




An Approach to Tight I/O Lower Bounds for Algorithms with Composite Procedures

Rui Xia^{1,2} , Ligang Cao^{1,2}, Huajian Zhang^{1,2}, Jihu Guo^{1,2}, Xiaowei Guo^{1,2,3} , Jie Liu^{1,2,4} , Huaimin Wang^{1,4}

¹ College of Computer Science and Technology, National University of Defense Technology

² Laboratory of Digitizing Software for Frontier Equipment, National University of Defense Technology

³ Institute for Quantum Information & State Key Laboratory of High Performance Computing, National University of Defense Technology

⁴ National Key Laboratory of Parallel and Distributed Computing, National University of Defense Technology

xiarui21@nudt.edu.cn, {ligangcao, zhjsag}@nudt.edu.cn, guojihu1998cs@gmail.com
{guoxiaowei, liujie}@nudt.edu.cn, whm_w@163.com

Abstract. Analyzing the I/O lower bound is crucial for enhancing dataflow performance within the memory hierarchy. Existing methods struggle with composite step programs, often underestimating I/O behavior during subcomputations. To address this, we introduce the (X_1, X_2) -Partition theory, providing tighter I/O lower bounds for statically analyzable programs. We evaluate this theory on stencil operators and show that the I/O lower bound becomes tighter compared to X -Partition theory. Furthermore, we develop a method for calculating I/O lower bounds for composite algorithms using the (X_1, X_2) -Partition theory. By analyzing the dataflow of the QR Decomposition algorithm, we demonstrate the effectiveness of our method in handling composite procedures. We also introduced this theory into the neural network architecture search task for the first time, and preliminary experimental results further demonstrated the broad application prospects of this theory.

Keywords: I/O Complexity · I/O Lower Bound · Red-blue Pebble Game · Network Architecture Search

1 Introduction

One of the major expenses in scientific computing programs is the overhead of input/output (I/O) operations across memory structures. Studying the I/O lower bound, as mentioned in [1], is crucial for achieving high-performance computing, minimizing I/O overhead, and efficiently managing I/O streams. This analytical theory is essential for designing parallel algorithms [2] and communication optimization algorithms [3], greatly benefiting scientific computing [4, 5].

The theory of I/O lower bounds has a rich history, with Hong and Kung et al. playing a pivotal role [6]. They introduced the influential Red-Blue Pebble Game model, upon which the S -Partition theory was formulated, marking the first I/O lower bound for matrix-matrix multiplication. Since then, subsequent studies have built upon this foundation. Auguste Olivry et al. [7] developed IOLB, a precise tool for calculating I/O lower bounds for affine programs. Zhang et al. [8] optimized the S -Partition theory to estimate I/O lower bounds of composite algorithms, especially for convolutional operations. Grzegorz Kwasniewski et al. [9] further advanced this field with the X -Partition theory and SOAP analysis, enhancing bounds for Polybench operators. Saachi Jain et al. [10] proposed a novel method to determine I/O bounds of computational graphs, focusing on the first few eigenvalues of the Laplacian operator. Their analysis covered algorithms like Bellman-Held-Karp and the Fast Fourier Transform, yielding precise bounds.

Previous theoretical approaches have limitations: they’re specific to certain operators and algorithms, limiting their applicability to composite ones. Additionally, they neglect I/O overhead during subcomputation, making it challenging to establish tighter I/O lower bounds for write-intensive programs. This paper addresses these issues by introducing a method to establish stricter I/O lower bounds for composite process algorithms. Specifically, the contributions of this paper are as follows.

- We propose introducing the (X_1, X_2) -Partition theory to derive more stringent I/O lower bounds by incorporating potential I/O behavior from intermediate subcomputations, surpassing previous theories.
- Using (X_1, X_2) -Partition, we introduce an analysis approach for composite process programs, which not only enables a tighter I/O lower bound but also reveals previously undetectable I/O behaviors.
- Utilizing the proposed theory and method, we analyzed both the stencil operator and QR decomposition, demonstrating its efficacy in both cases.
- We first incorporated the I/O lower bound theory into neural network architecture search, and preliminary experimental results showed that the I/O performance of the architectures on CIFAR10 and CIFAR100 was improved.

The article is organized as follows: Chapter 2 covers the background. Chapter 3 introduces the (X_1, X_2) -Partition theory and I/O lower bound analysis for composite algorithms. Chapter 4 applies this theory to stencil operator dataflow. Chapter 5 analyzes QR decomposition dataflow using the I/O lower bound method. Chapter 6 presents preliminary experimental results after introducing the I/O lower bound theory into neural network architecture search. Chapter 7 summarizes the paper and discusses future research.

