Tightening I/O Lower Bounds through the Hourglass Dependency Pattern

Lionel Eyraud-Dubois (Inria Bordeaux)
Guillaume Iooss (Inria Grenoble)
Julien Langou (University of Colorado Denver + Inria Lyon)
Fabrice Rastello (Inria Grenoble)

SPAA’24

18 June 2024
Modified Gram-Schmidt – QR factorization

for (j = 0; j < N; j++) {
    for (i = 0; i < j; i++) {
        R[i][j] = 0.0e+00;
        for (k = 0; k < M; k++)
            R[i][j] += A[k][i] * A[k][j];
        for (k = 0; k < M; k++)
            A[k][j] -= A[k][i] * R[i][j];
    }
    R[j][j] = 0.0;
    for (k = 0; k < M; k++)
    R[j][j] = sqrt(R[j][j]);
    for (k = 0; k < M; k++)
        A[k][j] /= R[j][j];
}

MGS

- “Everyone knows” it is memory-bound ⇒ many tricks to obtain efficient versions
- But no actual lower bound on communications!
Motivation

K-partitioning method

Hourglass pattern

Results

Modified Gram-Schmidt – QR factorization

```c
for (j = 0; j < N; j++) {
    for (i = 0; i < j; i++) {
        R[i][j] = 0.0e+00;
        for (k = 0; k < M; k++)
            R[i][j] += A[k][i] * A[k][j];
        for (k = 0; k < M; k++)
            A[k][j] -= A[k][i] * R[i][j];
    }
    R[j][j] = 0.0;
    for (k = 0; k < M; k++)
    R[j][j] = sqrt(R[j][j]);
    for (k = 0; k < M; k++)
        A[k][j] /= R[j][j];
}
```

MGS

- “Everyone knows” it is memory-bound ⇒ many tricks to obtain efficient versions
- But no actual lower bound on communications!
IOLB tool, https://iocomplexity.corse.inria.fr/iolb

Motivation

K-partitioning method

Hourglass pattern

Results

IOLB tool, https://iocomplexity.corse.inria.fr/iolb

Same bound as matrix multiplication

We improve it to $\Omega(MN^2)$ for $S \leq M^2$, and $\Omega(M^2N^2S)$ for $S \geq M^2$.

IOLB Demo - Input

```c
void qr_mgs_ll (int M, int N, double A[M][N], double R[N][N])
{
    int i, j, k;
    #pragma scop
    for (j = 0; j < N; j++) {
        for (i = 0; i < j; i++) {
            R[i][j] = 0.0e+00;
            for (k = 0; k < M; k++)
                R[i][j] += A[k][i] * A[k][j];
            for (k = 0; k < M; k++)
                A[k][j] += A[k][i] * R[i][j];
        }
    }
}
```

Enable small dimensions

Input of the IOLB tool - small dimensions

IO Complexity lower and upper bounds

$\Omega(MN^2)$

Full expression:

$MN + \max\left(0, \frac{2}{\sqrt{S}} - \frac{2M-N}{\sqrt{S}} - \frac{3M-N}{\sqrt{S}} + \frac{5M}{\sqrt{S}} - \frac{2M}{2} + \frac{7N}{2} - S - 6\right)$

Asymptotic expression:

$\frac{2}{\sqrt{S}}$
IOLB tool, https://iocomplexity.corse.inria.fr/iolb

Same bound as matrix multiplication

\[
\Omega(M^2N^2S) \quad \text{for} \quad S \leq M^2,
\]

\[
\Omega(M^2N^2S^{-1}) \quad \text{for} \quad S \geq M^2.
\]
IOLB tool, https://iocomplexity.corse.inria.fr/iolb

Same bound as matrix multiplication

- We improve it to $\Omega(MN^2)$ for $S \leq \frac{M}{2}$, and $\Omega\left(\frac{M^2N^2}{S}\right)$ for $S \geq \frac{M}{2}$. 
When optimizing for performance, many aspects to consider.

