Capítulo 5:
GRAPH TRAVERSE SCHEDULING, Vectorización de Recurrencias

GTS es una técnica que permite la vectorización de recurrencias obteniendo instrucciones vectoriales de máxima longitud. La distribución de operaciones se basa en la idea de ejecutar, en modo vectorial y para cada sentencia del bucle, tantas iteraciones posibles como permitan las dependencias 'que aparecen consigo misma a través de las recurrencias en que se ve involucrada.

En este capítulo se considera, en primer lugar, la aplicación de GTS a grafos que incluyen una única recurrencia hamiltoniana. En este caso, todas las sentencias del bucle pueden ser ejecutadas con la misma longitud vectorial. La aplicación de GTS a grafos hamiltonianos con varias recurrencias puede provocar que las sentencias tengan que ejecutarse con distintas longitudes vectoriales, dificultando la posterior generación de código vectorial. Con la finalidad de simplificarla, GTS se ve soportado por técnicas adicionales que permiten modificar los arcos existentes o añadir nuevos arcos al grafo de dependencias.

5.1 GRAFOS CON UNA ÚNICA RECURRENCIA HAMILTONIANA

La longitud vectorial que GTS asignaba cada una de las sentencias del bucle queda determinada por el pesó de la recurrencia hamiltoniana que cubre todos los nodos del grafo.

El algoritmo utilizado por GTS para realizar la distribución de operaciones para este tipo de máquinas es el siguiente:

(a) Con la finalidad de permitir la ejecución vectorial de la primera sentencia Si del bucle con su máxima longitud vectorial, GTS ejecuta todas aquellas iteraciones de sentencias que liberan la ejecución de todas las S_{ij} que no dependen de S_{11}.

De esta manera, se ejecutan, siguiendo el recorrido de la recurrencia hamiltoniana R, $w(C_{ij})$ iteraciones consecutivas de cada sentencia S_j del bucle ($1< j \leqq n$).

(b) A continuación se ejecutan $lv = w(R)$ iteraciones de cada sentencia S_j del bucle ($1 \leqq j \leqq n$), recorriendo la recurrencia hamiltoniana R.

La iteración inicial de cada sentencia a partir de la cual se puede iniciar su ejecución vectorial viene dada por $i_0 + w(C_{ij})$, siendo i_0 la iteración inicial del bucle.

La figura 5.1(b)muestra. La distribución de operaciones obtenida por GTS siguiendo el algoritmo anterior para el bucle con grafo de dependencias asociado mostrado en la figura. 5.l(a). Esta distribución muestra por filas el rango de iteraciones de cada sentencia que puede ser ejecutado en modo vectorial.

5.2 GRAFOS HAMILTONIANOS CON VARIAS RECURRENCIAS

Tal como se ha descrito en el apartado 3.2.2, cada sentencia Si de un bucle con grafo hamiltoniano y varias recurrencias, posee asociada una longitud vectorial $lv(S_i)$, determinada por el peso de la recurrencia más cerrada que la incluye.

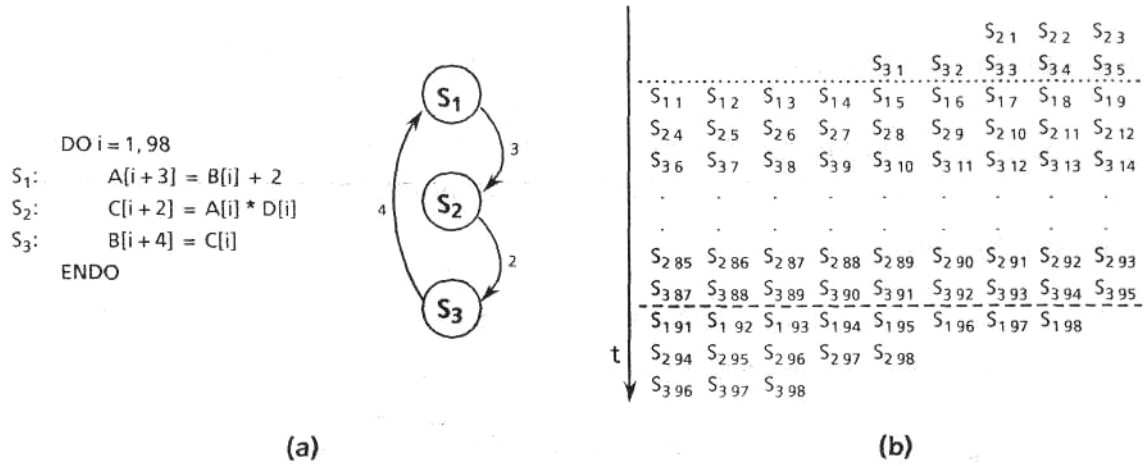


Figura 5.1: Vectorización de un bucle con una única recurrencia hamiltoniana.

