método borroso Fuzzy C-Means (Bezdek, 1981) emplea la función objetivo:

$$\text{Min} \sum_{j=1}^{P}\sum_{k=1}^{C}(y_{jk})^{f}\left\|\bar{a}_j - \bar{m}_k\right\|^2$$

en la que 1<f<4 es un parámetro de borrosidad, $\|\cdot\|$ es la norma euclídea, y $\bar{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{Mj} \end{bmatrix}$

es uno de los vectores columna de la matriz máquina-pieza y, por último,

$\bar{m}_k$ es el llamado, indistintamente, centroide, vector de referencia o prototipo de la familia k, y que se obtiene de la forma:

$$\bar{m}_k = \frac{\sum_{j=1}^{P}(y_{jk})^{f}\,\bar{a}_j}{\sum_{j=1}^{P}(y_{jk})^{f}}$$

Las iteraciones del algoritmo FCM consisten en modificar los valores y_{jk} y $\bar{m}_k$:

$$\bar{m}_k^{t+1} = \frac{\sum_{j=1}^{P}(y_{jk}^{t})^{f}\,\bar{a}_j}{\sum_{j=1}^{P}(y_{jk}^{t})^{f}}$$

Para cada uno de los problemas con los que se está ilustrando el algoritmo, aparecieron estas matrices de centroides entre las iteraciones:

Problema 1º:

0,700	0,501
0,871	0,715
0,655	0,570
0,387	0,385

Problema 2º:

$$\begin{vmatrix} 0{,}618 & 0{,}568 \\ 0{,}670 & 0{,}416 \\ 0{,}617 & 0{,}216 \\ 0{,}630 & 0{,}881 \end{vmatrix}$$

Problema 3º:

$$\begin{vmatrix} 0{,}828 & 0{,}495 \\ 0{,}495 & 0{,}0828 \\ 1 & 1 \end{vmatrix}$$

$$y_{jk}^{t+1} = \left[\frac{\frac{1}{\left\| \bar{a}_j - \bar{m}_k^t \right\|}}{\sum_{k'=1}^{C} \frac{1}{\left\| \bar{a}_j - \bar{m}_{k'}^t \right\|}} \right]^{\frac{2}{f-1}}$$

La convergencia, alcanzada en una iteración T, se consigue si la matriz de pertenencia cumple

$$\left| y_{jk}^{T+1} - y_{jk}^{T} \right| \leq e \qquad \forall j, k$$

Alcanzada la convergencia del algoritmo, las pertenencias de las piezas a las familias se hallan como sigue:

$$y^{*}_{jk} = \begin{cases} 1 & \text{si } k = \arg \max_{1 \le k' \le C} y^{T}_{jk'} \\ 0 & \text{si no} \end{cases}$$

Estas pertenencias son el resultado final de esta fase.

6.3 Programación de la Segunda Fase

El algoritmo necesita una matriz **y** de partida, que puede ser cualquiera, y que se consigue mediante la función **creay**. Para los tres casos, estas matrices de partida fueron:

Problema 1º:

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

Problema 2º:

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

Problema 3º:

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{vmatrix}$$

Con esta primera **y**, la función **fcm** halla la matriz **bin** de pertenencia de las piezas a las familias o células, aplicando el método citado: Fuzzy C-Means Modificado.

Para ello, en **fcm** se crean las matrices **mu** y **temp**, la primera conteniendo los centroides de las sucesivas iteraciones, y la segunda las matrices de pertenencia borrosa que se van generando en las distintas iteraciones, y que son copiadas en **y**. Con este fin, **fcm** llama alternativamente a **modificmu** y **modifictemp**, funciones que realizan las iteraciones correspondientes sobre los centroides y las matrices de pertenencia borrosa, respectivamente.

La función **modificmu** halla, para cada célula k, el denominador del cociente que proporciona. Después obtiene cada componente m_{ik} como el producto de cada componente elevada a **f** de la columna j de la matriz **y** por las componentes de la fila i de la matriz a , dividiéndolo por el denominador citado.

La otra función citada, **modifictemp**, calcula para cada j y para cada k el inverso de la norma euclídea de la diferencia entre los vectores $\vec{a}_j$ y $\vec{m}_k$, y los va sumando para todo k. Entonces, para el j ya citado, y un k determinado, halla ese mismo valor : el inverso de la norma euclídea de $\vec{a}_j$ menos $\vec{m}_k$, lo divide por la suma anterior, y lo eleva a 2/(**f**-1). Así obtiene cada componente y_{jk}.

Tras una llamada a **modificmu**, sigue una a **modifictemp**, y después otra a la función **compro**. En ésta se realiza la comprobación de que las diferencias entre todas las componentes entre dos matrices sucesivas (**temp** e **y**) de pertenencia borrosa sean menores que **E**. Si no lo son, **fcm** llama a la función **cambio**, que copia la nueva **temp** en **y**, y si lo son, se llama a **cambiabin**, que crea **bin** a partir de esa última **temp**.

Se muestran a continuación las últimas dos iteraciones de **temp** para los tres ejemplos anteriores, así como las consiguientes matrices **bin**:

Problema 1º:

0,515...	0,484...
0,503...	0,496...
0,510...	0,489...
0,490...	0,509...
0,491...	0,508...

0,509...	0,490...
0,505...	0,494...
0,512...	0,487...
0,492...	0,507...
0,490...	0,509...

1	0
1	0
1	0
0	1
0	1

Problema 2º:

0,531...	0,468...
0,487...	0,512...
0,516...	0,483...
0,473...	0,526...
0,478...	0,521...

0,524...	0,475...
0,494...	0,505...
0,514...	0,485...
0,475...	0,524...
0,471...	0,528...

1	0
0	1
1	0
0	1
0	1

Problema 3º:

$$\begin{vmatrix} 0{,}544... & 0{,}455... \\ 0{,}499... & 0{,}500... \\ 0{,}499... & 0{,}500... \\ 0{,}455... & 0{,}544... \end{vmatrix}$$

$$\begin{vmatrix} 0{,}539... & 0{,}460... \\ 0{,}499... & 0{,}500... \\ 0{,}499... & 0{,}500... \\ 0{,}460... & 0{,}539... \end{vmatrix}$$

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{vmatrix}$$

7. TERCERA FASE: 1ª VARIANTE

En este capítulo se describe la primera variante de la última fase del algoritmo, en la cual sólo se puede asignar un solo plan para cada pieza, y cada operación sólo puede ser realizada por una sola máquina. Así pues, se persiguen tres objetivos: la elección de los planes de proceso de cada pieza, la asignación de las máquinas a las operaciones de esos planes, y las células a las que se destinarán esas máquinas. Al igual que en los dos capítulos anteriores, se ilustrarán los distintos pasos para llegar a esos tres objetivos con resultados del programa sencillos de representar, y que partirán de los ejemplos usados en los dos capítulos anteriores y en la generación de problemas, y asimismo se detallará e ilustrará cómo se ha programado esta sección del algoritmo.

7.1 Modelo del Problema

El último paso de esta fase, en esta variante, es la asignación de las máquinas a la familia a la que pertenecerán. Esta asignación se expresa mediante una matriz binaria **xik**, con las filas representando a las máquinas y las columnas a las células, y en la que, si en la componente correspondiente a la fila (máquina) i y la columna (célula) k, hay un uno, dicha máquina ha sido destinada a dicha célula, habiendo un cero en caso contrario. Así, para el ejemplo que ilustra esta variante, una de las **xik** que aparecerán es:

$$\begin{vmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

Se observa que en estas matrices sólo puede haber un 1 por fila.

Para lograr la **xik** de cada solución, antes es necesario decidir mediante qué planes se fabricará cada pieza, y qué máquinas serán las encargadas de llevar a término las operaciones de dichos planes. El modo de expresar en el programa la asignación de planes a las piezas, y la adjudicación de máquinas a operaciones, se verá en el epígrafe

siguiente. En lo referente a la asignación de operaciones a máquinas, en un principio podría pensarse que lo mejor es que cada operación se realice en aquella máquina que lo hace al menor coste. Pero si efectuamos así la asignación de las operaciones a máquinas, nada impide que se sature la capacidad de las máquinas más eficientes, mientras que otras podrían quedar ociosas, resultando demasiado descompensadas las cargas de trabajo, lo cual es inadmisible. Parece, por ello, más lógico imponer restricciones de capacidad a las máquinas, y que las asignaciones de las operaciones a las máquinas sean variables del modelo. Esto se realiza en el programa mediante un vector, **capac**, que, para cada solución, guarda el tiempo de ocupación de cada máquina. Al hacer modificaciones en las soluciones, se comprueba si las componentes afectadas de **capac** exceden el valor **H** (de valor 1000, a lo largo de este ejemplo). El vector **capac** para la solución anterior vale:

696	686	757	792

En la función objetivo propuesta, se encuentran no sólo los costes de realización de las operaciones por parte de las máquinas, sino también los asociados a mover las piezas de unas células a otras, que se modelan como el producto de un valor fijo h_j para cada pieza j, por el número de operaciones que se efectúan sobre dicha pieza fuera de la célula a la que pertenece.

Establezcamos la notación que vamos a necesitar, para después formular el modelo. Así, tendremos

i índice de máquinas

j índice de piezas

k índice de familias/células

p índice de planes de proceso

l índice de operaciones

C Número de familias de piezas formadas

J(k) Conjunto de piezas pertenecientes a la familia k

D_j Demanda de la pieza j

I(j,p,l) Conjunto de máquinas que pueden realizar la operación l del plan p de la pieza j

O(i) Conjunto de operaciones que se pueden realizar sobre la máquina i

c_{ijpl} Coste de realizar la operación l del plan p de la pieza j en la máquina i

t_{ijpl} Tiempo de proceso de la operación l del plan p de la pieza j en la máquina i

H_i Límite de capacidad de la máquina i

h_j Coste unitario por desplazamiento intercelular de la pieza j

Las variables de decisión del modelo son

$$u_{jp} = \begin{cases} 1 & \text{si la parte j se procesa siguiendo el plan p} \\ 0 & \text{en caso contrario} \end{cases}$$

$$v_{ijpl} = \begin{cases} 1 & \text{si la operación l del plan p de la pieza j es realizada en máquina i} \\ 0 & \text{en caso contrario} \end{cases}$$

w_{ij} Número de operaciones que la máquina i realiza sobre la pieza j

$$x_{ik} = \begin{cases} 1 & \text{si máquina i se asigna a célula k} \\ 0 & \text{en caso contrario} \end{cases}$$

Dada la notación anterior, vamos a condiderar en esta variante de la tercera fase el modelo matemático de programación entera siguiente

$$\text{Minimizar} \quad \sum_i \sum_j \sum_p \sum_l D_j \, c_{ijpl} \, v_{ijpl} + \sum_{k=1}^{C} \sum_{j \in J(k)} h_j \sum_i D_j \, w_{ij} \, (1 - x_{ik})$$

sujeto a

$$\sum_{k=1}^{C} x_{ik} = 1 \qquad \forall i$$

$$\sum_{p} u_{jp} = 1 \qquad \forall j$$

$$\sum_{i \in I(j,p,l)} \mathrm{v}_{ijpl} = u_{jp} \qquad \forall j, p, l$$

$$\sum_{p} \sum_{l} \mathrm{v}_{ijpl} = w_{ij} \qquad \forall i, j$$

$$\sum_{(j,p,l) \in O(i)} D_j \, t_{ijpl} \, \mathrm{v}_{ijpl} \leq H_i \qquad \forall i$$

$$M_{\min} \leq \sum_{i} x_{ik} \leq M_{\max} \qquad \forall k$$

$$x_{ik}, u_{jp}, \mathrm{v}_{ijpl} = \{0,1\} \qquad w_{ij} \ \textit{entera}$$

Los costes de operación se hallan en el primer sumando de la función objetivo, así como en el segundo tenemos los costes asociados a transportar una pieza de la célula a la que pertenece a otra célula, porque una o varias de las operaciones que tiene que sufrir la pieza han sido asignadas a máquinas que se encuentran en células diferentes a las de la pieza de la que hablamos. Las restricciones se interpretan como sigue:

- Toda máquina ha de pertenecer a una y sólo a una célula.
- A toda pieza debe asignársele un y sólo un proceso.
- Si p es el plan de proceso asignado a la pieza j, cada operación l debe ser realizada por una y sólo una máquina i de las pertenecientes al grupo de las que pueden hacerlo. Si p no es el plan de proceso asignado, entonces no puede asignarse ninguna máquina a sus operaciones l.

- El nº de operaciones que realiza la máquina i sobre la pieza j, debe concordar con la suma de las operaciones l de los planes de proceso p de la pieza j asignadas a las máquinas i.

- Restricción que expresa que no debe sobrepasarse la capacidad de las máquinas.

- Ídem de las células. Además, las células también tienen un tope mínimo.

Mientras que todas las restricciones son lineales, así como el primer sumando de la función objetivo, el segundo sumando de ésta es cuadrático. Se va considerar como estrategia de resolución que el problema tiene dos niveles de programación. El primer nivel consiste en asignar valores a las variables de selección de planes de proceso y de asignación de operaciones a máquinas. En el segundo, se hallarían los valores de las variables de agrupamiento de máquinas $x_{ik.}$ Lo que se pretende es que, una vez asignados los planes de proceso y elegidas las máquinas que realizarán sus operaciones, el segundo nivel consistente en resolver los agrupamientos de las máquinas, se trate como un modelo lineal de minimización del coste del flujo en una red, cuya solución óptima puede encontrarse. Así, se podría usar una metaheurística que explorase el espacio de soluciones admisibles correspondientes al problema del primer nivel (selección de planes de proceso y asignación de operaciones a máquinas), y calcular el valor de la función objetivo tras resolver el problema de segundo nivel asociado al del primer nivel.

7.2 Búsqueda Tabú

Ésta es la metaheurística que vamos a usar. La codificación de las soluciones del espacio puede ser la que se detalla a continuación:

- Se disponen en un vector fijo de dimensión igual al número de piezas P, los valores $\pi(j)$ que señalan para cada pieza j cuál de sus N_j planes de proceso sería el que se emplease.

- Una matriz con tantas filas como piezas y tantas columnas como el número máximo de operaciones, indicando en cada fila (que representa a una pieza), qué máquinas realizan las operaciones del plan que se le ha asignado. Como

no todos los planes poseen el número máximo de operaciones, en dicha matriz hay espacios vacíos.

La asignación de planes queda recogida en el programa en un vector (**asigplan**) con tantas componentes como piezas. Para la solución del ejemplo visto en capítulos anteriores es:

2	2	3	1	2

cuya interpretación es que a la primera pieza se le asigna el segundo de sus tres planes, también el segundo plan para la segunda pieza, el tercero de sus planes a la tercera, etcétera.