Need to estimate some key program properties:
- Volume of computation? \(\Rightarrow\) Algorithmic complexity.
When optimizing for performance, many aspects to consider.

Need to estimate some key program properties:
- Volume of computation? \(\Rightarrow\) Algorithmic complexity.
- Volume of I/O to be transferred across memories? \(\Rightarrow\) I/O Complexity: minimal amount of I/O required.
When optimizing for performance, many aspects to consider.

Need to estimate some key program properties:
- Volume of computation \( \Rightarrow \) Algorithmic complexity.
- Volume of I/O to be transferred across memories \( \Rightarrow \) I/O Complexity: minimal amount of I/O required.

How to model & compute this I/O Complexity?
I/O Complexity

- 2-level memory model:

![Diagram showing 2-level memory model with Memory, Cache, and Computation connections]
I/O Complexity

- 2-level memory model:

```
Memory  ????  Cache  ????  Computation
∞       S      ????
```

- **Minimal** number of memory transfer, for all schedule
I/O Complexity

- 2-level memory model:

![Diagram showing a 2-level memory model with Memory, Cache, and Computation components.]

- **Minimal** number of memory transfer, **for all** schedule

- Direct computation not feasible
  - ⇒ **Lower bound (proof)** + upper bound (exhibit schedule)
I/O Complexity

- 2-level memory model:

![Diagram showing a 2-level memory model with Memory, Cache, and Computation nodes.](image)

**Minimal** number of memory transfer, *for all* schedule

- Direct computation not feasible
  - \( \Rightarrow \) **Lower bound (proof)** + upper bound (exhibit schedule)

- Focus on Reads + No recomputation
Motivation

K-partitioning method

Hourglass pattern

Results

Content of this presentation - Contributions

Motivation

State-of-the-art proof method: **K-partitioning**.

Why this is not optimal for some specific kernels?

Identify a pattern of dependence that causes the issue

⇒ **Hourglass pattern**.

Content of this presentation - Contributions
Content of this presentation - Contributions

- State-of-the-art proof method: **K-partitioning**.

- Why this is not optimal for some specific kernels? Identify a pattern of dependence that causes the issue
  ⇒ **Hourglass pattern**.

- Adapt K-partitioning to improve the bound
  Integrated in the IOLB automatic lower bound derivation tool
  ⇒ Improve the bounds of many kernels by asymptotic factor.
CDAG

- Need to reason about the computation of a program

**Computational Directed Acyclic Graph (CDAG):**
- **Node** = one computation
- **Edge** = dependence between computations
Need to reason about the computation of a program

**Computational Directed Acyclic Graph (CDAG):**
- Node = one computation
- Edge = dependence between computations

**Focus on polyhedral programs:**
- Loop indexes satisfies affine constraints (ex: “0 ≤ i < N”)
- Memory accesses are affine (ex: “A[2i − j + 1]”)

⇒ Many linear algebra kernels fit these criteria
**K-partitioning method**

- **Idea:** Partition the CDAG into convex $K$-sets

**K-set**
Set of nodes of the CDAG, such that the size of its \textit{inset} (input data) is $\leq K$. 

\[ \text{Theorem (Hong and Kung'81)} \]
With $S$ the cache size, for any $K$-partition:

\[ \# I/O \geq (K - S) \times \max(\text{Num\_Sets\_in\_Partition}) \]
\[ \geq (K - S) \times \text{Num\_Nodes\_CDAG} \]
\[ \times \max(\text{Size\_KSet}) \]

$\Rightarrow$ Convert an upper bound on $K$-set size into a lower bound on I/O.
**K-partitioning method**

- **Idea:** Partition the CDAG into convex $K$-sets

**K-set**
Set of nodes of the CDAG, such that the size of its *inset* (input data) is $\leq K$.

**Theorem (Hong and Kung’81)**

*With $S$ the cache size, for any $K$-partition:*

$$\#I/O \geq (K - S) \times \max(\text{Num\_Sets\_in\_Partition})$$

$$\geq (K - S) \times \frac{\text{Num\_Nodes\_CDAG}}{\max(\text{Size\_KSet})}$$

$\Rightarrow$ Convert an upper bound on $K$-set size into a lower bound on I/O.
Deriving an upper bound of a K-set

\( E \) K-set of arbitrary shape

Upper bound on \(|E|\) ?
Deriving an upper bound of a K-set

\( E \) K-set of arbitrary shape
Upper bound on \( |E| \) ?

