A Dynamic Programming Approach to Optimizing the Blocking Strategy for the Householder QR Decomposition

October 1st, 2008

iWAPT2008

Tsukuba International Congress Center,
EPOCHAL TSUKUBA, Japan

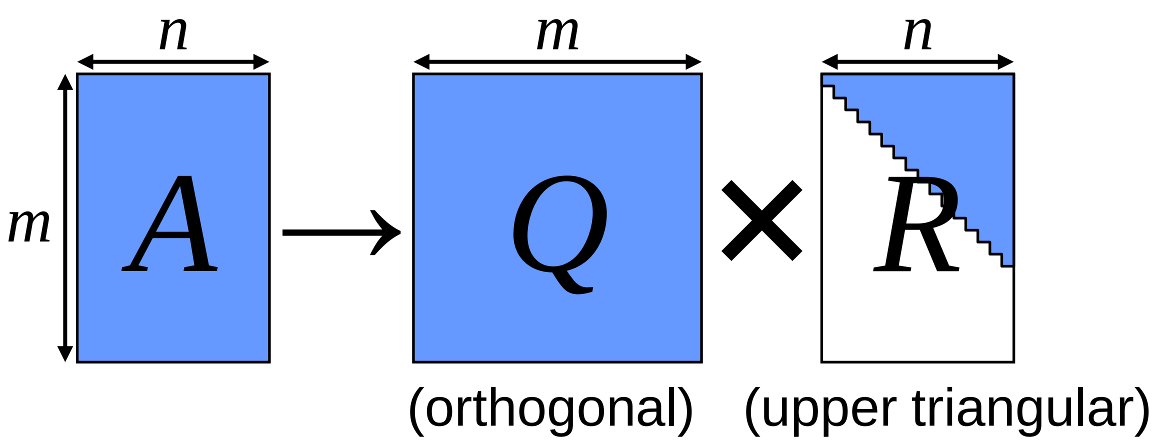
Takeshi Fukaya, Yusaku Yamamoto, Shao-Liang Zhang
(Nagoya University)

Outline

1. Introduction
2. Our Approach to Optimizing
   - ✓ Parameterization of blocking strategies
   - ✓ Proposal of an Optimization Algorithm
3. Performance Evaluation
4. Conclusion

Introduction

QR Decomposition



Algorithms

- Gram-Schmidt process
- Householder transformations **Our target**
- Givens rotations

Applications

- Least square problem
- Singular value decomposition of rectangular matrices

Householder QR Decomposition

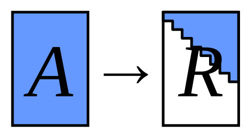
Householder transformation

$$Ha = \underbrace{\left(I - t y y^T\right)}_{\text{orthogonal matrix}} a = b$$

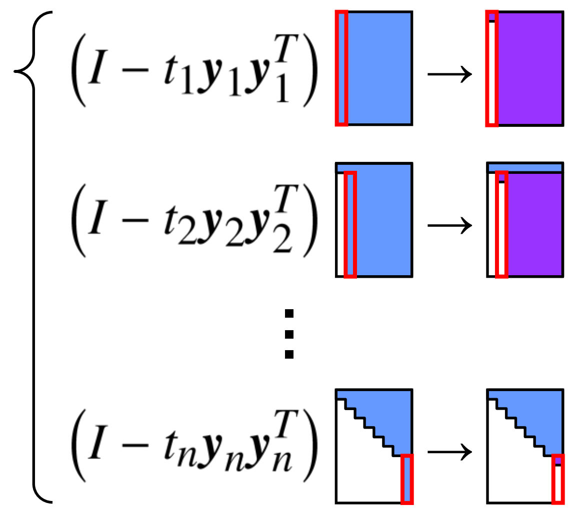
- $\|a\|_2 = \|b\|_2$
- $y = a - b$
- $t = \dfrac{2}{\|y\|_2^2}$

Householder QR decomposition

$$\underbrace{\left(I - t_n y_n y_n^T\right) \cdots \left(I - t_2 y_2 y_2^T\right)\left(I - t_1 y_1 y_1^T\right)}_{\text{product of orthogonal matrices}}$$



$$Q = \left(I - t_1 y_1 y_1^T\right)\left(I - t_2 y_2 y_2^T\right) \cdots \left(I - t_n y_n y_n^T\right)$$



Performance Bottleneck

How to compute $\left(I - t_i \boldsymbol{y}_i \boldsymbol{y}_i^T\right) A^{(i-1)} \rightarrow A^{(i)}$ **:**

- matrix-vector multiplication : $\boldsymbol{y}_i^T A^{(i-1)} \rightarrow \boldsymbol{w}^T$
- rank-1 update : $A^{(i-1)} - t_i \boldsymbol{y}_i \boldsymbol{w}^T \rightarrow A^{(i)}$

Both operations are Level-2 BLAS

BLAS (Basic Linear Algebra Subprograms)

