

El algoritmo de Allen-Kennedy

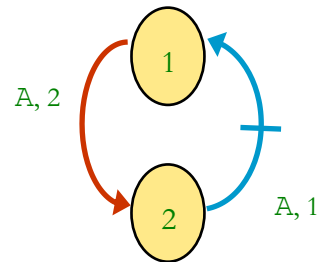
Como paralelizar (sistemáticamente) bucles

Referencia: A. Darte, Y. Robert, and F. Vivien, *Scheduling and Automatic Parallelization*, Chapter 5.

Vector distancia

En una relación de dependencia, definimos la **distancia** como aquel vector que contiene la diferencia entre origen y destino.

Este lo solemos indicar en la dependencia.

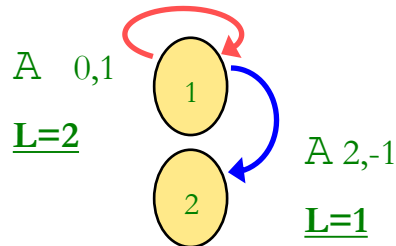


```
do i = 2, N-2
1  A(i) = B(i) + 2
2  C(i) = A(i-2) + A(i+1)
enddo
```

Profundidad o nivel

Para una relación de dependencia, se define **la profundidad o nivel** como el número del primer índice del vector distancia que es distinto de cero. Se denota por la letra L .

Nos indica sobre qué bucle (más externo o más interno) la dependencia actúa.



```
do i = 2, N-1
  do j = 1, N-2
    1 A(i,j) = A(i, j-1) * 2
    2 C(i,j) = A(i-2, j+1) + 1
  enddo
enddo
```

Allen-Kennedy para bucles simples

El algoritmo de Allen-Kennedy utiliza como entradas:

- El grafo de dependencias.
- El nivel de cada dependencia.

Pero para casos sin bucles anidados, es muy simple y solo requiere las distancias.

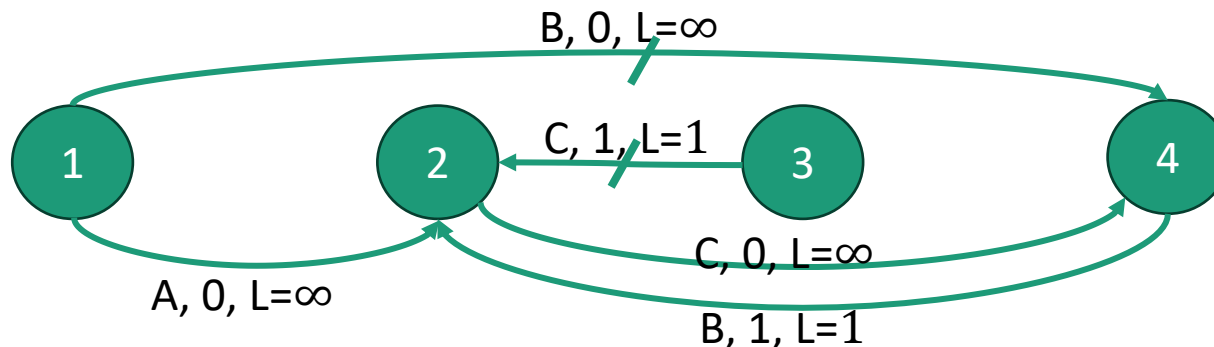
Allen-Kennedy para bucles simples

2.2.- El siguiente bucle se va a ejecutar en una máquina multithread:

```
do  $i = 1, N-2$   
  (1)  $A(i) = B(i)$   
  (2)  $C(i) = A(i) + B(i-1)$   
  (3)  $D(i) = C(i+1)$   
  (4)  $B(i) = C(i) * 2$   
enddo
```