Para un determinado **asigplan**, la información de qué máquinas serán las encargadas de realizar las operaciones de esos planes se registra en una matriz, de nombre **asigopermaq**, con tantas filas como piezas y tantas columnas como el número máximo de operaciones, indicando en cada fila (que representa a una pieza), qué máquinas realizan las operaciones del plan que se le ha asignado, tal como se describió más arriba. Para el ejemplo anterior, la matriz **asigopermaq** que se obtuvo es:

4	0	0	0
2	4	1	3
2	0	0	0
3	0	0	0
4	3	0	0

Puesto que no todos los planes poseen el número máximo de operaciones (como es el caso del ejemplo salvo para la segunda pieza), las filas correspondientes a la piezas a las que se les ha adjudicado uno de esos planes se rellenan de ceros. En nuestro ejemplo, la primera operación del plan asignado a la quinta pieza es efectuada por la cuarta máquina, y la segunda operación (y última) por la tercera máquina.

Hay dos tipos de movimientos para explorar la zona del espacio cercana a una solución:

- De corto alcance, en los que se cambia la asignación de una máquina a una operación.

- De largo alcance, en los que se modifica para una de las piezas el plan de proceso asignado. Esto significa que en la matriz de asignación de operaciones a máquina (**asigopermaq**) una fila varía completamente: las operaciones que representa pertenecen a un nuevo plan, que puede tener un número completamente distinto de aquéllas. Para obtener esas componentes, se propone una búsqueda aleatoria entre las máquinas que pueden realizar cada operación del plan de proceso asignado, lo que confiere al algoritmo de resolución una componente estocástica. Esto puede verse como un método de exploración parcial de la vecindad, debido a la enorme cardinalidad de ésta.

Como ejemplo, para la solución conformada por los **asigplan**, **asigopermaq** y **xik** siguientes:

$$\begin{vmatrix} 2 & 2 & 3 & 1 & 1 \end{vmatrix}$$

$$\begin{vmatrix} 3 & 0 & 0 & 0 \\ 2 & 4 & 1 & 3 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{vmatrix}$$

1	0
1	0
1	0
0	1

con valor de la función objetivo **fo** = 29575, el movimiento de largo alcance que consiste en asignar el segundo de sus planes a la quinta pieza, trae estos cambios para **asigplan** y **asigopermaq**, y supone también replantear **xik**, que en este caso resultó ser la misma que para la solución anterior:

2	2	3	1	2

3	0	0	0
2	4	1	3
2	0	0	0
3	0	0	0
4	3	0	0

1	0
1	0
1	0
0	1

El valor de la función objetivo para esta solución es:

fo = 29727

Si sobre esta solución se realiza un movimiento de corto alcance consistente en asignar la cuarta máquina a la operación del primer plan, las nuevas **asigopermaq** y **xik** son (**asigplan** no sufre, por definición, variaciones en los movimientos de corto alcance):

$$\begin{vmatrix} 4 & 0 & 0 & 0 \\ 2 & 4 & 1 & 3 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

El nuevo valor de la función objetivo resulta ser:

fo = 31565

En lo que respecta a los atributos de los dos tipos de movimientos para gestionar las respecivas listas tabú, podríamos tener:

- Respecto a los de corto alcance: la pieza, el plan de proceso y la operación cuya máquina es cambiada, e igualmente las máquinas elegidas antes y después del movimiento.

- Respecto a los de largo alcance: la pieza para la que cambiamos el plan de proceso, y los planes de proceso asignados a esa pieza antes y después del movimiento.

Para el programa, la lista tabú de los movimientos de corto alcance consiste en una matriz (**tabupeq**) con las mismas dimensiones de la matriz **opermaq**, vista en capítulos anteriores: un nº de filas igual al número máximo de máquinas que pueden ser asignadas a una operación (**maqs**), y un nº de columnas igual al nº total de operaciones. Esta matriz es de hecho un calco de **opermaq**, en el sentido de que las entradas de las dos matrices corresponden a asignaciones de máquinas a operaciones, en el caso de **opermaq** todas las posibles, y en el caso de **tabupeq** todas las prohibidas. Una entrada de **tabupeq** tiene como valor la iteración hasta la cual permanecerá prohibida la asignación de máquina a la que corresponde dicha entrada. Inicialmente, **tabupeq** está llena de ceros, y cada vez que se realiza una asignación de máquina a una operación, se coloca en el lugar correspondiente de **tabupeq** el valor de la iteración actual (almacenado en un contador llamado **contiter**) más un valor fijado por el usuario (**T**).

Si tomamos como ejemplo la serie de soluciones que ha servido para ilustrar los movimientos de corto y largo alcance, y consideramos que la primera solución de la serie es la primera que obtiene el algoritmo, para **T** = 10, la primera **tabupeq** (**contiter** = 0) resultante de las 8 asignaciones realizadas es:

0	0	0	10	0	0	0	10	10	0	0	0	0	0	0	0	0	0	10	0	0	0	0
0	0	0	0	0	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0

Al efectuar el cambio de plan citado más arriba en la quinta pieza, se aceptan dos nuevos movimientos, y como ya **contiter** = 1, se tiene la siguiente **tabupeq**:

0	0	0	10	0	0	0	10	10	0	0	0	0	0	0	0	0	0	10	0	0	0	0
0	0	0	0	0	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	10	11	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	11

El movimiento de corto alcance aprobado citado anteriormente, traerá una nueva asignación, y puesto que **contiter** sigue almacenando la iteración 1, la matriz anterior se verá así modificada:

0	0	0	10	0	0	0	10	10	0	0	0	0	0	0	0	0	0	10	0	0	0	0
0	0	0	11	0	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	10	11	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	11

Respecto a la lista tabú de los movimientos de largo alcance, cabe decir algo similar respecto a los de corto alcance: existe una matriz (**tabugrand**) con un nº de filas igual al nº de piezas y un nº de columnas igual al máximo nº de planes (**pla**). Cada componente de tabugrand corresponde a la asignación de un plan a una pieza (como no todas las pieza poseerán el máximo número de planes posible, los huecos se rellenan con ceros). Una entrada de **tabugrand** tiene como valor la iteración hasta la cual permanecerá prohibida la asignación de plan a la que corresponde dicha componente. Al comienzo de las iteraciones, **tabugrand** está llena de ceros, y cada vez que se realiza una asignación de un plan a una pieza, se sitúa en el lugar que corresponde en **tabupeq** el valor de la iteración actual (almacenado en el contador citado **contiter**) más un valor proporcionado por el usuario (**T2**).

Pongamos como ejemplo la serie de soluciones mostrada a lo largo de este apartado. Tras la primera asignación de planes, y para **T2** = 4, se tiene la siguiente **tabugrand** (**contiter** = 0):

0	4	0
0	4	0
0	0	4
4	0	0
4	0	0

El cambio de plan en la quinta pieza ya mencionado provocará una nueva anotación, para la que **contiter** vale 1:

0	4	0
0	4	0
0	0	4
4	0	0
4	5	0

La propuesta que se hace es una exploración a dos niveles de la vecindad. En el primer nivel, se exploran exhaustivamente los movimientos de corto alcance. Una vez estancada dicha exploración (y esto es detectado mediante un número de iteraciones que es superado sin que se obtengan mejoras en la función objetivo), se procede a explorar los movimientos de largo alcance, los cuales proporcionan una estrategia de diversificación a la exploración. Efectuado un movimiento de largo alcance, se exploran de nuevo los de corto alcance hasta que se dé otra vez el estancamiento. Se continúa de este modo hasta que se realicen un número máximo de iteraciones, o hasta que se supere un máximo de iteraciones sin que la función objetivo mejore. El programa posee por tanto tres contadores: para el estancamiento, es decir, para el número de soluciones sin que la función objetivo mejore para un mismo **asigplan** (**contest**), para el número de

iteraciones totales (**contiter**), y para el número de iteraciones sin que la función objetivo mejore (**contsegur**).

Por lo que se refiere a las limitaciones de capacidad de las máquinas, la propuesta es que se consideren en la propia definición de la vecindad, para que una solución que no cumpla alguna restricción de las máquinas nunca sea explorada. Si se opera así, queda garantizado que ninguna solución "caerá" fuera de la región de admisibilidad. Este control se lleva a cabo mediante el vector **capac**, como ya se ha mencionado, y será tratado con más detalle en el epígrafe siguiente.

7.3 Programación de la 1ª Variante de la 3ª Fase y Diagrama de Flujo

La función **fase31** es la encargada de gestionar esta variante de la tercera fase del algoritmo. Toma, como argumentos, parámetros de control, dimensiones del problema, y vectores y matrices resultantes, por un lado, de la generación de problemas, y por otro, de las fases primera y segunda. **fase31** devuelve una estructura (**resultado**) que contiene la solución al problema obtenida por el algoritmo con esta variante, además del valor de la función objetivo correspondiente a dicha solución. Es decir, en **resultado** se incluyen: un vector (**asigplanm**) que indica el plan asignado para fabricar cada pieza, una matriz (**asigopermaqm**) donde se muestra qué máquina es la elegida para realizar cada operación de los planes asignados, una matriz (**xikm**) con la información de a qué célula se destina cada máquina, y el valor (**fom**) que toma la función objetivo para la solución formada por los tres elementos anteriores. También se incluye en **resultado** el vector **capacm**, que tiene una componente por cada máquina, y en el que se guarda el tiempo de ocupación de las máquinas para la solución que se guarda en **resultado**. En cada instante de ejecución de esta fase del algoritmo, en **resultado** se almacena la mejor solución obtenida hasta dicho instante, y cuando se ha excedido el valor de alguno de los parámetros de control, **resultado** es devuelto a la función principal. Más precisamente, se devuelve **resultado** por parte de **fase31** cuando se ha superado, o bien un número máximo de iteraciones (cuya cuenta se lleva en la variable **contiter**), o bien, para asegurarnos de que el programa no realiza demasiadas iteraciones superfluas, un número máximo de iteraciones sin que mejore **fom** (dicho número de iteraciones sin mejora de la función objetivo se almacena en **contsegur**). Hay otra estructura, con los mismos elementos que **resultado**, de nombre **incumbent**, para cuyos vectores y matrices se reserva también memoria. En **incumbent** se encuentra la solución cuya

vecindad (para el movimiento de cambiar una asignación de una máquina a una operación) es explorada.

fase31 reserva memoria para las matrices y el vector citados, y también para **asigplan**, **asigopermaq** y **xik**, en los que se va almacenando la solución que en cada instante va configurando el programa, y para el vector **capac**, en el que se van anotando los nuevos tiempos de ocupación de las máquinas asociados a la nueva solución que se está creando. Una vez obtenida dicha solución, se evalúa la función objetivo para la misma, y si ese valor es menor que el que hay en **fom**, éste valor es sustituido por aquél, así como esta última solución obtenida reemplaza a la que contenían **asigplanm**, **asigopermaqm**, y **xikm**, y es sustituido asimismo el contenido de **capacm** por el de **capac**, y también reemplazan estos vectores y matrices a los que contenía **incumbent**. Igualmente se reserva memoria en **fase31** para las dos listas tabú, **tabugrand** (para planes y piezas) y **tabupeq** (para máquinas, operaciones, planes y piezas), y para dos matrices intermedias asociadas a la solución que va obteniendo el algoritmo, y que son necesarias para conseguir totalmente esa solución: **wij**, que contiene el número de operaciones que efectúa cualquier máquina i sobre cualquier pieza j, y **bik**, donde se guarda el ahorro en costes de transporte en términos absolutos resultante de destinar cada máquina i a cualquier célula k.

Para que el algoritmo propiamente dicho de esta fase en esta variante comience a operar es necesario construir una primera solución, lo que se logra con la llamada desde **fase31** a la función **inicial**.

Lo primero que realiza **inicial** es asignar un plan de proceso para cada pieza, es decir, configurar un primer **asigplan**. Así, para la ejecución del programa para el problema de 5 piezas y 4 máquinas del que se habló en capítulos anteriores, este primer **asigplan** resultó ser:

$$\begin{vmatrix} 2 & 2 & 3 & 1 & 1 \end{vmatrix}$$

cuya interpretación ya se ha mencionado: a la primera pieza se le asigna el segundo de sus tres planes, también el segundo plan para la segunda pieza, el tercero de sus planes a la tercera, etcétera.

Tras concluir la asignación de planes, en **inicial** se da paso a la adjudicación aleatoria de máquinas a operaciones, o lo que es lo mismo, a la configuración de **asigopermaq**. Cada vez que se elige al azar una máquina de entre las posibles para realizar una operación, se comprueba, sumando a la componente correspondiente del vector **capac** el tiempo que va a necesitar la máquina para realizar esa operación, si se viola la restricción de capacidad, y si es así se sortea otra máquina. Este tipo de repeticiones en los sorteos, cuya causa es la búsqueda lo más rápida posible de cualquier primera solución admisible, no se hará en las funciones (**cambiaplan** y **cambiamaq**) que realizan el algoritmo propiamente dicho, puesto que en éste interesa que se explore el menor número de soluciones posible, así que una solución (y también una zona de soluciones próximas a ella) será desechada en cuanto viole alguna restricción. Si, en cambio, se cumple la restricción de capacidad, se hacen las anotaciones correspondientes en **asigopermaq**, **capac** y **tabupeq**. Antes de elegir una máquina para que realice una operación, se comprueba (en previsión de que el programa no entre en un bucle sin fin) si entre todas las máquinas posibles para realizar la operación existe al menos una que no supere la capacidad máxima al adjudicársele aquélla. Si no existe ninguna, se borran las anotaciones efectuadas por **inicial** en las matrices y vectores citados, y se devuelve el control a **fase31** con el valor de retorno correspondiente para que se inicie de nuevo el proceso de búsqueda de la primera solución, de tal forma que **fase31** volverá a llamar a **inicial** hasta que la primera solución sea completada.