Distribución de operaciones

GTS intenta ejecutar tantas iteraciones de cada sentencia Si en forma vectorial como indique la máxima longitud vectorial lv_{max} de todas las sentencias del bucle.

Aquellas sentencias con una longitud vectorial inferior se ejecutarán en varias pasadas con su longitud vectorial asociada, de manera que se cubran las lv_{max} iteraciones.

La figura 5.2(b) muestra la distribución de operaciones obtenida para el bucle y grafo de dependencias con varias recurrencias mostrados en la figura 5.2(a). Observar que al ser la longitud vectorial de S_2 y S_3 la mitad de la longitud vectorial de S_1, es preciso realizar su ejecución en dos pasadas secuenciales, de manera que al final se ejecutan el mismo número de iteraciones de las tres sentencias.

Condiciones de vectorización

Dado un grafo G(V, E), GTS renumera sus nodos atravesando R_{sch} empezando por una sentencia Si que cumple

$$lv(S_1) = \mathbf{max}_{S_i \in V}\, lv(S_i).$$

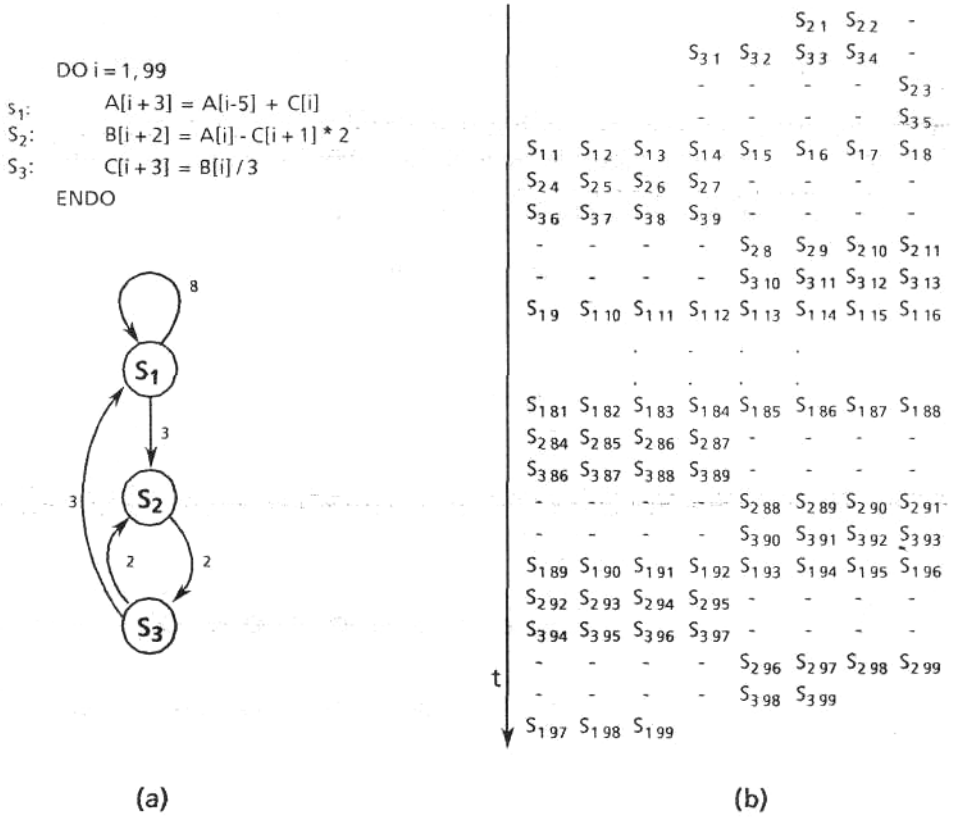


Figura 5.2: Vectorización de un bucle hamiltoniano con varias recurrencias.

