Bidiagonalization with Parallel Tiled Algorithms

Mathieu Faverge[1,2]
Julien Langou[3]
Yves Robert[2,4]
Jack Dongarra[2,5]

October 26, 2016

1: Bordeaux INP, CNRS, INRIA et Université de Bordeaux, France
2: University of Tennessee, Knoxville TN, USA
3: University of Colorado Denver, USA
4: Laboratoire LIP, École Normale Supérieure de Lyon et INRIA, France
5: University of Manchester, UK

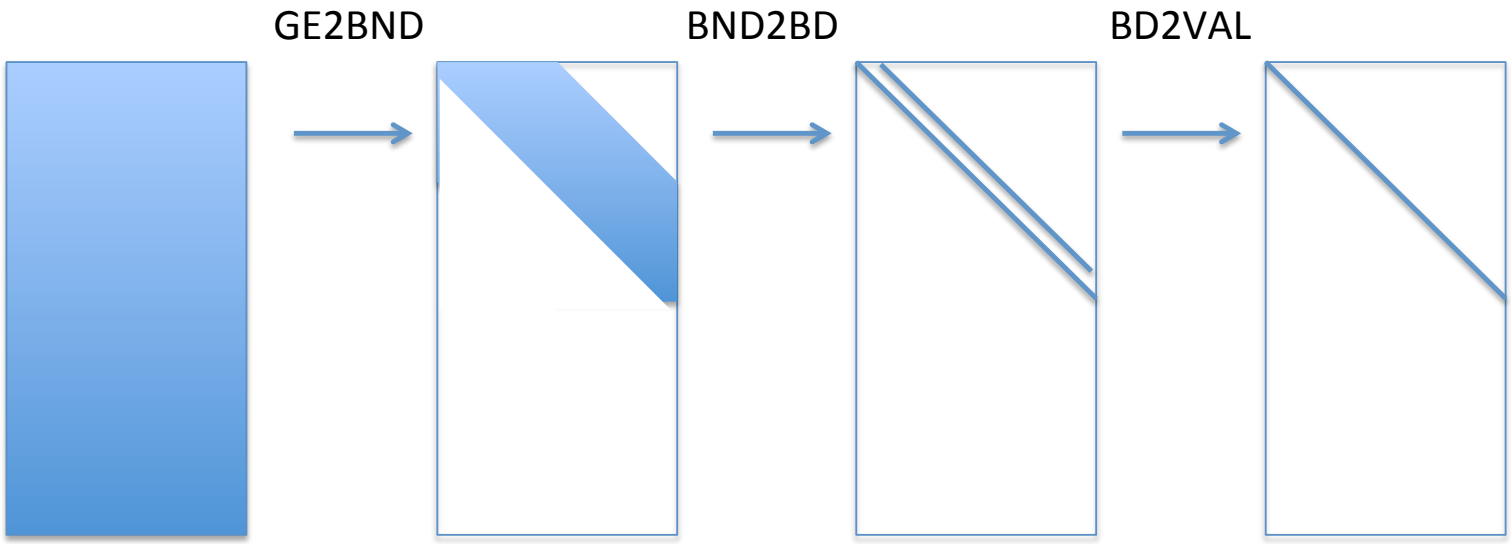
Context

- Computing singular values of matrices (GE2VAL)
- Use tiled algorithms
- Algorithms will execute over a runtime
- Consider the three-step approach:
  1. from full to band (GE2BND)
  2. from band to bidiagonal (BND2BD)
  3. from bidiagonal to singular values (BD2VAL).

  GE2VAL = GE2BND + BND2BD + BD2VAL

Three-step approach

$$\text{GE2VAL} = \text{GE2BND} + \text{BND2BD} + \text{BD2VAL}$$



Related work

B. Großer and B. Lang.
Efficient parallel reduction to bidiagonal form.
Parallel Comput., 25(8):969–986, Aug. 1999.

- Two-stage bidiagonalization (GE2BND + BND2BD) was first proposed by Großer and Lang (PARCO, 1999)
- Experiments in parallel distributed using ScaLAPACK presenting scalable algorithm and implementation of GE2BND and BND2BD

One comment

When singular vectors are requested,

One comment

When singular vectors are requested,

- The two-step approach requires more memory than “direct”

One comment

When singular vectors are requested,

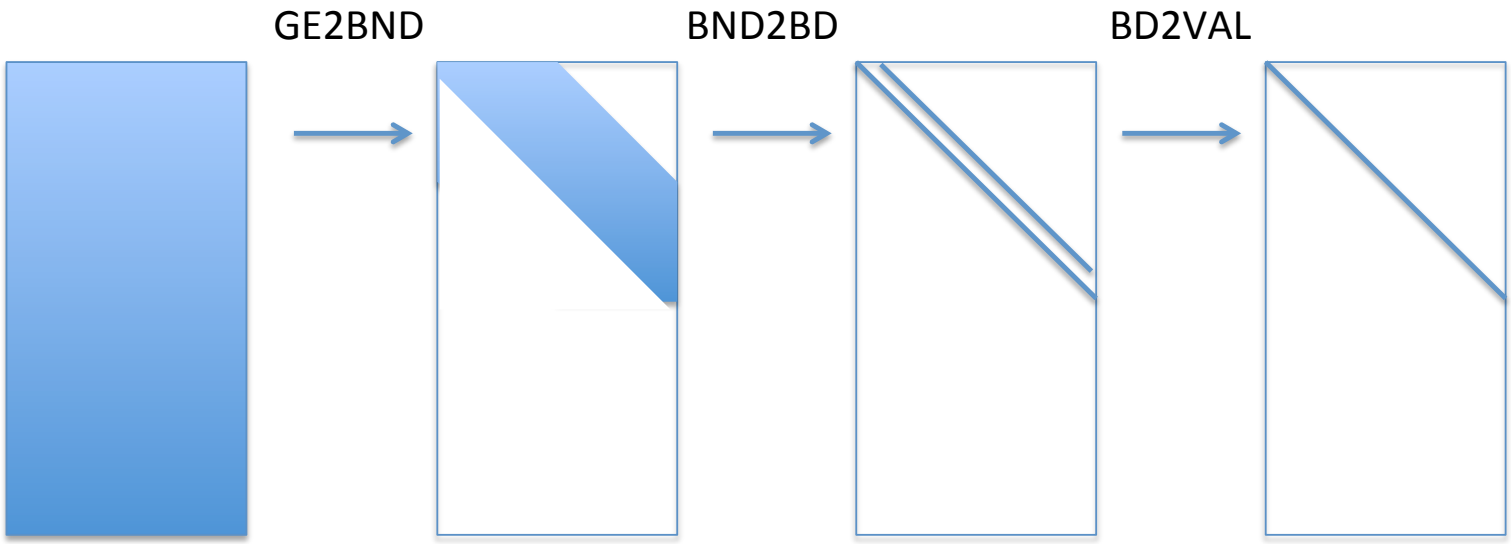
- The two-step approach requires more memory than "direct"
- The two-step approach requires more FLOPS than "direct".

