On the Optimality of Feautrier's Scheduling Algorithm

Frédéric Vivien

LIP, École normale supérieure de Lyon – INRIA

previously: ICPS/LSIIT, Université Louis Pasteur, Strasbourg

Frederic.Vivien@ens-lyon.fr

The subject

Feautrier's scheduling algorithm:

an algorithm to detect and extract parallelism.

Feautrier's algorithm is

1. The most powerful algorithm to extract parallelism.
2. A powerful tool to build optimization algorithms.

Conclusion

One can rely on Feautrier's algorithm:

1. To detect and extract parallelism.

2. To build other algorithms.

Talk overview

1. Importance of Feautrier's algorithm

2. Running Feautrier's algorithm on an example

   (a) Monodimensional schedules
   (b) Multidimensional schedules
   (c) The design "flaw"
   (d) The efficiency result

3. Conclusion

Parallelization overview

Original code

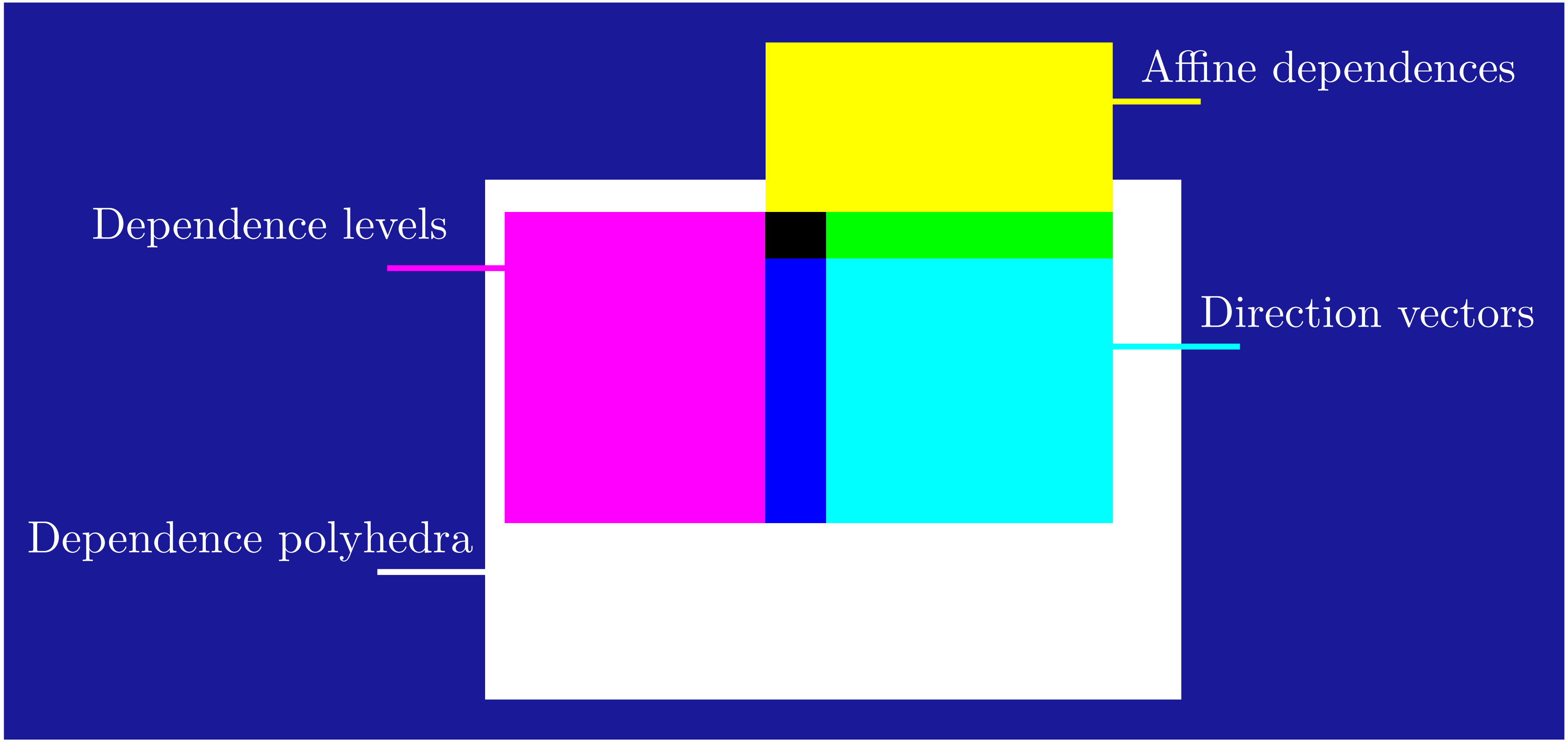
$\Downarrow$ Dependence analysis

Dependence representation

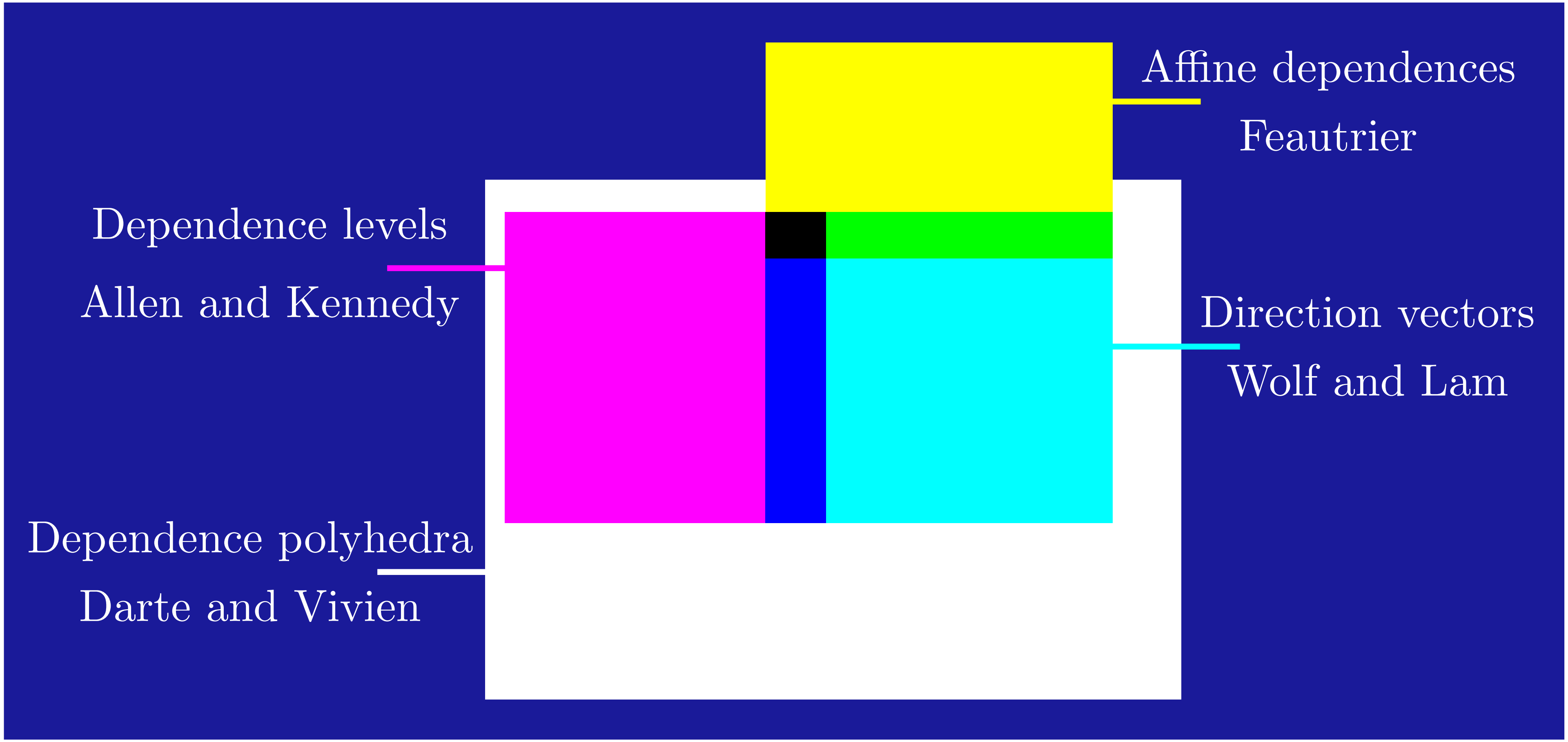
$\Downarrow$ Parallelism detection

Code transformations

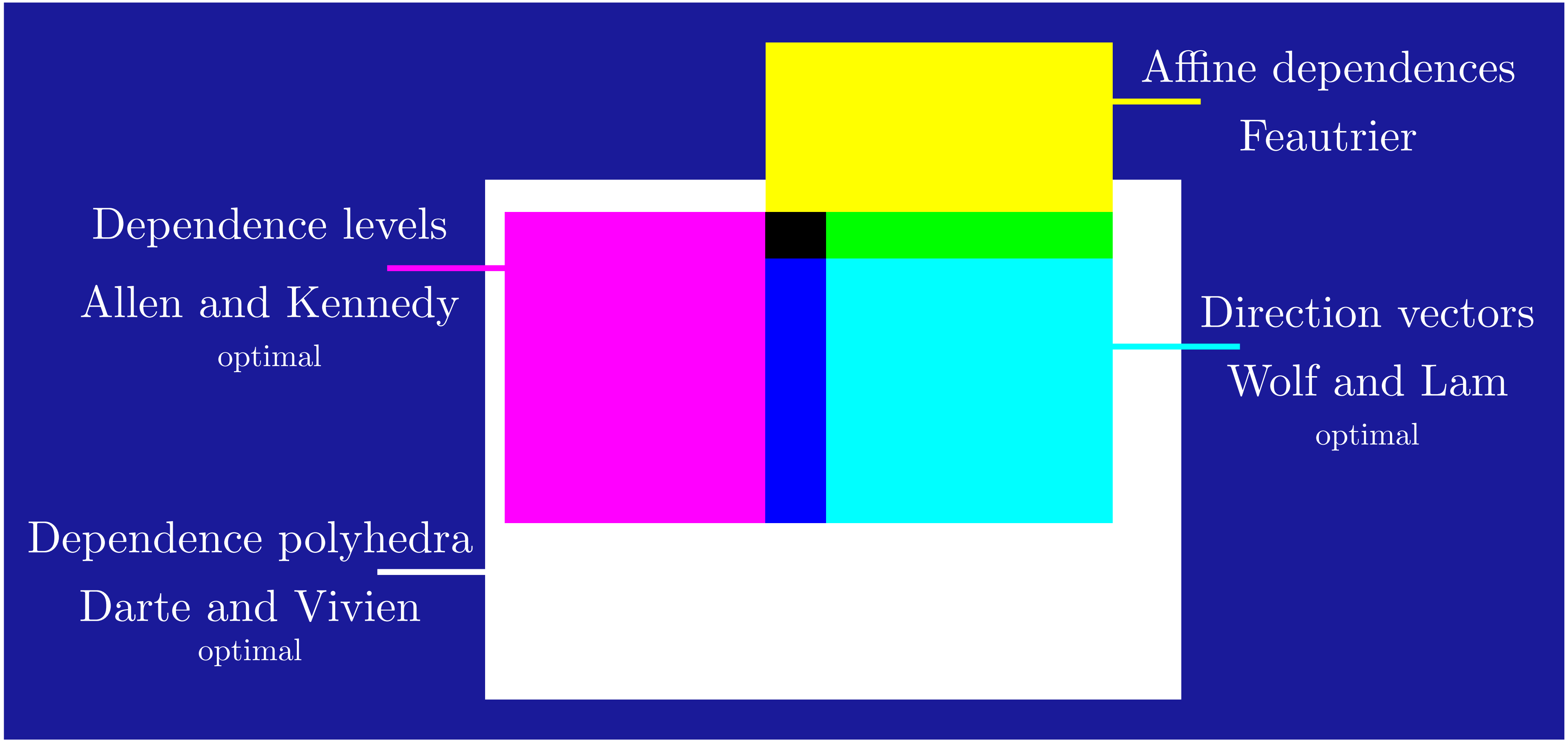
Dependence abstractions and parallelism detection



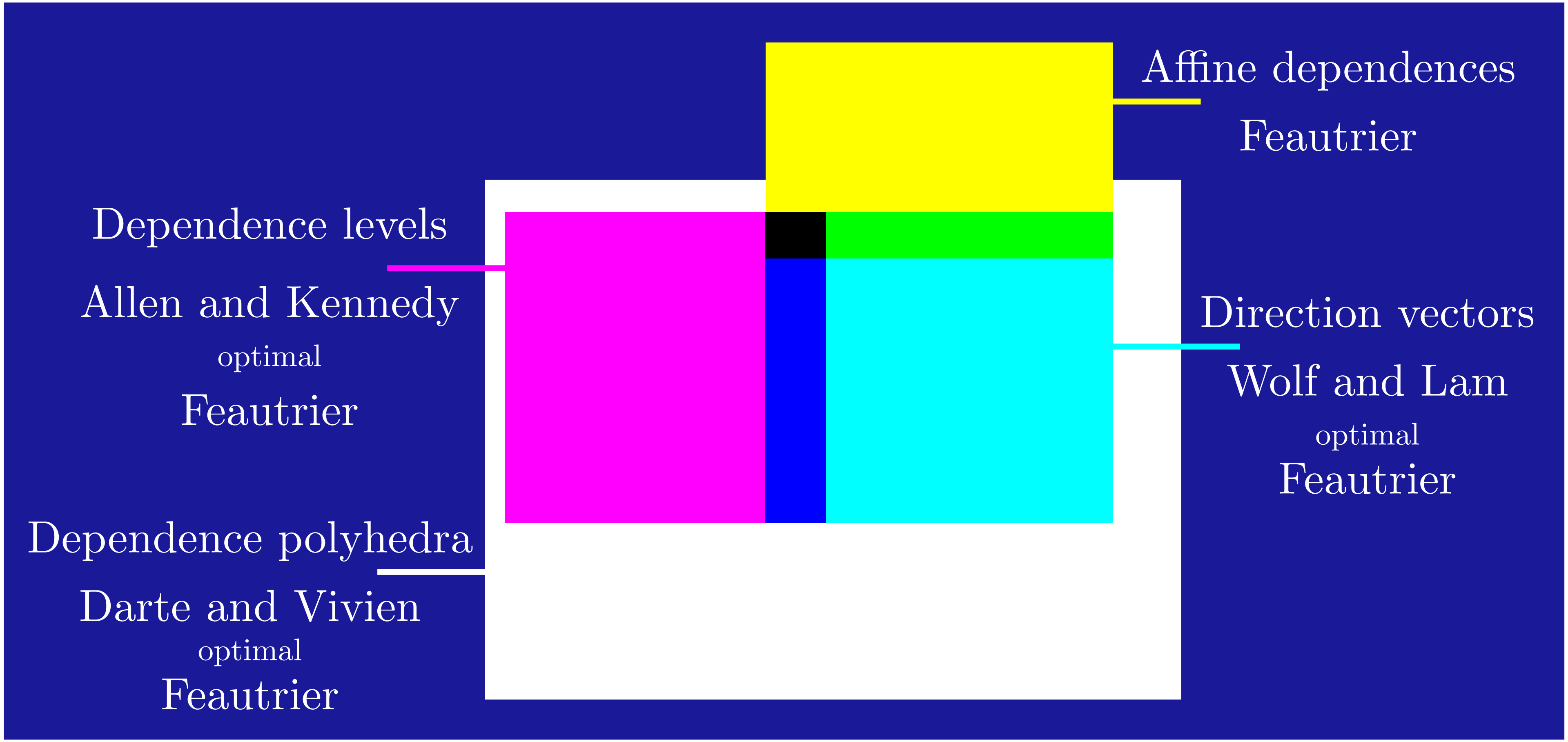
Dependence abstractions and parallelism detection