Las condiciones que deben de cumplirse para realizar una distribución de operaciones correcta, siguiendo el criterio mencionado al principio de esta sección, son las siguientes:

(a) El peso de la recurrencia R_{sch} debe ser igual a la máxima longitud vectorial, es decir,

$$w(\mathbf{R_{sch}}) = lv(S_1).$$

(b) Para cada dependencia $d_{ij} \notin R_{sch}$ debe cumplirse la siguiente condición:

$$\forall \mathbf{d_{ij} \notin R_{sch}} \qquad d_{ij} \geq \begin{cases} w(C_{ij}) & | \; C_{ij} \subset R_{sch} \quad \text{if } i<j \\ vl(S_i, S_j) - w(C_{ji}) & | \; C_{ji} \subset R_{sch} \quad \text{if } i \geq j \end{cases}$$

siendo

$$lv(S_i, S_j) = \mathbf{max}\ (lv(S_i), lv(S_j)).$$

Esta última condición es una consecuencia de imponer que cualquier arco de dependencia $d_{ij} \notin R_{sch}$ debe limitar menos que el camino alternativo a

través de R_{sch}, dado que en caso contrario se intentarían ejecutar iteraciones de sentencias no liberadas. Si dicho arco va en la misma dirección que R_{sch}, o sea i < j, tal como muestra el esquema de la figura 5.3(a), la última iteración de Si ejecutada debe liberar una iteración posterior a la última que se va a ejecutar de S_j en la misma pasada por R_{sch}. Por lo tanto, si *k* es la última iteración de S_i ejecutada en una pasada debe cumplirse que

$$k + d_{ij} \geq k + w(C_{ij}),$$

es decir,

$$d_{ij} \geq w(C_{ij}).$$

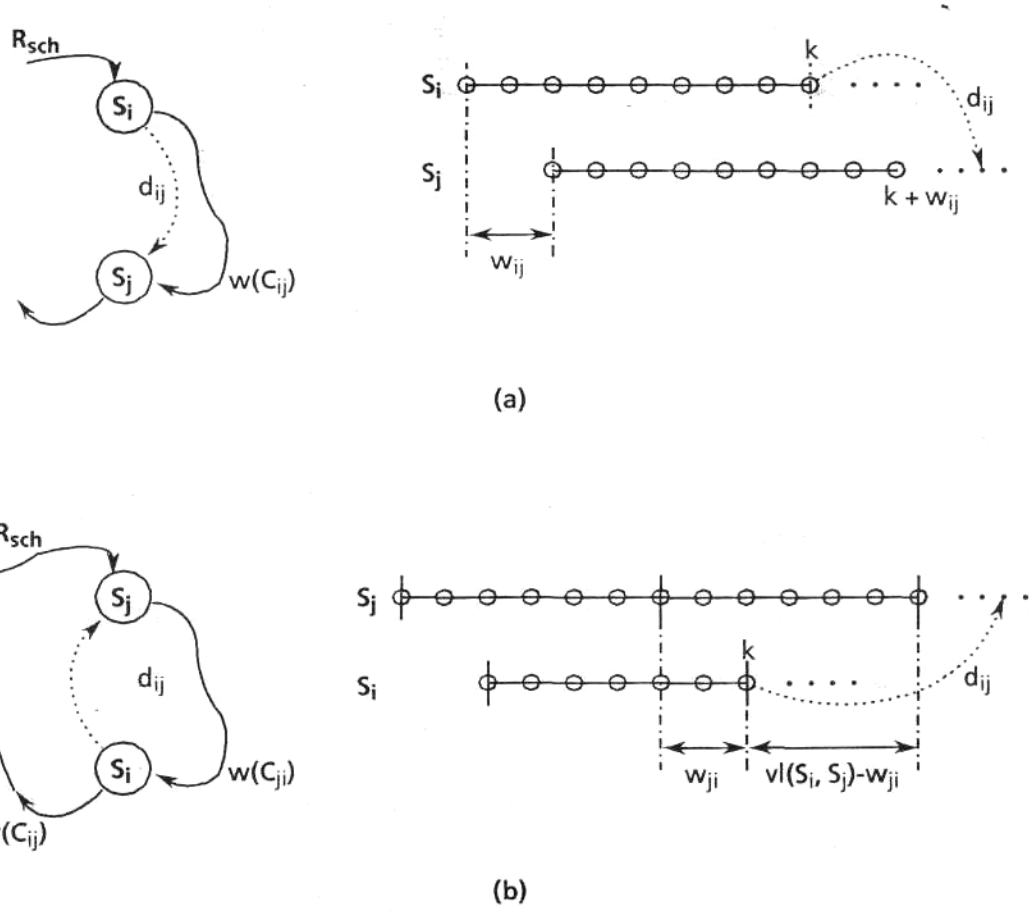


Figura 5.3 : Condición (b) de la sección 5.2 para la correcta distribución de operaciones aplicando GTS