Veamos cómo opera **inicial** para el ejemplo citado y para las dos primeras piezas:

Para la primera pieza, y para la única operación (op = 0) de su único plan, se le asigna la tercera máquina (i = 3), y, tomando datos de la matriz **tiempos** y el vector de demanda **D**, el resultado de la comprobación de capacidad es:

4 * 64 = 256

Con esta notación, haciendo lo mismo para la segunda pieza, cuyo plan tiene cuatro operaciones, resulta:

op = 0 , i = 2, 3 * 87 = 261

op = 1, i = 4, 9 * 87 = 783

op = 2, i = 1, 8 * 87 = 696

op = 3, i = 4, 783 + 9 * 87 = 1566 > 1000

Esta última asignación no es admisible, y el programa sortea una nueva máquina:

op = 3, i = 3, 256 + 4 * 87 = 604

Así, para el vector **asigplan** anterior, se llegó a la siguiente **asigopermaq**:

3	0	0	0
2	4	1	3
2	0	0	0
3	0	0	0
4	0	0	0

Con el correspondiente vector **capac**:

696	686	750	786

Las anotaciones para **tabupeq** quedaron de la forma:

0	0	0	10	0	0	0	10	10	0	0	0	0	0	0	0	0	0	10	0	0	0	0	
0	0	0	0	0	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	

Si se completa la asignación de máquinas a operaciones (o sea, si se obtiene una primera **asigopermaq**) puede garantizarse que la solución que se está formando será

completada. Así, tras hacer las anotaciones precisas en **tabugrand**, que, para nuestro ejemplo, quedaron de la forma:

0	4	0
0	4	0
0	0	4
4	0	0
4	0	0

se realiza desde **inicial** la llamada a la función **calcfo**, que evalúa, obteniendo de paso la matriz **xik**, la función objetivo para la solución creada. Para ello, **calcfo** evalúa previamente el coste total de fabricación de las piezas (sin tener en cuenta el transporte) para esos **asigplan** y **asigopermaq** (se obtiene un valor, para el ejemplo que estamos viendo, de 28994). Y entonces, para lograr el coste asociado al transporte de las piezas, **calcfo** efectúa las llamadas a **cambiawij**, **cambiabik**, **sumabik** e **intercel**.

La función **cambiawij** se encarga de obtener la matriz **wij** que corresponde a la **asigopermaq** que se ha logrado, y lo mismo cabe decir de **cambiabik** para conseguir **bik** a partir de **wij** y **bin**.

Para nuestro ejemplo, **wij** resultó ser:

0	1	0	0	0
0	1	1	0	0
1	1	0	1	0
0	1	0	0	1

Y **bik**:

$$\begin{vmatrix} -435 & 0 \\ -860 & 0 \\ -755 & -146 \\ -435 & -4 \end{vmatrix}$$

El coste que corresponde al transporte de piezas se puede expresar de la siguiente forma, en la que se define **bik**:

$$\sum_k \sum_{j\in J(k)} h_j \sum_i D_j \cdot w_{ij} \cdot (1 - x_{ik}) = \sum_k \sum_{j\in J(k)} \sum_i h_j \cdot D_j \cdot w_{ij} \cdot (1 - x_{ik})$$
$$= \sum_k \sum_i \left(\sum_{j\in J(k)} h_j \cdot w_{ij} \right) \cdot (1 - x_{ik}) = \sum_k \sum_i b_{ik} \cdot (1 - x_{ik})$$

que se descompone en los sumandos:

$$\sum_k \sum_i b_{ik} - \sum_k \sum_i b_{ik} \cdot x_{ik}$$

sumabik se encarga de añadir el primero de los sumandos a la función objetivo (en nuestro ejemplo, esta función arroja un valor de 2635, que sumado al valor acumulado de la función objetivo resulta 31629).

La misión de la función **intercel** es preparar la información contenida en **bik** para que una función de librería llamada **RelaxIV** resuelva un problema de minimización del coste del flujo en una red, así como traducir la salida de **RelaxIV** a una matriz **xik**, y calcular y añadir el segundo de los sumandos a la función objetivo. En el modelo que prepara **intercel**, cada máquina tiene un nodo que la representa, e igualmente hay un nodo por cada célula. El número de nodos se completa con un nodo de salida. Por los arcos circula un determinado número de máquinas. Los nodos correspondientes a cada máquina son los nodos de entrada, y reciben cada uno la inyección de una máquina (a la que representan), y hay un arco conectando cada uno de dichos nodos con cada uno de los nodos que representan a las células. El coste en esos arcos es la entrada correspondiente de **bik**, es decir, el ahorro absoluto que supone en términos de transporte el destinar la máquina i a la célula k. La capacidad máxima de

estos arcos es de una máquina, lo cual se cumplirá siempre por construcción, ya que sólo se inyecta una máquina en el origen de todos los arcos que parten del nodo de ésta. Que por el arco que conecta al nodo de la máquina i con el nodo de la célula k circule una máquina, significa que la máquina i ha sido destinada a la célula k, y por el resto de los arcos que conectan al nodo de esa máquina i con el resto de los nodos correspondientes a las células que no son la célula k, circularán (también por construcción) cero máquinas. Para que se cumplan las restricciones de capacidad de las células, de cada nodo que corresponde a cada una de ellas se detrae el número mínimo de máquinas que puede albergar una célula. Además, cada uno de dichos nodos se conecta al nodo de salida mediante un arco con capacidad mínima igual a cero (con lo cual, ya que se detrajo el número mínimo de máquinas por célula en su nodo de origen, dicho mínimo queda garantizado), y con capacidad máxima igual al número máximo de máquinas por célula menos el número mínimo, lo cual asegura que no se excede el máximo de máquinas por célula. El coste en estos arcos es cero. Al nodo de salida llegará el número de máquinas que se inyectó en la entrada (es decir, el número total de máquinas) menos el total que se detrajo en los nodos de las células.

Así, la función **RelaxIV** toma una serie de vectores y el número de nodos y arcos como parámetros. El primero de los vectores (**origen**), de dimensión igual al número de arcos, y con cada componente representando un arco, indica el nodo de origen de cada uno de los arcos. El orden puede ser cualquiera, y aquí tomamos primero el arco que sale de la primera máquina a la primera célula, después el que sale de la segunda máquina a la primera célula, y así sucesivamente hasta completar todos los arcos que unen a las máquinas con la primera célula. Después, de la misma forma, siguen los arcos que llegan a la segunda célula, después los de la tercera, etc. Terminada la serie de arcos que salen de las máquinas, le siguen el arco que sale de la primera célula, y después el de la segunda célula, hasta completar las células. Asimismo, numeramos los nodos de forma que el 1 corresponde a la primera máquina, el 2 a la segunda, etc. El siguiente número, acabada la numeración de las máquinas, corresponde a la primera célula, el siguiente a la segunda, etc. En el ejemplo se tienen 4 máquinas, luego al nodo correspondiente a la primera célula corresponde el número 5. El último nodo es el de salida (el 7, en el ejemplo). Puesto que tenemos 4 máquinas y 2 células, habrá 4 + 2 +1 = 7 nodos, y 4 * 2 + 2 = 8 arcos. Para la primera solución ejemplo, **origen** es:

1	2	3	4	1	2	3	4	5	6

Todos los vectores que aparecen a continuación también tienen una componente representando a cada arco, de forma coherente con el vector **origen**, es decir, por ejemplo, la tercera componente de todos estos vectores se refiere al arco que sale de la tercera máquina y llega a la primera célula. Así, hay otro vector (**destino**) con los nodos de destino de cada arco:

5	5	5	5	6	6	6	6	7	7

El vector **coste**, posee en cada componente el coste asociado al arco al que representa. Los costes, como se ha citado, se extraen de **bik**, que aparece junto al vector **coste** en esta ilustración:

-435	0
-860	0
-755	-146
-435	-4

-435	-860	-755	-435	0	0	-146	-4	0	0

Por fin, en el vector **capacidad**, se suministran a **RelaxIV** las capacidades de los arcos:

1	1	1	1	1	1	1	1	2	2

RelaxIV devuelve, entre otros, un vector con el flujo de máquinas que circulan por cada arco, y que es traducido por la función **intercel** a **xik**. Representamos el vector de flujo y la consiguiente matriz **xik**:

$$\begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{vmatrix}$$

Con **xik** y **bik**, **intercel** calcula el segundo sumando de los costes de transporte, que para nuestro ejemplo toma un valor de –2054, por lo que la función objetivo vale al fin para esta primera solución:

fo = 29575

La función **calcfo** devuelve **xik**, junto con el valor de la función objetivo para la solución obtenida, a **inicial**, que, puesto que se trata de la primera solución, llama a la función **copia**, que almacenará esta primera solución en **asigplanm**, **asigopermaqm** y **xikm**, y el valor correspondiente de la función objetivo en **fom**, así como el tiempo de ocupación de las máquinas en **capacm**. Tras esta llamada, **inicial** devuelve el control a **fase31**.

Obtenida la primera solución, no se volverá a llamar a **inicial,** y comienza el algoritmo propiamente dicho para esta variante de la tercera fase. **fase31** llamará, según el valor alcanzado por los parámetros de control, a las dos funciones, **cambiaplan** y **cambiamaq**, que realizan un cambio en la asignación de planes de proceso, y un cambio en la asignación de operaciones a máquinas, respectivamente. **fase31** llama primero a **cambiaplan**, lo que se considera el comienzo de una iteración, tras lo cual se realizan sucesivas llamadas a **cambiamaq** (o lo que es lo mismo: se explora la vecindad

asociada a un determinado **asigplan**), hasta que se supera un número máximo de soluciones (cuya cuenta se acumula en **contest**) que no mejoran el valor de **fom** para ese **asigplan**. Entonces **fase31** da comienzo a una nueva iteración llamando a **cambiaplan**.

cambiaplan sortea una pieza, y le asigna aleatoriamente un plan. Si la entrada de **tabugrand** correspondiente a esa pieza y ese plan contiene una iteración hasta la cual está prohibido efectuar ese movimiento, y esa iteración es posterior a la actual, se da por terminada la iteración (se incrementa **contiter** y **contsegur**) y se devuelve el control a **fase31**, que volverá a llamar otra vez a **cambiaplan**. Si el valor de la iteración contenido en la componente citada de **tabugrand** es anterior a la actual, **cambiaplan** anota la iteración, en esa componente de **tabugrand**, hasta la cual estará prohibido asignar ese plan a esa pieza, y llama a la función **copia** para que sitúe en **asigplan**, **asigopermaq**, y **capac** los valores que contienen **asigplanm**, **asigopermaqm**, y **capacm** de **resultado**, para efectuar en aquéllos el movimiento de cambiar la asignación de un plan a una pieza. Es decir: solución contenida en **resultado** es, en realidad, una "incumbent" o "actual" de los movimientos de cambiar la asignación de algún plan a alguna pieza.

cambiaplan realiza entonces la asignación aleatoria de máquinas a las operaciones del nuevo plan. **cambiaplan**, a diferencia de **inicial**, no comprueba si hay máquinas con capacidad suficiente para realizar adicionalmente la operación para la cual está sorteando una máquina. Si una máquina no cumple la restricción de capacidad al adjudicársele una operación, el control es devuelto, incrementando previamente **contiter** y **contsegur**, a **fase31**, que volverá a llamar a **cambiaplan** hasta que se logre completar una solución con un nuevo plan. Cabe decir lo mismo si la máquina sorteada, la operación, el plan y la pieza están prohibidos en **tabupeq** hasta una iteración posterior a la actual como atributos de un movimiento. Conforme se van asignando máquinas a las nuevas operaciones, se van corrigiendo **asigopermaq** y **capac**, y también se van haciendo las anotaciones precisas en **tabupeq**, aunque no llegue a completarse una solución, puesto que, como se mencionó anteriormente, interesa prohibir grandes zonas de soluciones, y se considera que, si una solución no llega a formarse por incumplir alguna restricción, es posible que algunas de sus vecinas también sean conflictivas. Lograr una solución completa para un nuevo **asigplan**, como ocurría en **inicial**, pasa por lograr una nueva **asigopermaq**.

Para el ejemplo que estamos siguiendo, a la quinta pieza se le asigna su segundo plan, con lo que la cuarta máquina queda liberada de la operación del primer plan de la quinta pieza, y la componente de **capac** correspondiente valdrá:

786 – 3 * 1 = 783

El nuevo plan tiene dos operaciones, de las cuales la primera resulta asignada a la cuarta máquina, de la que se realiza una comprobación de capacidad:

783 + 9 * 1 = 792

E igualmente para la segunda operación y la tercera máquina:

750 + 7 * 1 = 757

Puesto que se cumplen las restricciones de capacidad, tenemos unos nuevos **asigplan**, **asigopermaq** y **capac**, que son, respectivamente:

2	2	3	1	2

3	0	0	0
2	4	1	3
2	0	0	0
3	0	0	0
4	3	0	0

696	686	757	792

Y las nuevas **tabugrand** y **tabupeq**:

0	4	0
0	4	0
0	0	4
4	0	0
4	5	0

0	0	0	10	0	0	0	10	10	0	0	0	0	0	0	0	0	0	10	0	0	0	0
0	0	0	0	0	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	10	11	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	11

A continuación se realiza la llamada a **calcfo**. Si el valor que toma la función objetivo para la nueva solución es menor que el que se encuentra en **fom**, se llama a **copia** para que sitúe a los nuevos **asigplan**, **asigopermaq**, **xik** y **capac** en los **asigplanm**, **asigopermaqm**, **xikm** y **capacm** de **resultado**, y el valor **fo** correspondiente de la función objetivo en **fom**, y se incrementa **contiter**, mientras que **contsegur** se pone a cero. También los nuevos **asigplan**, **asigopermaq**, **xik**, **capac** y **fo** se copian en los **asigplanm**, **asigopermaqm**, **xikm**, **capacm** y **fom** de **incumbent**. Si no mejora **fom**, se incrementan **contiter** y **contsegur**, y sólo se copia la solución en **incumbent**. Esto es: tanto si mejora la función objetivo como si no lo hace, si se completa una nueva solución en **cambiaplan**, ésa pasará a ser la actual (que se almacena en la estructura **incumbent**) de los movimientos de corto alcance.

Éstas son las nuevas matrices **wij**, **bik** y **xik**, por este orden, que se obtienen:

$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{vmatrix}$$

$$\begin{vmatrix} -435 & 0 \\ -860 & 0 \\ -759 & -146 \\ -435 & -4 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{vmatrix}$$

Para esta solución, el nuevo valor de la función objetivo es:

fo = 29727

por lo que en este caso no se llama a **copia** para que inserte la nueva solución en **resultado**, pero sí para que lo haga en **incumbent**.