2 Background

2.1 Red and Blue Pebble Game

This section briefly introduces Red and Blue Pebble Games [11], reimaged as a directed acyclic graph (CDAG) $G = (V, E)$ with edges $(u, v) \in E$ representing

computation dependencies. For execution, the program adapts to a sequential machine with a two-tiered storage system: finite-sized fast memory (red pebbles) and infinite-sized slow memory (blue pebbles). Red pebbles indicate fast memory residence, while blue pebbles represent slow memory storage. This game follows four principles:

- **Load**: placing a red pebble on a vertex which has a blue pebble.
- **Store**: placing a blue pebble on a vertex which has a red pebble.
- **Compute**: placing a red pebble on a vertex whose parents have a red pebble.
- **Discard**: removing any pebble from a vertex.

The program starts with blue pebbles on all input vertices. Program execution ends when all output vertices have blue pebbles on them. The program's I/O lower bound Q is the minimum cost of all valid pebble configurations. The pebble configuration with cost Q is called the optimal configuration.

For convenience, in addition to the above constraints, the computer model in this paper also stipulates that the computer model does not output external results before all sub-computations are completed, that is, there is a barrier between the sub-computation and the final output. How to extend it to the situation where there is no barrier will be the direction of our future research.

2.2 X-Partition

The derivation of the X -Partition theory is detailed in Grzegorz Kwasniewski et al. [9]. Within this section, we present the pertinent conclusions directly. Prior to introducing definitions and theorems, we establish a comprehensive list of symbols used throughout the text in Table 1.

We state the X -Partition as Definition 1 and we can construct an X -Partition that satisfies Theorem 1. We will give a brief proof in Appendix A¹.

Definition 1. *For any directed acyclic graph $G = (V, E)$ of any program, for any $X \geq S$, if there is a set of vertex sets V_1, V_2, \dots, V_h that satisfies the following conditions, this set is called an **X -Partition** of this program:*

- **I**: $\forall i \neq j, V_i \cap V_j = \emptyset$, and $\cup_i V_i = V$.
- **II**: *There is no circular data correlation between V_i .*
- **III**: $\forall i, |In(V_i)| \leq X$.
- **IV**: $\forall i, |Out(V_i)| \leq X$.

Theorem 1. *Let $\mathcal{H}(X)$ be the minimum number of subcomputations in any valid X -Partition of the CDAG $G = (V, E)$ for any $X \geq S$. The minimum number of I/O operations Q for any valid execution of a CDAG $G = (V, E)$ is bounded by inequality(1) and Eq.(2). $\mathcal{R}(S)$ is the maximum reuse set size that satisfies $\forall i, \max\{|H_{R,i}|, |W_{R,i}|\} \leq \mathcal{R}(S)$. $\mathcal{T}(S)$ is the minimum I/O set size that satisfies $\forall i, \min\{|H_{BR,i}|, |W_{RB,i}|\} \geq \mathcal{T}(S)$.*

¹ We provide the appendix of the paper at <https://github.com/DongFengZero/COCOON24>.

Symbol	Meaning
$W_{B,i}$	The set of vertices with blue pebbles according to the storage rules in the i -th subcomputation.
$W_{R,i}$	The set of vertices with red pebbles at the end of the i -th subcomputation and at least one child will undergo pebbling in the $i + 1$ -th subcomputation.
$W_{RB,i}$	The set of vertices that transition from red to blue during the i -th subcomputation, regardless of their final color.
$H_{R,i}$	Before the i -th subcomputation, the set of red nodes with at least one child will be pebbled.
$H_{BR,i}$	The blue vertices required for the i -th subcomputation.
$In(V_i)$	The input set of the i -th subcomputation.
$Out(V_i)$	The output set of the i -th subcomputation.
R_j^i	The set of red nodes (retained in fast memory) before the $j + 1$ -th step of the i -th subcomputation.
I^i	The set of input nodes required for the i -th subcomputation.
I_j^i	The set of input nodes required for the j -th step of the i -th subcomputation.
\tilde{O}_j^i	The set of output nodes required for the j -th step of the i -th subcomputation.

Table 1. Explanation of some symbols used in this article.

$$Q \geq (X - \mathcal{R}(S) + \mathcal{T}(S)) \cdot (\mathcal{H}(X) - 1) \quad (1)$$

$$\mathcal{H}(X) = \frac{|V|}{|\mathcal{H}_{max}|} \quad (2)$$

Where $\mathcal{H}_{max} = \arg \max_{H_i \in \mathcal{S}(X)} |\mathcal{H}_i|$ is the largest subset of vertices in the CDAG schedule $\mathcal{S}(X) = \mathcal{H}_1, \dots, \mathcal{H}_h$.

