Fourier Motzkin

Announcements

- Reading for Monday: Legal reorderings paper, all but Sections 5, 6, and 7
- Tuesday April 18th, Assignment 3 due in D2L; demo, writeup, and slides

Today

- Fourier motzkin elimination

Fourier-Motzkin Elimination: The Algorithm

FM(P, i_k) => P'

Input:
$$P = \{(i_1, i_2, ..., i_d) \mid Q\vec{i} \geq (\vec{q} + B\vec{p})\}$$
$$i_k \text{ such that } 1 \leq k \leq d$$

Output:
$$P' = \{(i_1, ..., i_{k-1}, i_{k+1}, ..., i_d) \mid Q'\vec{i'} \geq (\vec{q'} + B'\vec{p})\}$$

Algorithm:

for each lower bound of $i_k, (L \leq c_1 i_k)$

$$P = P - \{L \leq c_1 i_k\}$$

for each upper bound of $i_k, (c_2 i_k \leq U)$

$$P = P - \{c_2 i_k \leq U\}$$

$$P' = P' \cup \{c_2 L \leq c_1 U\}$$

Distinguishing Upper and Lower Bounds

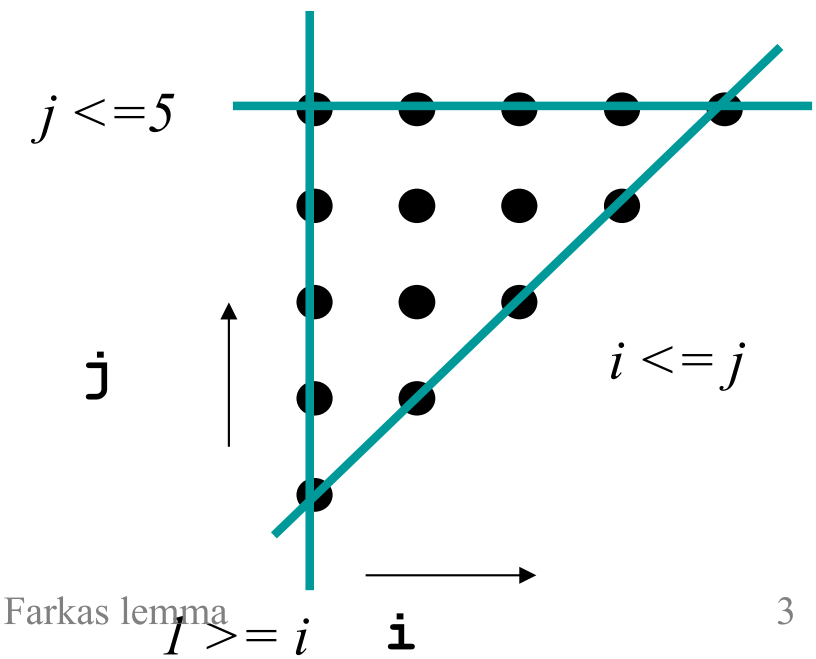
Simple Algorithm

– given that the polyhedron is represented as follows:

$$P = \{(i_1, i_2, ..., i_d) \mid Q\vec{i} \geq (\vec{q} + B\vec{p})\}$$

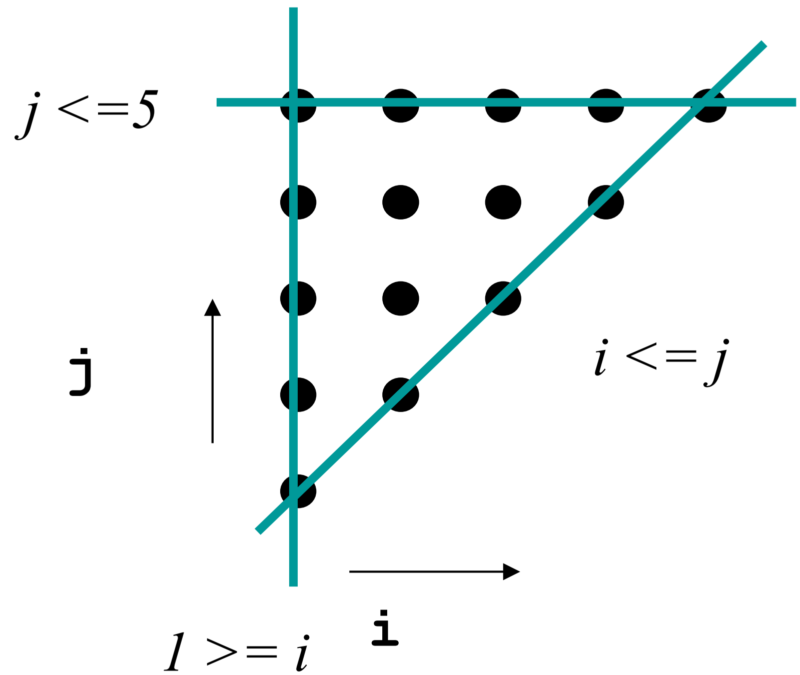
– any constraint with a positive coefficient for i_k is a lower bound

– any constraint with a negative coefficient for i_k is an upper bound



Triangular Iteration Space Example

(i, j) for target iteration space



(j, i) for target iteration space

General Algorithm for Generating Loop Bounds

Input: $P = \{(i_1, i_2, ..., i_d) \mid Q\vec{i} \geq (\vec{q} + B\vec{p})\}$

where the i vector is the desired loop order

Output: $L_{i_1}, L_{i_2}, ..., L_{i_d}$ such that $L_{i_k} = f(i_1, ...i_{k-1})$

$U_{i_1}, U_{i_2}, ..., L_{i_d}$ such that $U_{i_k} = g(i_1, ...i_{k-1})$

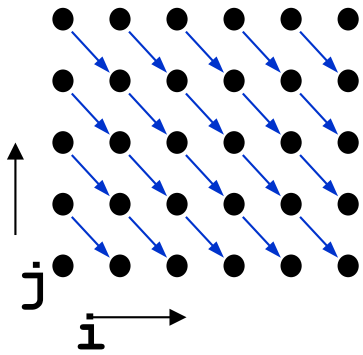
Algorithm:

$P_n = P$

for k = d to 1 by -1

$$
\begin{aligned}
L_{i_k} &= \text{all lower bounds for } i_k \text{ in } P_k \\
U_{i_k} &= \text{all upper bounds for } i_k \text{ in } P_k \\
P_{k-1} &= FM(P_k, i_k)
\end{aligned}
$$

Loop Skewing and Permutation (Remember me?)



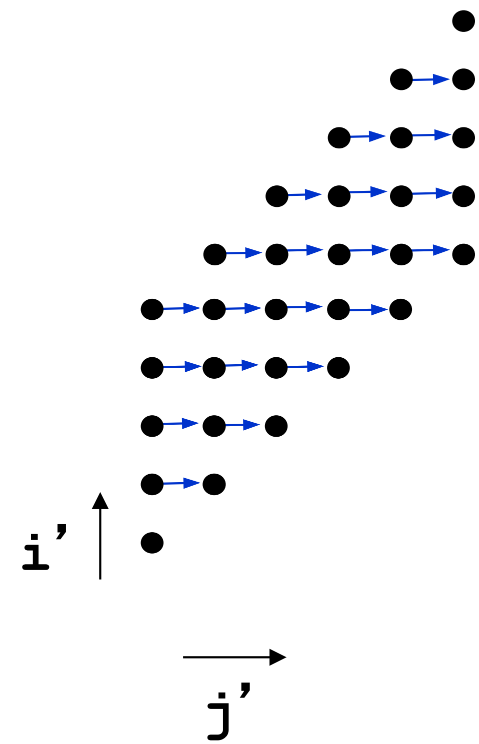
Original code

```
do i = 1,6
     do j = 1,5
          A(i,j) = A(i-1,j+1)+1
     enddo
enddo
```