For n-by-n matrix with reduction to bandwidth b:

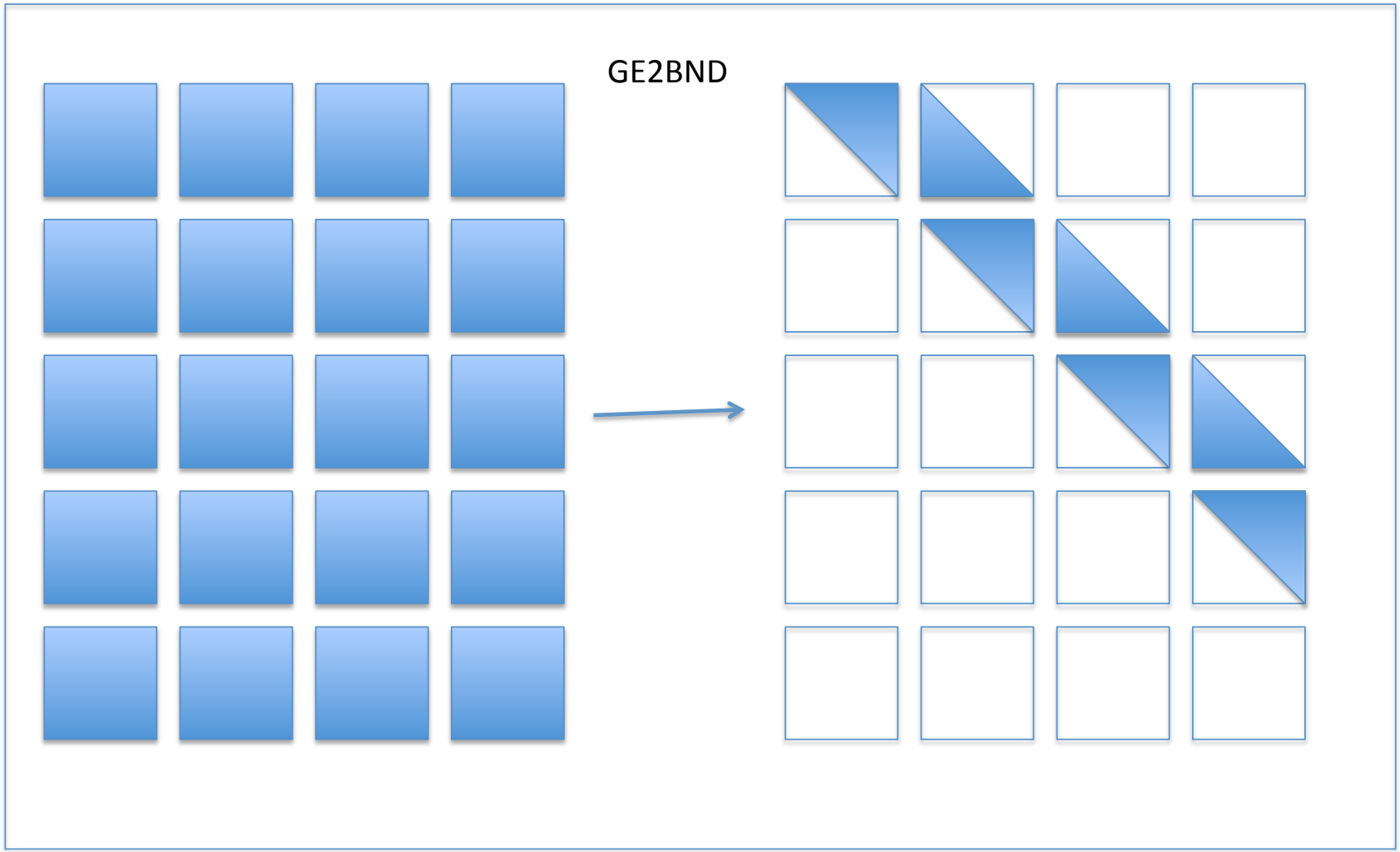
	Reduction of A	Update of U	Update of V
"direct" GE2BD	$\frac{8}{3}n^3$	$2n^3$	$2n^3$
GE2BND	$\frac{8}{3}n^3$	$2n^3$	$2n^3$
BND2BD	$8n^2b$	$2n^3$	$2n^3$