Further, we have the following definition. **Computational intensity** is defined as $\rho = \frac{|\mathcal{H}_i|}{X - |H_{R,i}| + |W_{B,i}|} \cdot \rho \leq \rho_{max} = \frac{|\mathcal{H}_{max}|}{X_0 - \mathcal{R}(S) + \mathcal{T}(S)}$ is called **maximum computational intensity**. Let $\chi(X) = |\mathcal{H}_{max}|$ be the maximum size of a single subcomputation. Q is constrained by the inequality(3).

$$Q \geq |V| \frac{X_0 - \mathcal{R}(S) + \mathcal{T}(S)}{\chi(X_0)} = \frac{|V|}{\rho_{max}} \quad (3)$$

Where $X_0 = \arg \min_X \frac{\chi(X)}{X - \mathcal{R}(S) + \mathcal{T}(S)}$, thus $\rho_{max} = \frac{\chi(X_0)}{X_0 - \mathcal{R}(S) + \mathcal{T}(S)}$.

3 Theory And Method

3.1 (X_1, X_2) -Partition

X-Partition theory gives a way to estimate the I/O lower bound of a program. However, there is still room for further improvement of this theorem. $W_{R,i} \cap W_{RB,i}$ is often not null, giving optimization ideas for evaluating write operations. Inequality (4) is held according to **the principle of inclusion-exclusion**.

$$|Out(V_i)| \leq |W_{R,i}| + |W_{RB,i}| - |W_{R,i} \cap W_{RB,i}| \quad (4)$$

Let $\mathcal{R}_1(S) = \max_i |H_{R,i}|$, $\mathcal{R}_2(S) = \max_i |W_{R,i}|$, $\mathcal{T}_1(S) = \min_i |H_{B,i}|$, $\mathcal{T}_2(S) = \min_i |W_{RB,i}|$, $\mathcal{T}_3(S) = \min_i |W_{R,i} \cap W_{RB,i}|$. $Q(V_1, S) = \dots = Q(V_{h-1}, S) = Y$, $Q(V_h, S) \leq Y$. Based on the above definitions, it is obvious that inequality (5)-(6) hold.

$$|In(V_i)| \leq |H_{R,i}| + |H_{B,i}| \leq \mathcal{R}_1(S) + Y - \mathcal{T}_2(S) \quad (5)$$

$$\begin{aligned} |Out(V_i)| &\leq |W_{R,i}| + |W_{RB,i}| - |W_{R,i} \cap W_{RB,i}| \\ &\leq \mathcal{R}_2(S) + Y - \mathcal{T}_1(S) - \mathcal{T}_3(S) \end{aligned} \quad (6)$$

Let $X_1 = \mathcal{R}_1(S) + Y - \mathcal{T}_2(S)$, $X_2 = \mathcal{R}_2(S) + Y - \mathcal{T}_1(S) - \mathcal{T}_3(S)$. Building upon the existing X-Partition Definition 1 and Theorem 1, we can construct a concise definition of the (X_1, X_2) -Partition, resulting in Definition 2 and Theorem 2 for enhanced clarity and brevity.

Definition 2. For any directed acyclic graph $G = (V, E)$ of any program, let $X_1 = \mathcal{R}_1(S) + Y - \mathcal{T}_2(S)$, $X_2 = \mathcal{R}_2(S) + Y - \mathcal{T}_1(S) - \mathcal{T}_3(S)$, for any $\min\{X_1, X_2\} \geq S$, if there is a set of vertex sets V_1, V_2, \dots, V_h that satisfies the following conditions, this set is called an (X_1, X_2) -**Partition** of this program:

- **I**: $\forall i \neq j, V_i \cap V_j = \emptyset$, and $\cup_i V_i = V$.
- **II**: There is no circular data correlation between V_i .
- **III**: $\forall i, |In(V_i)| \leq \max\{X_1, X_2\}$.
- **IV**: $\forall i, |Out(V_i)| \leq \max\{X_1, X_2\}$.

Theorem 2. Let $X_1 = \mathcal{R}_1(S) + Y - \mathcal{T}_2(S)$, $X_2 = \mathcal{R}_2(S) + Y - \mathcal{T}_1(S) - \mathcal{T}_3(S)$, $X_{max} = \max\{X_1, X_2\}$. $\mathcal{H}(X_{max})$ be the minimum number of subcomputations in any valid (X_1, X_2) -Partition of the CDAG $G = (V, E)$. $\chi(X_{max})$ is the same as the definition in Theorem 1. For any $\min\{X_1, X_2\} \geq S$, the minimum I/O operations Q for any valid execution of a CDAG $G = (V, E)$ is bounded by inequality (7) and Eq.(8).

