

SECUENCIACIÓN DE TAREAS EN EL ÁMBITO DE LA PRODUCCIÓN: UNA APLICACIÓN DEL ALGORITMO DEL RECOCIDO SIMULADO

Zuleyka Díaz Martínez ^(a)

José Fernández Menéndez ^(b)

Paloma Martínez Almodóvar ^(b)

^(a) *Departamento de Economía Financiera y Contabilidad I.*

^(b) *Departamento de Organización de Empresas.*

Universidad Complutense de Madrid.

RESUMEN

Un problema esencial en la Dirección de Operaciones en entornos industriales y manufactureros es la determinación de la secuencia óptima en la que ejecutar los distintos lotes de productos de manera que se minimicen los tiempos de preparación de máquinas. Se trata de un problema de optimización combinatoria que obliga a utilizar técnicas heurísticas ante la imposibilidad práctica de llevar a cabo búsquedas exhaustivas.

Un algoritmo muy conocido por su sencillez es el de Kaufmann, que tiene el inconveniente de que proporciona un óptimo local que puede ser poco adecuado. En nuestro trabajo hemos elaborado un programa de ordenador en lenguaje C que implementa dicho algoritmo con algunas modificaciones y otro programa que implementa el algoritmo del Recocido Simulado, el cual suele proporcionar buenos resultados en problemas de optimización combinatoria al conseguir evitar óptimos locales. Para la implementación se ha utilizado una serie de subrutinas en C proporcionadas por la GNU Scientific Library. Se ha comparado el desempeño de ambos algoritmos y se han buscado los parámetros que permiten ajustar de forma adecuada el algoritmo del Recocido Simulado para su uso eficiente en estos problemas de minimización de los tiempos de preparación de máquinas.

Palabras clave: Dirección de Operaciones, Recocido Simulado, tiempos de preparación de máquinas.

1. INTRODUCCIÓN.

La secuenciación adecuada de pedidos constituye un importante problema que se plantea dentro de la Dirección de Operaciones a corto plazo. El orden en que los pedidos serán atendidos o procesados, o en general el orden en el que cualquier tipo de tareas serán realizadas, no resulta indiferente, sino que determinará algún parámetro de interés cuyos valores convendrá optimizar en la medida de lo posible. Así podrá verse afectado el coste total de ejecución de las tareas, el tiempo necesario para concluir las o el stock de productos en curso que será generado. Esto conduce de forma directa al problema de determinar cuál será el orden más adecuado para llevar a cabo las tareas con vistas a optimizar alguno de los anteriores parámetros u otros similares. Se trata de un problema de secuenciación o *scheduling* que se presenta de forma habitual en la programación de operaciones a corto plazo en entornos industriales o manufactureros (Heizer y Render, 2001) y que puede adoptar una enorme variedad de formulaciones.

La gran dificultad para resolver el problema determinando una secuencia óptima, o al menos admisible, junto con la importancia de conseguirlo, han hecho proliferar reglas, más o menos complejas, muchas de ellas heurísticas, y algunas incluso empíricas, que proporcionan soluciones rápidas y fáciles de calcular destinadas a su uso en situaciones de trabajo reales (Fernández Sánchez y Vázquez Ordás, 1994).

El desarrollo de los ordenadores y la aparición de nuevas técnicas de simulación y optimización heurística que aprovechan plenamente las disponibilidades de cálculo intensivo que aquéllos proporcionan -optimización Montecarlo, *soft computing*, metaheurísticas, etc, (Melián *et al.*, 2003) - han abierto una nueva vía para abordar los problemas de secuenciación, suministrando un creciente arsenal de métodos y algoritmos cuyo uso se extiende paulatinamente sustituyendo a las sencillas reglas heurísticas usadas tradicionalmente.

El problema concreto al que aquí se prestará atención es el de determinar la secuencia óptima según la cual deben ser elaborados una serie de lotes distintos para minimizar los tiempos de preparación de máquinas.

En un sistema de producción intermitente o por lotes (también denominado con frecuencia *job-shop*) cada vez que finaliza la elaboración de uno de dichos lotes, para poder comenzar con la del siguiente se debe proceder a un reajuste o “preparación” de

las máquinas, lo que supondrá una paralización momentánea de la actividad de las máquinas reajustadas. Los tiempos de preparación de dichas máquinas dependerán de los distintos pares de lotes entre los que se produzca la preparación, es decir, del orden en que los lotes sean elaborados. Se plantea entonces el problema de determinar cuál será el orden más adecuado para minimizar el tiempo de preparación total (Domínguez Machuca *et al.*, 2001).