Dependence abstractions and parallelism detection



Dependence abstractions and parallelism detection



Feautrier's algorithm: the most powerful for parallelism detection.

The most powerful but...

Not optimal:

- It sometimes miss significant amount of parallelism

Questions:

- Why?
- May it be improved?
- How may it be improved?
- How can we build a better algorithm?

A powerful tool to build upon

Reason:

- Explicit all valid schedules (under some assumptions).

Applications:

- Scheduling with respect of a given computation mapping (Darte, Diderich, Gengler, and Vivien, Euro-Par'2000).
- Affine occupancy vectors (Thies, Vivien, Sheldon, and Amarasinghe, PLDI 2002).
- Schedules which reduce utility span of values (Clauss and Vivien, work in progress).

Motivating example

```
DO i=1, N
    DO j=1,i
        S : a(i, i+j+1) = a(i-1, 2*i-1) + a(j, 2*j)
    ENDDO
ENDDO
```

First dependence relation

```
DO i=1, N
    DO j=1,i
        S : a(i, i+j+1) = a(i-1, 2*i-1) + a(j, 2*j)
    ENDDO
ENDDO
```

$S(i,j)$ **reads** into memory location $a(i-1, 2*i-1)$
$S(i',j')$ **writes** into memory location $a(i', i'+j'+1)$

$S(i,j)$ depends on $S(i-1, j-1)$ if $2 \leq i \leq N, 1 \leq j \leq i$.

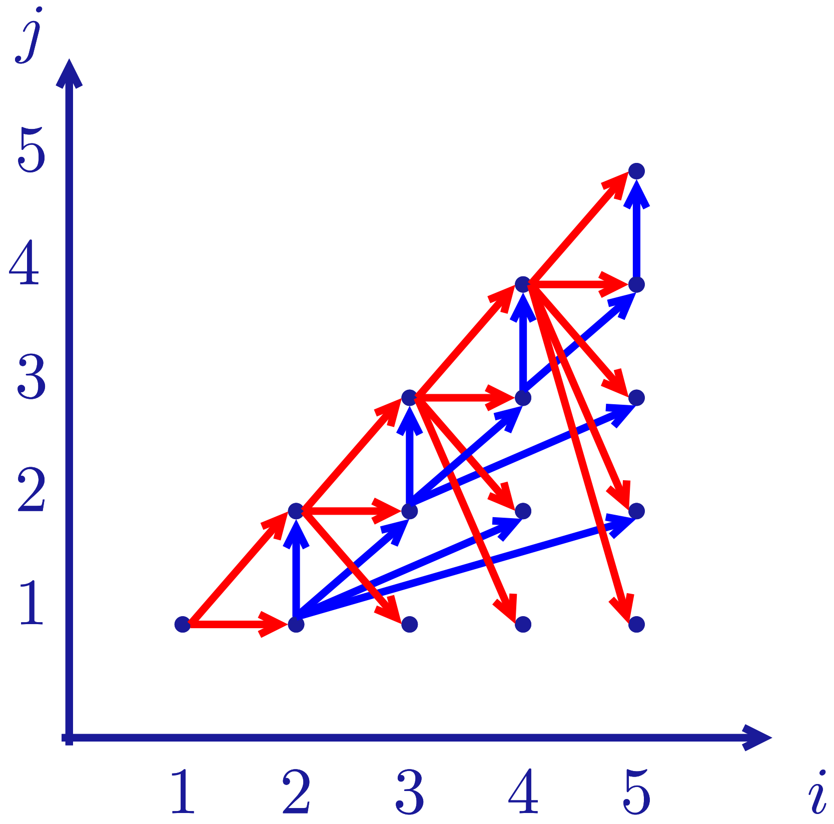
Second dependence relation

```
DO i=1, N
    DO j=1,i
        S : a(i, i+j+1) = a(i-1, 2*i-1) + a(j, 2*j)
    ENDDO
ENDDO
```

$S(i,j)$	**reads**	into memory location	$a(j, 2*j)$
$S(i',j')$	**writes**	into memory location	$a(i', i'+j'+1)$

$S(i,j)$ depends on $S(j, j-1)$ if $1 \leq i \leq N, 2 \leq j \leq i$.

