Resolución del *PRODUCT RATE VARIATION PROBLEM* (PRVP) de grandes dimensiones como un problema de afectación

Natalia Moreno Palli, Albert Corominas Subias
IOC / DOE / ETSEIB,UPC; Av.Diagonal 647, 08028 Barcelona, moreno@ioc.upc.es
IOC / DOE / ETSEIB,UPC; Av.Diagonal 647, 08028 Barcelona, corominas@oe.upc.es

RESUMEN

El PRVP es un problema que se presenta en las líneas de producción mixtas. Una de las formas de resolverlo consiste en reducirlo al problema de afectación y aplicar entonces algoritmos específicos para este último. Este trabajo presenta un estudio de los algoritmos y una experiencia computacional.

1. Introducción

El PRVP (*Production Rate Variation Problem*) se presenta en la determinación de secuencias regulares en contexto JIT y su objetivo es minimizar la variación de las tasas de producción a lo largo del tiempo para los diversos productos implicados.

El PRVP es un caso particular del problema de Monden [1] que consiste en encontrar secuencias con el consumo de componentes lo más regular posible. El propio Monden [1] presenta un procedimiento heurístico para resolución de dicho problema.

En 1989 Miltenburg [2] introduce el PRVP y lo formula como un problema de programación entera no lineal con el objetivo de minimizar la desviación total de las tasas de producción en una línea de producción mixta. Para ello, Miltenburg presenta un algoritmo exacto con requerimientos de tiempo elevados, por lo que propone también dos algoritmos heurísticos.

Para la resolución exacta se pueden aplicar algoritmos basados en la programación dinámica. El propuesto por Miltenburg, Steiner y Yeomans [3] sólo permite resolver prácticamente ejemplares de pequeñas dimensiones, puesto que los requerimientos de memoria y de tiempo crecen rápidamente a medida que aumenta el número total de unidades a secuenciar y, más aún, el número de tipos de productos. Bautista, Companys y Corominas [4] presentan un algoritmo de programación dinámica acotada (BDP) en el que la utilización de cotas y su comparación con el valor de una solución obtenida con un procedimiento heurístico hace posible la resolución en tiempos breves de ejemplares de mayores dimensiones.

En 1993 Kubiak [5] introduce la denominación PRVP. Kubiak y Sethi [6,7] demuestran que el PRVP puede reducirse al problema de afectación cuando las funciones que componen la función objetivo son no negativas y convexas. Bautista, Companys y Corominas [8] establecen un procedimiento general de cálculo para la matriz del problema de afectación, tanto para el caso *minisum* (minimización de una suma de funciones de discrepancia) como para el *minimax* (minimización del valor máximo de la función o funciones de discrepancia). No obstante, sólo muy recientemente [9] se ha publicado una limitada experiencia computacional de resolución del PRV como un problema de afectación.

En este trabajo se trata el caso *minisum* del PRVP como un problema de afectación (AP), con una matriz de costes dotada de propiedades específicas que, mediante las oportunas

adaptaciones, permiten obtener una mayor eficiencia de los algoritmos. En la Sección 2 se expone el modelo matemático del problema; la Sección 3 se refiere a los algoritmos de resolución del problema de afectación (AP); en la Sección 4 se describe el PRVP como un caso particular de AP; en la Sección 5 se presenta la experiencia computacional y la Sección 6 incluye las conclusiones e indicaciones sobre las líneas de trabajo futuras.

2. El modelo matemático

Se considera un producto del que existen V variantes o tipos, que se producen en una línea en la que los tiempos de cambio de un tipo a otro son despreciables y en la que cada unidad, independientemente de cuál sea la variante a que corresponda, requiere el mismo tiempo (tiempo de ciclo, que puede adoptarse como unidad de tiempo sin pérdida de generalidad). Se trata de secuenciar un total de U unidades, de las que u_i son de tipo i ($i=1,...,V$) de modo que las tasas de fabricación de los distintos tipos de producto se mantengan tan constantes como sea posible a lo largo del tiempo.

Las tasas ideales, r_i, son:

$$r_i = u_i / U \quad , i = 1 \ldots V \qquad (1)$$

Si se denomina x_{ih} al número de unidades de tipo i producidas tras h ciclos de fabricación, el problema se puede formalizar como sigue:

$$min\, z_S = \sum_{h=1}^{U} \sum_{i=1}^{V} f_i(x_{ih}, h) \qquad (2)$$

En [8] se demuestra que este problema es equivalente a un problema de afectación en el que los elementos de la matriz, $\varphi_{ik}(t)$, se pueden calcular mediante la expresión:

$$j_{ik}(t) = \sum_{h=t}^{U} (f_i(k,h) - f_i(k-1,h)) \qquad (3)$$

3. Algoritmos de resolución del problema de afectación (AP)

Los algoritmos para la resolución de AP pueden ser agrupados en tres clases: primales, primal-duales y duales.