Three-step approach

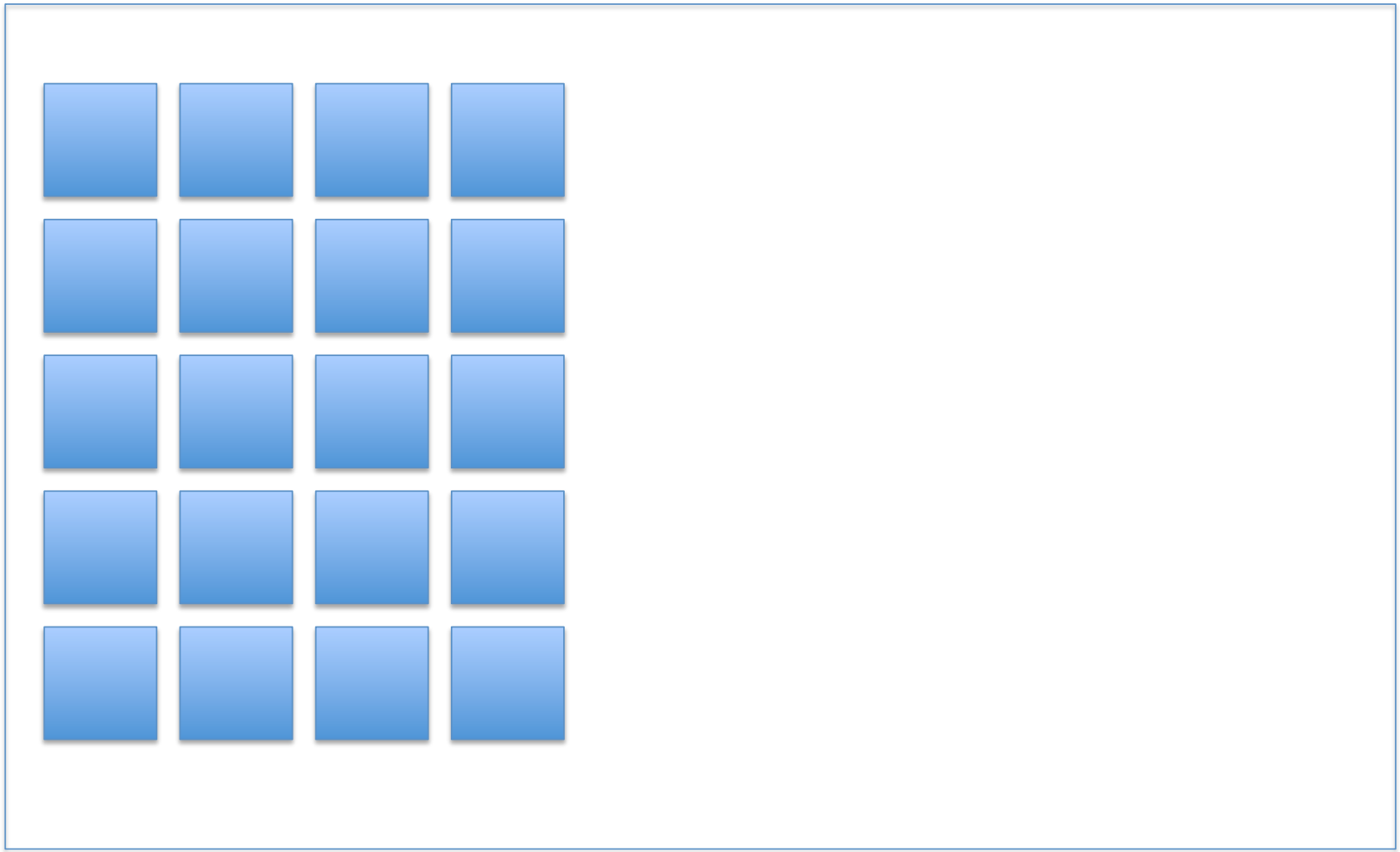
$$\text{GE2VAL} = \text{GE2BND} + \text{BND2BD} + \text{BD2VAL}$$