Aunque una búsqueda exhaustiva puede ser adecuada cuando el número n de lotes a considerar es muy reducido, el rápido crecimiento del número de operaciones de búsqueda y comparación a medida que crece el número de lotes la convierte rápidamente en inviable. Se trata el de la secuenciación de un problema muy similar al conocido como el problema del viajante - *Travelling Salesman Problem* o TSP-, es decir, determinar la secuencia en que deben ser recorridas una serie de ciudades para minimizar el recorrido total. Es éste un problema de complejidad exponencialmente creciente con el número de ciudades y que cae dentro de la categoría de los conocidos como problemas NP-completos (Fernández y Sáez Vacas, 1987). Para ellos no se conocen algoritmos que proporcionen soluciones óptimas en tiempos ni remotamente razonables, lo que obliga a recurrir a aproximaciones y métodos heurísticos que con cargas computacionales asumibles proporcionen soluciones ya que no óptimas, al menos sí aceptables.

Uno de tales algoritmos, de gran sencillez y ampliamente conocido es el de Kaufmann (1964), que consiste en comenzar por un lote cualquiera y elegir como siguiente aquél para el cual sea menor el tiempo de preparación y así sucesivamente. El inconveniente de este método, y de otros similares, es el de que proporciona un óptimo local que puede estar muy alejado del óptimo global, sin que haya además ninguna manera de estimar la distancia entre ambos.

Los modernos ordenadores ponen a nuestro alcance otros métodos, más sofisticados, que permiten ir sorteando, al menos en alguna medida, los óptimos locales y aproximarnos cada vez más al óptimo global (pero sin alcanzarlo salvo por puro azar). Entre estos métodos destaca por su amplia utilización y los buenos resultados que generalmente consigue el conocido como “Recocido Simulado”.

2. EL RECOCIDO SIMULADO.

Una técnica heurística ampliamente utilizada en el tratamiento de problemas de optimización combinatoria es la del Recocido Simulado (Díaz, 1996, Downsland y Díaz, 2003 y Ríos Insua *et al.*, 1997). El método consiste en ir recorriendo de forma aleatoria el espacio de configuraciones para seleccionar aquel punto de dicho espacio en el que alcance su valor óptimo una determinada función de coste (o de *energía*), que asumirá el papel de función objetivo que se pretende minimizar. Sin embargo la esencia del Recocido Simulado, y lo que lo distingue de una mera búsqueda aleatoria, es la regla que utiliza para ir saltando de una configuración a otra. Este paso de un punto a otro del espacio de configuraciones se hará siempre, es decir, con probabilidad 1, cuando el nuevo punto suponga un menor valor de la función a optimizar. Cuando el nuevo punto suponga un coste, o una energía, mayor que el del inicial, la transición aún podrá llevarse a cabo, pero sólo con una probabilidad $p < 1$ que vendrá dada de acuerdo con la conocida distribución de Boltzmann de la Mecánica Estadística Clásica.

Es este último aspecto, la probabilidad no nula de saltar hacia puntos de coste o energía mayores – salto sólo momentáneo, por supuesto, concebido como etapa intermedia necesaria para llegar a puntos de energía menor – lo que hace que el método evite quedar atrapado en el entorno de algún óptimo local.

La denominación del algoritmo, Recocido Simulado, o *Simulated Annealing* en inglés, proviene de la estrecha analogía, analogía que es la que le ha dado origen, que guarda con el proceso del Recocido tal y como se usa en metalurgia. Éste consiste en que un metal fundido se va enfriando lentamente de manera que sus moléculas van adoptando poco a poco una configuración de mínima energía. Cuando comienza el proceso, a alta temperatura, las moléculas vibran y se desplazan caóticamente adoptando todo tipo de configuraciones en la estructura del metal de la que forman parte. A medida que la temperatura disminuye se va ralentizando el movimiento de las moléculas y estas, de acuerdo con la Termodinámica, tienden a adoptar paulatinamente las configuraciones de menor energía, siendo ésta nula en el cero absoluto. Durante sus vibraciones en la red metálica las moléculas podrán saltar de una configuración a otra con una probabilidad que será directamente proporcional a la temperatura e

inversamente proporcional a la diferencia de energías entre las configuraciones inicial y final y que vendrá dada por la distribución de Boltzmann:

$$p = e^{-\left(\frac{E_F - E_I}{kT}\right)}$$

El número k será la conocida constante de Boltzmann. En los problemas de optimización que nos interesan carece de significado y se le da convencionalmente el valor 1.