Sin embargo, si dicho arco forma un ciclo con R_{sch}, o sea i≥j, tal como muestra el esquema de la figura 5.3(b), la última iteración de Si debe liberar una iteración posterior a la última que va a ser ejecutada de S_j en su siguiente ejecución vectorial. Por lo tanto, si *k* es la última iteración de Si ejecutada en

una pasada debe de cumplirse que

$$k + d_{ij} \geq k + lv(S_i, S_j) - w(C_{ji}).$$

es decir

$$d_{ij} \geq lv(S_i, S_j) - w(C_{ji}),$$

La siguiente condición es impuesta por GTS con la finalidad de facilitar la generación de código vectorial, descrita en la siguiente sección de este mismo capítulo. Las longitudes vectoriales de las sentencias del bucle deben cumplir que

$$\textbf{m.c.d}\ (lv(S_i), lv(S_j)) = \textbf{min}\ (lv(S_i), lv(S_j)) \qquad \forall S_i, S_j$$

de manera que se hagan un, número entero de pasadas para aquellas sentencias vectoriales con longitud vectorial inferior a la máxima.

Si alguna de las condiciones anteriores no se cumple, se pueden añadir nuevos arcos o modificar las distancias asociadas a los ya existentes a fin de asegurar su cumplimiento. Así por ejemplo será necesario reducir la longitud vectorial de algunas sentencias con la finalidad de cumplir la última condición descrita. Estos aspectos son considerados en la sección 5.4.

5.3 GENERACIÓN DE CÓDIGO VECTORIAL

El código vectorial generado por GTS consta de tres partes: prólogo, núcleo y epílogo.

Prólogo

En el prólogo se ejecutan de forma vectorial o secuencial (en función del número de iteraciones de las sentencias a ejecutar) aquellas instanciaciones de sentencias que liberan la ejecución de todas las S_{ij} que no dependen de Su, siguiendo el recorrido de R_{sch}. Por ejemplo, en las figuras 5.1(b) y 5.2(b) dichas instanciaciones se corresponden con las iteraciones de sentencias por encima de la línea de puntos.

La figura 5.4(b) muestra la estructura general del prólogo ejecutado de forma secuencial para el bucle general de la figura 5.4(a) que incluye una R_{sch} tal que $S_i \delta S_i+1$ $(1 \leq i \leq n-1)$ y $S_n \delta S_i$. Observar que las sentencias condicionales controlan que sólo se ejecuten las primeras $w(C_{ij})$ iteraciones de cada sentencia S_j tal que $2 \leq j \leq n$.

Núcleo

El núcleo está constituido por la ejecución de sentencias con su máxima longitud vectorial y cuya estructura general se describe a continuación.

Sea NL el número de longitudes vectoriales $lv(S_i)$ distintas en el grafo y lv_i, lv_2, ..., Iv_{NL} las distintas $lv(S_i)$ en orden decreciente. El código generado por GTS pretende ejecutar (a) las sentencias con longitud vectorial máxima de una sola pasada y el resto en varias pasadas con su longitud vectorial asociada.

En general, el código vectorial generado está constituido por NL bucles anidados L_1, L_2,..., L_{NL} siendo L_1 el bucle más externo. Cada bucle L_i lleva asociada una variable de control I_i con valor inicial I_{i-1}, valor final $I_{i-1} + 1_{Vi-1}$-1 e incremento lv_i. El bucle L_1 lleva asociada una variable de control I_1 con valor inicial 1, valor final

$$\lfloor (N - w(C_{1n}))/lv_1 \rfloor \cdot \quad lv_1$$

e incremento lv_i.

Cada sentencia S_j a lo largo de R_{Sch} queda incluida en un bucle L_k tal que $lv_k = lv(S_i)$. Su longitud vectorial es lv_k y la iteración inicial $I_k + w(C_{1i})$. Todas las sentencias aparecen en el mismo orden léxico en que aparecen al recorrer R_{sch} en el sentido indicado por los arcos de dependencia.