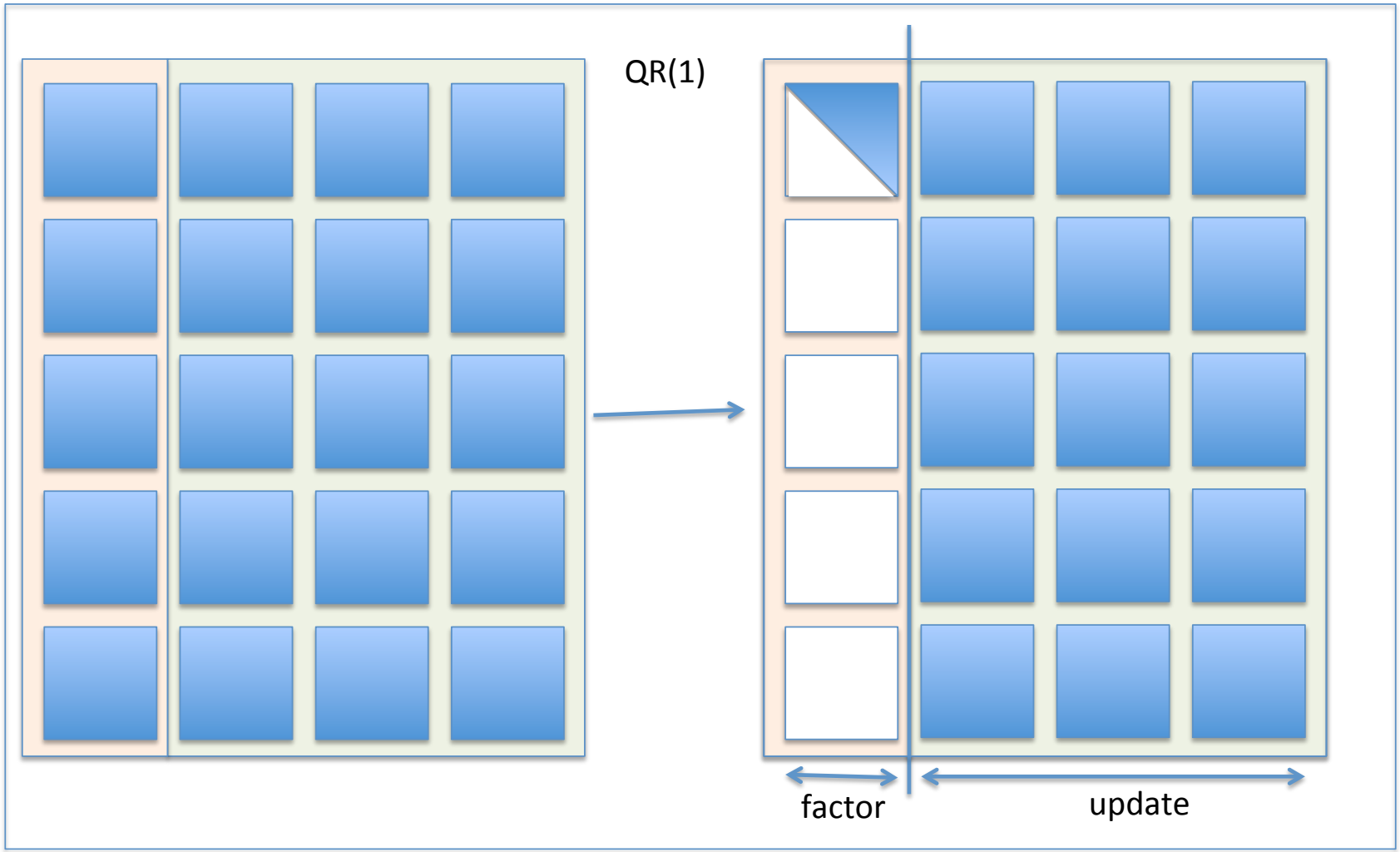
GE2BND: from full to band bidiagonal



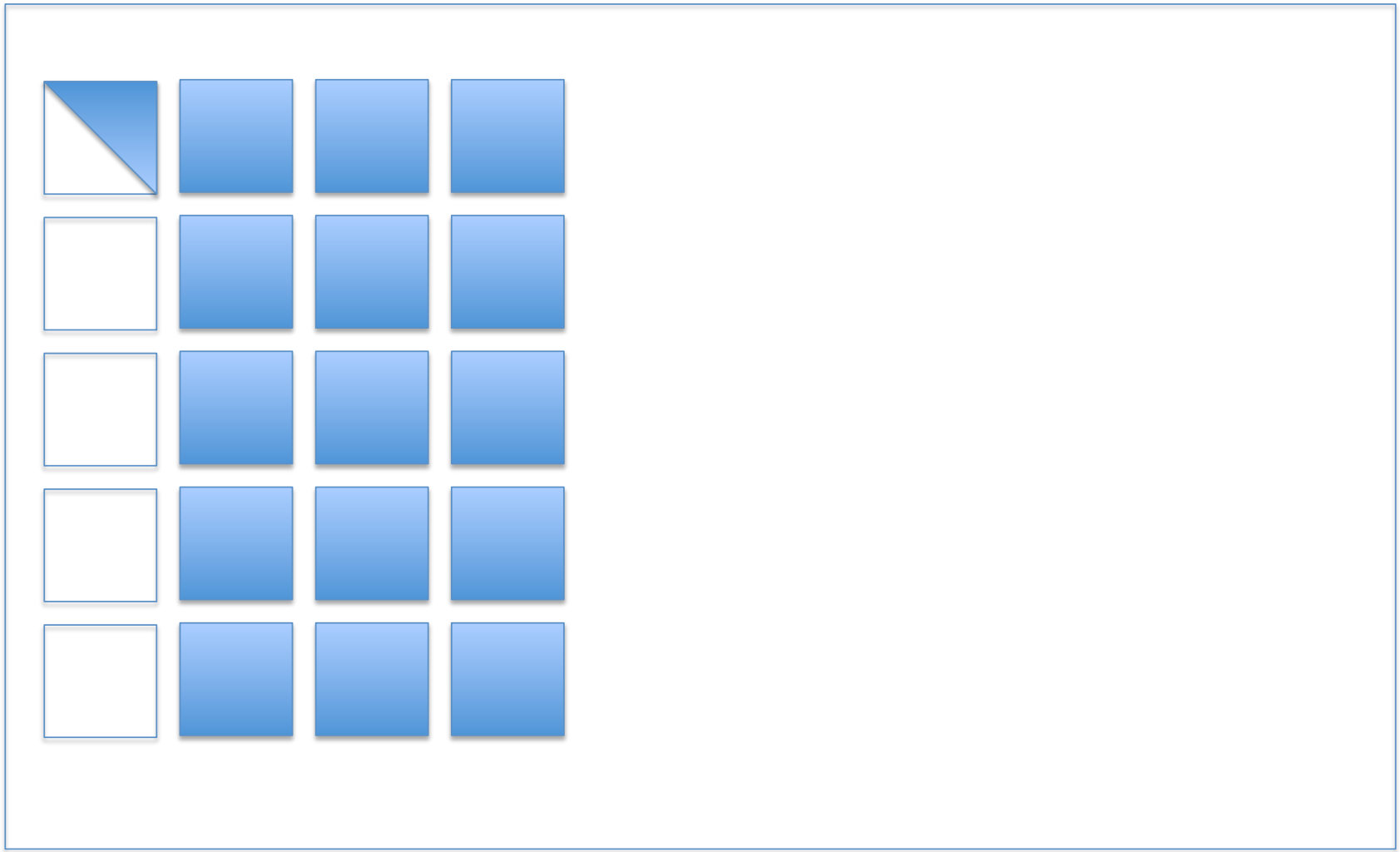
Band Tiled Bidiagonalization



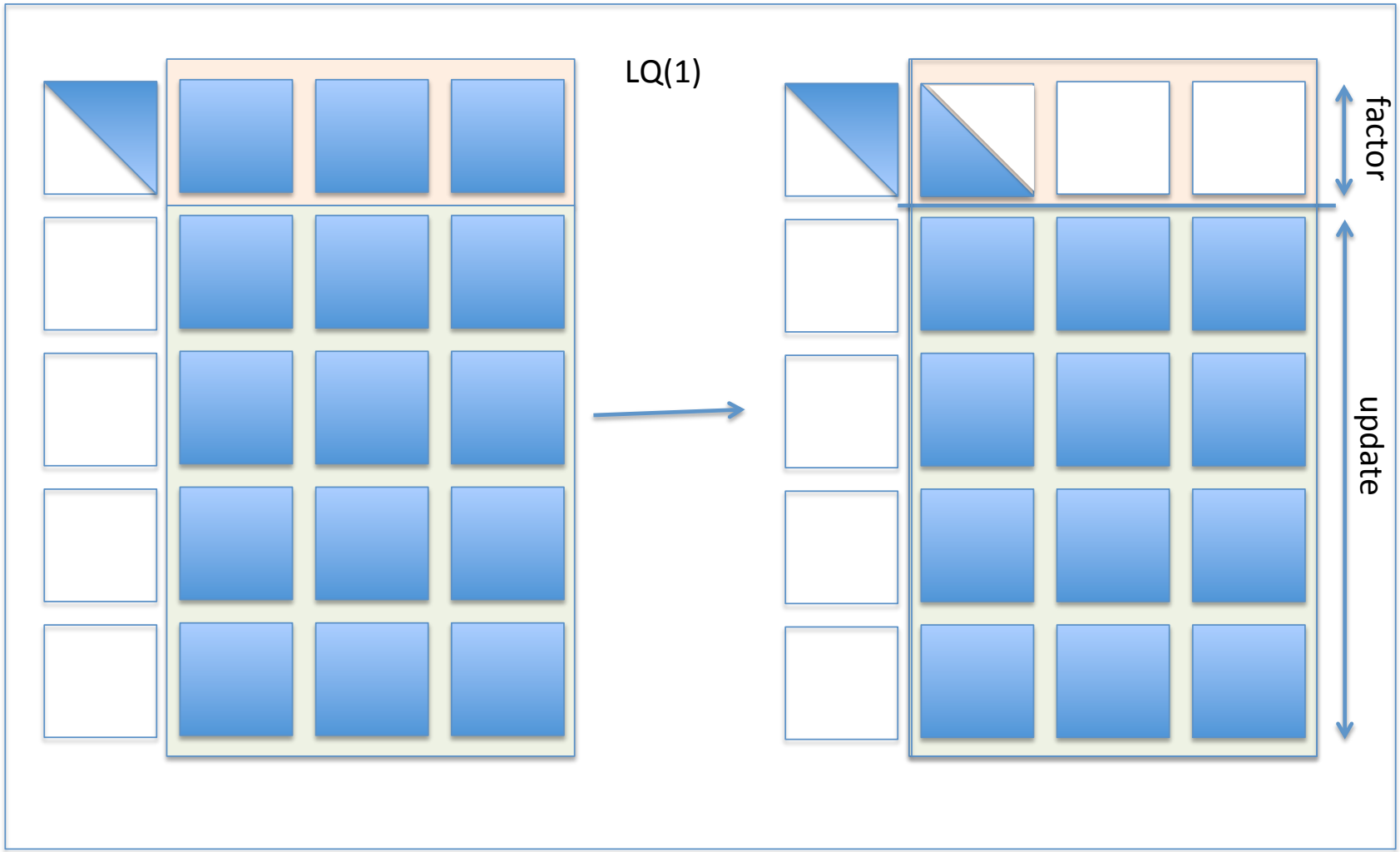