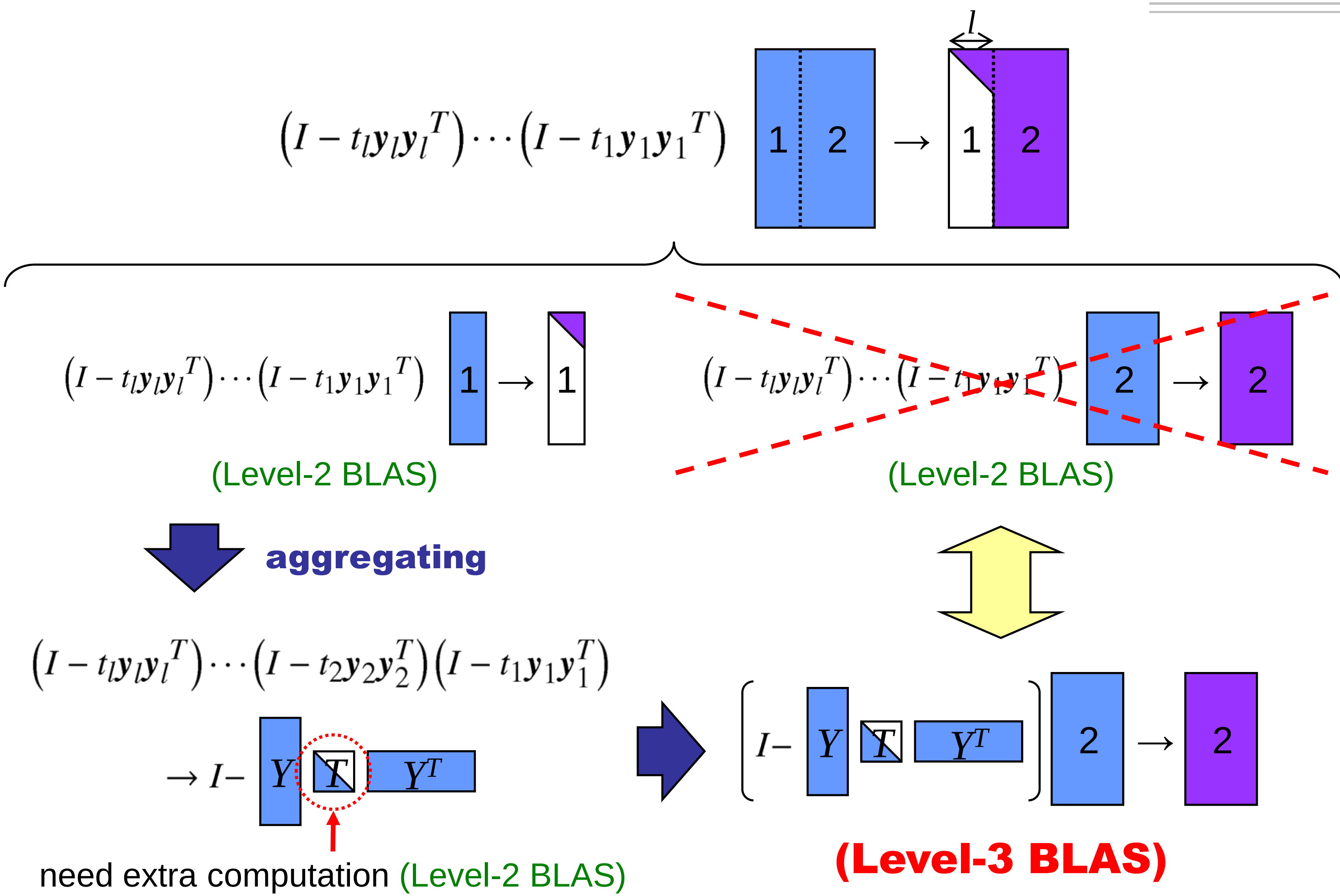
- Level-2 (matrix-vector multiplication) : $O(n^2)$ data, $O(n^2)$ FLOPs
- Level-3 (matrix multiplication) : $O(n^2)$ data, $O(n^3)$ FLOPs

Performance : Level-3 BLAS >> Level-2 BLAS

For exploiting the performance,

efficient use of Level-3 BLAS is necessary.