Distance vector: (1, -1)

Skewing followed by Permutation:

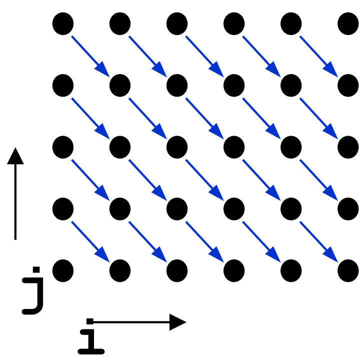
$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i' \\ j' \end{bmatrix}$$



Transforming the Dependences and Array Accesses

Original code

```
do i = 1,6
    do j = 1,5
        A(i,j) = A(i-1,j+1)+1
    enddo
enddo
```



Dependence vector:

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

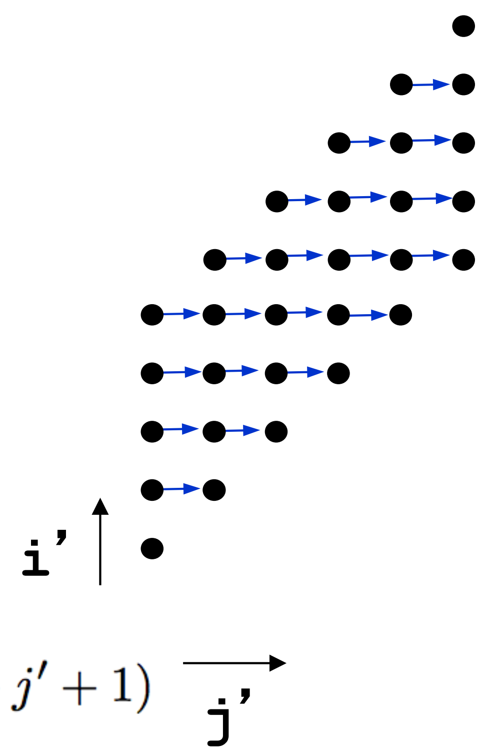
New Array Accesses:

$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = A(i,j)$$

$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} i' \\ j' \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = A(j', i'-j')$$

$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right) = A(i-1, j+1)$$

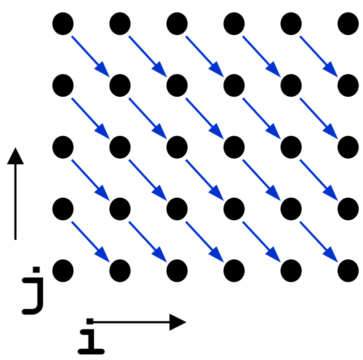
$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} i' \\ j' \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right) = A(j'-1, i'-j'+1)$$



Transforming the Loop Bounds

Original code

```
do i = 1,6
      do j = 1,5
            A(i,j) = A(i-1,j+1)+1
      enddo
enddo
```

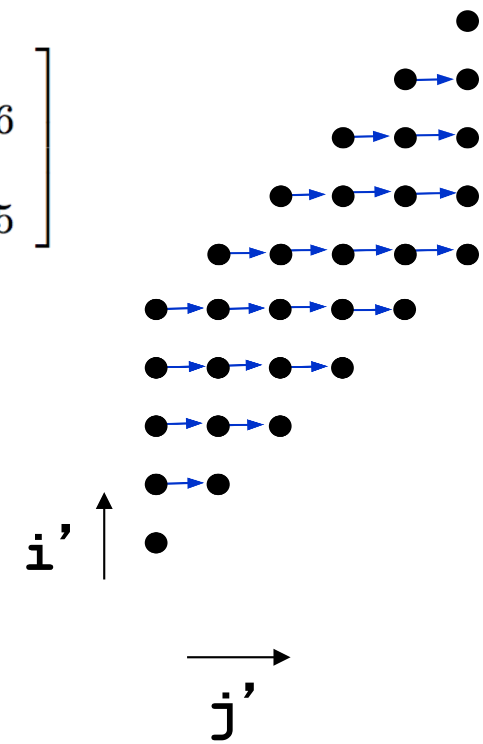


Bounds:

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \geq \begin{bmatrix} 1 \\ -6 \\ 1 \\ -5 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} \geq \begin{bmatrix} 1 \\ -6 \\ 1 \\ -5 \end{bmatrix}$$

Transformed code (use general loop bound alg)

```
do i' = 2,11
      do j' = max(i'-5,1), min(6,i'-1)
            A(j',i'-j') = A(j'-1,i'-j'+1)+1
      enddo
enddo
```



Affine Scheduling

A schedule maps each iteration to a virtual time

$$\theta(\vec{i}) = T \begin{pmatrix} \vec{i} \\ \vec{p} \\ 1 \end{pmatrix}$$

– The number of rows in T is the dimensionality of the schedule.
– The number of rows in T is also the number of outermost sequential loops.

Citation: http://www.cse.ohio-state.edu/~pouchet/lectures/doc/888.11.3.pdf

Scheduling in the Polyhedral Model

Legality

- The schedule must respect all the dependences.
- Let's turn dependence relations into constraints on the schedule solution set.
  - If iteration $\vec{i_R}$ of statement R needs to execute before iteration $\vec{i_S}$ of statement S, then the schedules for statement R and S need to satisfy the following constraint:

$$\theta_R(\vec{i_R}) \prec \theta_S(\vec{i_S})$$

One-dimensional schedules

$$\theta_R(\vec{i_R}) < \theta_S(\vec{i_S})$$

Constraint for schedule legality

Time delta

- between statement instances with dependences,
- needs to be non-negative over the dependence polyhedron

$$\Delta_{R,S} = \theta_S(\vec{i_S}) - \theta_R(\vec{i_R}) - 1 >= 0$$

<Example dependence polyhedron done on paper>

Turning this observation into scheduling constraints

Affine form of Farkas lemma

- Let D be a nonempty polyhedron defined by $A\vec{i} + \vec{b} \geq 0$.
- Any affine function f(i) is non-negative everywhere in D if and only if it is a positive affine combination of the constraints for D:

$$f(\vec{i}) = \lambda_0 + \vec{\lambda}^T(A\vec{i} + \vec{b})$$

$$\text{with } \lambda_0 \geq 0 \text{ and } \vec{\lambda} \geq \vec{0}$$

where λ_0 and $\vec{\lambda}^T$ are called the Farkas multipliers.