Dell'Amico y Toth [10] presentan un excelente resumen sobre el estado del arte del problema de afectación, en el que se describen las técnicas de resolución y algunos códigos disponibles. De la experiencia computacional incluida en dicho trabajo los autores concluyen que los códigos más eficientes corresponden a algoritmos primal-duales y duales. De hecho, en dicha experiencia no se utilizan códigos basados en algoritmos primales; los autores indican que no existen códigos disponibles eficientes, aunque la complejidad de algunos algoritmos primales es comparable a la de los primal-duales. Otros autores, como se expone en [11], coinciden en la apreciación de que los algoritmos primales son menos eficientes que los duales y los primal-duales.

Amatller [12] incluye también una síntesis sobre algoritmos y códigos para el AP e incorpora a la experiencia computacional un código basado en un algoritmo primal, el de Barr [13],

considerado como uno de los más eficientes entre los de su clase. La conclusión es que dicho código requiere tiempos de cálculo mayores que los correspondientes a algoritmos de tipo dual y primal-dual.

4. El PRVP como un caso particular del AP

Como se ha indicado anteriormente, el PRVP, considerado como un problema de afectación presenta propiedades específicas:

i) Es fácil obtener una solución de buena calidad mediante un procedimiento heurístico.

ii) Si consideramos una matriz de afectación en que las filas corresponden a unidades y las columnas a posiciones, en cada fila tenemos una (o, en caso de empate, dos) posición ideal y los valores de los elementos de la matriz son crecientes a derecha y a izquierda de dichas posiciones ideales.

De todo ello se desprende que:

i) A priori se puede pensar que los algoritmos primales pueden ser más eficientes que para los ejemplares del AP sin propiedades específicas, dada la disponibilidad de una solución inicial de buena calidad.

ii) En las soluciones óptimas probablemente intervienen únicamente elementos de la matriz situados en posiciones próximas a las ideales, por lo que parece razonable utilizar un algoritmo que sólo utilice los elementos que podemos llamar más prometedores, con una comprobación posterior del carácter óptimo de la solución obtenida (si no lo es, se incorporan nuevos elementos y se reitera).

En relación con esta última consideración, en Volgenant [14] se propone una mejora de uno de los algoritmos más eficientes para matrices completas (Jonker y Volgenant [11]). Dicha mejora consiste en resolver el AP con un subconjunto de los elementos de la matriz (por ejemplo, los *p* mejores de cada fila); de este modo se trabaja con una matriz poco densa, a la que se aplica la adaptación del algoritmo LAPJV para matrices poco densas (LAPJVsp [11]) ; se comprueba si la solución obtenida es óptima para el problema original y si no lo es se incorpora un nuevo elemento a la matriz y se reitera. Este enfoque permite resolver el AP en menos tiempo y con menores requerimientos de memoria. En el caso del PRVP los elementos a tener en cuenta se pueden elegir con mayor fundamento, la matriz poco densa es más fácil de almacenar y, finalmente, la comprobación de la optimalidad se puede simplificar.

Estas son las consideraciones que han orientado la experiencia computacional que se describe en la sección siguiente.

5. Experiencia computacional

Los cálculos se han realizado en una SUN 450 Ultra SPARC2 con 4 procesadores a 250 Mhz y 512 Mb de RAM.

Las funciones que se han utilizado en la experiencia computacional son:

$$f_i(x_{ih},h) = |x_{ih} - r_i h| \qquad (4)$$
$$f_i(x_{ih,}h) = (x_{ih} - r_i h)^2$$

Se han efectuado las pruebas con diversos ejemplares del PRVP, que se han generado fijando el número total de unidades y el número de tipos y obteniendo el número de unidades de cada tipo aletoriamente.

Unos primeros resultados pusieron de manifiesto que entre los algoritmos disponibles (BARR —primal—, NAUC —dual— y APC y LAPJV —primal-duales—) aplicados al PRVP el más rápido resultaba ser el algoritmo LAPJV, presentado en [11], seguido muy de cerca por el APC. La Figura 1 presenta la comparación de los tiempos medios de CPU (media de 5 ejemplares para cada dimensión —número de unidades—).