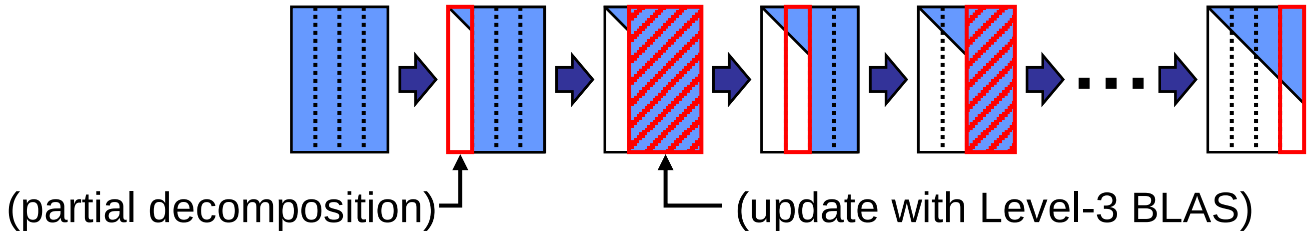
Basic Idea of Blocked Algorithms



Examples of Blocked Algorithms

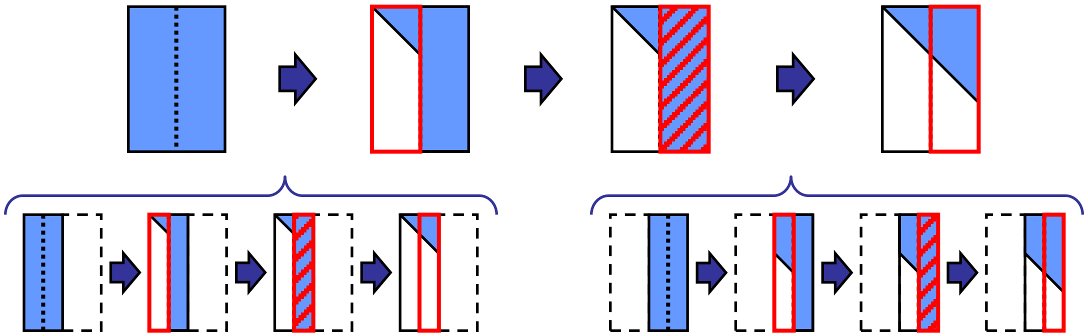
i) **Fixed-size Blocking** (used in LAPACK)

- Partition A into several blocks of the same width. **Optimal block width**
- Partial decompositions are computed with non-blocked algorithm.



ii) **Recursive Blocking** (E. Elmroth et al., 2000)

- Partition A into two blocks of the same width. **Optimal recursion level**
- Partial decompositions are computed with block algorithm recursively.



Optimization of the blocking strategy

Definition of the Blocking Strategy

- How to partition a matrix into blocks.
- How to compute the partial decomposition of the each blocks.

What is affected by the blocking strategy:

- Total amount of computational work.
- Proportion of the Level-3 BLAS operations to the total operations.
- Size of matrix in each Level-3 BLAS operation.

Computation time depends on the blocking strategy.

Optimization of the blocking strategy

- For the computational environment (CPU, BLAS library, …).
- For the size of problem (size of the target matrix).

How to optimize

Manual tuning (traditional)

- Consider the properties of the CPU (cache size, etc.)
- Use the experimental results.

Automatic tuning (our objective)

Optimization using an algorithm

1. Parameterize blocking strategies.
2. Propose an algorithm to optimize the parameters.

Parameterization of blocking strategies

Policy of parameterization

Examples of various blocking strategies

- Partition A into several (more than two) blocks recursively.
- Use blocks of unequal width. (C. Bischof et al., 1990)
- Hybrid of different blocking strategies. (E. Elmroth et al., 2000)

Policy of parameterization

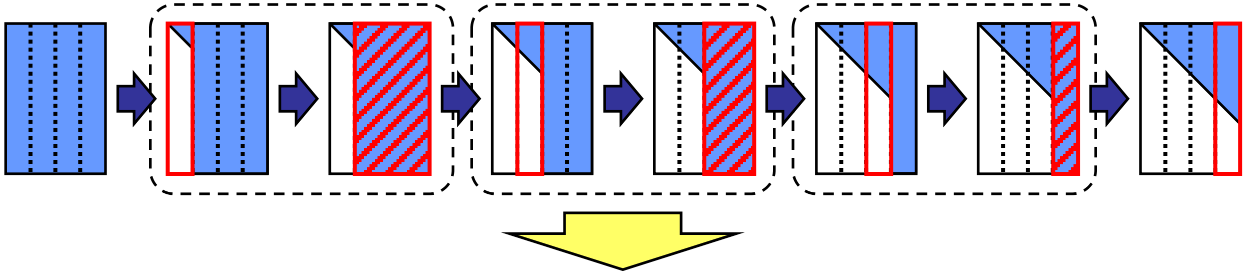
Choose parameters which are

- as simple as possible (because we have to optimize them),
- able to express as many blocking strategies as possible.

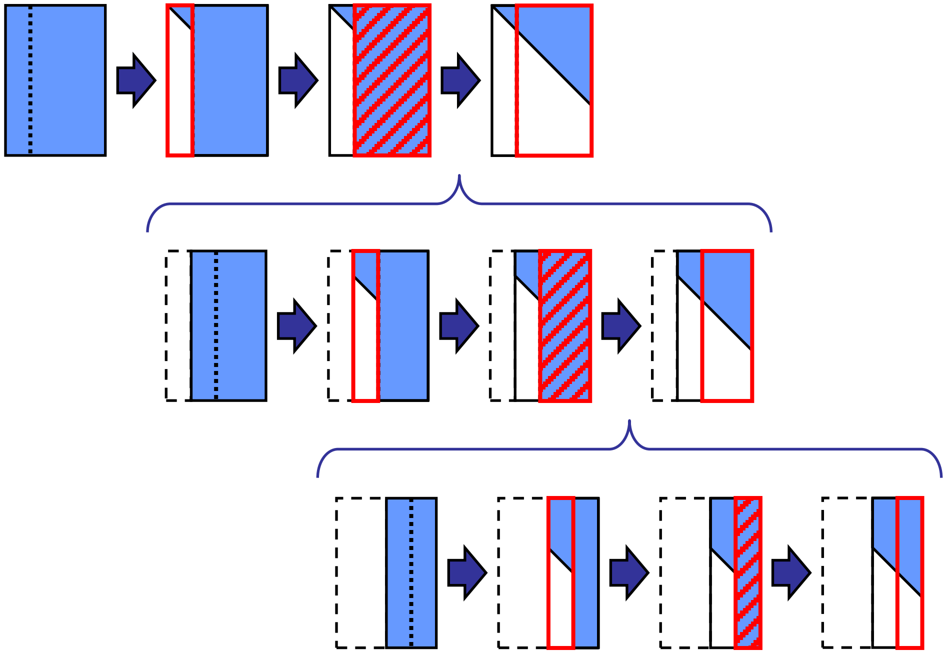
Express a complex strategy with simple parameters