El Recocido Simulado intenta realizar numéricamente un proceso análogo al del recocido metalúrgico. El espacio de configuraciones no vendrá ya dado por las posiciones de las moléculas, sino por los valores de una variable de interés, que en nuestro caso será la secuencia de lotes que deseamos procesar, mientras que el papel de la energía lo asumirá la función que intentamos minimizar, coste o tiempo de realización de tareas, por ejemplo. Mayores detalles pueden encontrarse en Robert y Casella (1999) o en Díaz (1996), donde puede verse también una buena revisión de las aplicaciones del Recocido a problemas prácticos del tipo más variado.

Este algoritmo no debe ser visto como una técnica completamente estandarizada, lista para ser aplicada de forma directa e invariable a no importa qué problema, sino que, debido al gran número de parámetros configurables que incluye, deberá ser adecuadamente implementada en cada situación concreta en la que vaya a ser utilizado. Será por tanto necesario precisar aspectos como los siguientes:

- La secuencia de enfriamiento. Serán las temperaturas inicial y final (la primera lógicamente más elevada que la segunda) y las sucesivas temperaturas recorridas para llegar de una a otra. Una temperatura inicial elevada facilita la exploración del espacio de configuraciones pues hace que casi todos los saltos de una configuración a otra estén permitidos. Idealmente la temperatura final tras el proceso de enfriamiento parece que debería ser cero, sin embargo no será necesario llegar a ese extremo, pues cuando la temperatura se haga lo suficientemente baja la probabilidad de salto a una configuración peor será virtualmente nula, con lo que el proceso quedará atrapado en el mejor óptimo local conseguido hasta el momento y ya no se producirá apenas ninguna mejora en el resultado alcanzado. Por otra parte, una diferencia muy acusada entre temperatura inicial y final hará que la ejecución del algoritmo sea demasiado lenta. Para lograr un equilibrio adecuado entre todos estos aspectos no cabe, en general, otra solución que

recurrir al ensayo y error. También es necesario determinar cuál es la secuencia de temperaturas recorridas. Se han propuesto muchos esquemas, siendo uno de los más aceptados el de mantener cada temperatura constante durante L iteraciones (con L parámetro a establecer en cada caso que se suele denominar *longitud de la temperatura*) y luego disminuirla dividiéndola por un factor constante μ_T (el *factor de enfriamiento*) ligeramente mayor que la unidad.

- La topología del espacio de configuraciones. En cada problema concreto habrá que determinar cuáles son las variables que describen adecuadamente el espacio de soluciones, lo cual puede no ser evidente *a priori*, y establecer la manera de ir saltando aleatoriamente de una configuración a otra de su entorno (lo que exige precisar la noción de entorno de una configuración). Puede ser conveniente trabajar tanto con entornos amplios como reducidos; ello dependerá de cada problema concreto, no siendo posible dar indicaciones generales al respecto. Más sencillo resultará establecer la función que hace el papel de la energía, pues se tratará en general del coste o duración que debe ser minimizado.
- La configuración de partida. Ésta puede ser una arbitraria u obtenida como resultado de un algoritmo previo que proporcione de forma sencilla alguna solución de bajo coste o energía.

En conjunto, la experiencia acumulada demuestra que aunque es relativamente sencillo implementar el Recocido Simulado, no lo es tanto hacerlo de forma adecuada, y en problemas reales siempre es necesaria una etapa de prueba y error para ir ajustando el algoritmo a la naturaleza de cada problema concreto.

3. IMPLEMENTACIÓN DEL ALGORITMO.

Para intentar resolver el problema de determinar la secuencia en la que deben ser ejecutados una serie de lotes o pedidos de forma que se minimicen los tiempos de preparación de máquinas se ha escrito un programa de ordenador en lenguaje C.

El programa ha sido compilado utilizando el compilador gcc 3.2, compilador que tiene un carácter estándar en el ámbito del sistema operativo Linux, pero para el que existen versiones (gratuitas) para Windows. Para la implementación se ha utilizado el conjunto de subrutinas para cálculo numérico escritas también en C, que integran la

librería de funciones GNU Scientific Library (GSL) y que son suministradas bajo licencia GPL por la Free Software Foundation.

La GSL incluye subrutinas que proporcionan un marco general para la implementación del recocido simulado, permitiendo la automatización de las operaciones más repetitivas, pero dejando en manos del programador la codificación de todos los aspectos específicos de cada problema. Esto incluirá no sólo el establecimiento de parámetros como temperatura inicial y final, longitud de temperatura y factor de enfriamiento, sino también, y muy especialmente, la codificación de la función energía, que para nuestro problema será el tiempo total de preparación de máquinas para cada secuencia concreta de procesamiento de los lotes; la definición del espacio de configuraciones, que en nuestro caso será el conjunto de todas las permutaciones de los lotes que deben ser secuenciados; y el establecimiento del mecanismo concreto de generación de saltos aleatorios de una configuración (permutación) a otra. Esto último obliga a determinar de alguna manera la distancia entre dos permutaciones distintas y con ello la estructura del sistema de entornos dentro de los cuales se producen los saltos aleatorios entre permutaciones.