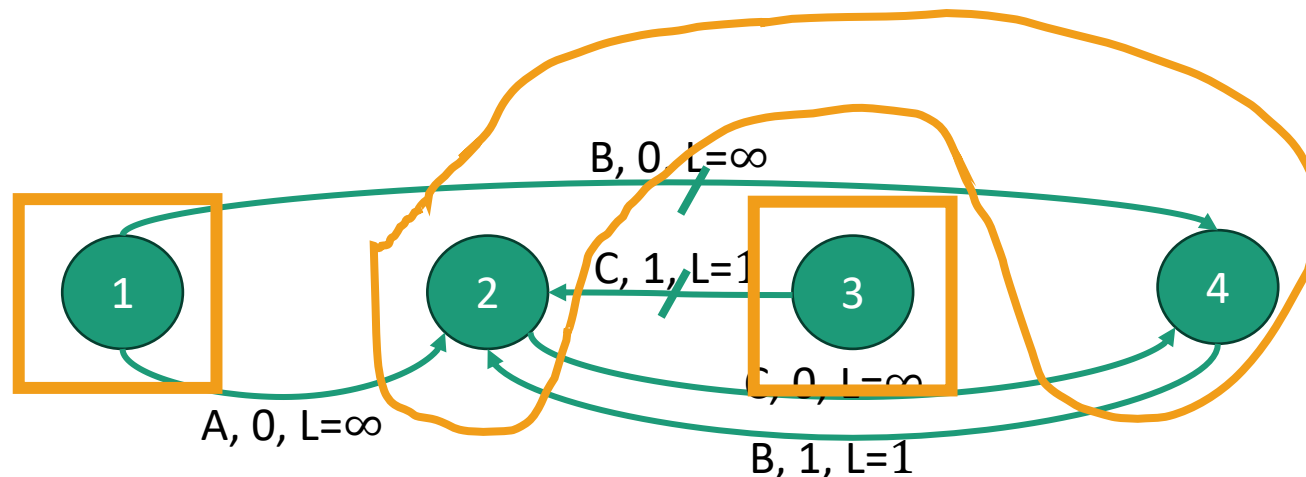
Genere el grafo de dependencias

Escribe una versión paralela de dicho bucle y suponiendo que el tiempo de ejecución de una instrucción es T , ¿Cuántas veces más rápido es la ejecución al paralelizar en P procesadores? ¿Cuál será la máxima aceleración con infinitos procesadores y despreciando el tiempo de sincronización.



Allen-Kennedy para bucles simples

1. Construimos el grafo de componentes fuertemente conexas
 - Dos elementos A y B pertenecen a la misma componente o cluster si desde A puedo ir a B y volver a A.
 - De 1 y 3, no hay flechas de entrada: Cluster 1 y Cluster 3
 - De 2 puedo ir a 4 y Volver: Cluster 2-4



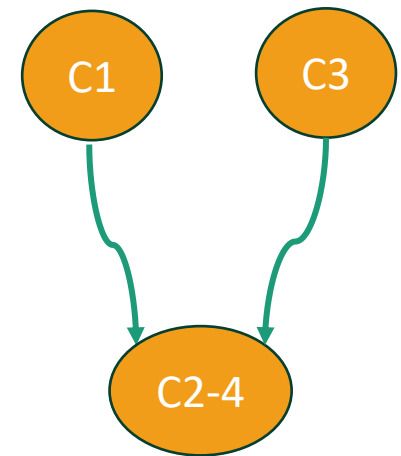
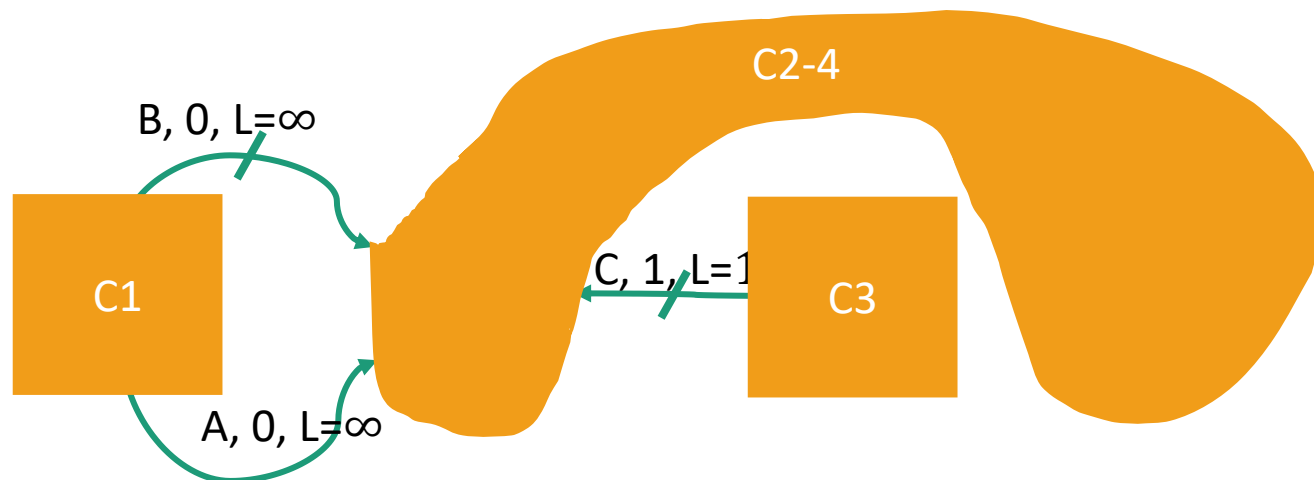
Allen-Kennedy para bucles simples