\( \text{InSet}(E) \): input data of \( E \)
\[ |\text{InSet}(E)| \leq K \]
Deriving an upper bound of a K-set

$E$ K-set of arbitrary shape
Upper bound on $|E|$?

$\text{InSet}(E)$: input data of $E$

$|\text{InSet}(E)| \leq K$

1) Derive paths that maps from $E$ to $\text{InSet}(E)$
Deriving an upper bound of a $K$-set

$E$ K-set of arbitrary shape
Upper bound on $|E|$ ?

$\text{InSet}(E)$: input data of $E$

$$|\text{InSet}(E)| \leq K$$

1) Derive paths that maps from $E$ to $\text{InSet}(E)$

2) Projections $\phi_x$ from paths

$$|\phi_x(E)| \leq |\text{InSet}(E)| \leq K$$
Deriving an upper bound of a K-set

$E$ K-set of arbitrary shape
Upper bound on $|E|$?

$\text{InSet}(E)$: input data of $E$

$|\text{InSet}(E)| \leq K$

1) Derive paths that maps from $E$ to $\text{InSet}(E)$

2) Projections $\phi_x$ from paths

$|\phi_x(E)| \leq |\text{InSet}(E)| \leq K$

3) Brascamp-Lieb theorem:

$|E| \leq |\phi_1(E)| \times |\phi_2(E)|$

$\Rightarrow |E| \leq K^2$
Example: Modified Gram-Schmidt

```
for (j = 0; j < N; j++) {
    for (i = 0; i < j; i++) {
        R[i][j] = 0.0e+00;
        for (k = 0; k < M; k++)
            R[i][j] += A[k][i] * A[k][j];
        for (k = 0; k < M; k++)
    }
    R[j][j] = 0.0;
    for (k = 0; k < M; k++)
    R[j][j] = sqrt(R[j][j]);
    for (k = 0; k < M; k++)
        A[k][j] /= R[j][j];
}
```
Example: Modified Gram-Schmidt

```
for (j = 0; j < N; j++) {
    for (i = 0; i < j; i++) {
        R[i][j] = 0.0e+00;
        for (k = 0; k < M; k++)
            R[i][j] += A[k][i] * A[k][j];
    }
    R[j][j] = 0.0;
    for (k = 0; k < M; k++)
    R[j][j] = sqrt(R[j][j]);
    for (k = 0; k < M; k++)
        A[k][j] /= R[j][j];
}
```

- **3 paths** $\Rightarrow$ **3 projections:**
  $$|\phi_{\cdot\cdot}(E)| \leq K$$
Example: Modified Gram-Schmidt

\[
\begin{align*}
\text{for} \ (j = 0; \ j < N; \ j++) \ { } \\
\text{for} \ (i = 0; \ i < j; \ i++) \ { } \\
R[i][j] &= 0.0e+00; \\
\text{for} \ (k = 0; \ k < M; \ k++) \ { } \\
R[i][j] &= A[k][i] \ast A[k][j]; \\
\text{for} \ (k = 0; \ k < M; \ k++) \ { } \\
A[k][j] &= A[k][j] - A[k][i] \ast R[i][j]; \\
R[j][j] &= 0.0; \\
\text{for} \ (k = 0; \ k < M; \ k++) \ { } \\
R[j][j] &= A[k][j] \ast A[k][j]; \\
R[j][j] &= \text{sqrt}(R[j][j]); \\
\text{for} \ (k = 0; \ k < M; \ k++) \ { } \\
A[k][j] &= R[j][j]; \\
\end{align*}
\]