$$Q \geq \mathcal{Z}(X_{max}, S) \cdot (\mathcal{H}(X_{max}) - 1) \quad (7)$$

$$\mathcal{Z}(X_{max}, S) = \begin{cases} X_2 - \mathcal{R}_2(S) + \mathcal{T}_1(S) + \mathcal{T}_3(S) & X_1 \leq X_2 \\ X_1 - \mathcal{R}_1(S) + \mathcal{T}_2(S) & X_1 > X_2 \end{cases} \quad (8)$$

Let $X_0 = \arg \min_{X_{max}} \frac{\chi(X_{max})}{\mathcal{Z}(X_{max}, S)}$, then $\rho_{max} = \frac{\chi(X_0)}{\mathcal{Z}(X_0, S)}$ and Q is constrained by $Q \geq \mathcal{X}(X_0, S) \cdot (\mathcal{H}(X_0) - 1) \approx \frac{|V|}{\rho_{max}}$.

3.2 (X_1, X_2) -Partition for Subcomputations With Combined Steps

In this section, we further introduce how to use the theory in the previous section to lower the I/O lower bound of the algorithm with n combined steps.

First, we need to separate sub-calculations with compound steps according to the granularity required. We estimate the maximum sub-computation domain size $|\mathcal{H}_{max}|$ and the calculation domain size $|V|$, and calculate $\mathcal{H}(X_{max})$ according to Equation (2). We provide a method in Appendix B.

Next, we need to calculate the five key sets. We can give the bounds of the five key sets $\mathcal{R}_1(S), \mathcal{R}_2(S), \mathcal{T}_1(S), \mathcal{T}_2(S), \mathcal{T}_3(S)$ required to solve the problem as shown in Eq.(9)-(13) and Fig. 1.

$$\mathcal{R}_1(S) = \max_i |R_0^i| \quad (9)$$

$$\mathcal{R}_2(S) = \max_i |R_n^i| \quad (10)$$

$$\mathcal{T}_1(S) = \min_i |I^i \setminus R_0^i| \quad (11)$$

$$\mathcal{T}_2(S) = \min_i \left| \bigcup_{j=1}^n (\tilde{O}_j^i \setminus R_j^i) \right| \quad (12)$$

$$\mathcal{T}_3(S) = \min_i \left| \bigcup_{j=1}^{n-1} (\tilde{O}_j^i \setminus R_j^i \cap \bigcup_{r=j+1}^n R_r^i) \right| \quad (13)$$

Finally, we get X_0 according to $X_0 = \arg \min_{X_{max}} \frac{\chi(X_{max})}{\mathcal{Z}(X_{max}, S)}$ and calculate the maximum calculation density ρ_{max} . We substitute it into the Theorem 2 to obtain the I/O lower bound Q .

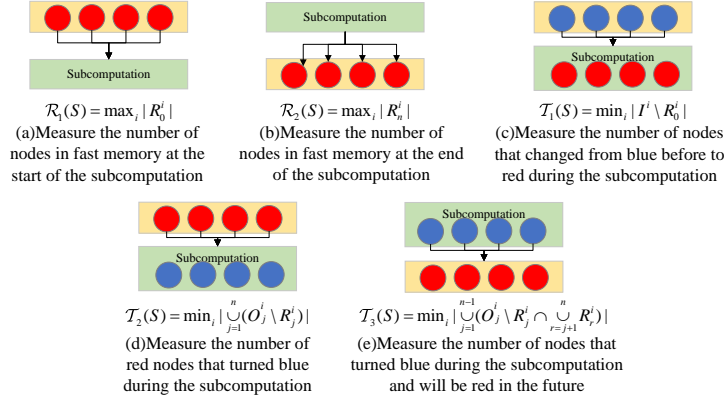


Fig. 1. Schematic diagram of the bound of the five sets

4 I/O Lower Bound Analysis for Stencil Operator

Taking *adi* as an instance, we analyze its I/O lower bound with (X_1, X_2) -Partition, utilizing a method akin to [9]. Our emphasis is on assessing Eq.(14)-Eq.(16), which is the contributor to I/O overhead Q . Since the processes of row scanning and column scanning are the same, only one of the processes is analyzed below, and is doubled when calculating Q in the end.