Epílogo

El epílogo está constituido por la ejecución de aquellas pasadas a través de R_{sch} para las cuales existe alguna sentencia que no puede ejecutarse con su máxima longitud vectorial. En el ejemplo de las figuras 5.1(b) y 5.2(b) se corresponden con las iteraciones de sentencias por debajo de la línea punteada. La figura 5.4(c) muestra la estructura general para la ejecución secuencial del epílogo para el bucle general mostrado en la figura 5.4(a).

Las figuras 5.5(a) y 5.5(b) muestran los códigos vectoriales generados para los bucles secuenciales respectivos de las figuras 5.1(a) y 5.2(a). Observar que en el primer caso, el epílogo y prólogo también son ejecutados en forma vectorial. El compilador debe de decidir su forma de ejecución en función de la longitud vectorial de dichas sentencias.

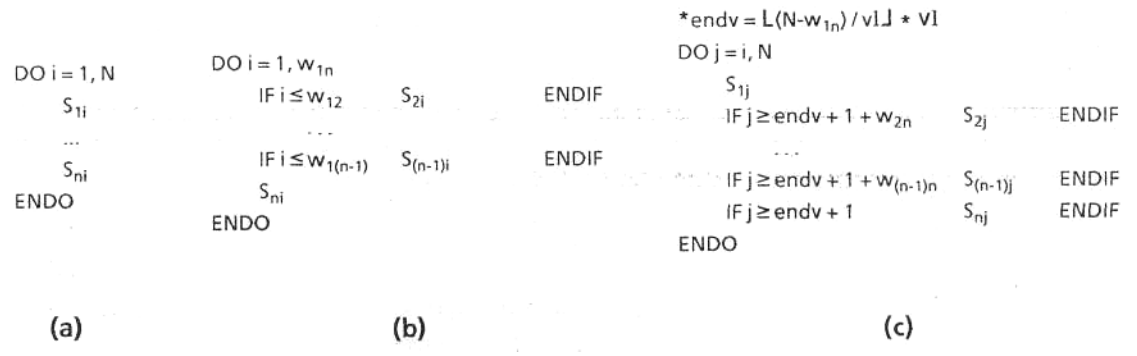


Figura 5.4: Estructura general para el prólogo y epílogo del código vectorial generado.

5.4 MODIFICACIONES DEL GRAFO DE DEPENDENCIAS

A continuación se presentan las técnicas que pueden aplicarse antes que GTS con la finalidad de realizar modificaciones en el grafo de dependencias. Estas técnicas permiten:

(a) obtener una R_{sch} a fin de poder aplicar GTS;

(b) asegurar el cumplimiento de las condiciones de vectorización descritas en la sección 5.2;

(c) facilitar la generación de código vectorial.

5.4.1 Obtención de Rsch

Se pueden considerar dos casos cuando no existe una R_{sch} que cubra todos los nodos del grafo: (a) la existencia de nodos no incluidos en ningún ciclo de dependencias y (b) la existencia de nblocks cíclicos sin recurrencia hamiltoniana.

Grafos con puntos aislados

En este caso, las sentencias asociadas a aquellos nodos del grafo no incluidos en ningún ciclo de dependencias pueden ejecutarse con longitud vectorial máxima igual al número de iteraciones del bucle., En este caso es preferible utilizar la técnica de distribución de bucles con la finalidad de separar puntos aislados del grafo de nblocks cíclicos, cuyas sentencias se deben de ejecutar con

```
      DO i = 1, 98
S1:       A[i + 3] = B[i] + 2
S2:       C[i + 2] = A[i] * D[i]
S3:       B[i + 4] = C[i]
      ENDO
```

```
S2:  C(4:6) = A(1:3) * D(1:3)                          *prólogo
S3:  B(3:7) = C(1:5)
.....
      DO i = 1, 90, 9                                  *núcleo
S1:       A(i + 3:i + 11) = B(i:i + 8) + 2
S2:       C(i + 5:i + 13) = A(i + 3:i + 11) * D(i + 3:i + 11)
S3:       B(i + 9:i + 17) = C(i + 5:i + 13)
      ENDO
.....
S1:  A(94:101) = B(91:98) + 2                          *epílogo
S2:  C(96:100) = A(94:98) * D(94:98)
S3:  B(100:102) = C(96:98)
```