All the dependences for $N = 5$



Extraction of parallelism

Parallelism extracted using **affine** schedules

All operations scheduled at the same date are executed in parallel.

Operation $S(i, j)$ is executed at date:

$$\Theta_S(i, j) = \begin{vmatrix} x_S \\ y_S \end{vmatrix} . \begin{vmatrix} i \\ j \end{vmatrix} + Y_S.N + \rho_S$$

Satisfying the dependences (theory)

If $T(i,j)$ depends on $S(i',j')$ then

$T(i,j)$ must be executed after $S(i',j')$ then

$$\Theta_T(i,j) \geq 1 + \Theta_S(i',j').$$

Satisfying the dependences (practice)

1. $S(i,j)$ depends on $S(i-1,i-1)$ with $2 \leq i \leq N, 1 \leq j \leq i$

   $$\left|\begin{array}{c} x \\ y \end{array}\right. . \left|\begin{array}{l} i \\ j \end{array}\right. \geq 1 + \left|\begin{array}{c} x \\ y \end{array}\right. . \left|\begin{array}{l} i-1 \\ i-1 \end{array}\right. \quad \Rightarrow \quad x + y(j-i+1) \geq 1$$

   especially: $x + y(2-N) \geq 1$ which implies $y \leq 0$.

2. $S(i,j)$ depends on $S(j,j-1)$ with $2 \leq i \leq N, 2 \leq j \leq i$

   $$\left|\begin{array}{c} x \\ y \end{array}\right| \begin{array}{l} i \\ j \end{array} \geq 1 + \left|\begin{array}{c} x \\ y \end{array}\right. . \left|\begin{array}{l} j \\ j-1 \end{array}\right. \quad \Rightarrow \quad x(i-j) + y \geq 1$$

   especially: $y \geq 1$

Satisfying the dependences (practice)

1. $S(i,j)$ depends on $S(i-1,i-1)$ with $2 \leq i \leq N, 1 \leq j \leq i$

   $$\left| \begin{array}{c} x \\ y \end{array} \right. . \left| \begin{array}{l} i \\ j \end{array} \right. \geq 1 + \left| \begin{array}{c} x \\ y \end{array} \right. . \left| \begin{array}{l} i-1 \\ i-1 \end{array} \right. \quad \Rightarrow \quad x + y(j-i+1) \geq 1$$

   especially: $x + y(2-N) \geq 1$ which implies $y \leq 0$.

2. $S(i,j)$ depends on $S(j,j-1)$ with $2 \leq i \leq N, 2 \leq j \leq i$

   $$\left| \begin{array}{c} x \\ y \end{array} \right| \begin{array}{l} i \\ j \end{array} \geq 1 + \left| \begin{array}{c} x \\ y \end{array} \right. . \left| \begin{array}{l} j \\ j-1 \end{array} \right. \quad \Rightarrow \quad x(i-j) + y \geq 1$$

   especially: $y \geq 1$

There is no solution!

Multidimensional schedules

The dates are no more **integers** but **vectors of integers**

A possible "view"

first	dimension =	hours
second	dimension =	minutes
third	dimension =	seconds

Dates are lexicographically ordered

The dependences must be respected

Satisfying the dependences

If $T(i,j)$ depends on $S(i',j')$

then $T(i,j)$ must be executed after $S(i',j')$

and in any case: $\Theta^1_T(i,j) \geq \Theta^1_S(i',j')$

If $\Theta^1_T(i,j) \geq 1 + \Theta^1_S(i',j')$ the dependence is satisfied by the first dimension of the schedule

If $\Theta^1_T(i,j) = \Theta^1_S(i',j')$ the dependence **must be** satisfied by the remaining dimensions of the schedule

Satisfying the dependences

If $T(i,j)$ depends on $S(i',j')$

then $T(i,j)$ must be executed after $S(i',j')$

and in any case: $\Theta_T^1(i,j) \geq \Theta_S^1(i',j')$

If $\Theta_T^1(i,j) \geq 1 + \Theta_S^1(i',j')$ the dependence is satisfied by the first dimension of the schedule

If $\Theta_T^1(i,j) = \Theta_S^1(i',j')$ the dependence **must be** satisfied by the remaining dimensions of the schedule

What dependences are we going to satisfy first?

Feautrier's approach

Greedy heuristic:

- The schedule first dimension satisfies as many dependences as possible (and so on for the remaining dimensions)

- If Θ^1 satisfies the dependence e
  If Θ'^1 satisfies the dependence f
  Then $(\Theta + \Theta')^1$ satisfies both e and f.

The algorithm (theory)

The dependences:

- e_1, ..., e_d are the dependence relations.
- e_k : $T_k(i, j)$ depends on $S_k(h_k(i, j))$ for $(i, j) \in \mathcal{D}_k$.

