Mecánica Computacional Volumen XV
Compilado por A. Larreteguy y M. Vénere
Bariloche, noviembre de 1995

ALGORITMOS EFICIENTES EN PROBLEMAS DE ASIGNACION

Ignacio Ponzoni
Departamento de Matemática
Universidad Nacional del Sur, Av. Alem 1253, 8000 Bahía Blanca, Argentina

Mabel C. Sánchez, Nélida B. Brignole*
Planta Piloto de Ingeniería Química,
UNS - CONICET, 12 de Octubre 1842, 8000 Bahía Blanca, Argentina

RESUMEN

Este trabajo contiene un estudio comparativo sobre metodologías para resolver problemas de asignación. Se implementaron en PC 486 varios algoritmos de reordenamiento que utilizan en forma separada y combinada técnicas basadas en la teoría de grafos y orientadas a ecuaciones. Además se desarrolló un nuevo algoritmo especialmente adaptado para aplicaciones en Instrumentación de Procesos. Las técnicas se evaluaron para varios casos de estudio correspondientes a sectores de planta existentes. Se comprobó que la generación de fronteras con filas y columnas spike no resulta adecuada, mientras que el nuevo enfoque es notoriamente eficiente.

ABSTRACT

A comparative study of methodologies to solve assignment problems has been carried out. Several reordering algorithms were implemented in a 486 PC. Both equation- oriented techniques and methods based on graph theory were considered not only individually but also in combination. A new algorithm is proposed besides, which is suitable for applications to process instrumentation. In all cases, the performance was assessed by trying out several case studies corresponding to existing plant sections. It was found out that the generation of boundaries with spike rows and columns is not adequate for this purpose, whereas the new approach is remarkably efficient.

INTRODUCCION

La evaluación de la performance, el control y la optimización de una planta química se basa en el conocimiento de su estado actual. Este queda determinado por los valores de las variables del proceso contenidas en el modelo propuesto para representar su operación y/o rentabilidad. Dicho modelo está compuesto por un conjunto de relaciones funcionales no lineales entre las variables. A este sistema le corresponde una matriz de ocurrencia rectangular con mayor número de variables que de ecuaciones, rala y de gran dimensión.

A fin de especificar algunas variables se realizan mediciones en el proceso. Por razones físicas, económicas o espaciales sólo se instalan instrumentos sobre un número reducido de corrientes.

* Autor a quien debe enviarse la correspondencia.

Algunas variables no medidas pueden calcularse utilizando el modelo funcional y las mediciones. Para establecer este subconjunto se requiere clasificar las variables no medidas en determinables e indeterminables. Una variable no medida es determinable si puede ser estimada en función de las observaciones utilizando las ecuaciones del modelo; en caso contrario será indeterminable. En esta última situación se evalúa la necesidad y factibilidad de incorporar nuevos instrumentos que brinden la información no disponible.

Una metodología muy difundida para la clasificación de variables consiste esencialmente en determinar qué información puede ser obtenida mediante la resolución de subconjuntos de dichas ecuaciones. Para ello es necesario efectuar un reacomodamiento estructural de la matriz de incidencia del proceso, disponiéndola en forma triangular en bloques. Esta etapa es clave porque condiciona la cantidad de instrumentación mínima para conseguir un conocimiento completo de las variables operativas requeridas.

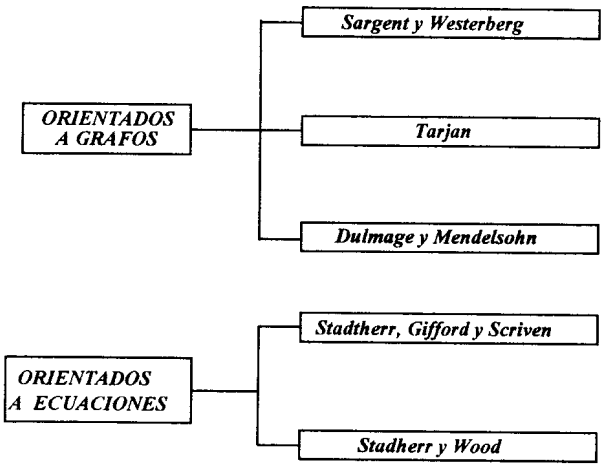
En la literatura referida a algoritmos de asignación pueden distinguirse dos líneas de trabajo. Algunas técnicas, tales como las propuestas en Stadtherr, Gifford y Scriven [1] y Stadherr y Wood [2], están orientadas a ecuaciones. En cambio, Sargent-Westerberg, Tarjan (implementaciones de Duff y Reid [3]) y Dulmage-Mendelsohn [4] utilizan grafos. En este trabajo se analizaron e implementaron los algoritmos existentes con el objeto de evaluar comparativamente su desempeño y de este modo poder seleccionar la estrategia más conveniente.