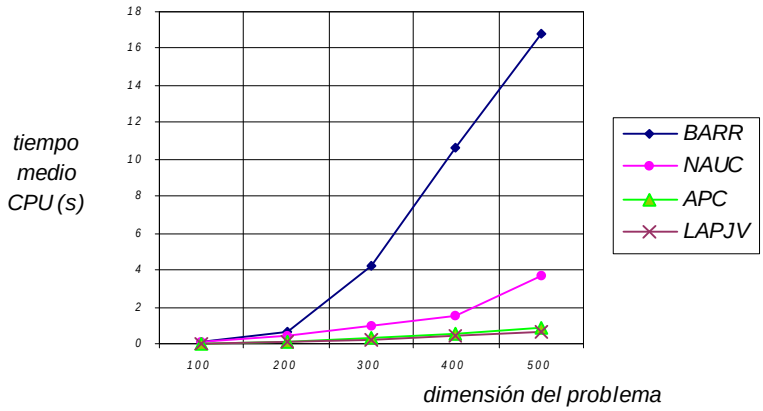

Figura 1 : Comparación de los tiempos para 20 tipos del producto

Se abandonaron los algoritmos menos eficientes y se profundizó el estudio de los más rápidos (APC y LAPJV).

Para matrices con elementos enteros se ha llegado a resolver ejemplares con *U=10000*, lo que no ha sido posible al utilizar variables de doble precisión, a causa de los mayores requerimientos de memoria (en este caso se ha llegado a *U=8500*). En los ejemplares de 10000 unidades, con variables enteras, el tiempo medio de CPU ha sido de 1042.34 segundos. Los cálculos en doble precisión requieren más tiempo (por ejemplo, con LAPJV y *U=5000*: 146.70 *s* con matriz entera y 201.69 *s* con matriz en doble precisión).

Para la aplicación del algoritmo modificado presentado en [14] se resuelve primero el problema haciendo intervenir sólo los elementos incluidos en la que hemos denominado *franja cuota*, (es decir, aquellos elementos que corresponden a una desviación de la producción real, en relación con la ideal, menor que la unidad); para la resolución se utiliza el algoritmo LAPJVsp (algoritmo LAPJV adaptado para matrices poco densas que se describe en [11]), en el que se ha adaptado al PRVP la rutina para la comprobación de la optimalidad. Para las funciones simétricas con que se ha realizado la experiencia computacional, se ha obtenido siempre la solución óptima en la primera iteración, es decir, en ningún caso han intervenido en la solución óptima elementos no pertenecientes a la franja cuota. El hecho de trabajar con un número reducido de elementos y de que con ellos baste habitualmente para

encontrar una solución óptima dan lugar muchas veces a una reducción apreciable de los tiempos de cálculo, tal como se refleja, a título de ejemplo, en la Tabla 1.

	Matriz poco densa (LAPJVsp)	Matriz completa (LAPJV)
Número de elementos	54946	5000 x 5000
Tiempo CPU (s)	0.38	24.9

Tabla 1: Comparación de LAPJVsp y LAPJV para 10 tipos del producto

Para los mejores algoritmos (LAPJV y LAPJVsp) se ha estudiado la influencia en el tiempo de cálculo del número de tipos de productos, del número total de unidades y de la desviación tipo, σ, de las u_i .

Se ha comparado (Fig.2) el comportamiento de ambos algoritmos para el mismo conjunto de ejemplares (número total de unidades igual a 500 en todos los casos y cinco ejemplares para cada uno de los valores utilizados del número de tipos de producto). El aumento del número de tipos influye especialmente en los tiempos de cálculo del algoritmo LAPJVsp, ya que el número de elementos de la matriz "poco densa" tiende a aproximarse al de la matriz completa cuando hay muchos tipos de producto con un número pequeño de unidades. En ciertos ejemplares (v.g.: 225 tipos de los cuales 223 tienen una unidad y los otros dos, 77 y 200 unidades) los tiempos de cálculo del algoritmo LAPJV son excepcionalmente elevados.