Band Tiled Bidiagonalization



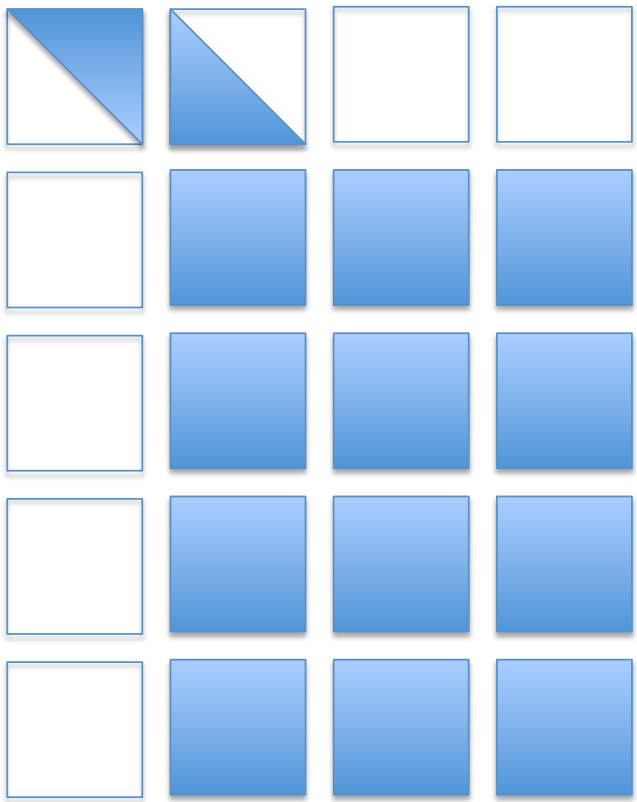
Band Tiled Bidiagonalization



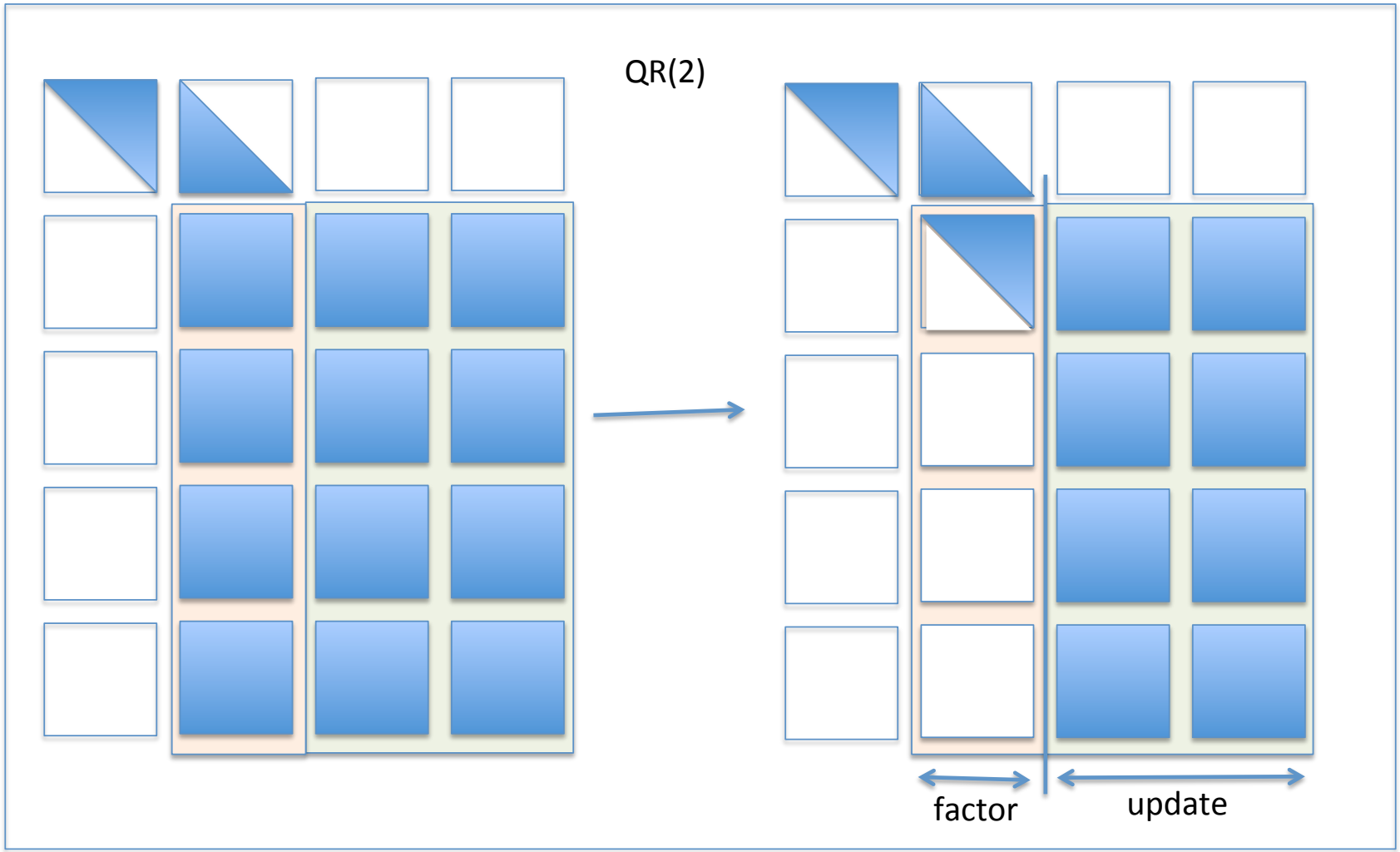
Band Tiled Bidiagonalization



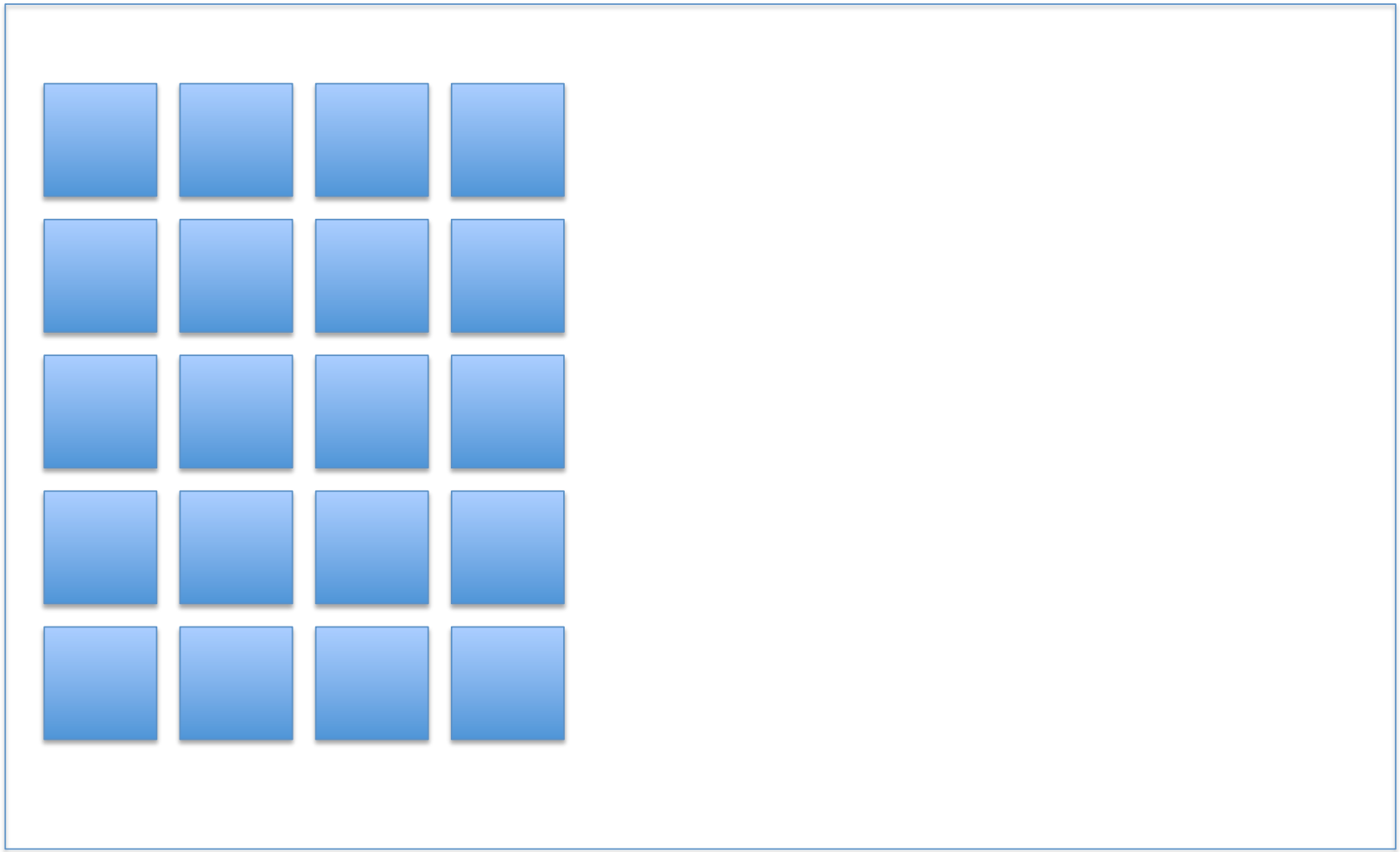