Con la solución para el nuevo **asigplan**, se realizan las sucesivas llamadas a **cambiamaq**.

cambiamaq comienza llamando a **copia** para que los **asigplanm**, **asigopermaqm** y **capacm** de **incumbent** sean copiados en **asigplan**, **asigopermaq** y **capac** para ser modificados. En **cambiamaq**, se sortean sucesivamente una pieza, una operación del plan que ésta tiene asignado, y una máquina para efectuar dicha operación. Si estos cuatro atributos del movimiento de corto alcance corresponden a una componente de **tabupeq** que almacena una iteración posterior a la actual, se devuelve el control a **fase31**, incrementando, para evitar ciclos sin fin, **contest**. Si la iteración almacenada en **tabupeq** es anterior a la actual, pero no se cumple la restricción de capacidad, se realiza la anotación correspondiente en la lista y se devuelve el control a **fase31**, también incrementando **contest**. Si la iteración es anterior, y la nueva máquina cumple la restricción de capacidad, se realiza el cambio correspondiente en **asigopermaq** y **capac**, y se prohíbe ese movimiento hasta **T** iteraciones posteriores en **tabupeq**, y se realiza la llamada a **calcfo**. Nuevamente, si el valor que toma la función objetivo para la nueva solución es menor que el que se encuentra en **fom**, se llama a **copia** para que sustituya la solución de **resultado** y la de **incumbent** por la nueva solución obtenida, y se pone a cero a **contest**, y también se incrementa **contiter**, puesto que hay una nueva solución en **resultado**. Si no mejora **fom**, simplemente, sin copiar la solución en ningún sitio, se incrementa **contest**.

Con el ejemplo que nos sirve de ilustración, se realiza una llamada a **cambiamaq**, en la que se intenta realizar el movimiento caracterizado por pieza, plan, operación y máquina: 5, 2, 0, 4, y que es rechazado por encontrarse en **tabupeq** prohibido hasta una iteración posterior.

En la segunda llamada a **cambiamaq**, se asigna la cuarta máquina para realizar la operación del plan de la primera pieza, atributos que no aparecen prohibidos en **tabupeq**. Se realiza la comprobación de capacidad:

792 + 1 * 64 = 856

con lo que el movimiento es aprobado. La tercera máquina es liberada de esta tarea, por lo que su tiempo de ocupación pasa a valer:

757 – 64 * 4 = 501

Los nuevos **asigopermaq** y **capac** son:

$$\begin{vmatrix} 4 & 0 & 0 & 0 \\ 2 & 4 & 1 & 3 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 \end{vmatrix}$$

$$\begin{vmatrix} 696 & 686 & 501 & 856 \end{vmatrix}$$

Se anota en **tabupeq** la iteración hasta la cual no son admisibles los atributos del movimiento realizado:

$$\begin{vmatrix} 0 & 0 & 0 & 10 & 0 & 0 & 0 & 10 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11 & 0 & 10 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 11 \end{vmatrix}$$

Y las matrices **wij**, **bik** y **xik** son, respectivamente:

$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{vmatrix}$$

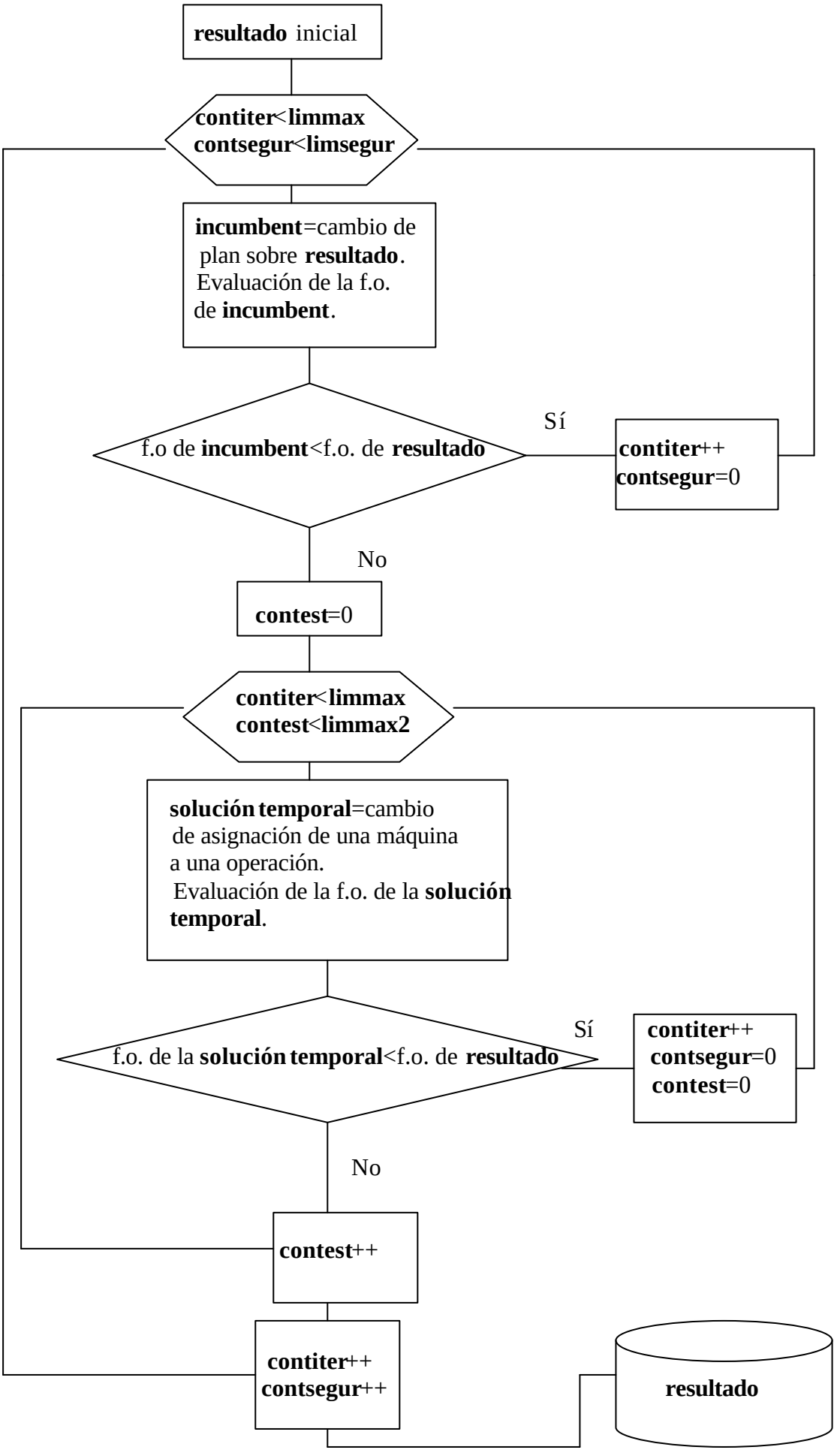
$$\begin{vmatrix} -435 & 0 \\ -860 & 0 \\ -435 & -150 \\ -755 & -4 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

El valor de la función objetivo para esta solución es:

fo = 31565

Por lo que tampoco en este caso sufre modificaciones la mejor solución obtenida.


resultado inicial
contiter<**limmax**
contsegur<**limsegur**
incumbent=cambio de plan sobre **resultado**. Evaluación de la f.o. de **incumbent**.
f.o de **incumbent**<f.o. de **resultado**
Sí
contiter++
contsegur=0
No
contest=0
contiter<**limmax**
contest<**limmax2**
solución temporal=cambio de asignación de una máquina a una operación. Evaluación de la f.o. de la **solución temporal**.
f.o. de la **solución temporal**<f.o. de **resultado**
Sí
contiter++
contsegur=0
contest=0
No
contest++
contiter++
contsegur++
resultado

8. TERCERA FASE: 2ª VARIANTE

En este capítulo se pretende, de forma similar al anterior, llevar a cabo una descripción de los aspectos conceptuales de la segunda variante de la tercera fase del algoritmo, que se distingue de la primera en que se permite realizar por varias máquinas cada operación de los planes de fabricación asignados a las piezas. El objetivo de esta variante de la última fase del algoritmo es, por tanto, determinar la célula de destino de cada máquina, elegir los planes de fabricación de cada pieza, y determinar qué fracción de las operaciones efectúa cada máquina. De la misma forma que en anteriores capítulos, la descripción mencionada irá acompañada de ilustraciones basadas en resultados del programa de sencilla representación, resultados que estarán originados en los ejemplos de las fases primera y segunda y de la generación de problemas, y se detallará e ilustrará la implementación en el programa de esta variante de la última fase del algoritmo.

8.1 Modelo del problema

Como se ha mencionado más arriba, dicha variante resulta de relajar condiciones del modelo de la primera variante. Así, se va a permitir que, aunque el plan de proceso asignado a una pieza sea único, cada una de sus operaciones pueda ser procesada por más de una máquina. Con ello se consiguen una mejor utilización de la capacidad de las máquinas, y reducciones en los costes de realización de las operaciones y del transporte de piezas entre células. Como desventaja, la gestión del sistema aumenta en complejidad, toda vez que podrán existir diferentes rutas de fabricación para fracciones de la demanda de un mismo tipo de pieza. En el programa, la asignación de los planes a las piezas se realizará, al igual que en la variante anterior, mediante un vector **asigplan** con tantas componentes como piezas, y en cada una de ellas el plan asignado a la pieza correspondiente, y para almacenar qué parte de la demanda de la pieza corresponde a cada una de las máquinas que pueden realizar las operaciones de los planes asignados, se dispone de matrices **demmaqsm**, con tantas columnas como operaciones totales haya, y tantas filas como máquinas puedan asignarse a una operación (**maqs**). Obsérvese que, como habrá operaciones de planes no asignados, en las columnas correspondientes a estas operaciones habrá ceros. Como ejemplos, que aparecerán más abajo, de **asigplan** y **demmaqsm**:

$$\begin{vmatrix} 2 & 2 & 2 & 1 & 2 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 0 & 4 & 4 & 0 & 0 & 26 & 88 & 0 & 0 & 0 & 20 & 0 & 0 & 40 & 0 & 33 \\ 0 & 4 & 0 & 0 & 0 & 0 & 62 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 88 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Para conseguir el modelo de esta variante, simplemente hay que realizar unas pequeñas modificaciones en el modelo de la variante anterior, concretamente en la función objetivo, en la definición de algunas variables, y en la formulación de ciertas restricciones.

Así pues, las variables de esta variante serán:

$$u_{jp} = \begin{cases} 1 & \text{si la parte j se procesa siguiendo el plan p} \\ 0 & \text{en caso contrario} \end{cases}$$

v'_{ijpl} Número de unidades de la pieza j que realizan la operación l de su plan de proceso p en la máquina i

w'_{ij} Número de operaciones que la máquina i realiza sobre el total de las D_j unidades de la pieza j

$$x_{ik} = \begin{cases} 1 & \text{si máquina i se asigna a célula k} \\ 0 & \text{en caso contrario} \end{cases}$$

Y en cuanto a las restricciones y a la función objetivo, se van a tener:

$$\text{Minimizar} \quad \sum_i \sum_j \sum_p \sum_l c_{ijpl} \, v'_{ijpl} + \sum_{k=1}^{C} \sum_{j \in J(k)} h_j \sum_i w'_{ij} (1 - x_{ik})$$

sujeto a

$$\sum_{k=1}^{C} x_{ik} = 1 \qquad \forall i$$

$$\sum_p u_{jp} = 1 \qquad \forall j$$

$$\sum_{i \in I(j,p,l)} v'_{ijpl} = D_j \, u_{jp} \qquad \forall j, p, l$$

$$\sum_p \sum_l v'_{ijpl} = w'_{ij} \qquad \forall i, j$$

$$\sum_{(j,p,l) \in O(i)} t_{ijpl} \, v'_{ijpl} \leq H_i \qquad \forall i$$

$$M_{\min} \leq \sum_i x_{ik} \leq M_{\max} \qquad \forall k$$

$$x_{ik}, u_{jp} = \{0,1\} \qquad w'_{ij}, v'_{ijpl} \geq 0$$

La aparición de variables continuas es lo que diferencia principalmente a esta formulación del modelo original. Puede ser usada de nuevo una estrategia de resolución a dos niveles. Las variables de selección de los planes de proceso para las piezas y asignación de máquinas a las células (es decir: las variables binarias) se exploran mediante Búsqueda Tabú, y esto representa el primer nivel de la estrategia de resolución, mientras que el segundo nivel consiste en resolver un modelo de programación lineal, determinando así las variables continuas. Lo que este enfoque a dos niveles tiene de particular es que nada impide que para ciertos valores de las variables binarias no puedan ser halladas soluciones admisibles en el problema

continuo, y entonces consideramos como no admisibles dichos valores de las variables binarias.

Para identificar a qué célula pertenecerá cada máquina, se dispone, como en la variante anterior, de una matriz binaria **xik** con tantas filas como máquinas y tantas columnas como células, de forma que, si una componente vale 1, significa que la máquina de la fila de esa componente pertenece a la célula de la columna de la componente citada, y que no pertenece si vale 0. Entre los ejemplos que veremos se tendrá esta **xik**:

$$\begin{vmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{vmatrix}$$

8.2 Programación de la 2ª Variante de la Tercera Fase y Diagrama de Flujo

La sección del programa correspondiente a la segunda variante de la tercera fase se gestiona mediante la función **fase32**. Entre los tipos de argumentos que toma esta función se encuentran parámetros de control, dimensiones del problema, y vectores y matrices resultantes de la generación de problemas y de las dos primeras fases. Devuelve una estructura (**resultado2**) que contiene a la mejor solución obtenida por el algoritmo con esta variante para la tercera fase, junto con el valor (**fom**) de la función objetivo para esa solución. La solución está formada por un vector y dos matrices: un vector (**asigplanm**) y una matriz (**xikm**) con el mismo significado que los elementos con el mismo nombre de la solución de la variante anterior, esto es, el primero indica el plan asignado para fabricar cada pieza, y la segunda a qué célula se destina cada máquina, y también se incluye en la solución una matriz (**demmaqsm**) donde se almacena qué parte de la demanda de la pieza corresponde a cada una de las máquinas que pueden realizar las operaciones de los planes asignados. **resultado2** es devuelto por **fase32**, al igual que ocurría en la primera variante, cuando se ha superado, o bien un

número máximo de iteraciones exploradas (cuya cuenta se lleva en la variable **contiter**), o bien, para asegurarnos de que el programa no realiza demasiadas iteraciones superfluas, si supera un número máximo de iteraciones exploradas sin que mejore **fom** (y la cuenta de esas iteraciones se almacena en **contsegur**).

fase32 reserva espacio para estos elementos, y, de forma equivalente a lo que ocurría en la variante anterior, para la estructura **incumbent2**, donde se almacenan los datos necesarios de la solución cuya vecindad es explorada, esto es, en **incumbent2** se incluyen también un vector **asigplanm** y una matriz **xikm**. Por no ser necesario, en **incumbent2** no se guarda el valor de la función objetivo para esa solución, ni una matriz **demmaqsm**. De forma análoga a como ocurría en la primera variante, se reserva memoria para **asigplan** y **xik**, que guardan en cada instante esos elementos que caracterizan a las soluciones, y que proceden de un movimiento efectuado, bien sobre la solución de **resultado2**, bien sobre la que se encuentra en **incumbent2**. Una vez efectuado el movimiento sobre la copia de **asigplanm** y **xikm** pertenecientes a **incumbent2** o **resultado2** que se guarda en **asigplan** y **xik**, se resuelve si es posible el problema de programación lineal que se forma para esos **asigplan** y **xik**, evaluando la función objetivo caso de ser resuelto, y si el valor obtenido es menor que el que existe en **fom**, esos dos elementos sustituyen a los que se encuentraban en **resultado2**, y **asigplan** y **xik** sustituyen a sus homólogos de **incumbent2**, y también se construirá **demmaqsm**. Asimismo, también en **fase32** se reserva memoria para **tabuplan** y **tabumaq**, listas tabú de asignación de planes, y de ubicación de máquinas en células, respectivamente.