Building intuition about the Farkas lemma

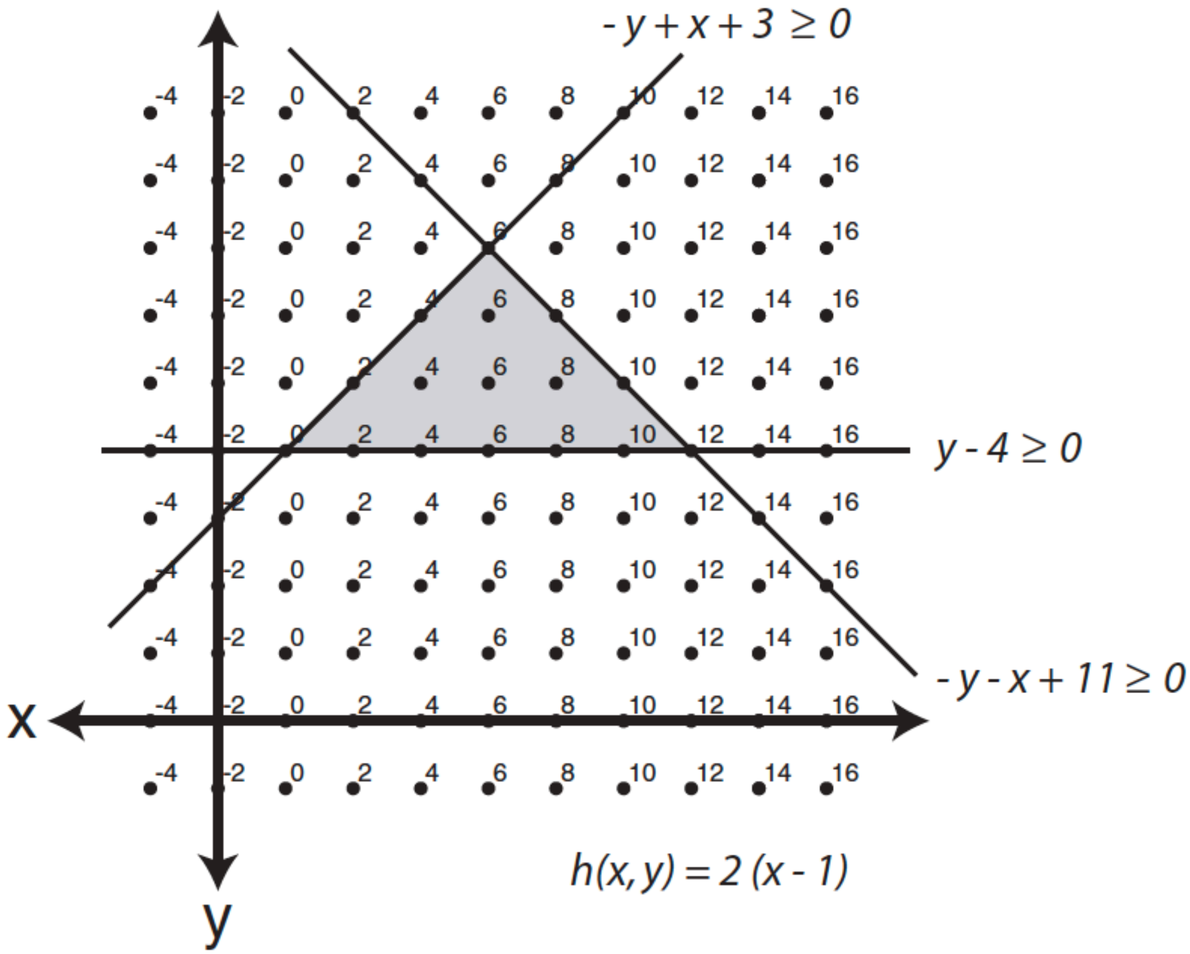


Fig. 2. An illustration of Farkas' lemma. The affine form $h(x, y) = 2 \cdot (x - 1)$ is nonnegative within the shaded polyhedron. Thus, it can be expressed as a nonnegative affine combination of the faces of that polyhedron: $h(x, y) = 2 \cdot (-y + x + 3) + 2 \cdot (y - 4)$.

Using the Farkas lemma

Assume the following dependence polyhedron

$$D_{R \to S} = \{[\vec{i} \to \vec{j} \mid A \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix} \geq \vec{0} \text{ and } B \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix} = \vec{0}\}$$

Assume a schedule function of the form

$$\theta_R(\vec{i}) = \vec{v}^T \vec{i} + \vec{b}$$
$$\theta_S(\vec{j}) = \vec{w}^T \vec{j} + \vec{c}$$

We need $\Delta_{R,S} = \theta_S(\vec{i}) - \theta_R(\vec{j}) - 1 >= 0$

The process of determining set of legal schedules

(1) Change all of the equality constraints in $D_{R \to S}$ to inequality constraints.

$$D_{R \to S} = \{[\vec{i} \to \vec{j} \mid A' \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix} \geq \vec{0}\}$$

(2) Use the Farkas lemma to create a set of constraints for the schedule.

$$\theta_S(\vec{j}) - \theta_R(\vec{i}) - 1 = \lambda_0 + \vec{\lambda}^T (A \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix})$$

$$\lambda_0 \geq 0 \text{ and } \vec{\lambda} \geq \vec{0}$$

$$\theta_R(\vec{i}) = \vec{v}^T \vec{i} + \vec{b}$$

$$\theta_S(\vec{j}) = \vec{w}^T \vec{j} + \vec{c}$$

(3) Collect coefficients for each term to create set of equalities.