$$P_{j,i} = -\frac{C_3}{C_1 P_{i,j-1} + C_2} \quad (14)$$

$$Q_{i,j} = \frac{-C_4 U_{j,i-1} + (1 + 2C_4)U_{j,i} - C_5 U_{j,i+1} - C_1 Q_{i,j-1}}{C_1 P_{i,j-1} + C_2} \quad (15)$$

$$V_{j,i} = P_{i,j} V_{j+1,i} + Q_{i,j} \quad (16)$$

Based on the Loomis-Whitney inequality, $|V_{max}| = 2\sqrt{a^3 b^3}$ and $\max\{R_1(S), R_2(S)\} \leq 4ab$. The required data for formula computation includes $U_{i,j}$, $P_{i,j}$, $V_{i,j}$ and $Q_{i,j}$ totaling $4ab$. Considering fast memory capacity, $\mathcal{T}_1(S) \geq \max\{4ab - S, 0\}$. Additionally, at least $12ab$ intermediate variables are generated, each operation yielding ab . Every $4ab$ initial variables can produce $12ab$ intermediate variables and $3ab$ final results. This results in $\mathcal{T}_2(S) \geq \max\{4ab - S, 0\} + \max\{12ab - \max\{S - 4ab, 0\}, 0\} + \max\{3ab - \max\{S - 16ab, 0\}, 0\}$. Assuming that starting from the calculation of the n -th group of intermediate variables will cause the initial variables that have not been used to be written back, then $S - 4(ab - n) \leq 12n$ must be satisfied, and then $8(ab - \frac{S-4ab}{8})$ variables are written back, they will still be read in future calculations, consistent with the definition of $\mathcal{T}_3(S)$. Incorporating the memory capacity constraint, we have $\mathcal{T}_3(S) \geq \max\{4ab - S, 0\} + \max\{3ab - S, 0\} + \max\{8(ab - \frac{S-4ab}{8}), 0\}$. This can be reformulated as a constrained optimization problem as Eq.(17).

$$\begin{aligned} \max \quad & \rho = \frac{2\sqrt{a^3 b^3}}{X_{max} + \min\{-\mathcal{R}_1(S) + \mathcal{T}_2(S), -\mathcal{R}_2(S) + \mathcal{T}_1(S) + \mathcal{T}_3(S)\}} \\ \text{s.t.} \quad & 4ab \leq S, 4ab \leq X_{max}, S \leq X_{max} \end{aligned} \quad (17)$$

The maximum density is $\rho_{max} = \frac{\sqrt{S}}{8}$. Combined with the conclusion of $Q \geq \frac{V}{\rho_{max}}$, we can get the I/O lower bound $Q = \frac{2 \cdot 2N^2 T}{\rho_{max}} = \frac{32N^2 T}{\sqrt{S}}$. Compared with $\frac{12N^2 T}{\sqrt{S}}$ proposed by the X -Partition theory, it is improved by $\frac{8}{3}$ times. Additionally, we list I/O lower bounds for other stencil operators in Table 2.

5 I/O Lower Bound Analysis for QR Decomposition

QR decomposition refers to decomposing the matrix into a regular orthogonal matrix Q and an upper triangular matrix R . QR decomposition based on

Kernel	I/O Lower Bound	Improv.	Kernel	I/O Lower Bound	Improv.
fdtd2d	$\frac{4\sqrt{3}N_x N_y T}{\sqrt{S}}$	2	heat3d	$\frac{6\sqrt[3]{9}N^3 T}{\sqrt[3]{S}}$	$\sqrt[3]{9}$
adi	$\frac{32N^2 T}{\sqrt{S}}$	$\frac{8}{3}$	jacobi1d	$\frac{4NT}{S}$	2
jacobi2d	$\frac{8N^2 T}{\sqrt{S}}$	2	seidel2d	$\frac{8\sqrt{2}N^2 T}{\sqrt{S}}$	$2\sqrt{2}$

Table 2. The leading term of the I/O lower bound of the stencil operator extracted using (X_1, X_2) -Partition is compared with previous work [9].

Householder transformation is a commonly used implementation. For household matrices, some researchers have proposed more efficient storage schemes. In the following, we show a Householder transform-based QR decomposition algorithm for Householder matrices based on the compact WY representation [12]. The matrix Q can be obtained by $Q = I + YTY^T$. The algorithm can be divided into four main steps.

- Step 1: $v_k = \frac{\omega}{\|\omega\|_2}$.
- Step 2: $A'_{k:m,k:n} = -2v_k v_k^T A_{k:m,k:n}$.
- Step 3: $A_{k:m,k:n} = A_{k:m,k:n} + A'_{k:m,k:n}$.
- Step 4: $Z = -2TY^T v_k$. $Y = [Y \ v_k]$. $T = \begin{bmatrix} T & Z \\ 0 & -2 \end{bmatrix}$ ($k > 1$).

We estimate the sizes of the sub-computational domain $|\mathcal{H}_{max}|$ and the global computational domain $|V|$ using a multi-variable constrained optimization approach in Appendix B. We found that with an input of $(X_{max}, 0, 0, 0)$ across four steps, the sub-computation domain achieves its maximum size. Utilizing these findings, we calculate $|V|$ and substitute them into Eq.(2) to obtain $\mathcal{H}(X_{max})$.