También en esta variante se tiene una función (**inicial2**) que se encarga de caracterizar una primera solución, aunque no de completarla totalmente: sólo se configuran unos primeros **asigplan** y **xik**, pero no se halla una matriz **demmaqs**. Para ello, **inicial2** sortea, para cada máquina, la célula a la que irá destinada. Cuando es sorteada una célula para una máquina, se comprueba si ésta se encuentra al máximo de su capacidad. Si es así, **inicial2** sortea otra célula para esa máquina. Si la célula no contiene el número máximo permitido de máquinas, se asigna la máquina en cuestión a esa célula, poniendo un 1 en el lugar correspondiente de **xik**, que inicialmente sólo contiene ceros. También se prohíbe asignar dicha máquina a esa célula por un número **T** de iteraciones, lo cual se realiza sumando **T** a la componente que corresponde de **tabumaq**, que en estos momentos se encontrará a cero. Así se opera sucesivamente

hasta haber destinado todas las máquinas. Se comprueba entonces si la matriz **xik** que se acaba de configurar indica que alguna célula se encuentra por debajo del mínimo de su capacidad, y si es así, se vuelve a comenzar desde el principio el proceso de formación de la primera **xik**. Si se vuelve a iniciar este proceso de obtención de la primera **xik**, son borradas las anotaciones que se habían hecho en **tabumaq**, puesto que no tiene sentido prohibir movimientos de soluciones que no han sido exploradas. Respecto al proceso de configuración de un primer **asigplan**, éste es mucho más sencillo que el anterior, puesto que no hay comprobaciones que realizar, y será completado a la primera. Sencillamente se trata de sortear un plan de entre los posibles para cada una de las piezas, y prohibir esas asignaciones en **tabuplan** por un número **T2** de iteraciones. Tras la llamada a **inicial2**, **fase32** llama a la función **copia2** para que sitúe estos primeros **xik** y **asigplan** en **xikm** y **asigplanm** de **resultado2**.

Para el ejemplo que estamos analizando, **xik**, **asigplan**, **tabumaq** y **tabuplan** quedan, en este orden y tras la llamada a **inicial2**, de esta forma:

$$\begin{vmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{vmatrix}$$

$$\begin{vmatrix} 2 & 2 & 2 & 1 & 2 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 2 \\ 2 & 0 \\ 0 & 2 \\ 0 & 2 \end{vmatrix}$$

0	3	0
0	3	0
0	3	0
3	0	0
0	3	0

Tras la obtención de los primeros **xikm** y **asigplanm**, comienza la parte del algoritmo propiamente dicha para esta variante de la tercera fase. Ello consiste en que el programa realice movimientos en la solución contenida en **resultado2**, movimientos que serán las iteraciones. También en las iteraciones, si se dan las condiciones que se verán, se anotará la solución modificada en la estructura **incumbent2** para que sea explorada la vecindad de ésta.

Las iteraciones comienzan con un movimiento sobre la solución guardada en **resultado2**. Los movimientos son de dos tipos, y en cada iteración se realiza uno solo de ellos, realizándose el otro tipo en la iteración siguiente. Uno de esos tipos de movimientos estriba en modificar la célula a la que pertenece una máquina (lo cual es llevado a cabo por la función **cambiacel**), y el otro tipo consiste en cambiar la asignación de un plan a una pieza (ídem **cambiaplan2**).

Como ilustración, para el ejemplo anterior la primera iteración consistió en trasladar la tercera máquina de la segunda a la primera célula. Como es lógico, esto tiene su efecto en la lista tabú, que también modifica **cambiacel**, por lo que **xik** y **tabumaq** quedan de la forma:

0	1
1	0
1	0
0	1

$$\begin{vmatrix} 0 & 2 \\ 2 & 0 \\ 3 & 2 \\ 0 & 2 \end{vmatrix}$$

Aparte de la llamada a **cambiacel** o **cambiaplan2**, los pasos dados en las iteraciones por el programa para ambos tipos de movimientos son los mismos. En primer lugar, en cada iteración se llama a **copia2** para situar **xikm** y **asigplanm**, pertenecientes a **resultado2**, en **xik** y **asigplan**. Después, tras la llamada a **cambiacel** o **cambiaplan2** para que se efectúe respectivamente un movimiento sobre **xik** o **asigplan**, se llama a la función **gestsimp2**, cuyo funcionamiento se explicará a continuación, y que es la encargada de preparar y resolver el problema de programación lineal que resulta de esos **xik** y **asigplan**. Si el problema de programación lineal es incompatible (acotado siempre lo será, puesto que hay una capacidad límite para todas las máquinas), se considera que otras soluciones vecinas a la que está caracterizada por esos **xik** y **asigplan** pueden ser también conflictivas, por lo que, en ese caso, no se explora la vecindad de dicha solución, se incrementan **contiter** y **contsegur**, y se da paso al comienzo de una nueva iteración. Si el problema puede resolverse, caben dos posibilidades: una, que el valor de la función objetivo para esa solución sea menor que el que se encuentra en **fom**, y entonces dicha solución se guarda en **resultado2**, y **xik** y **asigplan** en **incumbent2** para que sea explorada la vecindad de la solución, y dos, que no sea menor, y entonces sólo se hace la copia en **incumbent2**. En la primera de las dos posibilidades, se incrementa **contiter** y se pone a cero **contsegur**, y en la segunda se incrementan ambos. Únicamente si se hace, por una u otra causa, la copia en **resultado2**, se traduce el resultado de aplicación del simplex a la matriz **demmaqsm**. De modo que, para la parte entera de la solución (**xik** y **asigplan**), que ha resultado compatible y forman parte de la primera solución que evalúa el algoritmo, se tiene esta **demmaqsm**:

0	0	4	4	0	0	26	88	0	0	0	20	0	0	40	0	33
0	4	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	88	0	0	0	0	0	0	0	0	0	0	0

donde se observa que la segunda operación del plan asignado a la segunda pieza es realizada, sobre 88 piezas, 26 veces en la máquina segunda y 62 en la tercera.

Si procede, es decir, si se ha realizado la copia mencionada en **incumbent2**, se explora la vecindad de la solución ahí almacenada, iniciándose dicha exploración con la puesta a cero de **contest**. Los pasos que sigue el programa en la exploración de una vecindad son muy similares a los dados en las iteraciones, con la solución caracterizada por los elementos de **incumbent2** en el papel que hacía la que se encontraba en **resultado2**. Así, primero se copia **xikm** y **asigplanm** de **incumbent2** en **xik** y **asigplan**, para ser modificada **xik** por **cambiacel**, y **asigplan** por **cambiaplan2**, cada vez uno de los dos y alternativamente. Tras la llamada, bien a **cambiacel**, bien a **cambiaplan2**, se llama a **gestsimp2** para que resuelva el problema de programación lineal asociado a **xik** y **asigplan**. Si el problema creado es incompatible, o si no lo es pero el valor de la función objetivo asociado a la solución completa es menor que el que se encuentra en **fom**, lo único que ocurre es que se incrementa **contest**. Si el problema es resoluble y el valor obtenido de la función objetivo para la solución es menor que el valor de **fom**, es como si diera comienzo una nueva iteración: se copian **xik** y **asigplan** de esa solución en **resultado2** y en **incumbent2**, y el valor asociado de la función objetivo en **fom**, y se incrementa **contiter**, mientras que **contest** y **contsegur** se ponen a cero, pasando a ser explorada la vecindad de esa solución.

En nuestro ejemplo, con la llamada a **cambiaplan**, se producen los cambios en **asigplan** y **tabuplan** siguientes:

$$\begin{vmatrix} 0 & 3 & 0 \\ 5 & 3 & 0 \\ 0 & 3 & 0 \\ 3 & 0 & 0 \\ 0 & 3 & 0 \end{vmatrix}$$

$$\begin{vmatrix} 2 & 1 & 2 & 1 & 2 \end{vmatrix}$$

Para este asigplan (con la matriz **xik** anterior) se tiene el siguiente valor de la función objetivo:

fo = 8065

Por lo que se llama a crearresul, que, entre las tareas mencionadas, creará la matriz **demmaqsm** para esta solución:

$$\begin{vmatrix} 0 & 0 & 4 & 4 & 88 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 40 & 0 & 33 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Seguidamente se detallan los funcionamientos de **cambiacel**, **cambiaplan2** y **gestsimp2**.

cambiacel, cuya misión es cambiar la pertenencia de una máquina a una célula, o lo que es lo mismo: cambiar de columna un 1 de **xik**, sortea primero la máquina a la que se cambiará de célula. Comprueba entonces si la célula en la que se encuentra la máquina posee el número mínimo de máquinas. Si no es así, sortea la nueva célula a la que será destinada la máquina, y si es así, sortea otra máquina. Cuando se ha sorteado una nueva célula para una máquina, **cambiacel** comprueba si esta célula está al máximo

de su capacidad. Si está al máximo, sortea otra célula, y si no lo está se mira por fin en **tabumaq** si el movimiento está prohibido. Si está prohibido, todo el proceso vuelve a comenzar desde el principio, y si no lo está, se modifica **xik** y se realiza la anotación oportuna en **tabumaq**.

El cometido de **cambiaplan2** consiste en efectuar un cambio en la asignación de un plan a una pieza. Para ello, sortea la pieza, y después sortea un plan para la misma. Si ese plan no está prohibido en **tabuplan** para la iteración en curso, se efectúa el cambio en **asigplan**, y se prohíbe el movimiento para las siguientes **T2** iteraciones en **tabuplan**.

gestsimp2, una vez obtenidos los nuevos **asigplan** y **xik**, llama a las funciones encargadas de preparar el problema para obtener la matriz **demmaqs** óptima para esos **asigplan** y **xik**. En concreto, se trata de formular un problema de programación lineal, con un vector de costes (**costesfo**), una matriz tecnológica (**a**), y un vector de términos independientes (**be**), para ser resuelto mediante el método simplex. Las funciones encargadas de preparar el modelo son:

- **cuentavars2**, que determina el número de variables principales o significativas asociadas a **asigplan**, y que será igual a la suma de todas las asignaciones posibles de máquinas que puedan hacerse a las operaciones de los planes asignados. Cada una de las máquinas que puedan realizar alguna operación de los planes asignados, posee una de esas variables principales del problema. Ese número se sitúa en la variable **numvars2**. Además de estas variables, hay una holgura para cada una de las máquinas, asociadas a la capacidad de éstas.

  Para el primer ejemplo, resulta:

  numvars2 = 14

- **creacosts2**, que halla el coste asociado a cada una de las variables principales anteriores. El coste asociado a que una máquina realice una operación se encuentra en la matriz **cosoper1**, y se le suma el coste de transporte asociado a la pieza si ésta y la máquina citada pertenecen a células distintas, información que se encuentra en **bin** y **xik**, respectivamente. Los

costes asociados a las holguras son nulos. Esos costes se guardan en el vector **costesfo**, que en el ejemplo vale:

29	30	105	57	100	47	102	41	15	39	59	47	24	41	0	0	0	0

- **cuentam**, que determina el número, asociado a **asigplan**, de operaciones. Por cada operación habrá una restricción. En el caso que estamos tratando:

  m = 9

- **crearestybe2**, que como su nombre indica crea la matriz tecnológica (**a**) y el vector de términos independientes (**be**). Respecto a la primera, puesto que una operación, que pertenece a un plan, y éste a una pieza, debe realizarse un nº de veces (y ese número es la suma de lo que realiza cada máquina que puede efectuar esa operación, o sea, la suma de los valores que toman las variables principales asociadas a esa operación) igual al valor de la demanda de la pieza. Los coeficientes de las variables implicadas en la restricción de una operación serán unos, y el término independiente será igual a la demanda de la pieza, por tanto. A esas variables también les corresponde un coeficiente en la restricción de capacidad de la máquina que realiza esa operación, y que es el tiempo que tarda en realizarla, valor que se encuentra en la matriz **tiempos**. Cada máquina, como se dijo anteriormente, lleva asociada una holgura, de coeficiente uno.