En este tipo de problemas resulta habitual (Díaz, 1996) el salto de una permutación a otra que se obtiene intercambiando dos y sólo dos elementos de la permutación inicial. Sin embargo, aquí se implementará un esquema más general que permite una mayor variedad de saltos.

Para ello, y atendiendo a la naturaleza de nuestro problema, parece oportuno medir la distancia entre dos secuencias distintas de lotes por el número de *transiciones*, o pasos de un lote a otro, diferentes entre las dos secuencias. De este modo si tenemos las permutaciones L1, L2, L3, L4 y L1, L2, L4, L3, su distancia será 2, pues tienen una transición igual, de L1 a L2, y dos diferentes, de L2 a L3 y de L3 a L4 en la primera y de L2 a L4 y de L4 a L3 en la segunda.

Para medir de una forma cómoda esta distancia se procederá de la siguiente manera: se sabe que cualquier permutación σ de N elementos corresponde a una biyección del conjunto $\{1, 2, \dots, N\}$ en sí mismo que puede ser representada por una matriz P de $N \times N$ cuyos elementos p_{ij} valdrán 1 si $\sigma(i) = j$ y 0 en caso contrario. Una matriz de permutación tendrá un único 1 en cada fila y cada columna. Se puede generar a partir de ella otra matriz, a la que podemos denominar *de transiciones*, T , que tendrá un valor 1 en su elemento t_{ij} cuando la permutación suponga una transición del elemento (en nuestro caso del lote) i al j y 0 cuando dicha transición no esté presente. Así por

ejemplo, si tenemos la permutación L2, L1, L3 (representada por la biyección $\sigma(1)=2$, $\sigma(2)=1$, $\sigma(3)=3$) sus matrices P y T serán:

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

pues en la permutación L2, L1, L3 se producen las transiciones de los lotes L2 a L1 y de L1 a L3, con lo que tendrán valor 1 los elementos t_{21} y t_{13} en T.

La matriz T puede “ampliarse” dando el valor 1 a su elemento t_{32} , lo que la convertiría en una matriz de transiciones “ampliada”:

$$T^* = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

que como vemos es una matriz de permutaciones normal por tener un 1 y sólo uno en cada fila y cada columna. Para pasar de T a T* será necesario igualar a 1 el elemento correspondiente a la transición del último al primer elemento de la permutación (transición “virtual” entonces); en nuestro ejemplo, al ser dicha permutación L2, L1, L3 se tratará de la transición de L3 a L2 y por lo tanto $t^*_{23} = 1$.

A partir de las matrices de transiciones, T_1 y T_2 , de dos permutaciones distintas se puede calcular su distancia de forma sencilla multiplicando ambas matrices elemento a elemento y sumando los elementos de la matriz resultante, lo que dará el número de transiciones comunes y a partir de ahí el de las diferentes. Todas estas operaciones se ven facilitadas por las subrutinas que ofrece la librería GSL.

Para saltar de una permutación P_1 a otra P_2 situada a una distancia menor o igual que d resulta apropiado partir de la matriz T^*_1 de vínculos ampliada correspondiente a P_1 y en ella seleccionar $n-(d+1)$ transiciones, es decir, $n-(d+1)$ unos de la matriz (que no incluirán el correspondiente a la transición virtual) que permanecerán fijos mientras los demás se cambian aleatoriamente para generar una nueva matriz T^*_2 correspondiente a la permutación P_2 (serán necesarias algunas precauciones para conseguir que T^*_2 corresponda a una matriz de transiciones ampliada válida). Operando así se conseguirá una nueva permutación P_2 a una distancia menor o igual que d de P_1 .

Para implementar este proceso puede ser conveniente utilizar alguna manera sencilla de pasar de la matriz P a la T^* . Se puede comprobar con facilidad que $T^* = P' \cdot R^*_I \cdot P$, siendo P' la transpuesta de P y R^*_I la matriz de transiciones ampliada correspondiente a la permutación identidad, $\sigma(i)=j$. Por ejemplo, si $n = 4$ resultará ser:

$$R^*_I = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Aunque esta manera de calcular T^* puede ser sencilla de codificar, si los tamaños de las matrices involucradas son grandes se generará código poco eficiente, especialmente en nuestro caso debido a que la librería *GSL* utilizada no incorpora facilidades para el tratamiento de matrices dispersas (aquéllas cuyos elementos son mayoritariamente ceros), que acelerarían notablemente las operaciones matriciales.