- 3 paths ⇒ 3 projections:
  \[|\phi_{i,j}(E)| \leq K\]
- Brascamp-Lieb:
  \[|E| \leq |\phi_{i,j}(E)|^{\frac{1}{2}} \times |\phi_{i,k}(E)|^{\frac{1}{2}} \times |\phi_{k,j}(E)|^{\frac{1}{2}}\]
  \[\Rightarrow |E| \leq K^{\frac{3}{2}}\]
Example: Modified Gram-Schmidt

```c
for (j = 0; j < N; j++) {
  for (i = 0; i < j; i++) {
    R[i][j] = 0.0e+00;
    for (k = 0; k < M; k++)
      R[i][j] += A[k][i] * A[k][j];
  } // end i loop
  R[j][j] = 0.0;
  for (k = 0; k < M; k++)
    R[j][j] += A[k][j] * R[i][j];
  R[j][j] = sqrt(R[j][j]);
  for (k = 0; k < M; k++)
    A[k][j] /= R[j][j];
} // end j loop
```

- **3 paths** ⇒ **3 projections:**
  \[
  |\phi_i, j(E)| \leq K
  \]
- **Brascamp-Lieb:**
  \[
  |E| \leq |\phi_{i,j}(E)|^{\frac{1}{2}} \times |\phi_{i,k}(E)|^{\frac{1}{2}} \times |\phi_{k,j}(E)|^{\frac{1}{2}}
  \]
  \[\Rightarrow |E| \leq K^{\frac{3}{2}}\]
- **\(Q_{MGS} \geq \Omega\left(\frac{MN^2}{\sqrt{S}}\right)\)**
Example: Modified Gram-Schmidt

```c
for (j = 0; j < N; j++) {
    for (i = 0; i < j; i++) {
        R[i][j] = 0.0e+00;
        for (k = 0; k < M; k++)
            R[i][j] += A[k][i] * A[k][j];
    }  
    R[j][j] = 0.0;
    for (k = 0; k < M; k++)
    R[j][j] = sqrt(R[j][j]);
    for (k = 0; k < M; k++)
        A[k][j] /= R[j][j];
}
```

- **3 paths ⇒ 3 projections:**
  \[ |\phi_{i,j}(E)| \leq K \]

- **Brascamp-Lieb:**
  \[ |E| \leq |\phi_{i,j}(E)|^{1/2} \times |\phi_{i,k}(E)|^{1/2} \times |\phi_{k,j}(E)|^{1/2} \]
  \[ \Rightarrow |E| \leq K^{3/2} \]

- **\( \Omega(MN^2) \)**

... but best known schedule in \( O(MN^2) \). Can we do better?
Motivation

K-partitioning method

Hourglass pattern

Results

Example: Modified Gram-Schmidt

```c
for (j = 0; j < N; j++) {
    for (i = 0; i < j; i++) {
        R[i][j] = 0.0e+00;
        for (k = 0; k < M; k++)
            R[i][j] += A[k][i] * A[k][j];
    }
    R[j][j] = 0.0;
    for (k = 0; k < M; k++)
    R[j][j] = sqrt(R[j][j]);
    for (k = 0; k < M; k++)
        A[k][j] /= R[j][j];
}

R[j][j] = 0.0;
for (k = 0; k < M; k++)
R[j][j] = sqrt(R[j][j]);
for (k = 0; k < M; k++)
    A[k][j] /= R[j][j];
```

\[ \phi_{k,j} - \phi_{k,i} \cdot \phi_{i,j} \]

\[ \Rightarrow |E| \leq K \cdot Q_{MGS} \geq \Omega \left( MN^{2} \sqrt{S} \right) \]

Reduction

Broadcast

... but best known schedule in \( O(MN^{2}) \). Can we do better?
The Hourglass pattern