Idea to express a complex strategy

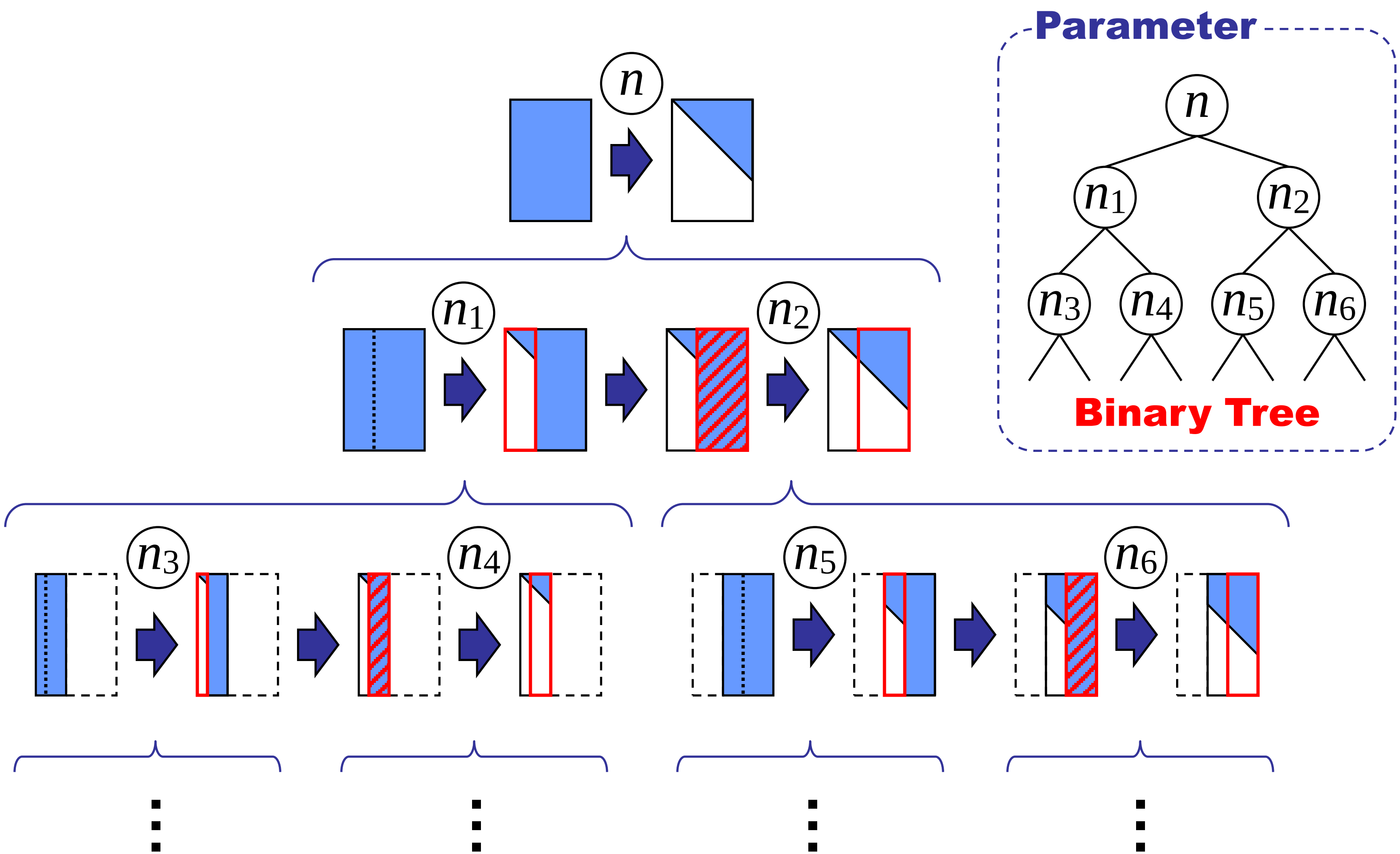
Partitioning into more than two blocks at one time