The algorithm (theory)

The dependences:

- e_1, ..., e_d are the dependence relations.
- e_k : $T_k(i,j)$ depends on $S_k(h_k(i,j))$ for $(i,j) \in \mathcal{D}_k$.

Constraints on the schedules:

- $\forall k \in [1,d], \forall (i,j) \in \mathcal{D}_k, \quad \Theta^1_{T_k}(i,j) \geq \Theta^1_{S_k}(h_k(i,j)) + z_k$
- $\forall k \in [1,d], \quad 0 \leq z_k$

The algorithm (theory)

The dependences:

- e_1, ..., e_d are the dependence relations.
- e_k : $T_k(i,j)$ depends on $S_k(h_k(i,j))$ for $(i,j) \in \mathcal{D}_k$.

Constraints on the schedules:

- $\forall k \in [1,d], \forall (i,j) \in \mathcal{D}_k, \quad \Theta^1_{T_k}(i,j) \geq \Theta^1_{S_k}(h_k(i,j)) + z_k$
- $\forall k \in [1,d], \quad 0 \leq z_k \leq 1$

The algorithm (theory)

The dependences:

- e_1, ..., e_d are the dependence relations.
- e_k : $T_k(i,j)$ depends on $S_k(h_k(i,j))$ for $(i,j) \in \mathcal{D}_k$.

Constraints on the schedules:

- $\forall k \in [1,d], \forall (i,j) \in \mathcal{D}_k, \quad \Theta^1_{T_k}(i,j) \geq \Theta^1_{S_k}(h_k(i,j)) + z_k$
- $\forall k \in [1,d], \quad 0 \leq z_k \leq 1$
- $maximize\ \sum_{k=1}^{d} z_k$

The algorithm (theory)

The dependences:

- e_1, ..., e_d are the dependence relations.
- $e_k : T_k(i,j)$ depends on $S_k(h_k(i,j))$ for $(i,j) \in \mathcal{D}_k$.

Constraints on the schedules:

- $\forall k \in [1,d], \forall (i,j) \in \mathcal{D}_k, \quad \Theta^1_{T_k}(i,j) \geq \Theta^1_{S_k}(h_k(i,j)) + z_k$
- $\forall k \in [1,d], \quad 0 \leq z_k \leq 1$
- $maximize\ \sum_{k=1}^{d} z_k$
- Recursive call on unsatisfied dependence relations

The algorithm (practice)

1. $S(i,j)$ depends on $S(i-1,i-1)$ with $2 \le i \le N, 1 \le j \le i$

$$\left| \begin{array}{c} x \\ y \end{array} \right. \cdot \left| \begin{array}{c} i \\ j \end{array} \right. \ge \left| \begin{array}{c} x \\ y \end{array} \right. \cdot \left| \begin{array}{c} i-1 \\ i-1 \end{array} \right. + z_1,\ 0 \le z_1 \le 1 \quad \Rightarrow \quad x + y(j-i+1) \ge z_1 \ge 0$$

especially: $x + y(2-N) \ge z_1 \ge 0$ which implies $y \le 0$.

2. $S(i,j)$ depends on $S(j,j-1)$ with $2 \le i \le N, 2 \le j \le i$

$$\left| \begin{array}{c} x \\ y \end{array} \right| \begin{array}{c} i \\ j \end{array} \ge \left| \begin{array}{c} x \\ y \end{array} \right. \cdot \left| \begin{array}{c} j \\ j-1 \end{array} \right. + z_2,\ 0 \le z_2 \le 1 \quad \Rightarrow \quad x(i-j) + y \ge z_2 \ge 0$$

especially: $y \ge z_2 \ge 0$ (when $i = j$) thus $y = 0$, thus $z_2 = 0$

Solution: $x = 1$, $y = 0$, $z_1 = 1$, $z_2 = 0$.

What does that mean?

Solution: $x = 1$, $y = 0$, $z_1 = 1$, $z_2 = 0$.

What does that mean?

Solution: $x = 1$, $y = 0$, $z_1 = 1$, $z_2 = 0$.

- $z_1 = 1$: the first dependence relation is satisfied by the first dimension of the schedule.

  No need to be considered by the remaining dimension of the schedule.

What does that mean?

Solution: $x = 1$, $y = 0$, $z_1 = 1$, $z_2 = 0$.

- $z_1 = 1$: the first dependence relation is satisfied by the first dimension of the schedule.

  No need to be considered by the remaining dimension of the schedule.

- $z_2 = 0$: the second dependence relation is **not** satisfied by the first dimension of the schedule.

  It **must be satisfied** by the second dimension of the schedule.

The algorithm is called recursively on the second dependence relation.