Next, we calculate the five key sets. If the size of the fast memory space S is large enough to store all the data needed for the next k -th step of sub-computation, then $\mathcal{R}_1(S) = \mathcal{R}_2(S) \leq \max_k \{\lceil mn - (k-1)(n - \frac{k-1}{2}) \rceil\} = mn$. We consider that the size of the fast storage is S , so we can conclude that $\mathcal{R}_1(S) = \mathcal{R}_2(S) = \min\{S, mn\}$.

By definition, $\mathcal{T}_1(S)$ represents the minimum number of blue nodes transitioning to red nodes during a subcomputation. The input necessitates a total of $mn - (k-1)(n - \frac{k-1}{2})$ elements. The minimal condition for $\mathcal{T}_1(S)$ occurs when fast memory usage is maximized, ensuring all data is utilized for the subsequent subcomputation, thus $\mathcal{T}_1(S) = \min_k \{\max\{\lfloor mn - (k-1)(n - \frac{k-1}{2}) - S \rfloor, 0\}\}$.

We analyze $\mathcal{T}_2(S)$ and $\mathcal{T}_3(S)$ in detail, examining each step individually. Initially, a vector of size $(m - k + 1)$ and $2(m - k)$ intermediate variables are generated. Taking inputs into account, the maximum efficient storage capacity with fast storage is limited to $mn - (k-1)(n - \frac{k-1}{2})$. Unused fast memory space is designated as $S_{re}^0 = S - (mn - (k-1)(n - \frac{k-1}{2}))$. To minimize writeback operations, data is preferentially stored in this cache space. The minimum data to

be written out in the first step is $\mathcal{T}_2^1(S, k) = \max\{3(m-k)+1-\max\{\lceil S_{re}^0 \rceil, 0\}, 0\}$. The minimum data to be written out in the first step is $\mathcal{T}_2^1(S, k) = \max\{3(m-k)+1-\max\{\lceil S_{re}^0 \rceil, 0\}, 0\}$. Let $S_{rem}^1 = S_{re}^0 - (3m-3k+1)$ represent the residual fast memory capacity after writing intermediate variables and the final result. $S_{re}^1 = S_{re}^0 - (m-k+1)$ represents the amount of redundancy after removing data in fast memory that can be reused in the future. Reusable data amount is $\mathcal{T}_3^1(S, k) = \max\{(m-k+1) - \max\{\lceil S_{re}^0 \rceil, 0\}, 0\}$.

In the second step, a matrix of size $(m-k+1)(n-k+1)$ is generated, along with $(4m-4k+1)(n-k+1)$ intermediate variables. Taking inputs into account, the maximum available space with fast storage is $S_{re}^2 = S_{re}^1 - (m-k+1)(n-k+1)$. The residual fast memory capacity when intermediate variables and the final result are written is $S_{rem}^2 = S_{rem}^1 - (5m-5k+2)(n-k+1)$. Based on these definitions, $\mathcal{T}_2^2(S, k) = \max\{(5m-5k+2)(n-k+1) - \max\{\lceil S_{rem}^1 \rceil, 0\}, 0\}$ and $\mathcal{T}_3^2(S, k) = \max\{(m-k+1)(n-k+1) - \max\{\lceil S_{re}^1 \rceil, 0\}, 0\}$ can be derived.

In the third step, without generating new variables or intermediate variables, writeback ideally occurs in the memory space occupied by the second step. Writeback is feasible only when S is insufficient to store both $A'_{k:m,k:n}$ and $A_{k:m,k:n}$, where $A_{k:m,k:n}$ is reusable data. Following previous definitions, $S_{rem}^3 = S_{rem}^2$. Based on these definitions, $\mathcal{T}_2^3(S, k) = \max\{2(m-k+1)(n-k+1) - S, 0\}$ and $\mathcal{T}_3^3(S, k) = \max\{(m-k+1)(n-k+1) - S, 0\}$ can be derived.

The fourth step focuses on updating existing variables and producing $2mk$ intermediate variables. Given the limited capacity of fast memory S , it may not accommodate all $mn - k(n - \frac{k}{2})$ outputs. Additionally, $m+n+1$ data is written back but will not be reused. Based on these definitions, $\mathcal{T}_2^4(S, k) = \max\{\lceil mn - k(n - \frac{k}{2}) - S \rceil, 0\} + \max\{2mk - \lceil S_{rem}^3 \rceil, 0\} + (m+n+1)$, $\mathcal{T}_3^4(S, k) = \max\{\lceil mn - k(n - \frac{k}{2}) - S \rceil, 0\}$.