  En el ejemplo, la matriz **a** y el vector **be** resultaron ser:

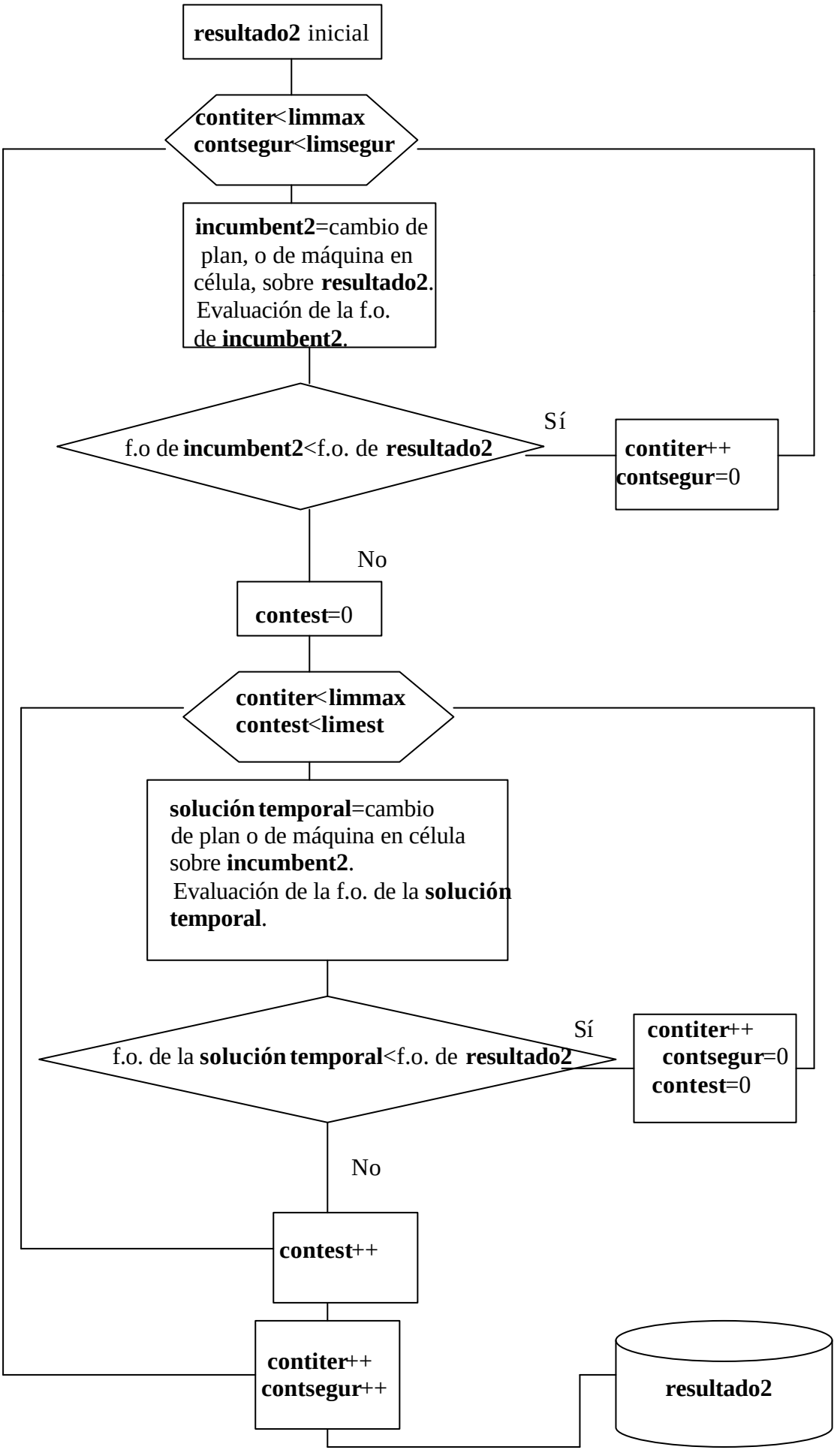
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	3	0	0	10	0	5	0	0	0	0	1	0	0
0	10	0	9	0	0	5	0	0	10	0	0	0	0	0	0	1	0
0	0	0	0	8	0	0	3	0	0	0	8	4	3	0	0	0	1

4	4	4	88	88	88	20	40	33	700	700	700	700

En el ejemplo, las cuatro últimas filas de **a** representan las restricciones de capacidad de las máquinas, y el resto a las restricciones de las operaciones. Si, a modo de aclaración, tomamos la cuarta restricción, que corresponde a la primera de las operaciones del plan asignado a la segunda pieza, cada variable (con un 1 como coeficiente) está asociada a una de las tres máquinas que pueden efectuar esa operación. La suma de esas tres variables tiene que ser igual al número de veces totales que se realiza la operación: como una operación se lleva a cabo sólo una vez por pieza, dicho número será igual a la producción de la pieza, es decir, a la demanda (88, en este caso). Cada una de esas variables tendrá un coeficiente (obtenido de la matriz **tiempos**) en la restricción de capacidad de la máquina a la que se refiera. Al ser la segunda, la tercera y la cuarta las máquinas implicadas, “debajo” de cada variable (de cada 1) aparece el tiempo que tarda en llevar a

cabo la operación en la restricción (o fila) de la máquina afectada. Las restricciones de capacidad de las máquinas presentan una holgura cada una, y están todas igualadas a **H** (capacidad máxima de las máquinas).

Con **costesfo**, **a** y **be**, **fase32** llama a **simplex1**, función que resuelve el problema de programación planteado mediante el método simplex. Si el problema era compatible, **gestsimp2** llama a **evalfo2** para que calcule el valor de la función objetivo con los valores de las variables principales que se han obtenido. Sólo si nos encontramos en la primera solución que se evalúa (la que halla **inicial2**), o si el valor de la función objetivo calculado por **evalfo2** es menor que **fom**, **gestsimp2** llamará a **crearresul2**, función que copia todos los elementos de la nueva solución en **resultado2**, y que además traduce los valores de las variables principales del problema de programación lineal a la matriz **demmaqsm**.


resultado2 inicial
contiter<**limmax**
contsegur<**limsegur**
incumbent2=cambio de plan, o de máquina en célula, sobre **resultado2**. Evaluación de la f.o. de **incumbent2**.
f.o de **incumbent2**<f.o. de **resultado2**
Sí
contiter++
contsegur=0
No
contest=0
contiter<**limmax**
contest<**limest**
solución temporal=cambio de plan o de máquina en célula sobre **incumbent2**. Evaluación de la f.o. de la **solución temporal**.
f.o. de la **solución temporal**<f.o. de **resultado2**
Sí
contiter++
contsegur=0
contest=0
No
contest++
contiter++
contsegur++
resultado2

9. TERCERA FASE: 3ª VARIANTE

El objetivo de este capítulo es describir la tercera y última variante de la tercera fase en sus aspectos teóricos. En esta variante, no sólo hay libertad para que una operación pueda ser llevada a cabo por diferentes máquinas, sino que también se permite que una pieza pueda ser fabricada por distintos planes. Se busca pues, determinar las células a las que pertenecen las distintas máquinas, e identificar qué parte de la demanda de las piezas se obtiene mediante los distintos planes, y qué fracción del total de cada operación a realizar es efectuada por las diferentes máquinas. Como en capítulos anteriores, la descripción de esta variante se ilustrará con resultados del programa sencillos de representar, y que tienen su origen en los ejemplos de las dos primeras fases y la generación de problemas, y también en este capítulo se hará explícita y se ilustrará la programación de esta última variante de la tercera fase.

9.1 Modelo del Problema

Como se ha mencionado en el párrafo anterior, la tercera variante que consideramos para esta fase surge de relajar (a partir del modelo de la segunda variante, o sea: considerando que distintas unidades de un mismo tipo de pieza pueden seguir distintas rutas de fabricación) la condición de que todas las piezas de un mismo tipo hayan de ser fabricadas según un mismo plan de proceso. De forma similar a la variante anterior, una matriz **demmaqsm**, con tantas columnas como nº total de operaciones haya y con tantas filas como nº máximo de máquinas asignables a una operación (**maqs**), indica qué parte del número de operaciones que le corresponde a cada plan satisface cada máquina. Como muestra de una de estas matrices **demmaqsm** que nos servirán de ejemplo:

40	40	41	0	0	0	67
0	0	0	21	0	0	0
0	0	0	0	0	0	0

Se puede decir lo mismo que para la variante anterior: que se mejora de nuevo tanto el uso de la capacidad de las máquinas como los costes de operación y de

transporte entre células, pero que vuelve a aumentar la complejidad de la gestión del sistema. Esta vez las modificaciones consisten únicamente en retocar las definiciones de algunas variables y la formulación de las restricciones afectadas, quedando intacta la función objetivo.

Por tanto, las variables del modelo son las que siguen:

u'_{jp} Número de unidades de la pieza j que se fabrican según el plan de proceso p

v'_{ijpl} Número de unidades de la pieza j que realizan la operación l de su plan de proceso p en la máquina i

w'_{ij} Número de operaciones que la máquina i realiza sobre el total de las D_j unidades de la pieza j

$$x_{ik} = \begin{cases} 1 & \text{si máquina i se asigna a célula k} \\ 0 & \text{en caso contrario} \end{cases}$$

Y el correspondiente modelo para esta variante resuta ser:

$$\text{Minimizar} \quad \sum_i \sum_j \sum_p \sum_l c_{ijpl}\, v'_{ijpl} + \sum_{k=1}^{C} \sum_{j \in J(k)} h_j \sum_i w'_{ij}(1 - x_{ik})$$

sujeto a

$$\sum_{k=1}^{C} x_{ik} = 1 \qquad \forall i$$

$$\sum_p u'_{jp} = D_j \qquad \forall j$$

$$\sum_{i \in I(j,p,l)} v'_{ijpl} = u'_{jp} \qquad \forall j, p, l$$

$$\sum_p \sum_l v'_{ijpl} = w'_{ij} \qquad \forall i, j$$

$$\sum_{(j,p,l)\in O(i)} t_{ijpl}\, v'_{ijpl} \le H_i \qquad \forall i$$

$$M_{\min} \le \sum_i x_{ik} \le M_{\max} \qquad \forall k$$

$$x_{ik} = \{0,1\} \qquad u'_{jp}, w'_{ij}, v'_{ijpl} \ge 0$$

Como puede verse, para este modelo son continuas casi todas las variables. Mientras que en la primera variante de esta fase la asignación de planes de proceso a las piezas, la elección de máquinas para la realización de las operaciones, y la asignación de las máquinas a las células eran descritas por variables enteras, sólo las relativas a la elección de las células en las que se incluirán las máquinas siguen siéndolo. El programa usará matrices binarias **xik** para guardar la información de a qué célula se destina cada máquina. En dichas matrices, con tantas filas como máquinas, y tantas columnas como células, un uno significa que la máquina de la fila que se trate pertenece a la célula de la columna correspondiente, y un cero si no pertenece. Una muestra de **xik** que usaremos es:

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

Se va a seguir una estrategia de resolución muy parecida a las de las dos variantes anteriores. Así pues, existen dos niveles en dicha resolución: Un primer nivel en el que se exploran, siguiendo la metaheurística Búsqueda Tabú, las variables que corresponden a las pertenencias de las máquinas a las células, esto es: las variables enteras, y un segundo nivel en el que se hallarían, mediante la resolución de un problema de programación lineal, los valores de las variables continuas.

9.2 Programación de la 3ª Variante de la 3ª Fase y Diagrama de Flujo

fase33 es la función que gestiona la tercera variante de la tercera fase del algoritmo. Entre sus argumentos se encuentran parámetros de control, dimensiones del problema, y vectores y matrices resultantes de la generación de problemas y de las dos primeras fases. Devuelve, como era el caso de las otras variantes, una estructura (**resultado3**) con la solución hallada por el algoritmo con esta variante de la tercera fase, así como el valor (**fom**) de la función objetivo para esa solución. En **resultado3**, se incluyen, por tanto: el valor **fom** citado, una matriz (**xikm**) con el mismo significado que en las variantes primera y segunda, es decir, indica a qué célula se destina cada máquina, y una matriz (**demmaqsm**) con la interpretación que se ha dado más arriba en este capítulo: qué parte del número de operaciones que le corresponde a cada plan satisface cada máquina. **resultado3** es devuelto por **fase33**, al igual que ocurría en el resto de variantes, cuando se ha superado, o bien un número máximo de iteraciones exploradas (cuya cuenta se lleva en la variable **contiter**), o bien, para asegurarnos de que el programa no realiza demasiadas iteraciones superfluas, si excede un número máximo de iteraciones exploradas sin que mejore **fom** (y la cuenta de esas iteraciones se almacena en **contsegur**).

fase33 reserva memoria para estos elementos, y, de forma similar a lo que ocurría en las variantes anteriores, para la matriz **incxik**, que caracteriza a la solución actual o incumbent cuya vecindad es explorada. De forma también análoga a lo que ocurría en las variantes primera y segunda, se reserva memoria para **xik**, que guarda en cada instante la matriz de asignación de máquinas a células de la solución que está explorando el algoritmo, y que procede de un movimiento efectuado, bien sobre la matriz **xikm** de **resultado3**, bien sobre **incxik**. Una vez efectuado el movimiento sobre la copia de **xikm** de **resultado3** o de **incxik** situada en **xik**, se evalúa, si es posible, la función objetivo para esa solución, y si el valor obtenido es menor que el que existe en **fom**, las matrices **demmaqsm** y **xik** de esta solución sustituirán a **demmaqsm** y **xikm** de **resultado3**, y además **xik** pasa a ser la nueva **incxik**. También se reserva memoria para **tabumaq**, lista tabú de ubicación de máquinas en células.

En esta variante, al igual que en la anterior, también es necesario resolver un problema de programación lineal para completar la solución, pero al contrario de lo que ocurría en la segunda variante, la matriz tecnológica es la misma para todas las

soluciones del espacio, por lo que sólo se construye una vez. Para crearla, **fase33** llama a las siguientes funciones:

- **cuentavars3**, cuyo cometido consiste en contar las variables significativas o principales del problema lineal, y cuyo número será igual al número total de asignaciones posibles de máquinas a las operaciones de todos los planes. El resultado se anota en **numvars3**. El número total de variables se completa sumándole al anterior el número de planes y el número de máquinas. Para el ejemplo que vamos a tratar:

numvars3 = 11

- **creare stybe3**, que se encarga de crear la matriz tecnológica **a** y el vector **be** de términos independientes. Las restricciones son de tres tipos:

1. El número del primer tipo de restricciones será igual al número total de operaciones, y dichas restricciones consisten en lo siguiente: la suma del número total de veces que se realiza una determinada operación por las máquinas a las que puede ser asignada, debe ser igual a un número (que será variable del modelo) que será idéntico para todas las restricciones correspondientes a todas las operaciones de un mismo plan. El valor del término independiente es cero en estos casos.

2. La suma de los números anteriormente mencionados, de los que hay uno por cada plan, para todos los planes de una misma pieza debe ser igual a la demanda de dicha pieza. Por tanto, el número de estas restricciones será igual a **P**, el número de piezas. El valor del término independiente es la demanda de la pieza en cuestión.

3. Las veces que una máquina realiza una operación acarrean un aumento del tiempo de ocupación de la máquina, que, en total, debe ser menor o igual que **H** (capacidad máxima de las máquinas). El valor del término independiente es **H**.

La matriz **a** y el vector **be** que resultaron para el ejemplo anterior fueron:

$$
\begin{vmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 9 & 5 & 0 & 7 & 8 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{vmatrix}
$$

$$
\begin{vmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 40 & 41 & 21 & 67 & 800 & 800 & 800
\end{vmatrix}
$$

Se puede decir, por lo que se acaba de exponer, que las siete primeras restricciones corresponden a las operaciones, las cuatro siguientes a los planes, y las tres últimas a las máquinas. Las restricciones quinta y sexta corresponden cada una a una operación del primer plan de la cuarta pieza. Si nos fijamos en la sexta, hay dos variables implicadas con un 1 como coeficiente: son variables principales, y el valor de cada una de ellas significa el número de veces que se lleva a cabo esa operación por una determinada máquina. La suma de esos dos números de veces que se realiza la operación debe ser igual a un número (que es variable del modelo, como se dijo anteriormente), y por tanto, al restarse este número a la suma citada, se obtendrá el valor cero, por lo que, expresando de esa forma la restricción, dicho número tendrá coeficiente –1, y el término independiente será cero. El

número citado es el mismo para cada plan, y por ello en la quinta restricción se tiene el –1 en la misma columna, y en la séptima restricción aparece en otra columna porque se trata, aunque de la misma pieza, de otro plan. La undécima restricción expresa que la suma de esos dos números (que aluden cada uno a la cantidad de la cuarta pieza que se fabrican por el plan al que representan) debe ser igual a la demanda de la pieza, y ésta es por tanto el término independiente. Asimismo cada una de las variables principales tendrá un coeficiente extraído de la matriz **tiempos** en la restricción de capacidad de la máquina a la que esté asociada. En la séptima restricción, por ejemplo, vemos, en la columna de cada 1 y en la restricción de capacidad de la máquina implicada el coeficiente tomado de **tiempos**. En cada restricción de capacidad de las máquinas hay una holgura de coeficiente 1, y el término independiente es en esas restricciones la capacidad máxima de las máquinas: **H**.