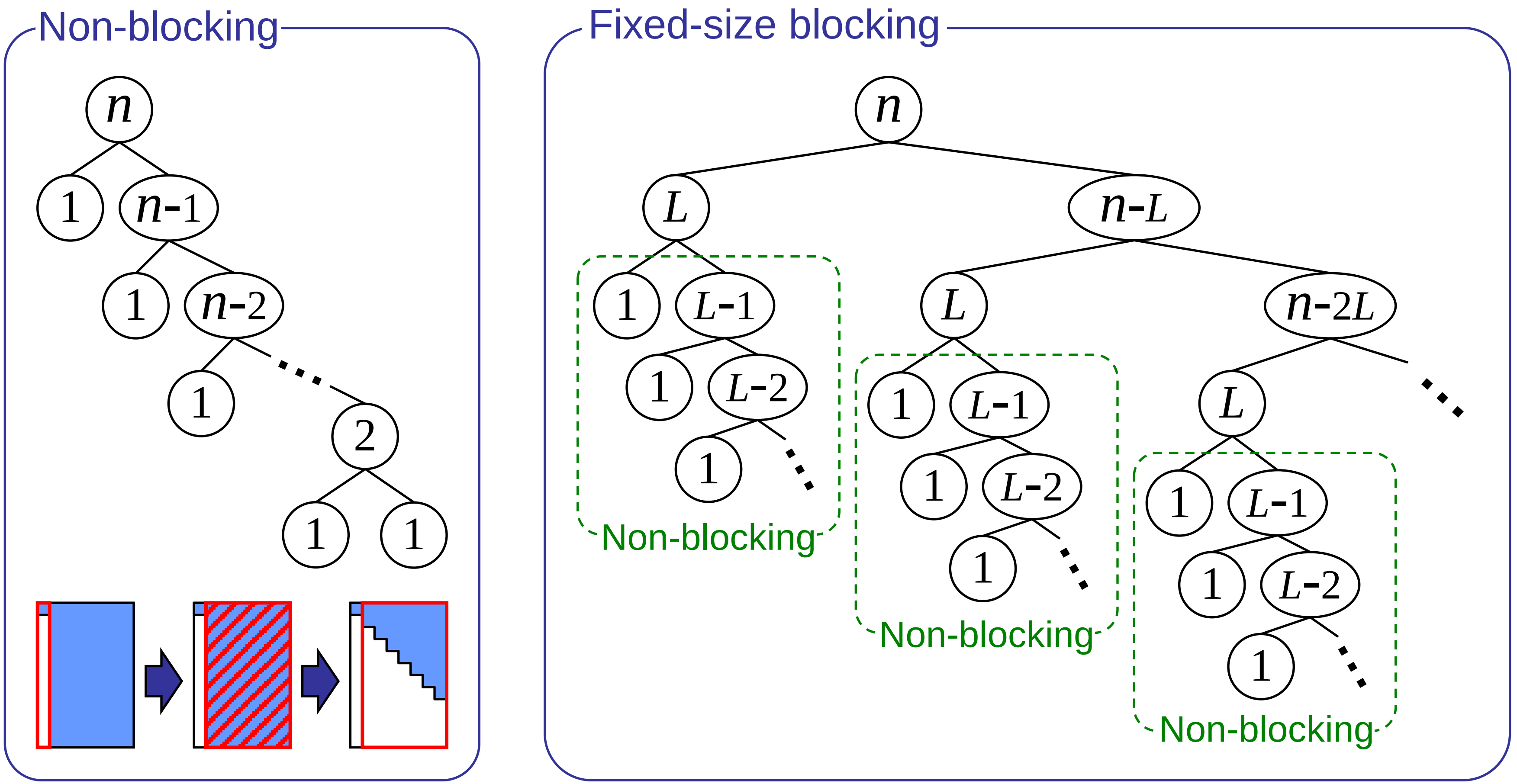
Recursion of partitioning into two blocks



Generalized Recursive Blocking



Representation with a binary tree



A blocking strategy for a matrix with n columns **one-to-one** A binary tree with n leaves

Proposal of an Optimization Algorithm

Policy of optimization

Policy of optimization

Analytical optimization is difficult.

Estimate total computation time and **search for the shortest one**.

Performance model of BLAS　　**Proposed algorithm**

Performance model of BLAS

Estimating the computation time of the BLAS operation

- Input : size of each matrix in BLAS operation

Multilinear interpolation using sampled performance data.

- Output : estimated computation time

Formulation of the optimization

Minimization of the estimated computation time

$$T_{\mathrm{QR}}^{\mathrm{best}}(m,n) = \min_{t_n \in \mathcal{T}_n} T_{\mathrm{QR}}(m,n,t_n)$$

- $\mathcal{T}_n$: a set of all the binary trees with n leaves
- t_n : an individual binary tree belonging to $\mathcal{T}_n$
- $T_{\mathrm{QR}}(m,n,t_n)$: estimated time to compute $\left(I - YTY^T\right)A = R$ with t_n

  (A is an m × n matrix, using generalized recursive blocking)

$$|\mathcal{T}_n| = \frac{2^{n-1}(2n-3)!!}{n!} >> O(2^n)$$

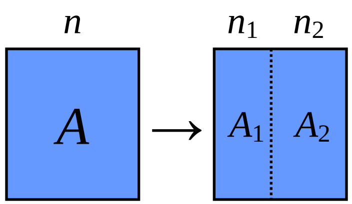
too large for exhaustive search

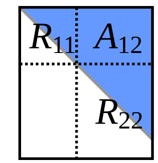
search a nearly optimal one with practical cost

Approach for optimization (1/3)

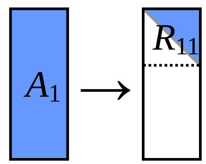
Recall : Generalized Recursive Blocking (GeRB)

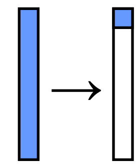
Input : an $m \times n$ matrix A ⇨ Output : Y, T and R satisfying $(I - Y T Y^T) A = R$



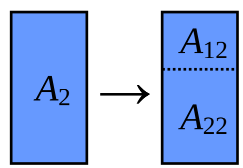
$Y = [Y_1|\bar{Y}_2]$ $\quad T = \begin{bmatrix} T_{11} & O \\ T_{21} & T_{22} \end{bmatrix}$ $\quad R =$ 

partial decomposition

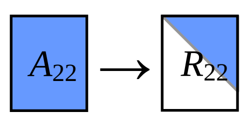
[step 1] $\left(I - Y_1 T_{11} Y_1^T\right)$ 