Band Tiled Bidiagonalization



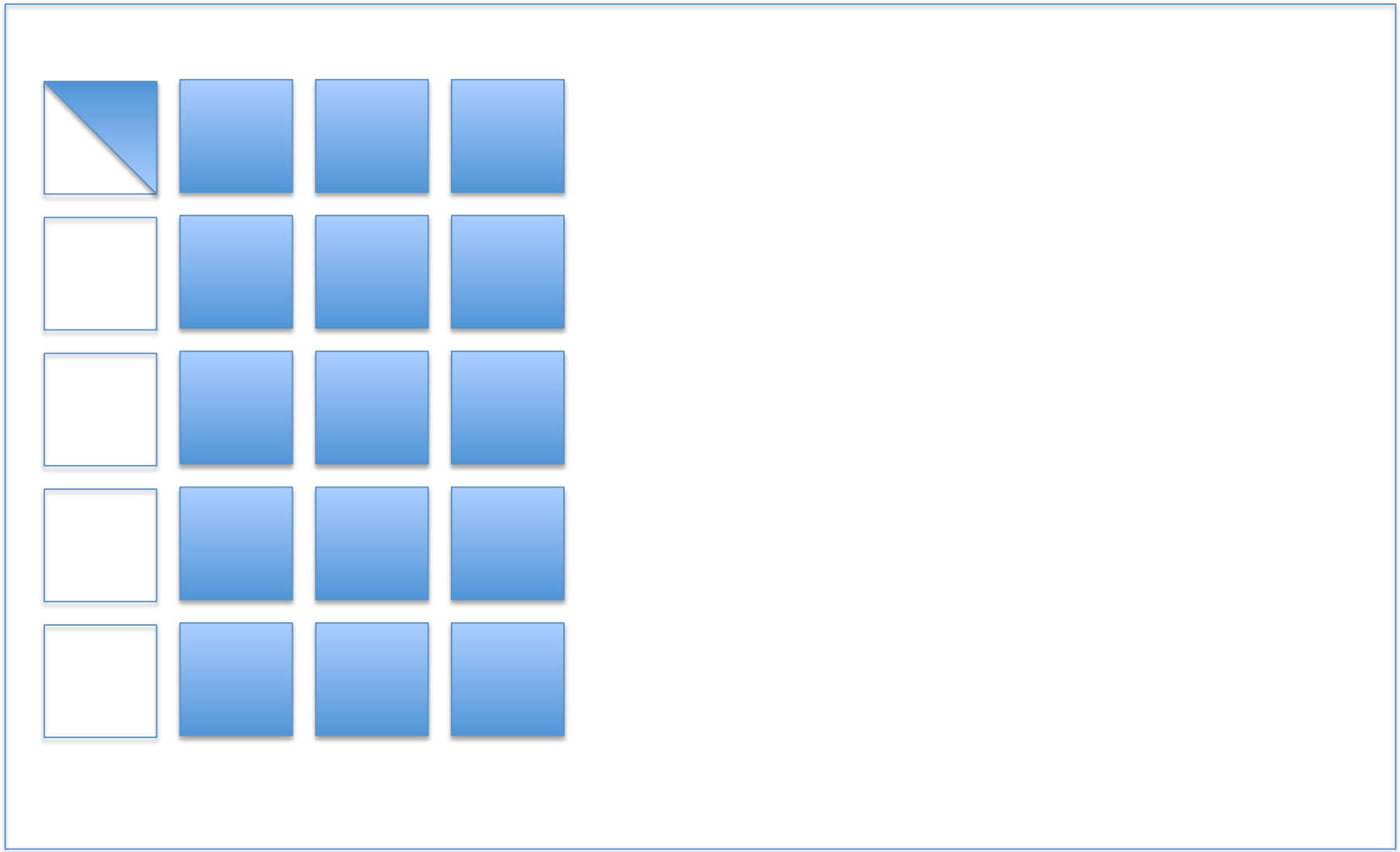
Band Tiled Bidiagonalization



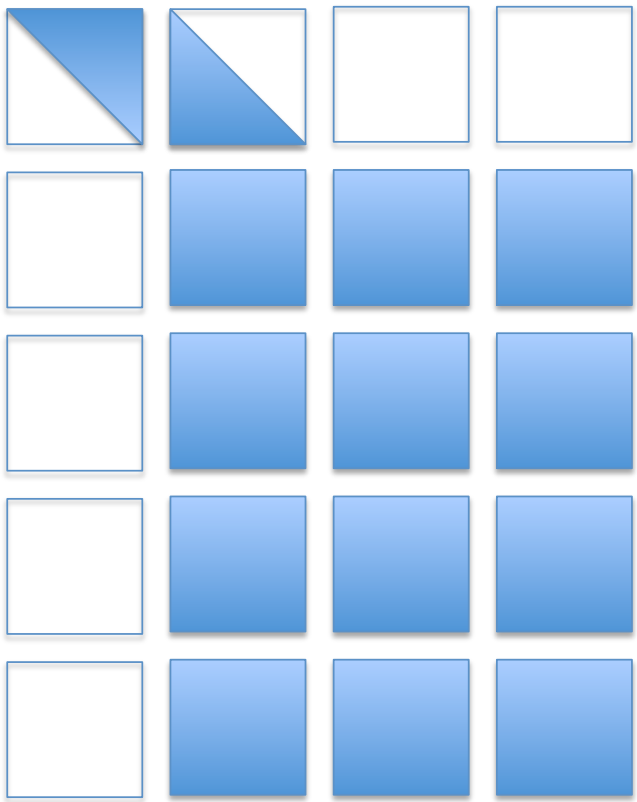
Band Tiled Bidiagonalization