No obstante, sí se dispone de un gran número de subrutinas para la manipulación de *arrays* y matrices, generación de números aleatorios, de permutaciones, ordenación, muestreo, etc, que han sido ampliamente utilizadas en la escritura del código del programa facilitando su implementación.

4. RESULTADOS.

Para comprobar la eficacia del algoritmo implementado se han generado una serie de tablas o matrices de números aleatorios que representan tiempos de preparación de máquinas. En concreto, se han utilizado matrices de 10×10 y 20×20 correspondientes respectivamente al caso de 10 y 20 lotes a secuenciar.

A las matrices obtenidas se les ha aplicado el Recocido Simulado y, como elemento de comparación, el algoritmo de Kaufmann, pero éste ligeramente mejorado, ya que en lugar de comenzar por un lote elegido aleatoriamente, como en su formulación original, se ha ensayado comenzando con cada uno de los lotes existentes, seleccionando el mejor resultado obtenido.

En conjunto se han alcanzado las siguientes conclusiones:

- Los mejores resultados parecen obtenerse con una temperatura inicial en torno a 20 y final en torno a $2 \cdot 10^{-3}$, aunque bastante antes de alcanzar esta temperatura ya se produce una estabilización de los resultados.

- Como distancia de salto la más adecuada parece ser una ligeramente por encima de la mínima posible (que es 2, por lo tanto conviene tomar 3 ó 4).

- El Recocido Simulado mejora virtualmente siempre los resultados del algoritmo de Kaufmann, siendo esta mejora del orden del 20% o superior. Como ejemplo veamos los resultados obtenidos con 5 matrices de prueba distintas de 10×10 :

<u>Kaufmann</u>	<u>Recocido Simulado</u>
86,57	84,33
190,60	120,17
105,22	91,94
128,00	88,99
129,76	108,52

que suponen una mejora promedio del 21%.

- Como conclusión podemos afirmar que el Recocido Simulado constituye una alternativa excelente para resolver problemas de secuenciación de pedidos, ya que permite obtener unos resultados muy satisfactorios con una implementación relativamente sencilla.

5. REFERENCIAS BIBLIOGRÁFICAS.

- DÍAZ, A.(coord.) (1996). *Optimización Heurística y Redes Neuronales*. Ed. Paraninfo. Madrid.

- DOMÍNGUEZ MACHUCA, J.A., GARCÍA GONZÁLEZ, S., DOMÍNGUEZ MACHUCA, M.A., RUÍZ JIMÉNEZ A. y ÁLVAREZ GIL, M.J. (2001). *Dirección de Operaciones. Aspectos tácticos y operativos en la producción y los servicios*. McGraw-Hill. Madrid.

- DOWNSLAND, K.A. y DÍAZ, A. (2003). "Heuristic design and fundamentals of the Simulated Annealing". *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, nº 19, pp. 93-102.

- FERNÁNDEZ, G. y SÁEZ VACAS, F. (1987). *Fundamentos de Informática*. Alianza Editorial. Madrid.
- FERNÁNDEZ SÁNCHEZ, E. y VÁZQUEZ ORDÁS, C. (1994). *Dirección de la Producción. II Métodos Operativos*. Civitas. Madrid.
- GOUGH, B. (ed.) (2003). *GNU Scientific Library. Reference Manual. Ed. 1.3*. Disponible en <http://www.network-theory.co.uk/gsl/manual/>.
- HEIZER, J. y RENDER, B. (2001). *Dirección de la Producción. Decisiones Tácticas*. Prentice-Hall. Pearson Education. Madrid.
- KAUFMANN, A. (1964). *Méthodes et Modèles de la Recherche Opérationnelle. Tome II*. Dunod. París.
- KERNIGHAN, B.W. y RITCHIE, D.M. (1991). *El Lenguaje de Programación C*. Prentice-Hall Hispanoamericana. México.
- MELIÁN, B., MORENO PÉREZ, J.A. y MORENO VEGA, J.M. (2003). "Metaheuristics: A global view". *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, nº 19, pp. 7-28.
- ROBERT, C.P. y CASELLA, G. (1999). *Monte Carlo Statistical Methods*. Springer-Verlag. New York.
- RÍOS INSUA, D., RÍOS INSUA, S., MARTÍN, J. (1997). *Simulación. Métodos y Aplicaciones*. Ra-Ma Editorial. Madrid.