Summarizing the above results, we can get the formula of $\mathcal{T}_2(S)$ as $\mathcal{T}_2(S, k) = \sum_{i=1}^4 \mathcal{T}_2^i(S, k)$ and $\mathcal{T}_3(S)$ as $\mathcal{T}_3(S, k) = \sum_{i=1}^4 \mathcal{T}_3^i(S, k)$. We use matrices of different sizes and fast memories of different sizes to perform theoretical derivation and analyze the results.

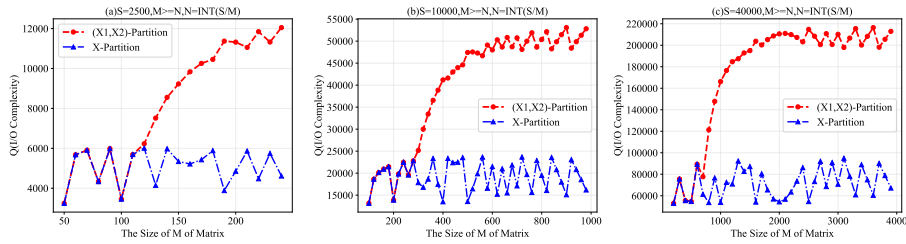


Fig. 2. The I/O lower bounds of QR decomposition in various cases are analyzed by using the method based on (X_1, X_2) -Partition we proposed. The figure shows the advantages of this approach over the method based on X -Partition. M and N are the number of rows and columns of the matrix respectively.

M	N	S	Improv.	M	N	S	Improv.	M	N	S	Improv.
100	25	2500	1.00	200	50	10000	1.00	400	100	40000	1.00
150	16	2500	1.73	300	33	10000	1.41	800	50	40000	2.26
190	13	2500	2.93	400	25	10000	3.05	1200	33	40000	2.60
200	12	2500	2.33	500	20	10000	3.50	1500	26	40000	2.24
220	11	2500	2.65	700	14	10000	2.96	2000	20	40000	3.87
240	10	2500	2.61	800	12	10000	2.58	2500	16	40000	3.93

Table 3. This table shows the improvement in I/O lower bound for QR decomposition using the analysis method proposed in this article compared to using the method based on X -Partition theory in some cases of Fig. 2.

Fig. 2(a)-(c) and Table 3 demonstrate the (X_1, X_2) -Partition’s superiority over X -Partition when $N = \lfloor S/M \rfloor$ and $M \geq N$. Initially, the curve is fully consistent with the X -Partition theory within a certain range, which also indicates that the fast memory capacity is sufficient at this time. However, as M rises, the impact of I/O operations on intermediate results becomes evident, favoring the (X_1, X_2) -Partition theory. Specifically, for $S = 2500$, $S = 10000$, and $S = 40000$, the later curve increases by a factor of 2-4, peaking at 2.93, 3.50, and 3.93, respectively. This is because the length of the calculation vector in the first step gradually increases, so the I/O overhead becomes more prominent. Obviously, the larger the fast memory, the more obvious the effect.

It shows that the theoretical method we proposed can better explore the impact of potential I/O behavior on I/O performance than previous theories, thereby making the estimate of the lower bound tighter. Therefore, this theoretical method is more suitable as a criterion for evaluating program I/O performance. In the next section, we introduce it into the neural network to search for a neural network architecture with better I/O performance.

6 I/O Lower Bound Analysis for Network Architecture Search

In addition to the above work, we also introduced the I/O lower bound theory into the graph neural network and used it to search for neural network structure configurations with excellent I/O performance and accuracy. We describe the network design in Appendix C in detail. Based on this design, we conducted some simple preliminary tests on the CIFAR10 and CIFAR100 datasets.

Table 4 presents the experimental results. The data reveals that when incorporating the I/O lower bound theory as the loss term, the prediction model exhibits a slight decrease of 1.87% in accuracy on the CIFAR100 dataset but a notable increase of 4.36% on the CIFAR10 dataset, creating a balanced outcome. Furthermore, the model utilizing I/O lower bound theory demonstrates clear advantages in model parameter size, memory usage, as well as memory reading and writing capabilities on both datasets. In terms of the floating-point operations required for a single image, the improved model is only 68.53% of the

	Dataset	Accuracy	Params	Memory	FLOPs	MemR+W
GCN	CIFAR100	55.56%	2414720	13.13MB	117.59MFlops	35.51MB
GCN+IOLB	CIFAR100	53.69%	1693440	9.38MB	80.59MFlops	25.26MB
GCN	CIFAR10	81.37%	3266240	21.10MB	167.61MFlops	54.69MB
GCN+IOLB	CIFAR10	85.73%	2652480	13.13MB	143.27MFlops	36.42MB

Table 4. Performance comparison of whether to use I/O lower bound theory (IOLB) in graph convolutional networks (GCN) to search network structures on CIFAR10 and CIFAR100. The memory required for node inference is referred to as Memory. MemR+W denotes the aggregate size of data being read from and write into memory. FLOPs is floating point operations required for inference of an image. All models are trained for 100 epochs and feature four convolutional layers with the same generated structure, having output channel counts of 64, 128, 256, and 512. A linear layer is added for classification.

original model on CIFAR100 (80.59/117.59), and 85.48% of the original model on CIFAR10 (143.27/167.61). These findings suggest that by incorporating the I/O lower bound theory as a loss term, we can devise a model that not only maintains comparable accuracy but also boasts smaller parameter size and superior I/O performance. Due to the limitation of the paper length, there is still room for further exploration, which is also the direction of future efforts.