La mejor técnica de análisis estructural para resolver el problema planteado es aquella que logre formar la mayor cantidad de conjuntos de asignación respetando una estructura estrictamente triangular inferior con bloques cuadrados, a excepción del último que está asociado a las variables indeterminables. Ninguno de los métodos existentes cumplió satisfactoriamente estos objetivos cuando se los aplicó individualmente a problemas reales. Por esta razón surgió la necesidad de desarrollar una nueva estrategia especialmente adaptada para diseño de sistemas de monitoreo. En este contexto, se intentó en primer lugar, realizar modificaciones de los procedimientos orientados a las ecuaciones. Si bien mediante metodologías de este estilo se lograron ciertas mejoras, no se alcanzó el nivel de robustez requerido. Fue necesario entonces desarrollar un nuevo algoritmo que se basó en recorridos recursivos de árboles.

El desarrollo de estrategias tendientes a resolver sistemas de ecuaciones no lineales ralos es útil para solucionar una gran variedad de problemas prácticos. En ingeniería de procesos, estos algoritmos son potencialmente aplicables no sólo a cuestiones de instrumentación, sino también a modelamiento para simulaciones estacionarias y dinámicas de equipos individuales o sectores de planta y para optimización. Las técnicas de solución numérica revisten particular importancia en el caso de los nuevos simuladores combinados discreto-continuos [5]. En estos paquetes es fundamental diagnosticar automáticamente singularidades estructurales y guiar al usuario en la selección de especificaciones con eficiencia.

DESCRIPCIÓN DE LOS ALGORITMOS EXISTENTES

Se han desarrollado varios algoritmos de reordenamiento concernientes a la resolución de problemas de asignación. Estos pueden clasificarse en dos grupos:



Algoritmos basados en la teoría de grafos

Los algoritmos de Sargent-Westerberg y Tarjan trabajan mediante permutaciones simétricas, por lo que se requiere que las matrices sean cuadradas. Esto impone una fuerte restricción que podría salvarse mediante el uso de filas llenas complementarias y transposición de matrices. No obstante ello, no es siempre posible obtener la forma deseada, es decir *triangular inferior en bloques*. Además debemos considerar el costo adicional de almacenar y procesar estas filas llenas.

Por último el algoritmo de Dulmage y Mendelsohn trabaja con grafos bipartitos, permitiendo así su aplicación a matrices generales. Sin embargo, el reordenamiento obtenido por este método no se adapta para la selección de instrumentación porque origina una forma triangular inferior con el primer bloque rectangular.

Algoritmos orientados a las ecuaciones

Stadtherr, Gifford y Scriven presentaron una estrategia de reordenamiento que obtiene una estructura de fronteras con filas y columnas spike. A pesar de no ser la forma buscada, algunas de las técnicas para la detección de conjuntos de asignación fueron utilizadas y modificadas en el algoritmo desarrollado por Sánchez y otros [6], el cual se presenta a continuación:

Algoritmo de Sánchez y otros

Repetir
1 - Triangularizar hacia adelante.
2 - Buscar todos los conjuntos de 2x2 mediante el Algoritmo 1.
    Si existen conjuntos de 2x2 volver al Paso 1.
    En cualquier otro caso ir al Paso 3.

3 - n = 3,
  Repetir
    Buscar todos los posibles conjuntos de nxn mediante el Algoritmo 2.
    Si existen conjuntos de nxn volver al Paso 1.
    En cualquier otro caso n = n+1.
  hasta (n > tope)
hasta (n > tope) ó (se asignan todas las variables)

Algoritmo 1 : Búsqueda de conjuntos de asignación de 2x2.