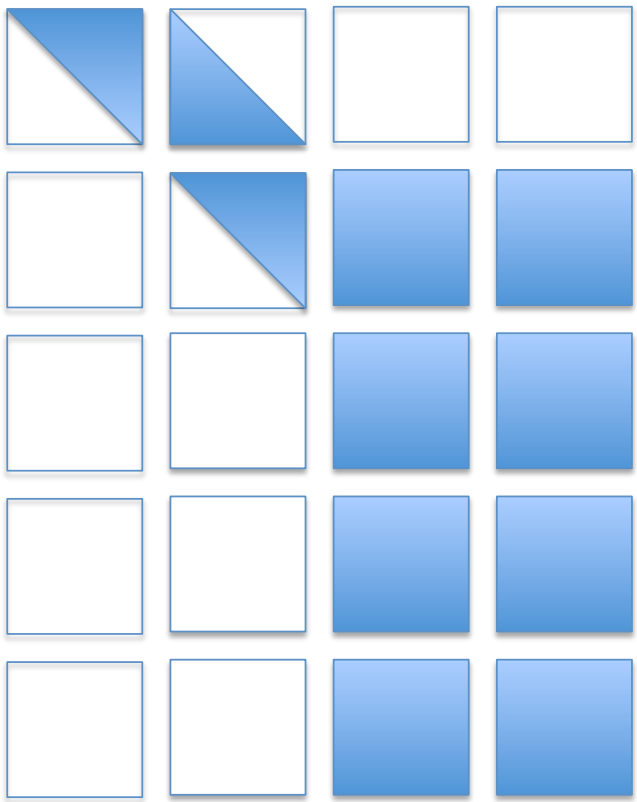
Band Tiled Bidiagonalization



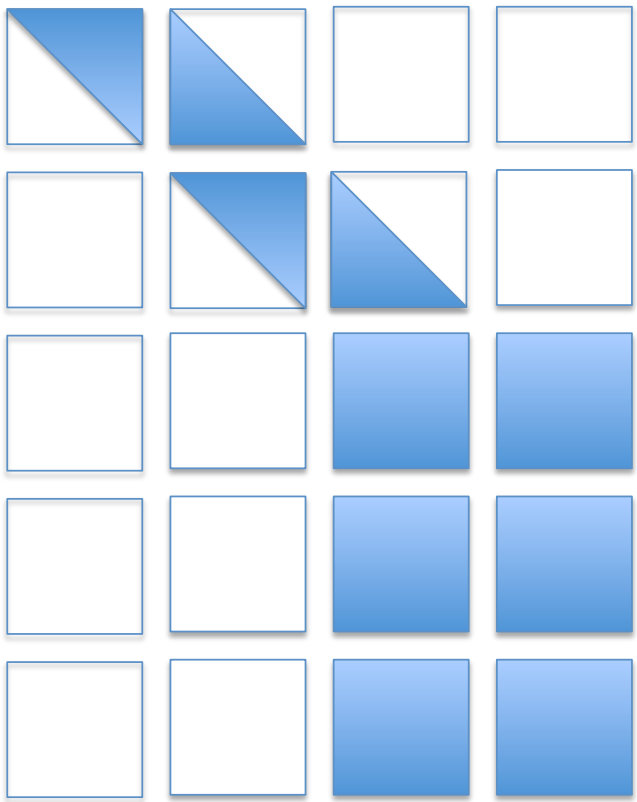
Band Tiled Bidiagonalization



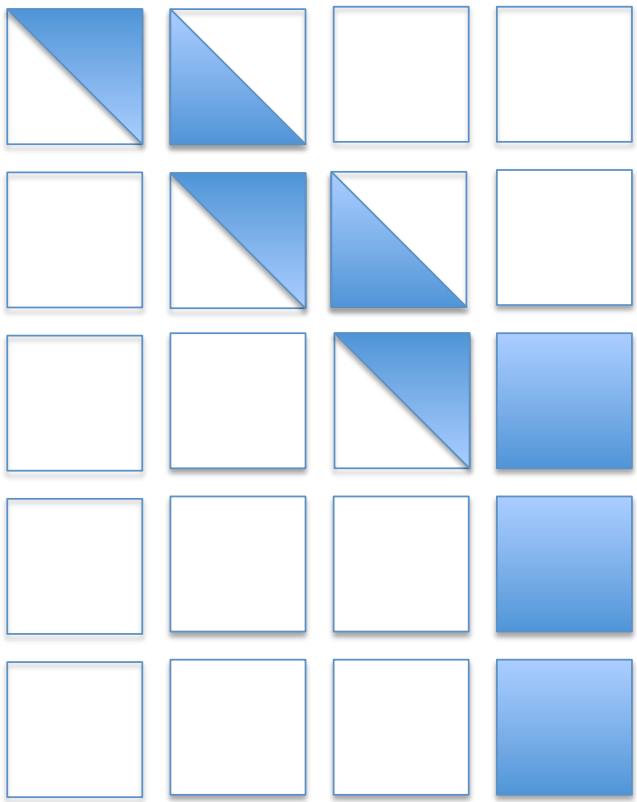
Band Tiled Bidiagonalization