(a)

```
      DO i = 1, 99
S1:       A[i + 3] = A[i-5] + C[i]
S2:       B[i + 2] = A[i] - C[i + 1] * 2
S3:       C[i + 3] = B[i] / 3
      ENDO
```

```
      DO i = 1, 5                                      *prólogo
          IF i ≤ 3
S2:           B[i + 2] = A[i] - C[i + 1] * 2
          ENDIF
S3:       C[i + 3] = B[i] / 3
      ENDO
.....
      DO i = 1, 88, 8                                  *núcleo
S1:       A[i + 3:i + 10] = A[i-5:i-6] + C[i:i + 7]
          DO j = i, i + 4, 4
S2:           B[j + 5:j + 8] = A[j + 3:j + 6] - C[j + 4:j + 7] * 2
S3:           C[j + 8:j + 11] = B[j + 5:j + 8] / 3
          ENDO
      ENDO
.....
      DO i = 89, 99                                    *epílogo
S1:       A[i + 3] = A[i-5] + C[i]
          IF i ≥ 92
S2:           B[i + 2] = A[i] - C[i + 1] * 2
          ENDIF
          IF i ≥ 94
S3:           C[i + 3] = B[i] / 3
          ENDIF
      ENDO
```

(b)

Figura 5.5: Códigos vectoriales generados por GTS para los bucles de las figuras 5.1 (a) y 5.2(b).

la longitud vectorial determinada por la recurrencia más restrictiva que las incluya.

Πblocks cíclicos sin recurrencia hamiltoniana

El segundo caso aparece al tratar Πblocks cíclicos que contienen recurrencias, aunque ninguna de ellas hamiltoniana. Para este tipo de grafos, GTS debe añadir dependencias ficticias a fin de aplicar el algoritmo anterior. Estas

dependencias ficticias no deben limitar la longitud vectorial de ninguna de las sentencias del bucle.

La figura 5.6(a) muestra el grafo de dependencias asociado a un bucle que ejecuta 100 iteraciones, en el que puede observarse que el nodo asociado a S_1 no queda incluido en ningún ciclo del grafo y que los nodos asociados a S_2, S_3 y S_4 forman un ciclo maximal. A fin de aplicar GTS es preferible aplicar distribución de bucles generando dos Πblocks: $\Pi_1 = \{S_1\}$ y $\Pi_2 = \{S_2, S_3, S_4\}$. Las longitudes vectoriales asociadas a cada sentencia del bucle son $lv(S_1) = 100$, $lv(S_2) = 6$ y $lv(S_3) = lv(S_4) = 4$. La vectorización de Π_2 aplicando GTS requiere una R_{Sch} que puede obtenerse añadiendo un arco ficticio d42 con distancia asociada 1, tal y como se muestra en la figura 5.6(b), de manera que no se limitan las longitudes vectoriales originales de las sentencias del bucle.

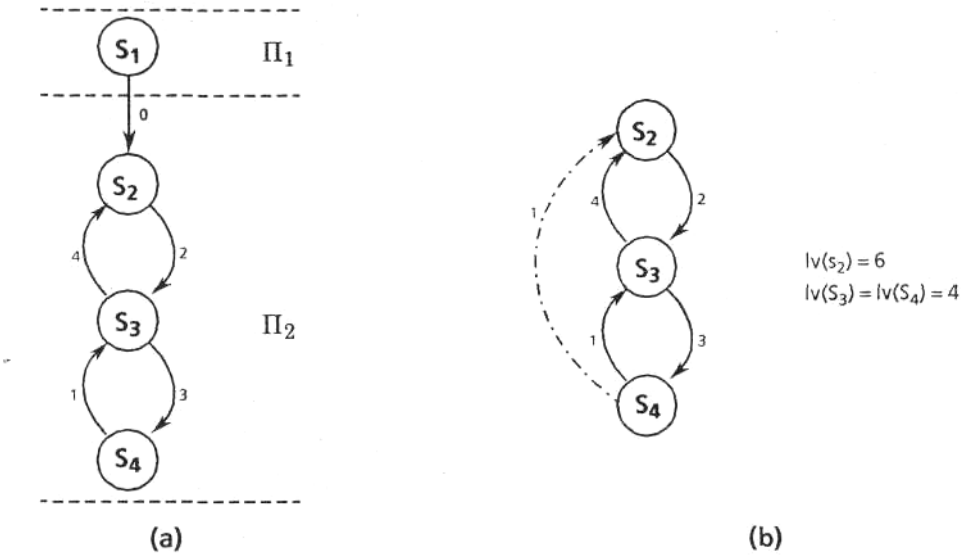


Figura 5.6: Obtención de una R_{sch} a fin de aplicar GTS.