Para este algoritmo se utilizó la Subrutina 2 presentada por Stadtherr, Gifford y Scriven; se da aquí una breve descripción de la misma :

1- Formar la submatriz consistente de todas las filas con sólo dos entradas
  y las columnas en que dichas entradas aparecen.
2- Repetir
    Repetir
      Buscar un columna con una sola entrada.
      Borrar dicha columna y la fila asociada.
      Borrar cualquier columna que no contenga entradas.
    hasta que no queden columnas con menos de dos entradas.
    Buscar la columna con el mayor número de entradas.
    Borrar esta columna y sus filas asociadas.
    Buscar una columna en la cual el número de entradas se halla reducido en dos y borrarla.
    Las dos columnas borradas en este paso junto con las filas que contienen en común estas columnas forman un conjunto removible de 2x2, que luego puede chequearse para ver si es una asignación permitida.
    Si dos o más columnas han reducido sus entradas en dos
      entonces
        más de un conjunto de 2x2 ha sido encontrado y sólo uno puede ser removido.
    Borramos toda columna que quede vacía.
  hasta que la submatriz este vacía.

Para la obtención de los bloques de nxn (para n > 2) el algoritmo básico es:

1- Formar la submatriz con todas las filas con a lo sumo n elementos distintos de cero y las columnas en que dichos elementos aparecen.
2- Borrar toda columna con un solo elemento distinto de cero.
3- Si hay n columnas que tengan todos sus elementos distintos de cero en las mismas n filas entonces se detecta un bloque de nxn.

Lamentablemente este último algoritmo, denominado Básico, es muy expansivo. Si la cantidad de filas y columnas de la submatriz obtenida en el Paso 1 es s y r respectivamente; para conjuntos de tamaño *n* tenemos que el número de combinaciones posibles de filas y columnas para un conjunto de asignación de *nxn* es :

$$C_r^n \times C_s^n = \frac{r! \times s!}{2 \times n! \times (r-n)! \times (s-n)!} \qquad (1)$$

Debido al elevado costo de este método surgió la necesidad de utilizar el siguiente algoritmo.

Algoritmo 2 : Búsqueda de conjuntos de asignación de nxn (n > 2).

1- Formar la submatriz con todas las filas con a lo sumo n elementos distintos de cero y las columnas en que dichos elementos aparecen.
2- Repetir
  Borrar toda columna con un solo elemento distinto de cero.
  Elegir la columna con mayor cantidad de entradas.
  Borrar dicha columna y temporalmente las filas asociadas a sus entradas
  Si hay n-1 columnas cuyo número de entradas se haya reducido en dos o más
    entonces
      Se remueven dichas columnas temporalmente.
      Si hay n filas vacías
        entonces
          se identifica un conjunto de nxn y se borran dichas filas y columnas.
        si no
          se restauran las filas y columnas borradas temporalmente.
      fin Si
    si no
      se restauran las filas borradas temporalmente.
  fin Si.
 hasta que la submatriz este vacía.

Con esta estrategia se reduce el número de combinaciones drásticamente, pero en contrapartida se pierde robustez, por ejemplo : sea A la submatriz obtenida en el primer paso del Algoritmo 2 cuando se considera la búsqueda de grupos de 4x4.

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

En el siguiente paso se selecciona la columna número 4 por ser la de mayor cantidad de entradas. Luego se borran temporalmente las filas 1, 2 y 3. Como las columnas 1, 2 y 3 pierden solamente una entrada, no son consideradas candidatas a formar un conjunto de asignación, y por este motivo no se encuentran conjuntos de 4x4.

Sin embargo este grupo de filas y columnas forman un conjunto removible de 4x4 que surgiría si se emplease el Algoritmo Básico. ¿Cómo se mejora el Algoritmo 2 para que tenga la robustez del primero sin pagar un costo computacional tan elevado? Una aproximación sería modificar el Paso 2 de la siguiente manera :

2- Repetir
  Borrar toda columna con un solo elemento distinto de cero.
  Elegir la columna con mayor cantidad de entradas.
  Borrar dicha columna y temporalmente las filas asociadas a sus entradas
  Si hay n-1 columnas cuyo número de entradas se haya reducido en *uno* o más

```
    entonces
      Se remueven dichas columnas temporalmente.
      Si hay n filas vacías
        entonces
          se identifica un conjunto de nxn y se borran dichas filas y columnas.
        si no
          se restauran las filas y columnas borradas temporalmente.
      fin Si
    si no
      se restauran las filas borradas temporalmente.
  fin Si.
hasta que la submatriz este vacía.
```

El Algoritmo Modificado logra identificar grupos como los existentes en el ejemplo previo sin aumentar significativamente la cantidad de combinaciones recorridas. ¿Tenemos ahora la robustez del Algoritmo Básico? La respuesta es no, esto se observa en el siguiente bloque del ejemplo presentado por Pissanetzki [7]:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Este bloque de 7x7 es hallado tanto por el Algoritmo Básico como por los de grafos, sin embargo el Algoritmo Modificado no lo encuentra. ¿Por qué? La explicación es sencilla: ninguna de las siete columnas tiene filas en común con cada una de las restantes. Por ejemplo : si elegimos a la columna 1 como primera, las columnas 3, 5, 6 y 7 no poseen filas en común con ésta, razón por la cual no serán consideradas para la formación del conjunto de asignación. Lo mismo ocurre para las demás columnas.