2. Ordenamos las components para su ejecución (ahora es un DAG) basándonos en sus dependencias. En el ejemplo:

C1;

C3;

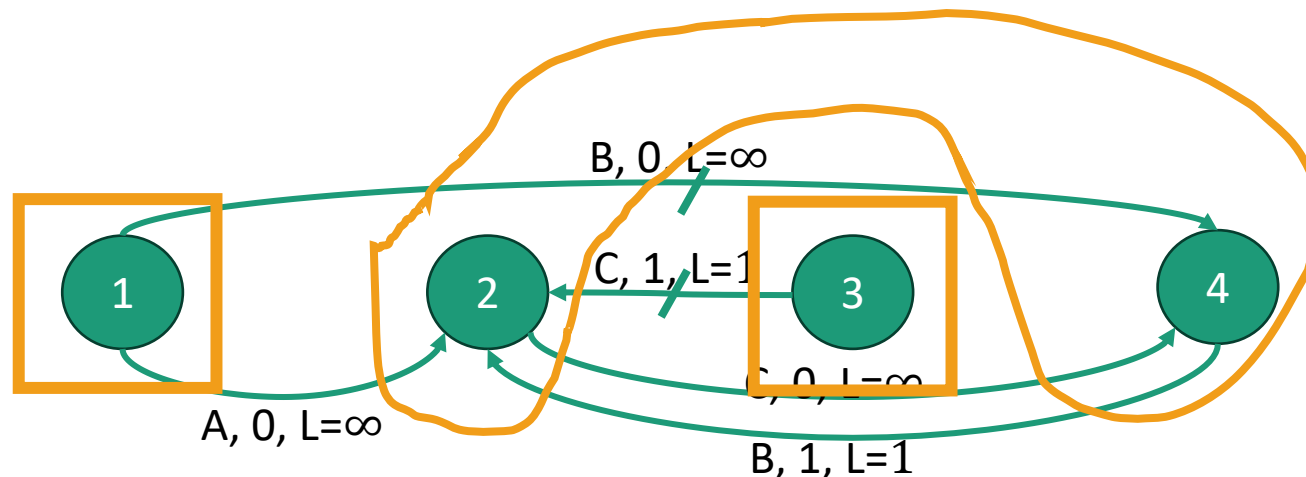
C2-4 :



Allen-Kennedy para bucles simples

3. Para cada Cluster, determinamos si se ejecuta en paralelo o en serie.

- Si es paralelo, todas las relaciones intra-cluster son de distancia cero.
- Si es serie, existe una relación intra-cluster de distancia mayor que cero.

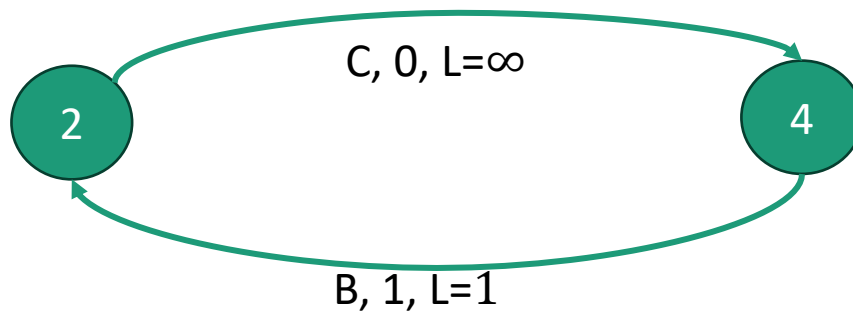


Cluster 1 y Cluster 3 no tienen dependencias intra-cluster => son paralelos

Allen-Kennedy para bucles simples

3. Para cada Cluster, determinamos si se ejecuta en paralelo o en serie.

- Si es paralelo, todas las relaciones intra-cluster son de distancia cero.
- Si es serie, existe una relación intra-cluster de distancia mayor que cero.