7 Conclusion

This paper introduces the (X_1, X_2) -Partition theory, which offers tighter I/O lower bounds by analyzing subcomputations’ potential I/O behavior. We showcase its superiority by calculating the stencil operator’s I/O lower bound. Furthermore, we present an I/O lower bound analysis for algorithms with compound subcomputations, illustrated using the QR decomposition to demonstrate its advantages over prior theories. In addition, we preliminarily introduced this theory into neural network architecture search and achieved satisfactory results.

Currently, the automated application of this method remains unexplored. Our future plans involve using this theoretical framework as a performance benchmark and integrating it with machine learning and reinforcement learning to develop automated analysis tools. This integrated strategy will offer crucial insights for selecting operators, algorithms, and critical performance parameters of neural networks, ultimately facilitating the optimal design of dataflows.

Acknowledgment

The work is supported by the National Key Research and Development Program of China (No.2021YFB0300101, No.2023YFA1011704) and the National Natural Science Foundation of China (No.12102468, No.12002380) and the National University of Defense Technology Foundation (No.ZK21-02, No.ZK20-52).

References

1. Ballard, G., Demmel, J., Holtz, O., Schwartz, O.: Graph expansion and communication costs of fast matrix multiplication. *Journal of the ACM (JACM)* **59**(6), 1–23 (2013)
2. Elango, V., Rastello, F., Pouchet, L.N., Ramanujam, J., Sadayappan, P.: On characterizing the data movement complexity of computational dags for parallel execution. In: *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*. pp. 296–306 (2014)
3. Ballard, G., Demmel, J., Holtz, O., Lipshitz, B., Schwartz, O.: Communication-optimal parallel algorithm for strassen’s matrix multiplication. In: *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*. pp. 193–204 (2012)
4. Yan, J., Chen, X., Liu, J.: Csr&rv: An efficient value compression format for sparse matrix-vector multiplication. In: Liu, S., Wei, X. (eds.) *Network and Parallel Computing*. pp. 54–60. Springer Nature Switzerland, Cham (2022)
5. Chen, X., Gong, C., Liu, J., Pang, Y., Deng, L., Chi, L., Li, K.: A novel neural network approach for airfoil mesh quality evaluation. *Journal of Parallel and Distributed Computing* **164**, 123–132 (2022)
6. Jia-Wei, H., Kung, H.T.: I/o complexity: The red-blue pebble game. In: *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*. pp. 326—333. STOC ’81, Association for Computing Machinery, New York, NY, USA (1981). <https://doi.org/10.1145/800076.802486>
7. Olivry, A., Langou, J., Pouchet, L.N., Sadayappan, P., Rastello, F.: Automated derivation of parametric data movement lower bounds for affine programs. In: *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. pp. 808—822. PLDI 2020, Association for Computing Machinery, New York, NY, USA (2020). <https://doi.org/10.1145/3385412.3385989>
8. Zhang, X., Xiao, J., Tan, G.: I/o lower bounds for auto-tuning of convolutions in cnns. In: *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. pp. 247—261. PPOPP ’21, Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3437801.3441609>
9. Kwasniewski, G., Ben-Nun, T., Gianinazzi, L., Calotoiu, A., Schneider, T., Ziogas, A.N., Besta, M., Hoefer, T.: Pebbles, graphs, and a pinch of combinatorics: Towards tight i/o lower bounds for statically analyzable programs. In: *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*. pp. 328—339. SPAA ’21, Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3409964.3461796>
10. Jain, S., Zaharia, M.: Spectral lower bounds on the i/o complexity of computation graphs. In: *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. pp. 329–338 (2020)
11. Demaine, E.D., Liu, Q.C.: Red-blue pebble game: Complexity of computing the trade-off between cache size and memory transfers. In: *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. pp. 195–204 (2018)
12. Schreiber, R., Van Loan, C.: A storage-efficient \$wy\$ representation for products of householder transformations. *SIAM Journal on Scientific and Statistical Computing* **10**(1), 53–57 (1989). <https://doi.org/10.1137/0910005>