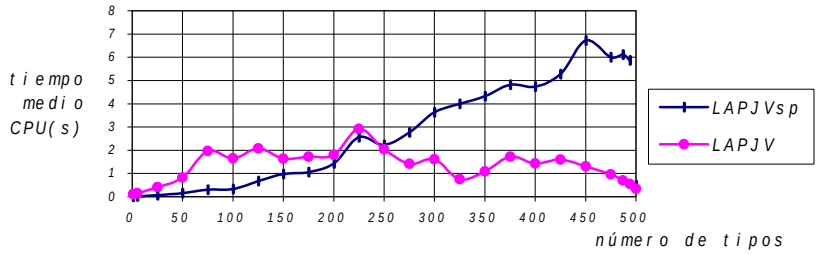

Figura 2 : Comparación de los tiempos de LAPJVsp y LAPJV para diferentes números del tipo del producto

Se ha estudiado el comportamiento del LAPJVsp, con 10 tipos de producto, para diferentes valores del número total de unidades (5 ejemplares para los valores menores o iguales a 6500 y 2 para los demás). Por supuesto, los tiempos (Fig. 3 y Fig. 4) aumentan a medida que crece el número total de unidades, con un cambio de tendencia a partir, aproximadamente, de U = 5000, ya que los ejemplares grandes requieren la utilización de memoria externa.

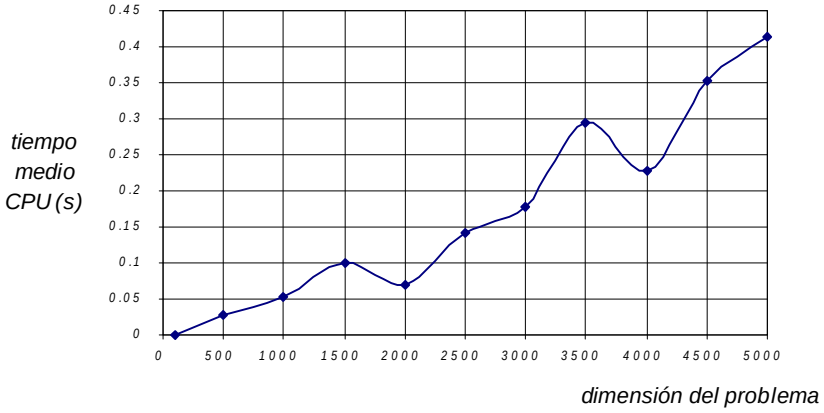

Figura 3: Comparación de los tiempos medios para LAPJVsp

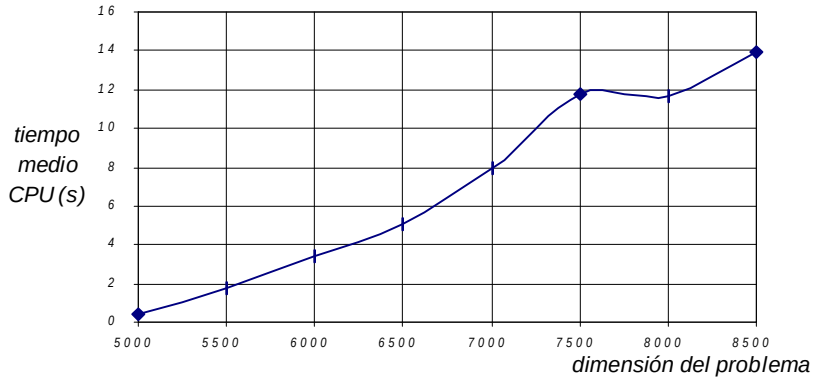

Figura 4: Comparación de los tiempos medios para LAPJVsp

Por lo que respecta al parámetro σ, las pruebas se han realizado con más de 500 ejemplares con 10 tipos de producto y 500 unidades, para cada uno de los algoritmos LAPVJ y LAPVJsp. Los tiempos que se reflejan en las figuras 5 y 6 corresponden a la ejecución del algoritmo, sin incluir los de cálculo de la matriz. La resolución en la medida de tiempos es de una centésima de segundo, por lo cual, en el caso del LAPJVsp, el número de valores distintos registrados es pequeño y los tiempos aparentemente coinciden para numerosos ejemplares.

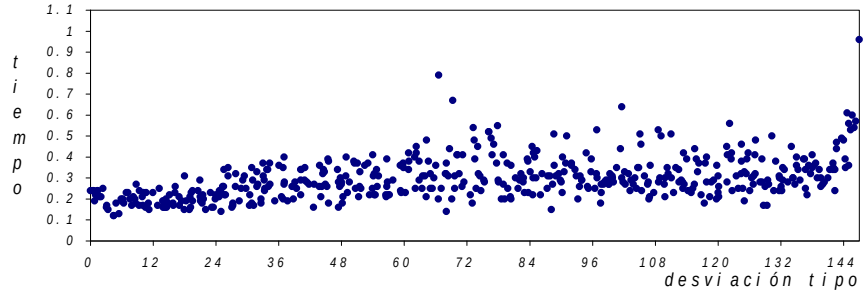

Figura 5 : Comparación de los tiempos (en segundos) para diferentes σ (LAPJV)