Cluster 2-4 tiene una relación de distancia 0 (no impide paralelismo), pero una de distancia 1 (impide paralelismo) => es serie

Allen-Kennedy para bucles simples

4. Generamos el Código. Es necesario usar barriers después de cada for paralelo.

```
doall i=1,N-2
```

```
    S1
```



C1

```
barrier
```

```
doall i=1,N-2
```

```
    S3
```

C3

```
barrier
```

```
for i=1,N-2
```

```
    S2
```

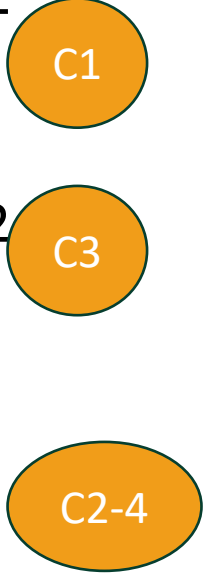
```
    S4
```

C2-4

Allen-Kennedy para bucles simples

Al final, revisamos barreras innecesarias

```
doall i=1,N-2
    S1
barrier
doall i=1,N-2
    S3
barrier
for i=1,N-2
    S2
    S4
```



```
doall i=1,N-2
    S1
    S3
barrier
for i=1,N-2
    S2
    S4
```

Si C1 no depende de C3, la barrera **no es necesaria** (**C1 y C3 se pueden ejecutar concurrentemente**).

Si todas las dependencias entre C1 y C3 son de distancia 0, hay dos opciones:

- Mantener la barrera.
- Fusionar los bucles.

Si hay una **dependencia** entre C1 y C3 y **distancia > 0**, **mantenemos la barrera (no se pueden fusionar los bucles)**.

Allen-Kennedy para bucles anidados

2.4. Represente el grafo de dependencias de los siguientes bucles:

a)

do $i = 1, N-2$

(1) $A(i) = B(i) + 2$

(2) $C(i) = A(i-2) + A(i+1)$

enddo

b)

do $i = 1, N-2$

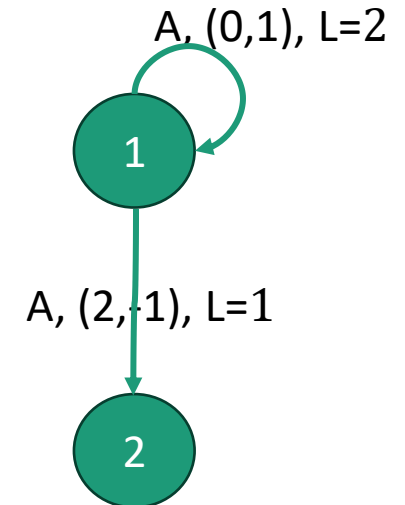
do $j = 1, N-2$

(1) $A(i,j) = A(i,j-1) * 2$

(2) $C(i,j) = A(i-2,j+1) + 1$

Enddo

enddo

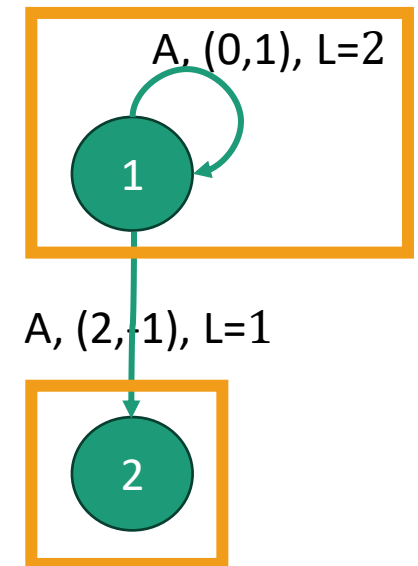


Allen-Kennedy para bucles anidados

1. Construimos el grafo de componentes fuertemente conexas (**a nivel 1**).

Dos elementos A y B pertenecen a la misma componente conexas si desde A puedo ir a B y volver a A.

Da igual el nivel de las dependencias en este paso.