$\left(I - t\boldsymbol{y}\boldsymbol{y}^T\right)$  (block width = 1)

with GeRB (otherwise)

[step 2] $\left(I - Y_1 T_{11} Y_1^T\right)$ 

partial decomposition

[step 3] $\left(I - Y_2 T_{22} Y_2^T\right)$ 

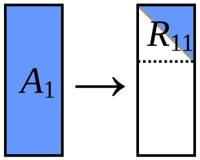
(as same way as step 1)

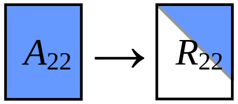
[step 4] $-T_{22}\left(\bar{Y}_2^T Y_1\right) T_{11} \rightarrow T_{21} \quad \left(\bar{Y}_2 = \begin{bmatrix} O \\ Y_2 \end{bmatrix} \right)$

Approach for optimization (2/3)

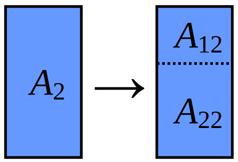
Expand T_{QR} as follows:

$$T_{\mathrm{QR}}(m,n,t_n) = T_{\mathrm{QR}}(m,n_1,\tilde{t}_{n_1}) + T_{\mathrm{QR}}(m-n_1,n_2,\tilde{t}_{n_2}) + T_{\mathrm{other}}(m,n_1,n_2)$$

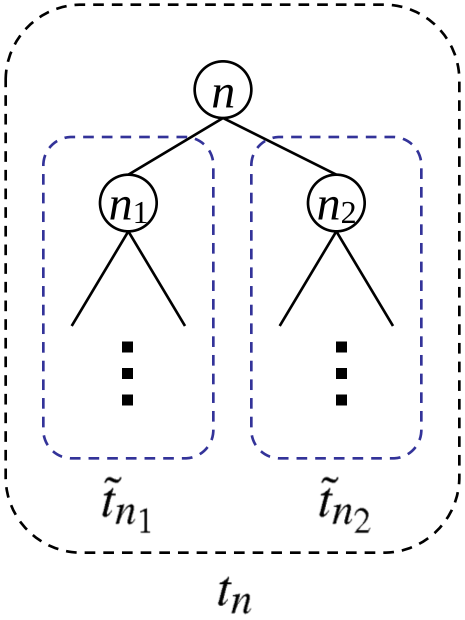
- $T_{\mathrm{QR}}(m,n_1,\tilde{t}_{n_1})$: time for $\left(I - Y_1 T_{11} {Y_1}^T\right)$ 

- $T_{\mathrm{QR}}(m-n_1,n_2,\tilde{t}_{n_2})$: time for $\left(I - Y_2 T_{22} {Y_2}^T\right)$ 

- $T_{\mathrm{other}}(m,n_1,n_2)$: time for
  estimated with the
  BLAS performance model

$\left(I - Y_1 T_{11} {Y_1}^T\right)$ 

$-T_{22}\left(\bar{Y}_2^T Y_1\right) T_{11} \rightarrow T_{21}$



Approach for optimization (3/3)

$$
\begin{aligned}
T_{\mathrm{QR}}^{\mathrm{best}}(m,n) &= \min_{t_n \in \mathcal{T}_n} T_{\mathrm{QR}}(m,n,t_n) \\
&= \min_{\substack{1 \le n_1 \le n-1 \\ (n_2 := n-n_1)}} \left[\min_{\substack{t_{n_1} \in \mathcal{T}_{n_1} \\ t_{n_2} \in \mathcal{T}_{n_2}}} \left\{ T_{\mathrm{QR}}(m,n_1,t_{n_1}) + T_{\mathrm{QR}}(m-n_1,n_2,t_{n_2}) + T_{\mathrm{other}}(m,n_1,n_2) \right\} \right] \\
&= \min_{\substack{1 \le n_1 \le n-1 \\ (n_2 := n-n_1)}} \left\{ \min_{t_{n_1} \in \mathcal{T}_{n_1}} T_{\mathrm{QR}}(m,n_1,t_{n_1}) + \min_{t_{n_2} \in \mathcal{T}_{n_2}} T_{\mathrm{QR}}(m-n_1,n_2,t_{n_2}) + T_{\mathrm{other}}(m,n_1,n_2) \right\} \\
&= \min_{\substack{1 \le n_1 \le n-1 \\ (n_2 := n-n_1)}} \left\{ T_{\mathrm{QR}}^{\mathrm{best}}(m,n_1) + T_{\mathrm{QR}}^{\mathrm{best}}(m-n_1,n_2) + T_{\mathrm{other}}(m,n_1,n_2) \right\}
\end{aligned}
$$

$$
\therefore T_{\mathrm{QR}}^{\mathrm{best}}(m,n) \simeq \min_{\substack{1 \le n_1 \le n-1 \\ (n_2 := n-n_1)}} \left\{ T_{\mathrm{QR}}^{\mathrm{best}}(m,n_1) + \frac{m-n_1}{m} T_{\mathrm{QR}}^{\mathrm{best}}(m,n_2) + T_{\mathrm{other}}(m,n_1,n_2) \right\}
$$

approximation

(Bellman equation) can be solved by dynamic programming

Algorithm with Dynamic Programming

Input: m, n

Output: I ; $I[k]$ contains the optimal value of n_1 for an $m \times k$ matrix.