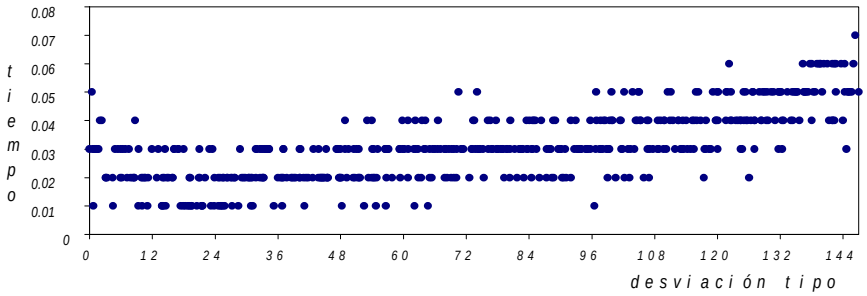

Figura 6 : Comparación de los tiempos (en segundos) para diferentes σ (LAPJVsp)

6. Conclusiones y líneas de trabajo futuras

Aunque la experiencia computacional todavía es limitada parece ya suficiente para indicar que el tratamiento del PRVP como un problema de afectación permite resolver ejemplares de dimensiones considerables en tiempos suficientemente breves para las aplicaciones reales del problema.

Además, los tiempos son menos sensibles al número de tipos que en los algoritmos basados en la programación dinámica.

Por otra parte, el tratamiento del PRVP como un AP se puede aplicar a cualquier tipo de función de discrepancia convexa, con independencia de que la función sea o no la misma para los distintos tipos de productos, con sólo modificar en cada caso el cálculo de los elementos de la matriz, lo cual resulta inmediato, como se ha expuesto en la Sección 2.

A corto plazo, está previsto ampliar la experiencia computacional y el análisis de los resultados de la misma y abordar la resolución del problema $min\ max_{i,h}\ f(x_{ih},h)$ como un problema de afectación de cuello de botella.

Referencias

[1] Monden,Y. (1983) "Toyota Production System", *Institute of Industrial Engineers Press*, Norcross, GA

[2] Miltenburg, J.G. (1989) "Level schedules for mixed-model assembly lines in just-in-time production systems", *Management Science*, 35, 2, 192-207.

[3] Miltenburg, J.G., Steiner G., Yeomans S. (1990) "A dynamic programming algorithm for sheduling mixed-model, just-in-time production systems", *Mathl. Comput. Modelling*, 13, 3, 57-66.

[4] Bautista, J., Companys, R., Corominas, A. (1996a) "Heuristic and exact algorithms for solving the Monden problem", *Eur. J. Opl. Res.* 88, 101-113.

[5] Kubiak, W. (1993) "Minimizing variations of productions rates in just-in-time systems: A survey", *Eur. J. Opl. Res.*, 66, 259-271.

[6] Kubiak, W., Sethi,S. (1991) "A note on 'Level schedules for mixed-model assembly lines in just-in-time production systems'", *Management Science*, 37, 1, 121-122.

[7] Kubiak, W., Sethi,S. (1994) "Optimal just-in-time schedules for flexible transfer lines", *International Journal of Flexible Manufacturing Systems*, 6, 137-154.

[8] Bautista, J., Companys, R., Corominas, A. (1997) "Modelling and solving the production rate variation problem", *TOP* , vol. 5, 2, 221-239.

[9] Korkmazel, T., Meral, S. (2001) "Bicriteria sequencing methods for the mixed-model assembly line in just-in-time production systems", *Eur. J. Opl. Res.* 131, 188-207.

[10] Dell'Amico, M., Toth P. (1998) "Algorithms and codes for dense assignment problems: The state of the art", *DEP, Universitá di Modena, DEIS, Universitá di Bologna.*

[11] Jonker, R., Volgenant,A. (1987) "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems", *Computing* 38, 325-340.

[12] Amatller,J. (1999) " El problema d'afectació", documento de trabajo no publicado

[13] Barr, R.S., Glover F., Klingman, D. (1977) "The alternating basis algorithm for assignment problem", *Math. Program.*,13:1-13

[14] Volgenant, A. (1996) "Linear and Semi-assignment problems: A core oriented approach", *Computers Ops Res.* Vol. 23, No.10, 917-932