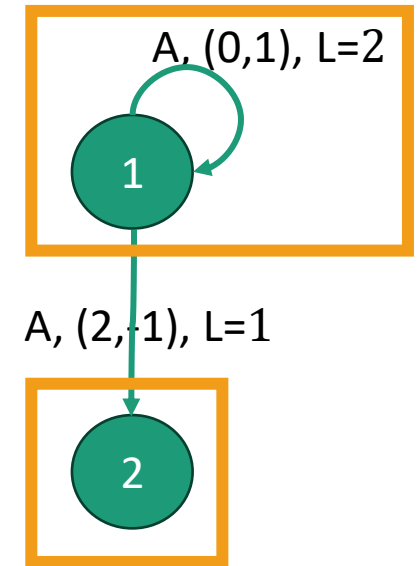


Allen-Kennedy para bucles anidados

2. Ordenamos las componentes para su ejecución (ahora es un DAG) basándonos en sus dependencias. En el ejemplo:

C1;

C2;

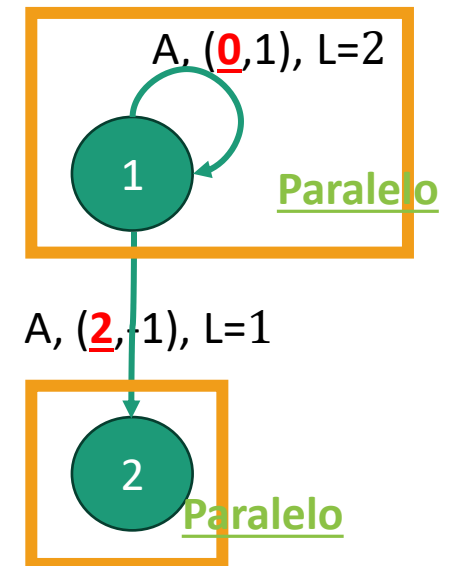


Allen-Kennedy para bucles anidados

3. Para cada Cluster, determinamos si se ejecuta en paralelo o en serie.

- Si es paralelo, todas las relaciones intra-cluster son de distancia cero en el nivel que estamos analizando.
- Si es serie, existe una relación intra-cluster de distancia mayor que cero en el nivel que estamos analizando.

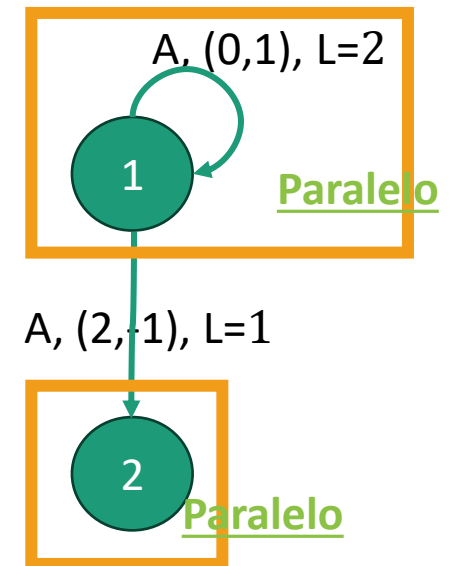
Bonus: aunque ambos clusters son paralelos, no se pueden ejecutar en el mismo doall (requiere dos doall separados mediante barrier) por la dependencia entre 1 y 2 de distancia (2,-1).



Allen-Kennedy para bucles anidados

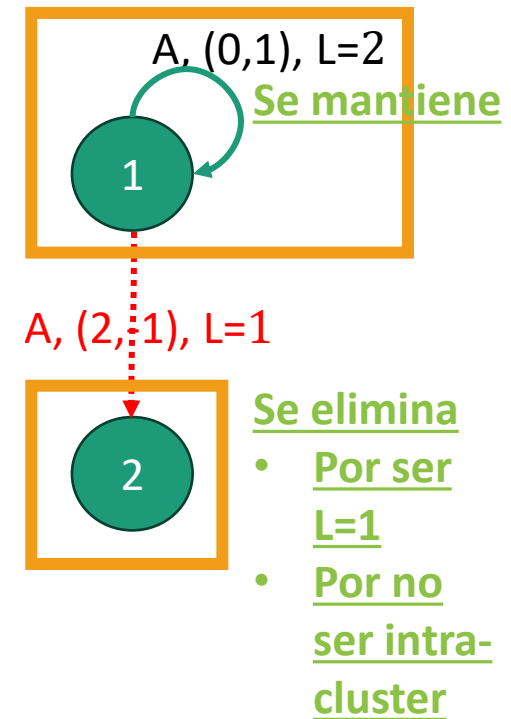
4. Generamos el código en el nivel estudiado. Es necesario usar barriers después de cada for paralelo.

```
doall i=1,N-2 # nivel L=1
    C1
barrier
doall i=1,N-2 # nivel L=1
    C2
```



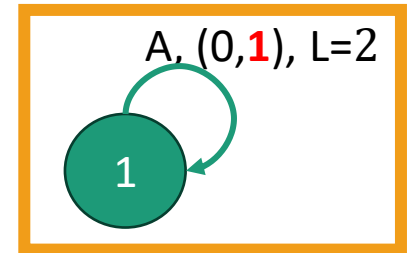
Allen-Kennedy para bucles anidados

5. Eliminamos las dependencias de nivel igual o inferior al analizado y aplicamos recursivamente sobre cada cluster al siguiente nivel (solo consideramos dependencias intra-cluster).