Looking closer at the second dependence relation

$S(i, j)$ depends on $S(j, j-1)$ with $2 \leq i \leq N, 2 \leq j \leq i$

$\Theta^1(i, j) - \Theta^1(j, j-1)$

Looking closer at the second dependence relation

$S(i,j)$ depends on $S(j,j-1)$ with $2 \leq i \leq N, 2 \leq j \leq i$

$$\begin{aligned}
\Theta^1(i,j) - \Theta^1(j,j-1) &= \begin{vmatrix} x \\ y \end{vmatrix} . \begin{vmatrix} i \\ j \end{vmatrix} - \begin{vmatrix} x \\ y \end{vmatrix} . \begin{vmatrix} j \\ j-1 \end{vmatrix} \\
&= \begin{vmatrix} 1 \\ 0 \end{vmatrix} . \begin{vmatrix} i \\ j \end{vmatrix} - \begin{vmatrix} 1 \\ 0 \end{vmatrix} . \begin{vmatrix} j \\ j-1 \end{vmatrix} \\
&= i - j
\end{aligned}$$

Looking closer at the second dependence relation

$S(i,j)$ depends on $S(j,j-1)$ with $2 \leq i \leq N, 2 \leq j \leq i$

$$
\begin{aligned}
\Theta^1(i,j) - \Theta^1(j,j-1) &= \left|\begin{matrix} x \\ y \end{matrix}\right. \cdot \left|\begin{matrix} i \\ j \end{matrix}\right. - \left|\begin{matrix} x \\ y \end{matrix}\right. \cdot \left|\begin{matrix} j \\ j-1 \end{matrix}\right. \\
&= \left|\begin{matrix} 1 \\ 0 \end{matrix}\right. \cdot \left|\begin{matrix} i \\ j \end{matrix}\right. - \left|\begin{matrix} 1 \\ 0 \end{matrix}\right. \cdot \left|\begin{matrix} j \\ j-1 \end{matrix}\right. \\
&= i - j \\
&= 0 \qquad \text{if } 2 \leq i \leq N, j = i
\end{aligned}
$$

Looking closer at the second dependence relation

$S(i,j)$ depends on $S(j,j-1)$ with $2 \leq i \leq N, 2 \leq j \leq i$

$$
\begin{aligned}
\Theta^1(i,j) - \Theta^1(j,j-1) &= \left| \begin{matrix} x \\ y \end{matrix} \right. . \left| \begin{matrix} i \\ j \end{matrix} \right. - \left| \begin{matrix} x \\ y \end{matrix} \right. . \left| \begin{matrix} j \\ j-1 \end{matrix} \right. \\
&= \left| \begin{matrix} 1 \\ 0 \end{matrix} \right. . \left| \begin{matrix} i \\ j \end{matrix} \right. - \left| \begin{matrix} 1 \\ 0 \end{matrix} \right. . \left| \begin{matrix} j \\ j-1 \end{matrix} \right. \\
&= i - j \\
&= 0 \qquad \text{if } 2 \leq i \leq N, j = i \\
&\geq 1 \qquad \text{if } 3 \leq i \leq N, 2 \leq j \leq i-1
\end{aligned}
$$

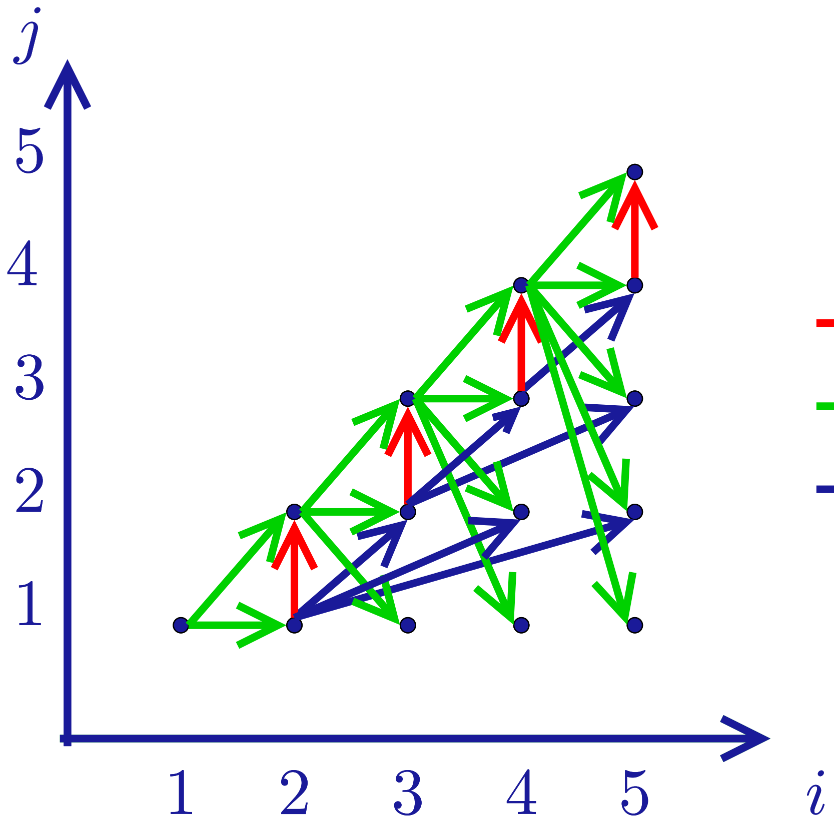
Looking closer at the second dependence relation