5.4.2 Condiciones de vectorización

A fin de asegurar las condiciones (a) y (b) de la sección 5.2 es posible que se tengan que reducir las distancias asociadas a algunos de los arcos del grafo de dependencias. En ningún caso, esta reducción de las distancias debe limitar la longitud vectorial de ninguna de las sentencias del bucle.

Condición (a): peso de la R_{sch}

En el grafo de la figura 5.7(a) puede observarse que las longitudes vectoriales asociadas a las sentencias son $lv(S_1) = 8$ y $lv(S_2) = lv(S_3) = 4$. Si la recurrencia

es $R_{sch} = \{d_{12}, d_{23}, d_{31}\}$ se tiene que la condición (a) no se cumple ya que $w(R_{sch}) = 10 \neq vl_{max} = 8$. Por lo tanto, previamente a la aplicación de GTS debe de reducirse alguno de los arcos de R_{sch} a fin de asegurar esta condición y sin modificar las longitudes vectoriales originales. La solución se muestra en la figura 5.7(b) en cuyo grafo de dependencias se ha reducido la distancia asociada al arco d_{31} a 1.

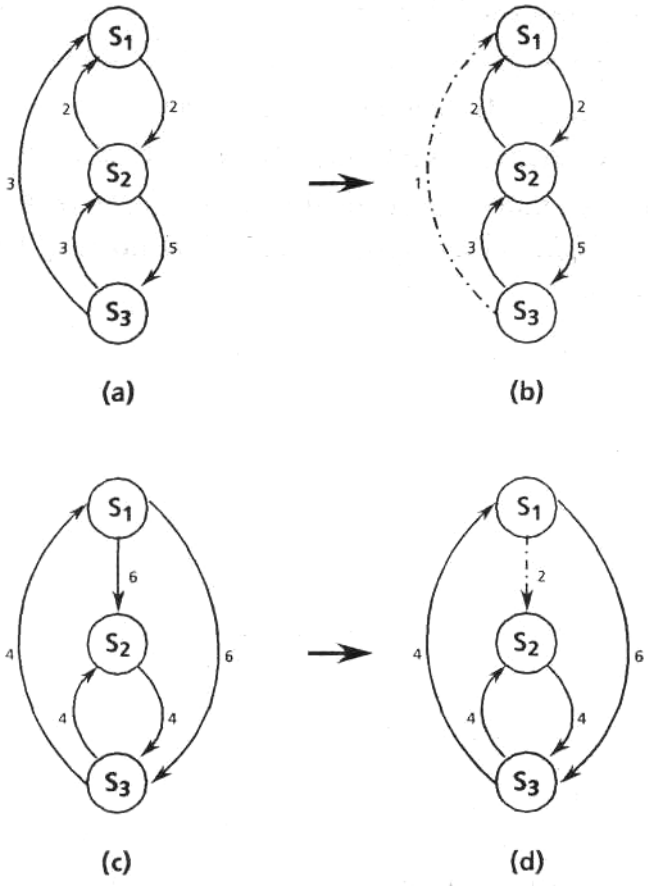


Figura 5.7: Modificaciones en el grafo a fin de asegurar las condiciones de vectorización.

Condición (b): arcos no incluidos en R_{sch}

Si se considera el grafo de la figura 5.7(c) puede observarse que las longitudes vectoriales de las sentencias son $lv(S_1) = 10$, y $lv(S_2) = lv(S_3) = 8$. A parte de no cumplir con la condición adicional que permite generar código de forma fácil, tema que se considera en el siguiente apartado, el arco d_{13} no cumple la condición (b), ya que su distancia es más restrictiva que el camino alternativo C_{13} a través de R_{sch}. A fin de asegurar su cumplimiento sin reducir las longitudes vectoriales originales de las sentencias se puede reducir la distancia asociada a d_{12} a un valor de 2, tal como muestra la figura 5.7(d).