Construidos la matriz **a** y el vector **be**, **fase33** llama a la función **simplex13** para que los transforme en su forma canónica, resolviendo para ello la Fase I del simplex, cosa que, al igual que la construcción de **a** y **be**, sólo es necesario efectuar una vez.

De forma similar a lo que sucedía en las otras variantes, una función (**inicial3**) tiene como cometido caracterizar una primera solución, para lo cual sólo es necesario crear una primera **xik**, sin necesidad de construir su **demmaqsm** correspondiente. Con ese objetivo, y el proceso es idéntico al de búsqueda de la primera **xik** de la variante anterior, **inicial3** sortea, para cada máquina, la célula a la que pertenecerá. Cuando se obtiene por sorteo una célula para una máquina, se comprueba si ésta se halla ocupada ya por el número máximo de máquinas posible. De ser así, **inicial3** vuelve a sortear una célula para esa máquina. Si la célula contiene un número de máquinas inferior al máximo permitido, se destina la máquina citada a esa célula, poniendo un 1 en la componente apropiada de **xik**, matriz que inicialmente se encuentra con ceros en todas sus entradas. Realizada esa asignación, se prohíbe destinar la máquina citada a la célula anterior por un número **T** de iteraciones, lo cual se consigue sumando **T** a la componente de **tabumaq** que corresponde a las mismas fila y columna donde se ha realizado la asignación en **xik**, y que en este instante tendrá valor nulo. Esta es la forma en la que se destinan todas las máquinas a las células por primera vez. Una vez concluidas las asignaciones de todas las máquinas, se comprueba si la matriz **xik** que se

acaba de crear contiene alguna célula con un número de máquinas inferior al mínimo permitido, y si es así, se vuelve a empezar desde el principio el proceso de configuración de la primera **xik**. Si se recomienza este proceso de creación de la primera **xik**, también las anotaciones realizadas hasta el momento en **tabumaq** son eliminadas, puesto que sólo tiene significado prohibir movimientos que han llevado a soluciones que han sido exploradas. Una vez conseguida la primera **xik**, **fase33** llama a **copia3** para que guarde a dicha matriz en **xikm** de **resultado3**.

Para el ejemplo de esta variante, la primera asignación de la matriz **xik** y la lista **tabumaq** fueron:

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

$$\begin{vmatrix} 2 & 0 \\ 0 & 2 \\ 2 & 0 \end{vmatrix}$$

Cuando se ha conseguido la primera **xikm**, da comienzo verdaderamente la parte del algoritmo para esta variante de la tercera fase. Es entonces cuando el programa empieza a efectuar movimientos en la solución que se halla en **resultado3**, esto es: es cuando comienza a realizar iteraciones, y, si se cumplen ciertos requisistos, anotará la matriz **xik**, extraída de **resultado3** y modificada, en la matriz **incxik**, para que sean exploradas las soluciones vecinas a la que caracteriza dicha matriz.

Las iteraciones resultan de efectuar un movimiento sobre la solución que se halla en **resultado3**. Dicho movimiento, que es realizado por la función **cambiacel3**, consiste en cambiar la pertenencia de una máquina a una célula. Ya en la primera iteración se

efectúa un movimiento sobre la matriz obtenida de **inicial3**, y para el ejemplo el resultado en **xik** y **tabumaq** fue:

$$\begin{vmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

$$\begin{vmatrix} 2 & 3 \\ 0 & 2 \\ 2 & 0 \end{vmatrix}$$

En cada iteración el programa sigue una serie de pasos. El primero de esos pasos consiste en llamar a **copia3** para colocar en **xik** la matriz **xikm** que se encuentra en **resultado3**. Después, tras la llamada a **cambiacel3** para que perturbe **xik**, se produce la llamada a la función **gestsim3**, cuyos pormenores se detallarán seguidamente, y que tiene por misión resolver el problema de programación lineal asociado a esa **xik**. Si el problema de programación lineal es irresoluble, se considera que otras soluciones vecinas a la caracterizada por **xik** pueden ser igualmente problemáticas, por lo que, si eso sucede, no se copia **xik** en **incxik** para que sea explorada la vecindad de esa solución, y se incrementan **contiter** y **contsegur** para dar paso a lo que ya se ha mencionado como el inicio de una nueva iteración. Si el problema es compatible, existen, como en la variante anterior, dos posibilidades: 1) que al evaluar la función objetivo para esa solución, el valor obtenido sea menor que el que se guarda en **fom**, y entonces la solución se copia a **resultado3**, y **xik** en **incxik** para explorar su vecindad, y 2) que el valor de la función objetivo resultante no sea menor, y entonces sólo se hace la copia en **incxik**. Para 1) se incrementa **contiter** y se vuelve a poner a cero **contsegur**, y para 2) ambos son incrementados.

Como se ha mencionado, si se ha realizado la copia de **xik** en **incxik**, la vecindad de la solución asociada a dicha matriz pasa a ser explorada, poniendo **contest**

previamente a cero. Los pasos dados por el programa en la exploración de las vecindades son prácticamente una repetición de los que se han detallado para las iteraciones, siendo en este caso **incxik** la matriz copiada en **xik** y modificada, en vez de su homóloga de **resultado3**. Por tanto, primero se realiza la copia citada, y después se llama a **cambicel3** para la modificación de **xik**. Para la iteración que nos sirvió de ejemplo, se obtuvieron estas nuevas **xik** y **tabumaq**:

0	1
1	0
1	0

2	3
4	2
2	0

Tras esta llamada, **fase33** llama a **gestsimp2** para que resuelva el problema de programación lineal ligado a **xik**. Si el problema resultante no es compatible, o si lo es pero el valor de la función objetivo asociado a la solución que se ha completado es menor que el que se encuentra en **fom**, lo único que efectúa el programa es un incremento de **contest**. Si el problema es compatible y el valor obtenido de la función objetivo evaluada para esa la solución es menor que el valor de **fom**, es como si comenzara una nueva iteración, por lo que se copia **xik** en **resultado2** y en **incxik**, y el valor mencionado de la función objetivo en **fom**, y se incrementa **contiter** y se ponen a cero **contest** y **contsegur**, iniciándose la exploración de la vecindad de la nueva solución asociada a **incxik**.

A continuación se explica cómo operan las funciones **cambiacel3** y **gestsimp3**.

cambiacel3 tiene como objetivo realizar un cambio en la asignación de una máquina a una célula, usando siempre la información de **xik**, en la que a la postre

efectuará una perturbación. Con ese objetivo, sortea primero la máquina a la que cambiará de célula. El sorteo de la máquina se repite, si es necesario, hasta que se encuentre una cuya célula en la que se encuentra no posea el número mínimo permitido de máquinas. Una vez obtenida la máquina, **cambiacel3** prosigue con el sorteo de la nueva célula de destino para la misma. Este sorteo es también repetido, si es preciso, hasta encontrar una célula que no esté al máximo de su capacidad. Una vez hallada, **cambiacel3** comprueba si el movimiento no está prohibido en la iteración en curso, y si no lo está, hace el cambio preciso en **xik** y prohíbe el movimiento para las **T** iteraciones sucesivas en **tabumaq**. Si el movimiento está prohibido, se vuelve a iniciar el proceso desde su comienzo.

Respecto a **gestsimp3**, la primera instrucción que realiza es llamar a **creacosts3**, cuya finalidad es análoga a la de **creacosts2** para la variante anterior: halla el coste asociado a **xik** de las variables del problema. El coste de las variables auxiliares y de las holguras es cero. El resultado es un vector, **costesfo**, que para la primera iteración que tomamos como ejemplo resultó ser:

15	96	90	100	42	90	73	95	17	95	45	0	0	0	0	0	0	0	0

Y para la solución vecina a la anterior:

11	96	90	102	42	90	73	98	17	98	45	0	0	0	0	0	0	0	0

Para estos costes, **fase33** llama a **arreglacosts**, con el objeto de que realice el pivoteo en los mismos para obtener los costes relativos asociados a la base de la matriz tecnológica **a**.

Se llama entonces a la función **simplex21**, que resuelve la Fase II del simplex, y después a **evalfo3**, que obtiene el valor de la función objetivo para los valores obtenidos en el problema resuelto por el método simplex. Si se trata de la primera solución que se obtiene (la que crea **inicial3**) o el valor obtenido es menor que **fom**, **gestsimp3** llama a **crearresul3** para que anote todos los elementos de la nueva solución en **resultado3**. Además, **crearresul3** traduce y copia los valores obtenidos por el simplex en la matriz **denmaqsm** de **resultado3**.

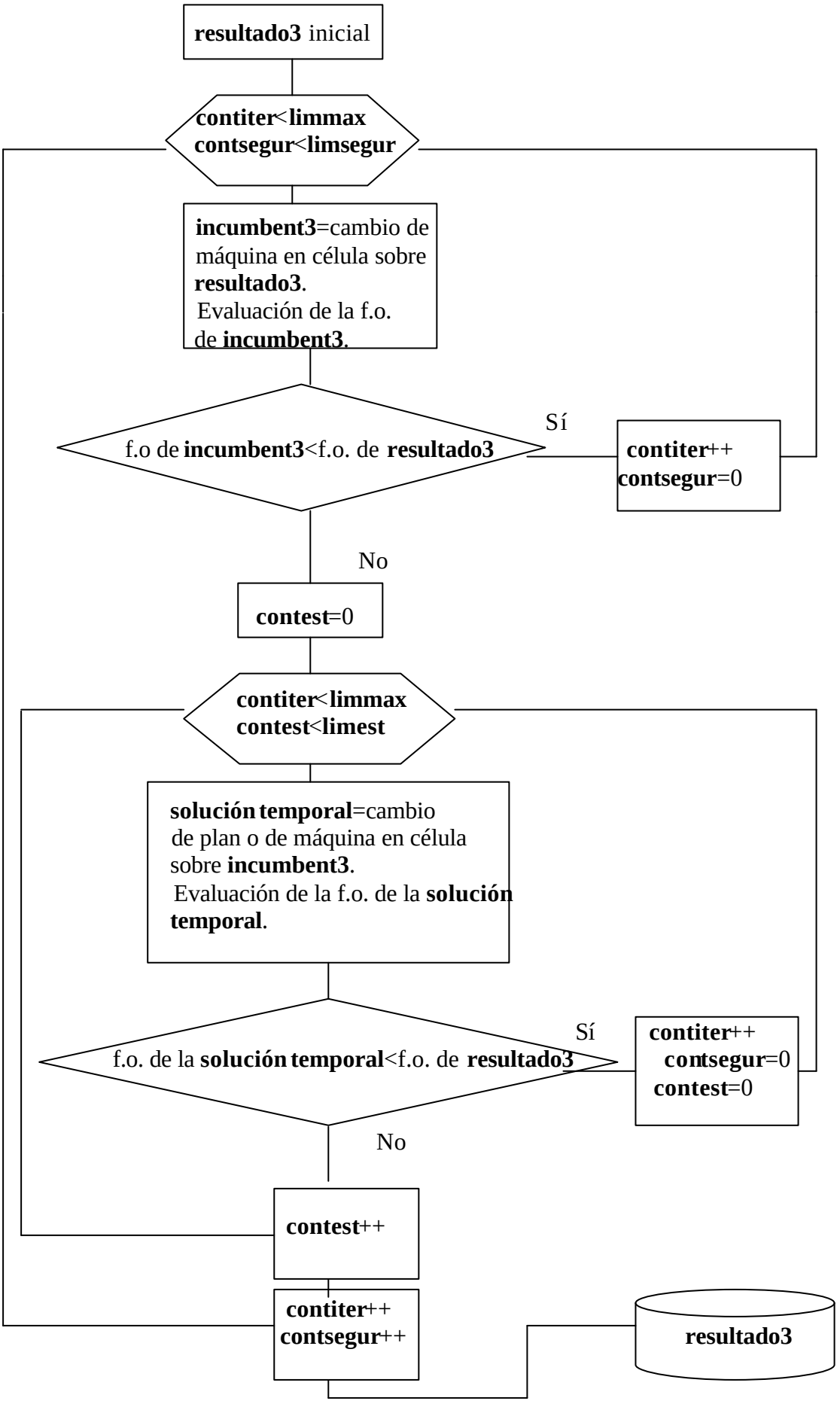
Para la solución que ilustró la iteración se obtuvieron esta **demmaqsm** y este valor de la función objetivo:

40	40	41	0	0	0	67
0	0	0	21	0	0	0
0	0	0	0	0	0	0

fo = 10151

Mientras que para la solución vecina a ésta, se obtuvo como **demmaqsm** una matriz idéntica a la anterior, pero con un valor menor de la función objetivo, debido a una reducción de los costes de transporte debida a su vez a la distinta distribución de la máquinas en las células:

fo = 9991


resultado3 inicial
contiter<**limmax**
contsegur<**limsegur**
incumbent3=cambio de máquina en célula sobre **resultado3**.
Evaluación de la f.o. de **incumbent3**.
f.o de **incumbent3**<f.o. de **resultado3**
Sí
contiter++
contsegur=0
No
contest=0
contiter<**limmax**
contest<**limest**
solución temporal=cambio de plan o de máquina en célula sobre **incumbent3**.
Evaluación de la f.o. de la **solución temporal**.
f.o. de la **solución temporal**<f.o. de **resultado3**
Sí
contiter++
contsegur=0
contest=0
No
contest++
contiter++
contsegur++
resultado3

10. ANÁLISIS DE RESULTADOS

Bajo este epígrafe se realiza un compendio de los resultados obtenidos en las pruebas efectuadas con el programa, así como un análisis de los mismos. Una de las pruebas efectuadas se emplea como ilustración.