$S(i,j)$ depends on $S(j,j-1)$ with $2 \leq i \leq N, 2 \leq j \leq i$

$$\begin{aligned}
\Theta^1(i,j) - \Theta^1(j,j-1) &= \left| \begin{array}{l} x \\ y \end{array} \cdot \right| \begin{array}{l} i \\ j \end{array} - \left| \begin{array}{l} x \\ y \end{array} \cdot \right| \begin{array}{l} j \\ j-1 \end{array} \\
&= \left| \begin{array}{l} 1 \\ 0 \end{array} \cdot \right| \begin{array}{l} i \\ j \end{array} - \left| \begin{array}{l} 1 \\ 0 \end{array} \cdot \right| \begin{array}{l} j \\ j-1 \end{array} \\
&= i - j \\
&= 0 \qquad \text{if } 2 \leq i \leq N, j = i \\
&\geq 1 \qquad \text{if } 3 \leq i \leq N, 2 \leq j \leq i-1
\end{aligned}$$

The dependences are sometimes satisfied...

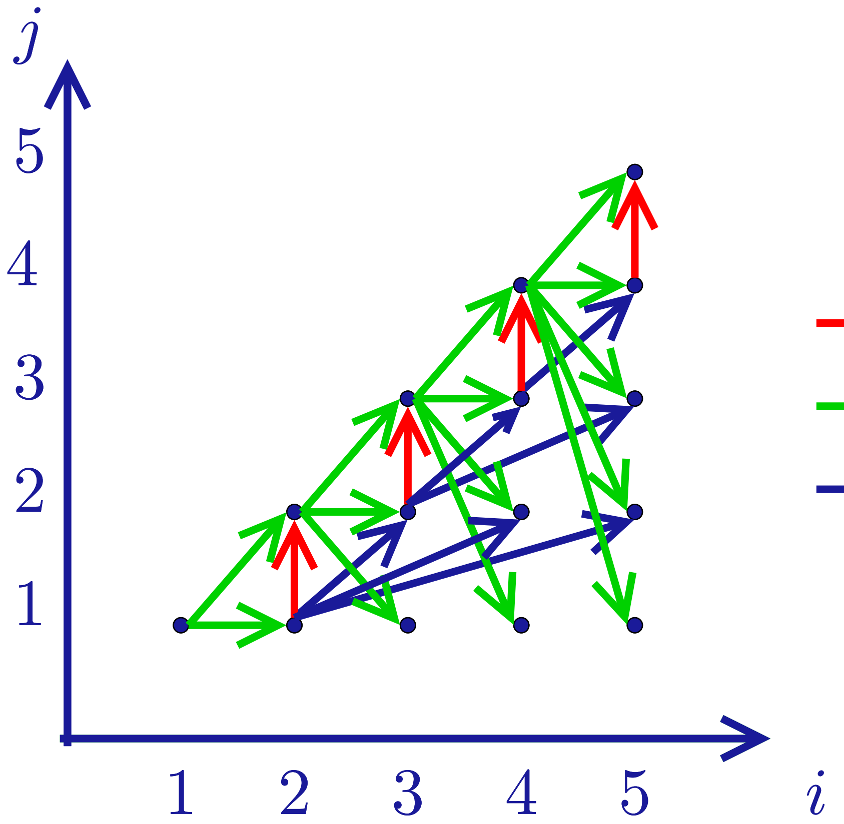
Dependences kept for the schedule second dimension



Dependences

- Unsatisfied, thus kept
- Satisfied and not kept
- Satisfied but kept

Dependences kept for the schedule second dimension



Dependences

Unsatisfied, thus kept

Satisfied and not kept

Satisfied but kept

Problem overconstrained

The question

The problem

Feautrier's algorithm is overconstraining its search of schedules.

The question

May it be the cause of a loss of parallelism?

The question

The problem

Feautrier's algorithm is overconstraining its search of schedules.

The question

May it be the cause of a loss of parallelism?

The answer

No: the dimension of the schedules built by Feautrier is minimal.

The exact result

Hypotheses:

1. All dependences are represented by affine functions.
2. We are looking for one affine schedule per statement of the loop nest.

Theorem:

The dimension of the schedules built by Feautrier is minimal for each statement of the loop nest.

$\Rightarrow$ no significant loss of parallelism due to the algorithm design

Going further

This result is valid

- for any set of unperfectly nested loops of any depths (in a static control flow program)
- for rational schedules

<u>One can build a greedier scheduling algorithm</u>

The schedule first dimension satisfies
as many *operation to operation dependences* as possible
(and not as many *dependence relations* as possible).

Conclusion

Feautrier's algorithm

- The most powerful algorithm to extract parallelism
- Do not miss any significant amount of parallelism *because* of its design

To improve it one must get rid of some of its hypotheses

- Affine schedules
- One scheduling function per statement

Feautrier's *greedy heuristic* is an efficient algorithm.