5.4.3 Facilitar la generación de código vectorial

La última condición de la sección 5.2 se impone a fin de facilitar la generación de código vectorial descrita en la sección 5.3

Existen dos posibilidades para generar código vectorial que no requiere prólogos y epílogos en los bucles internos: (a) ejecución con longitudes vectoriales múltiplo de la mínima de todas ellas y (b) ejecución con la mínima longitud vectorial.

Longitudes vectoriales múltiplo

Por ejemplo, para el grafo de la figura 5.8(a) se tiene que las sentencias del bucle poseen asociadas $lv(S_1)=12$ y $lv(S_2) = lv(S_3) = 8$, que no cumplen dicha condición. Una primera solución consiste en reducir la longitud vectorial de S_1 a 8 a base de reducir, por ejemplo, las distancias asociadas a los arcos d_{12} y d_{31} a 2, tal y como muestra la figura 5.8(b). Otra posibilidad mostrada en la figura 5.8(c) es reducir las longitudes vectoriales asociadas a S_2 y S_3 a 6 a base de reducir, por ejemplo, la distancia asociada a d_{32} a 2. De todas las posibles soluciones, el compilador debe de ser capaz de evaluar su eficiencia y elegir la óptima.

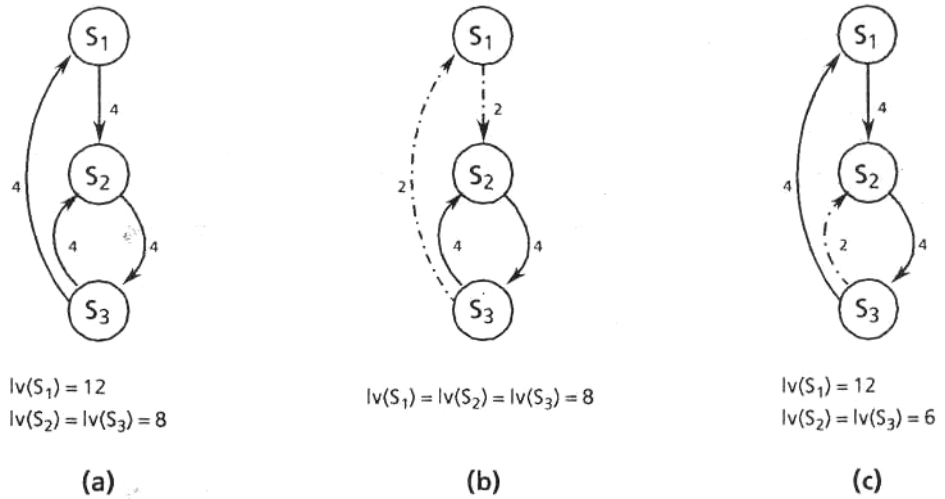


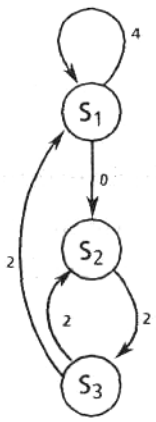
Figura 5.8: Modificaciones en el grafo a fin de facilitar la generación de código vectorial.

Longitud vectorial mínima

Otra posibilidad consiste en, aún cumpliéndose la condición (c), en reducir la longitud vectorial de todas las sentencias a

$$lv = \min_{S_i \in V} lv(S_i).$$

Para ello, deben de modificarse las distancias asociadas a las dependencias del grafo de manera que todas las recurrencias posean, el mismo peso. La figura 5.9 muestra el grafo de dependencias-modificado y «nuevo código vectorial generado para el bucle de la figura 5.2(a).



(a)

```
DO i = 1, 2
    C[i + 3] = B[i] / 3
ENDO
DO i = 1, 96, 4
    A[i + 3:i + 6] = A[i-5:i-2] + C[i:i + 3]
    B[i + 2:i + 5] = A[i:i + 3] - C[i + 1:i + 4] * 2
    C[i + 5:i + 8] = B[i + 2:i + 5] / 3
ENDO
DO i = 97, 100
    A[i + 3] = A[i-5] + C[i]
    B[i + 2] = A[i] - C[i + 1] * 2
    IF i ≥ 99
        C[i + 3] = B[i] / 3
    ENDIF
ENDO
```

(b)

Figura 5.9: Reducción de la longitud vectorial de las sentencias a fin de simplificar la generación de código vectorial.