**Iteration** $t$

- Reduction
- Broadcast

**Iteration** $(t + 1)$

- Reduction
- Broadcast

Constraints:
- Time dimension (often outer) ($i$)
- Broadcast/Red dimension (often inner) ($k$)
- Other dimensions: neutral ($j$)

And Width of hourglass is large (ex: $M$)

If reduction is large, not tilable! ⇒ Strongly constraints shape of a (convex) $K$-set.
The Hourglass pattern

Constraints:
\[
\begin{align*}
\text{Time dimension (often outer)} & \quad (i) \\
\text{Broadcast/Red dimension (often inner)} & \quad (k) \\
\text{Other dimensions: neutral} & \quad (j)
\end{align*}
\]

And Width of hourglass is large (ex: \(M\))
The Hourglass pattern

Constraints:
\[
\begin{align*}
& \text{Time dimension (often outer)} \quad (i) \\
& \text{Broadcast/Red dimension (often inner)} \quad (k) \\
& \text{Other dimensions: neutral} \quad (j)
\end{align*}
\]

And Width of hourglass is large (ex: $M$)

If reduction is large, not tilable!
⇒ Strongly constraints shape of a (convex) $K$-set.
Implication on the shape of $E$

Split the connected components of $E$ in 2 parts:

- **Thick** along temporal dimension ($E_1$)
  - $\Rightarrow$ Must cover all the Red/Bcst dim
- **Flat** along temporal dimension ($E_2$)
Implication on the shape of $E$

Split the connected components of $E$ in 2 parts:

- **Thick** along temporal dimension ($E_1$)
  - $\Rightarrow$ Must cover all the Red/Bcst dim
- **Flat** along temporal dimension ($E_2$)

\[ |\phi_i(E_1)| = M \]

If $\phi_{\bullet,i}$ is one projection:

\[ |\phi_{\bullet}(E_1)| \leq \frac{K}{M} \]
Implication on the shape of $E$

Split the connected components of $E$ in 2 parts:
- **Thick** along temporal dimension ($E_1$)
  - $\Rightarrow$ Must cover all the Red/Bcst dim
- **Flat** along temporal dimension ($E_2$)

\[
|\phi_i(E_1)| = M \\
|\phi_k(E_2)| \leq 2
\]
Implication on the shape of $E$

Split the connected components of $E$ in 2 parts:
- **Thick** along temporal dimension ($E_1$)
  - Must cover all the Red/Bcst dim
- **Flat** along temporal dimension ($E_2$)

$|\phi_i(E_1)| = M$  \[\leq 2\]

If $\phi_{i,j}$ is one projection:
- $|\phi_{i}(E_1)| \leq \frac{K}{M}$
- $|\phi_{k}(E_2)| \leq 2$

$\Rightarrow$ New constraints on projections sizes to exploit, for both parts.
Putting things together

**Example - Modified Gram-Schmidt.**
By adapting the list of projections given to Brascamp-Lieb:

- **First part (Thick):**

  Instead of:  
  \[ |E_1| \leq |\phi_{i,j}(E_1)|^{\frac{1}{2}} \times |\phi_{i,k}(E_1)|^{\frac{1}{2}} \times |\phi_{j,k}(E_1)|^{\frac{1}{2}} \leq K^3 \]

  We have:  
  \[ |E_1| \leq |\phi_i(E_1)| \times |\phi_j(E_1)| \times |\phi_k(E_1)| \leq M \times \frac{K}{M} \times \frac{K}{M} = \frac{K^2}{M} \]
Example - Modified Gram-Schmidt.
By adapting the list of projections given to Brascamp-Lieb:

- First part (Thick):
  
  Instead of: \[ |E_1| \leq |\phi_{i,j}(E_1)|^{1/2} \times |\phi_{i,k}(E_1)|^{1/2} \times |\phi_{j,k}(E_1)|^{1/2} \leq K^{3/2} \]
  
  We have: \[ |E_1| \leq |\phi_i(E_1)| \times |\phi_j(E_1)| \times |\phi_k(E_1)| \leq M \times \frac{K}{M} \times \frac{K}{M} = \frac{K^2}{M} \]

- Second part (Flat):
  
  Instead of: \[ |E_2| \leq K^{3/2} \]
  
  We have: \[ |E_2| \leq |\phi_k(E_2)| \times |\phi_{i,j}(E_2)| \leq 2K \]
Putting things together

Example - Modified Gram-Schmidt.