Allen-Kennedy para bucles anidados

- Para C1
 1. Grafo de componentes fuertemente conexas.
 2. Ordenamos DAG resultante
 3. Analizamos cada componente:
 - En este caso, a nivel $L=2$, hay una dependencia de distancia mayor que cero => bucle serie
 4. Código



for $j=1, N-2$
 S1

Allen-Kennedy para bucles anidados

- Para C2
 1. Grafo de componentes fuertemente conexas.
 2. Ordenamos DAG resultante
 3. Analizamos cada componente:
 - No hay dependencias \Rightarrow doall
 4. Código

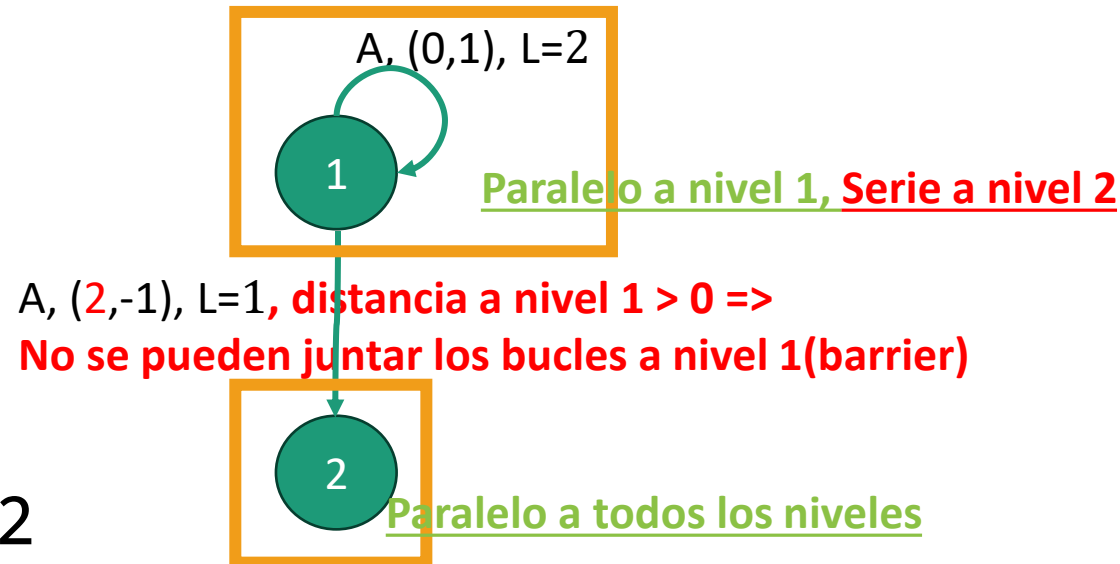


doall $j=1, N-2$
S2

Allen-Kennedy para bucles anidados

- Resultado final

```
doall i=1,N-2
  for j=1,N-2
    S1
barrier
doall i=1,N-2
  doall j=1,N-2
    S2
```



Resultados de Allen-Kennedy

El algoritmo de Allen-Kennedy es óptimo bajo las hipótesis habituales.

Pero:

- No tiene en cuenta posibilidad de reducir dependencias de distancia > 0 a dependencias de distancia 0. Habría que estudiar el posible *peeling* (u otras técnicas) para eliminar barriers.
 - Puede generar barriers eliminables.
- Tiende a producir paralelizaciones de grano fino (fisionar bucles).
- No distingue entre dependencias verdaderas o antidependencias/de salida. Los renombrados pueden reducir el tiempo de ejecución.

¿Quieres saber más?

En el capítulo 5 *Parallelism Detection in Nested Loops* de A. Darte, Y. Robert, and F. Vivien, *Scheduling and Automatic Parallelization* tenéis:

- Aproximaciones más generales al problema.
- Otros algoritmos (Wolf-Lam, Darte-Vivien).
- Definición y resultados acerca de la optimalidad de los algoritmos.