(4) Solve for v, w, b, and c vector constraints by projecting out lambdas.

Example of using the Farkas lemma

Original code

```
do i = 0,N-1
  do j = 0,N-1
        A(i,j) = A(i-1,j-1)*.05
     enddo
 enddo
```

(1) Dependence polyhedron $D_{I \to I} = \{[[i_1, j_1] \to [i_2, j_2] \mid \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 & -1 \end{bmatrix} \begin{pmatrix} i_1 \\ j_1 \\ i_1 \\ j_2 \\ N \\ 1 \end{pmatrix} \geq \vec{0}\}$

(2) Farkas lemma to set up constraints

(3) Collect coefficients for each term to create set of equalities

(4) Project out lambdas to determine set of legal schedules

Example problem continued

$$\theta(i_2, j_2) - \theta(i_1, j_1) - 1 = \lambda_0 + \vec{\lambda}^T A' \begin{pmatrix} i_1 \\ j_1 \\ i_1 \\ j_2 \\ N \\ 1 \end{pmatrix}$$

$$\text{With } \theta(i,j) = a*i + b*j + c \qquad \lambda_x \geq 0, \ \forall 0 \leq x \leq 12$$

(2) Farkas lemma to set up constraints

$$a\, i_2 + b j_2 + c - a i_1 - b j_1 - c - 1 = \lambda_0 + \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \\ \lambda_9 \\ \lambda_{10} \\ \lambda_{11} \\ \lambda_{12} \end{bmatrix}^T \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 & -1 \end{bmatrix} \begin{pmatrix} i_1 \\ j_1 \\ i_2 \\ j_2 \\ N \\ 1 \end{pmatrix}$$

(3) Collect coefficients for each term to create set of equalities

$$\begin{aligned} -a &= \lambda_1 - \lambda_2 + \lambda_5 - \lambda_9 \\ -b &= \lambda_3 - \lambda_4 + \lambda_6 - \lambda_{10} \\ a &= -\lambda_1 + \lambda_2 + \lambda_7 - \lambda_{11} \\ b &= -\lambda_3 + \lambda_4 + \lambda_8 - \lambda_{12} \\ 0 &= \lambda_9 + \lambda_{10} + \lambda_{11} + \lambda_{12} \\ -1 &= \lambda_0 + \lambda_1 - \lambda_2 + \lambda_3 - \lambda_4 - \lambda_9 - \lambda_{10} - \lambda_{11} - \lambda_{12} \\ & \lambda_x \geq 0, \ \forall 0 \leq x \leq 12 \end{aligned}$$

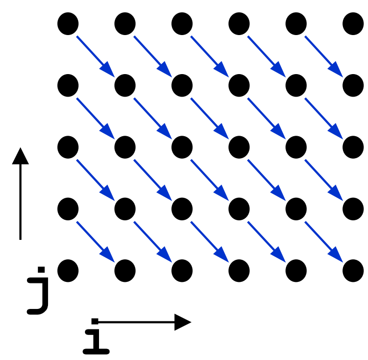
(4) Project out lambdas to determine set of legal schedules

```
{[a,b,c]: 1 <= a+b}
```

Another example of using the Farkas lemma

Original code

```
do i = 1,6
    do j = 1,5
        A(i,j) = A(i-1,j+1)+1
    enddo
enddo
```



(1) Dependence polyhedron

(2) Farkas lemma to set up constraints

(3) Collect coefficients for each term to create set of equalities

(4) Project out lambdas to determine set of legal schedules

Fourier Motzkin for Scheduling

Recall

- We need to project out the lambdas
- Now we know how to do that automatically

Next Time

Reading

- Reading for Monday: Legal reorderings paper, all but Sections 5, 6, and 7

Homework

- Start working on Assignment 3!

Lecture

- Finish up these notes
- Pointer Analysis