Debido a lo expuesto se ensayaron distintas modificaciones y combinaciones de las estrategias anteriores. Si bien dichos métodos lograron ciertas mejoras, no se pudo obtener una solución que cumpliera simultáneamente con los siguientes objetivos :

- Robustez: implica encontrar el mayor número de bloques posibles
- Eficiencia: significa resolver en un tiempo de cómputo aceptable

Teniendo en cuenta dichas metas se ha desarrollado un nuevo algoritmo denominado *"Búsqueda Arbórea"*. Este método logra la robustez del Básico con un tiempo de búsqueda inferior en varios órdenes de magnitud.

UN NUEVO ALGORITMO DE RESOLUCIÓN

A continuación se presenta un nuevo método para la asignación de grupos de nxn. Este sustituye al Algoritmo 2 en la estrategia de Sánchez y otros.

Algoritmo Búsqueda Arbórea : Búsqueda de conjuntos de asignación de nxn.

El algoritmo propuesto se basa en un recorrido recursivo en profundidad de un árbol. Los nodos de dicho árbol representan las columnas de la submatriz, mientras que la presencia de una arista desde el nodo n_i al nodo n_j indica que las columnas i y j poseen una fila en común.

Búsqueda Arbórea (submatriz,n,k,éxito)

```
Si  k = n
  entonces
    Elegir como k-ésima columna alguna que tenga una entrada en común con la (k-1)-ésima.
    Si no existe tal columna
      entonces
        éxito = falso.
      si no
        Si al borrar las k columnas elegidas quedan n filas vacías
          entonces { Se encontró un conjunto de asignación de nxn  formado por
                     las n filas vacías y las k columnas seleccionadas. }
            Asignar las n filas vacías al conjunto de nxn.
            éxito = verdadero.
          si no
            Búsqueda Arbórea (submatriz,n,k,éxito)
        fin del Si.
    fin del Si.
  si no      { Es el caso en que k < n }
    Si k = 1
      entonces
        Elegir como columna la de mayor cantidad de entradas.
      si no
        Elegir como k-ésima alguna que tenga una entrada en común con la (k-1)-ésima.
    fin Si.
    Si no existe tal columna
      entonces
        éxito = falso.
      si no
        Búsqueda Arbórea (submatriz,n,k-1,éxito)
        Si éxito
          entonces
            Asignar la k-ésima columna al conjunto de nxn.
          si no
            Búsqueda Arbórea (submatriz,n,k,éxito).
        fin Si.
    fin Si.
fin Si.
```

Donde *submatriz* es la hallada en el paso previo;
n el tamaño del conjunto buscado;
k la profundidad de búsqueda actual;
éxito un dato lógico que retorna : verdadero si se hallo el conjunto de asignación de *nxn* y falso en caso contrario.

EVALUACIÓN DEL DESEMPEÑO DEL ALGORITMO

La evaluación del desempeño del algoritmo propuesto se efectúa comparando su grado de eficiencia y robustez con respecto a otros métodos.

Para medir la eficiencia del nuevo método se requiere determinar la cantidad de combinaciones de columnas que genera el Algoritmo Básico y el de Búsqueda Arbórea. Es necesario utilizar como referencia el Algoritmo Básico porque proporciona la misma robustez. Si implementamos este último algoritmo siguiendo una política de recorrido en profundidad (similar a la utilizada por la búsqueda arbórea), el número de combinaciones de columnas es:

$$C_r^n = \frac{r!}{n! \times (r-n)!} \tag{2}$$

mientras que la cantidad de combinaciones que recorre la búsqueda arbórea está dada por:

$$C_{ba} = r \times (p-1)^{n-1} \tag{3}$$

donde r es el número de columnas en la submatriz,
p es el promedio de elementos distintos de cero en una columna, y
n es el tamaño del conjunto de asignación buscado.