$T[1] \leftarrow 0$
for $k = 2$ to n **do**
  $T[k] \leftarrow +\infty$
  for $i = 1$ to $k - 1$ **do**
    $t = T[i] + \frac{m-i}{m} T[k-i] + \underline{T_{\mathrm{other}}(m, i, k-i)}$
    if $t < T[k]$ **then**
      $T[k] \leftarrow t$
      $I[k] \leftarrow i$
    end if
  end for
end for

estimated with the performance model

execution cost

$$\sum_{k=2}^{n} (k-1) \simeq O(n^2)$$

Summary of proposed algorithm

Steps for computing the QR decomposition

1. Sampling the execution time of BLAS operations, and construct the performance model.
2. Execute the optimization algorithm, and find nearly optimal parameters.
3. Compute the QR decomposition with GeRB using parameters optimized in step2.

Cost for Optimization

- Step1 takes several hours, but we have to do it only once.
- Step2 takes several minutes, but we can obtain optimal strategies for all value of n' from 1 to n.

Our approach requires little running cost for optimization

Performance Evaluation

Evaluation Description

Test problem

- Measure the time for computing Y, T and R satisfying $\left(I - YTY^T\right)A = R.$
- Elements of A are random numbers.
- Matrix size : m, n = ($1000, 3000, 6000, m \geqq n$)

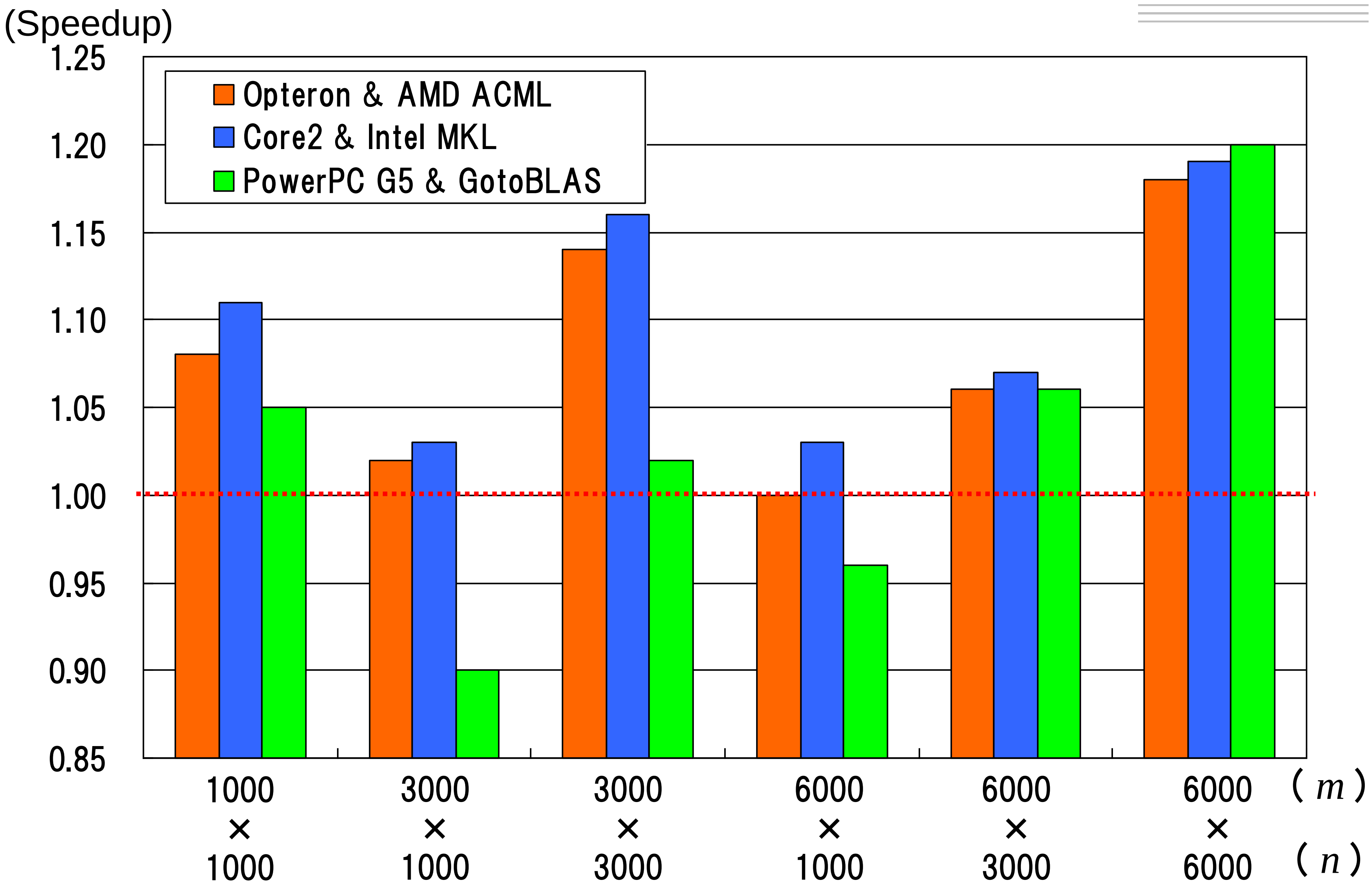
Target of evaluation

- Speedup over the (original) Recursive Blocking.
- Recursion level in original RB is optimized by manual tuning.