By adapting the list of projections given to Brascamp-Lieb:

- **First part (Thick):**
  
  Instead of: \( |E_1| \leq |\phi_{i,j}(E_1)|^{\frac{1}{2}} \times |\phi_{i,k}(E_1)|^{\frac{1}{2}} \times |\phi_{j,k}(E_1)|^{\frac{1}{2}} \leq K^{\frac{3}{2}} \)

  We have: \( |E_1| \leq |\phi_i(E_1)| \times |\phi_j(E_1)| \times |\phi_k(E_1)| \leq M \times \frac{K}{M} \times \frac{K}{M} = \frac{K^2}{M} \)

- **Second part (Flat):**
  
  Instead of: \( |E_2| \leq K^{\frac{3}{2}} \)

  We have: \( |E_2| \leq |\phi_k(E_2)| \times |\phi_{i,j}(E_2)| \leq 2K \)

- **Total:** \( |E| = |E_1| + |E_2| \leq \frac{K^2}{M} + 2K \). (instead of: \( |E| \leq K^{\frac{3}{2}} \))

  \( \Rightarrow \) When \( M \) is big, we gain a \( \sqrt{K} \) factor in the asymptotic bound.
Proof automated/integrated to IOLB [Olivry et al, PLDI'20]
Demo: https://iocomplexity.corse.inria.fr/
## Results

- Proof automated/integrated to IOLB [Olivry et al, PLDI’20]
  Demo: [https://iocomplexity.corse.inria.fr/](https://iocomplexity.corse.inria.fr/)

- Kernels with an hourglass + asymptotic bounds

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Old bound</th>
<th>New bound (hourglass)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGS</td>
<td>$\Omega \left( \frac{MN^2}{\sqrt{S}} \right)$</td>
<td>$\Omega \left( \frac{M^2N(N-1)}{S+M} \right)$</td>
</tr>
<tr>
<td>QR HH A2V</td>
<td>$\Omega \left( \frac{MN^2}{\sqrt{S}} \right)$</td>
<td>$\Omega \left( \frac{MN^2(N-M)}{N-M-S} \right)$</td>
</tr>
<tr>
<td>QR HH V2Q</td>
<td>$\Omega \left( \frac{MN^2}{\sqrt{S}} \right)$</td>
<td>$\Omega \left( \frac{MN^2(N-M)}{N-M-S} \right)$</td>
</tr>
<tr>
<td>GEBD2</td>
<td>$\Omega \left( \frac{MN^2}{\sqrt{S}} \right)$</td>
<td>$\Omega \left( \frac{MN^2(M-N+1)}{8(S+M-N+1)} \right)$</td>
</tr>
<tr>
<td>GEHD2</td>
<td>$\Omega \left( \frac{N^3}{\sqrt{S}} \right)$</td>
<td>$\Omega \left( \frac{N^4}{N+2S} \right)$</td>
</tr>
<tr>
<td>SYTD2 (new)</td>
<td>$\Omega \left( \frac{N^3}{\sqrt{S}} \right)$</td>
<td>$\Omega \left( \frac{N^4}{N+2S-2} \right)$</td>
</tr>
</tbody>
</table>
Width of hourglass might vary (with temporal dim).
  + For our bound, need to use the minimum of the width.
⇒ Issue when this minimum is 1.
Width of hourglass might vary (with temporal dim).
  + For our bound, need to use the minimum of the width.

⇒ Issue when this minimum is 1.

**Solution:** loop splitting transformation.
  - Does not change the CDAG.
  - Hourglass detected on the “wide” part of the split.
  - Adjust where to split to deduce the best bound.