El valor de p puede expresarse como:

$$p = f_{red}(s \times pr/100) \tag{4}$$

donde s es el número de filas,
pr es el porcentaje de elementos distintos de cero de la matriz, y
$fred$ es una función de redondeo, que asigna a un número real el entero más cercano.

Es posible establecer una comparación entre ambos métodos mediante los valores de la Tabla I, obtenidos para $s = 50$, $r = 50$, $pr = 6.5$ %, $p = 3$ y donde n es el tamaño de bloque buscado. Evidentemente se observa una reducción de varios órdenes de magnitud en el número de combinaciones exploradas por el Algoritmo de Búsqueda Arbórea.

En cuanto a la robustez del método, varios ejemplos reflejan las ventajas de una búsqueda que explore todas las combinaciones factibles. En este trabajo se analizan: un ejemplo presentado por Pissanetzki, otro proveniente de un sector de planta existente [8] y un caso de estudio elaborado por los autores. En la Tabla II se observa que el Método de Búsqueda Arbórea es el más robusto, puesto que identifica todos los conjuntos de asignación posibles.

Tabla I
Comparación del número de combinaciones exploradas

n	Búsqueda Arbórea	Básico
3	200	19.600
4	400	230.000
5	800	2.118.760
6	1.600	15.890.700
7	3.200	99.884.400
8	6.400	536.878.650
9	12.800	2.505.433.700
10	25.600	10.272.278.170

Tabla II
Número de bloques hallados

Ejemplos	Alg. Sánchez y otros.	Alg. Modificado	Alg. Búsqueda Arbórea
Pissanetzki (pág. 163)	1 de (15x15)	1 de (5x5) 1 de (1x1) 1 de (9x9)	1 de (5x5) 1 de (1x1) 1 de (7x7) 1 de (2x2)
Joris	2 de (1x1) 1 de (12x12)	2 de (1x1) 3 de (4x4)	2 de (1x1) 3 de (4x4)
Caso de Estudio	1 de (14x14)	1 de (14x14)	1 de (6x6) 3 de (1x1) 1 de (2x2) 1 de (3x3)

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

Caso de Estudio

CONCLUSIONES

Se realizó un estudio comparativo de algoritmos de asignación, que permitió establecer claramente los alcances de las distintas metodologías existentes con respecto a su aplicación específica en problemas de instrumentación. Debido a las limitaciones encontradas tanto en lo referente a la calidad de las particiones como en cuanto a la eficiencia de búsqueda, fue necesario plantear nuevas propuestas.

Por razones estructurales se descartaron las técnicas de grafos tradicionales. Se propusieron, entonces, modificaciones a los procedimientos orientados a ecuaciones. Dado que, en algunos casos, esta estrategia omitía asignaciones factibles para conjuntos mayores de 3x3, se desarrolló un nuevo algoritmo basado en recorrido de árboles (backtracking). El método es robusto y eficiente. Esta última característica se logra limitando el espacio de búsqueda únicamente a las combinaciones factibles.

REFERENCIAS

[1] Stadtherr M.A., Gifford W.A. y Scriven L.E., *Efficient Solution of Sparse Sets od Design Equations.* Chemical Engineering Science, 1974. Vol. 29.

[2] Stadtherr M. A. and Wood E. S., *Sparse Matrix Methods for Equation-Based Chemical Process Flowsheeting - I,* Computers & Chemical Engineering, Vol. 8, 1984, págs 9-18.

[3] Duff I.S., Reid J.K., *An Implementation of Tarjan's Algoritm for the Block Triangularization of a Matrix.* ACM Transactions on Mathematical Software June 1978, Vol. 4, N°2.

[4] Dulmage A.L., Mendelsohn N.S., *Two Algorithms for Bipartite Graphs.* J.Soc. Indust. Appl. Math. Vol.11, N°1, March, 1963.

[5] Pantélides C.C., Barton P.I., *Equation-oriented Dynamic Simulation: Current Status and Future Perspectives.* Comp. Chem. Eng., 1992, S263-S285.

[6] Sánchez M., Bandoni A. y Romagnoli J., *PLADAT - A Package for Process Variable Classification and Plant Data Reconciliation.* Comput. Chem. Eng., 616 (Suppl), 1992, S499-506

[7] Pissanetzky S., *Sparse Matrix Technology.*, Academic Press, Inc., 1984

[8] Joris P., Kalitventzeff B., *Process Measurement Analysis and Validation*, Proceedings of XVIII Congress: The Use of Computer in Chemical Engineering, Italia, 1987, págs 41-46.