10.1 Ejemplo ilustrativo

Se han resuelto cuatro tipos de problemas, identificados por la pareja de valores que se le han dado al nº máximo de planes (**pla**) junto con el nº máximo de máquinas que pueden realizar una operación (**maqs**). Tanto en lo que respecta a **plan** como a **maqs**, los valores que podían tomar eran 3 ó 5. Se han efectuado 5 pruebas para cada uno de los tipos de problemas. Se tomó un número de iteraciones adecuado para cada variante, de forma que la causa de mayor probabilidad de parada del algoritmo en cada una de ellas fuera exceder el número máximo de iteraciones sin mejora de la f.o. El número de iteraciones empleado en la primera variante para obtener mejoras similares en porcentaje a las de la variante segunda fue mucho más elevado que el usado en ésta, debido a una cardinalidad muy superior del espacio de soluciones considerado en la variante primera.

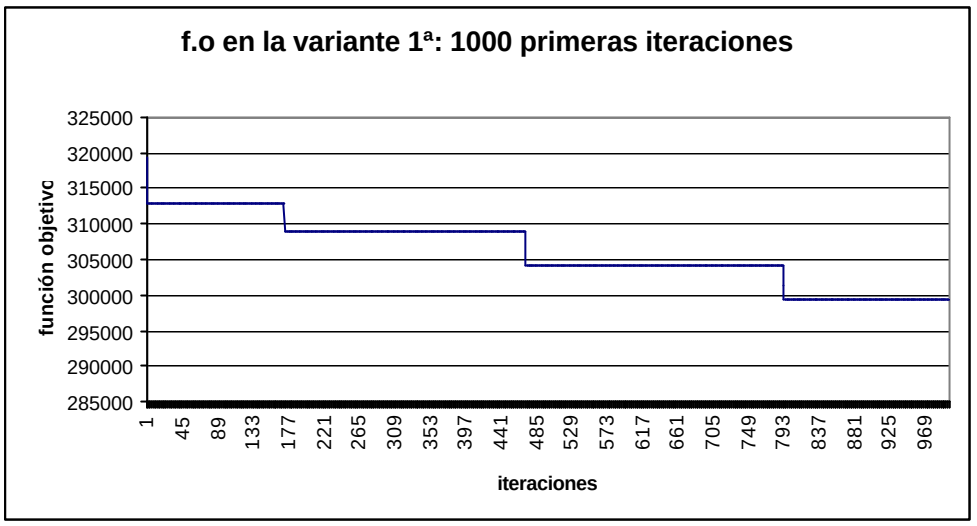
La salida del programa consiste en una serie de ficheros: los que contienen los datos del problema generado, los que guardan las tres soluciones, y otros de análisis de resultados. Seguidamente se muestra, como ejemplo de los ficheros de análisis que crea el programa para cualquier prueba, el correspondiente a una de las mismas. En concreto, se trata de una prueba con **pla** = 5 y **maqs** = 5.

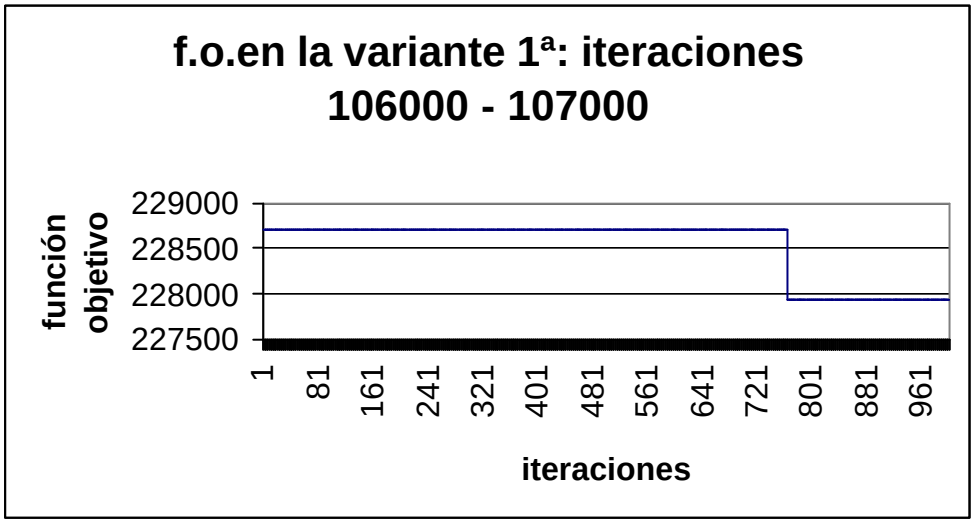
```
fo1:

227921.000

fo2:

109234.000

fo3:

101945.000

fo1/fo1ini:
```

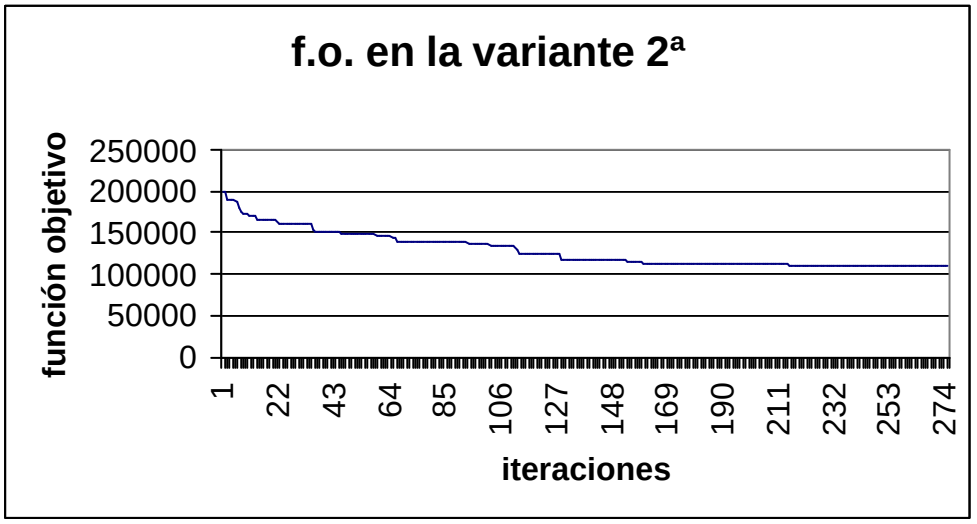
```
     0.714
tanto por uno de coincidencia de planes iniciales y finales en 1:
     0.720
fo2/fo2ini:
     0.549
tanto por uno de coincidencia de planes iniciales y finales en 2:
     0.500
fo2/fo1:
     0.479
tanto por uno de coincidencia de planes entre 1 y 2:
     0.680
fo3/fo3ini:
     0.950
tanto por uno de coincidencia de maqs iniciales y finales en 3:
     0.253
fo3/fo1:
     0.447
tanto por uno de coincidencia de maqs entre 2 y 3:
     0.822
tiempo de ejecución de la variante 2 partido por el de la variante 3:
     0.188
```

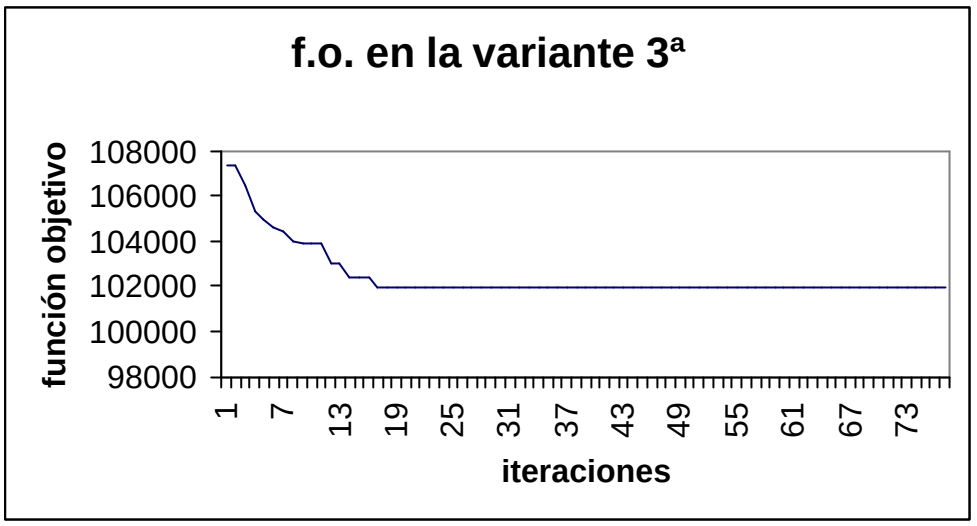
Otros ficheros creados por el programa almacenan el valor de la función objetivo para las sucesivas iteraciones. Para el ejemplo propuesto, siguen ahora las representaciones gráficas, para cada una de las variantes, de la evolución de la función objetivo con las iteraciones. Dado el elevado número de iteraciones necesario en la

primera variante, para ésta sólo se representan las 1000 primeras iteraciones, y las 1000 en las que se produce la estabilización de la función objetivo.









10.2 Resumen de los resultados

A continuación se presentan unas tablas con la media aritmética de los resultados que se han obtenido de todas las pruebas de los cuatro tipos de problemas.

Valores de la función objetivo correspondientes a la 1ª variante:

	pla = 5	**pla** = 3
maqs = 5	226196.8	235182.0
maqs = 3	245303.2	274033.4

Valores de la función objetivo correspondientes a la 2ª variante:

	pla = 5	**pla** = 3
maqs = 5	117567.8	138842.0
maqs = 3	146512.2	208044.8

Valores de la función objetivo correspondientes a la 3ª variante:

	pla = 5	**pla** = 3
maqs = 5	107243.4	123365.8
maqs = 3	139525.6	188546.0

Valores de la función objetivo de la última iteración partidos por los valores correspondientes a la primera iteración (1ª variante):

	pla = 5	**pla** = 3
maqs = 5	0.684	0.704
maqs = 3	0.729	0.797

Porcentajes de coincidencia de planes entre la primera y la última iteración (1ª variante):

	pla = 5	**pla** = 3
maqs = 5	70.8	79.2
maqs = 3	70.8	84.2

Valores de la función objetivo de la última iteración partidos por los valores correspondientes a la primera iteración (2ª variante):

	pla = 5	**pla** = 3
maqs = 5	0.626	0.712
maqs = 3	0.637	0.780

Porcentajes de coincidencia de planes entre la primera y la última iteración (2ª variante):

	pla = 5	**pla** = 3
maqs = 5	53.6	67.2
maqs = 3	56.8	74.2

Valores finales de la f.o. de la 2ª variante partidos por los valores finales de la f.o. de la 1ª variante:

	pla = 5	**pla** = 3
maqs = 5	0.526	0.591
maqs = 3	0.597	0.768

Porcentajes de coincidencia de planes entre los resultados de la 1ª y la 2ª variante:

	pla = 5	**pla** = 3
maqs = 5	70.4	68.4
maqs = 3	66.0	77.2

Valores de la función objetivo de la última iteración partidos por los correspondientes a la primera iteración (variante 3ª):

	pla = 5	**pla** = 3
maqs = 5	0.959	0.967
maqs = 3	0.972	0.971

Porcentajes de coincidencia de cargas de trabajo en las máquinas entre la primera y la última iteración de la variante 3ª:

	pla = 5	**pla** = 3
maqs = 5	24.8	36.6
maqs = 3	25.6	44.8

Valores finales de la f.o. de la 3ª variante partidos por los valores finales de la f.o. de la 1ª variante:

	pla = 5	**pla** = 3
maqs = 5	0.483	0.525
maqs = 3	0.568	0.695

Porcentajes de coincidencia de cargas de trabajo para las máquinas a las que se les ha asignado las mismas operaciones en las variantes 2ª y 3ª:

	pla = 5	**pla** = 3
maqs = 5	87.7	84.6
maqs = 3	89.2	83.6

10.3 Análisis

A tenor de los valores numéricos anteriores, se puede concluir que:

1) La libertad de conceder la posibilidad de que una operación pueda ser efectuada por varias máquinas, tiene más peso en la función objetivo que la exhaustividad en la exploración del espacio de soluciones. El espacio de soluciones de la 1ª variante es explorado en mayor medida que el de la variante segunda, y sin embargo sus mejores soluciones, para un tipo de problema, dan valores mayores de la función objetivo incluso que los de las primeras iteraciones de la segunda variante.

2) Cabe decir lo mismo respecto de la segunda y la tercera variante: la libertad de fabricar cada pieza mediante varios de los planes de proceso posibles, provoca valores mejores de la función objetivo, aunque con diferencias mucho menores que entre la primera y la segunda variante.

3) Podrían separarse las variantes en dos grupos, atendiendo a los valores obtenidos de la función objetivo. En el primer grupo estaría la primera variante, con resultados de la función objetivo sensiblemente mayores que el otro grupo, formado por las variantes segunda y tercera, con valores de dicha función más similares. Esto es así porque el segundo grupo aprovecha mejor la capacidad de las máquinas para reducir los costes de fabricación, cuyo peso sobre la f.o. es mucho mayor que el de los costes de transporte.

4) El espacio de soluciones resulta muy “plano” en la tercera variante. Ello es debido a que, para cada solución, cuenta con todas las máquinas para la fabricación de las piezas, optimizando el coste de fabricación en cada solución. No se garantiza el óptimo de los costes de transporte, pero éstos, como se acaba de decir, tienen poco peso en la f.o. con respecto a los de fabricación. Además, el coste de transporte tiene poca variabilidad (también comparándolo con el de fabricación) de una solución cualquiera a otra. En nuestros problemas, el coste de realizar una operación por una máquina puede variar entre 0 y 100, mientras que el coste de transportar una pieza de una célula a otra cualquiera puede hacerlo entre 0 y 10. La segunda variante también optimiza el coste de fabricación para cada solución, pero no cuenta

con todas las posibilidades de fabricación de las piezas, que están restringidas por los planes de proceso elegidos.

5) Atendiendo a los resultados de los ficheros de análisis que se encuentran en los Anexos, se observa que la variación de los valores numéricos de las distintas variables consideradas es muy pequeña, por lo que puede decirse que esos valores están asociados a las características de cada una de las variantes, y que los problemas generados aleatoriamente resultan muy similares entre sí.

6) Los resultados de la función objetivo empeoran al reducir el número de máquinas que pueden efectuar una operación y el número de planes por el que puede ser fabricada una pieza, debido a que el espacio de soluciones admisibles se reduce en todas las variantes.

11. ANEXOS

En esta sección se sitúan el código del programa y los ficheros de análisis de todas las 20 pruebas realizadas

11.1 Código y Ficheros

INGENIERÍA INDUSTRIAL, ESPECIALIDAD DE ORGANIZACIÓN INDUSTRIAL

Proyecto de Fin de Carrera:

FORMACIÓN DE CÉLULAS DE FABRICACIÓN CON MÚLTIPLES PLANES DE PROCESO PARA CADA PIEZA

Alumno: D. Rafael García Díaz Profesor: D. Sebastián Lozano Segura