Computational environments

- Opteron (2.0GHz) & AMD ACML ver. 3.6.0
- Core2 (1.86GHz) & Intel MKL ver. 8.1
- PowerPC G5 (2.5GHz) & GotoBLAS ver. 1.02

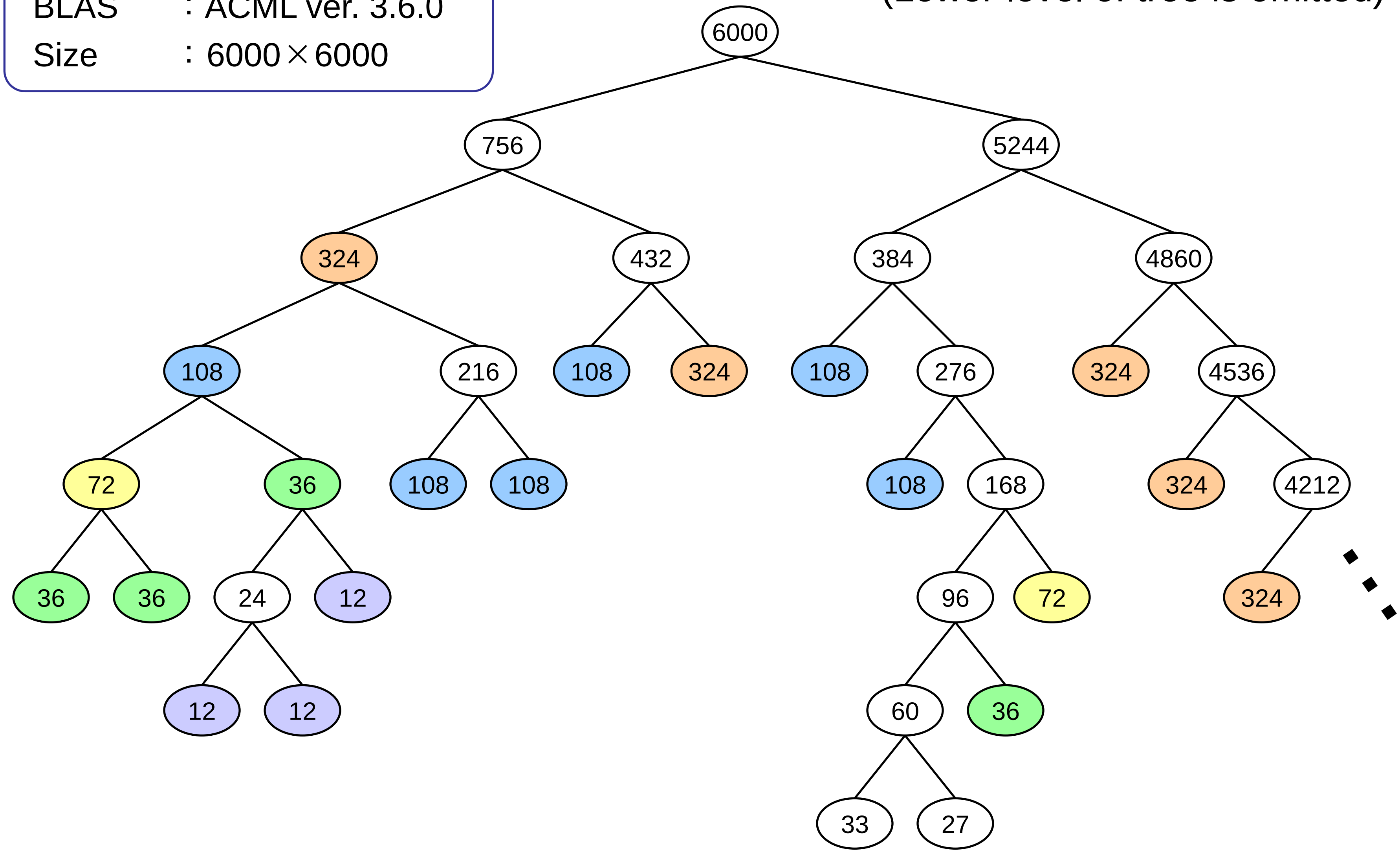
Speedup


(Speedup)
1.25
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
Opteron & AMD ACML
Core2 & Intel MKL
PowerPC G5 & GotoBLAS
1000 × 1000
3000 × 1000
3000 × 3000
6000 × 1000
6000 × 3000
6000 × 6000
(m)
(n)

Example of the optimized binary tree

CPU	:	Opteron (2.0GHz)
BLAS	:	ACML ver. 3.6.0
Size	:	6000×6000

(Lower level of tree is omitted)



Conclusion

Summary

- We aimed for the automatic tuning of block algorithms for the Householder QR decomposition.
- We showed that each blocking strategy can be represented with a binary tree.
- We proposed an algorithm for finding the optimal blocking strategy using the dynamic programming.
- The performance achieved by our approach is better than or as good as that obtained by manual tuning.

Future works

- Performance evaluation on various kinds of computational platforms (especially on novel architectures).
- Validation of our approximation used in proposed algorithm.
- Investigation into the effect of accuracy of the BLAS performance model.

Thank you for attention.