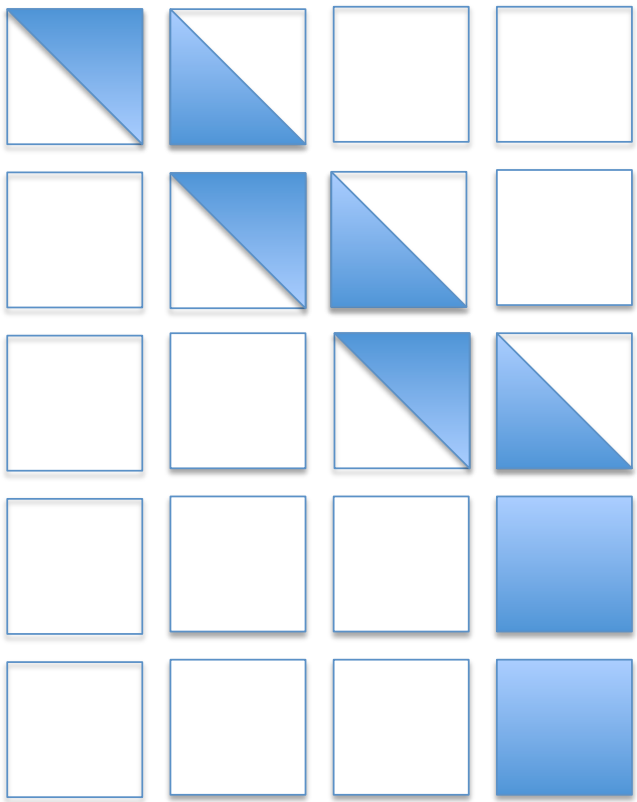
Band Tiled Bidiagonalization



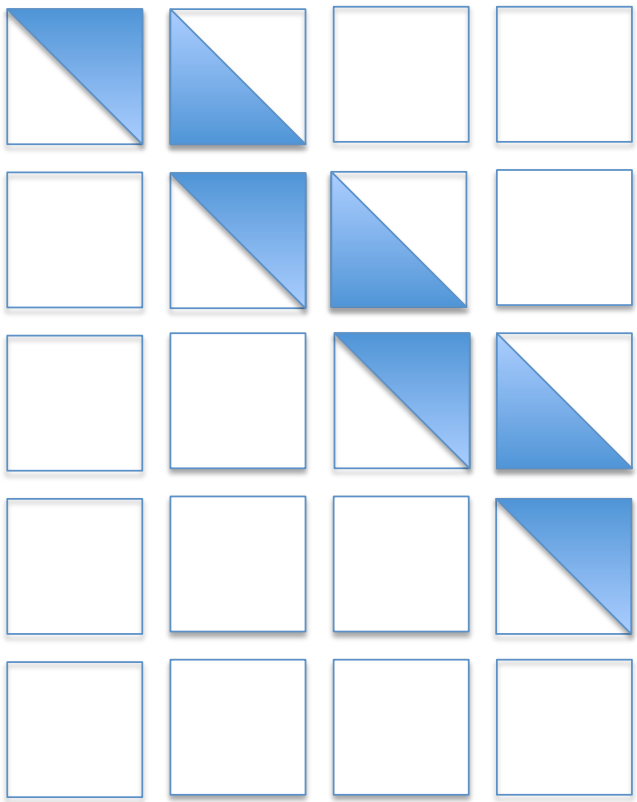
Band Tiled Bidiagonalization



Band Tiled Bidiagonalization



Band Tiled Bidiagonalization

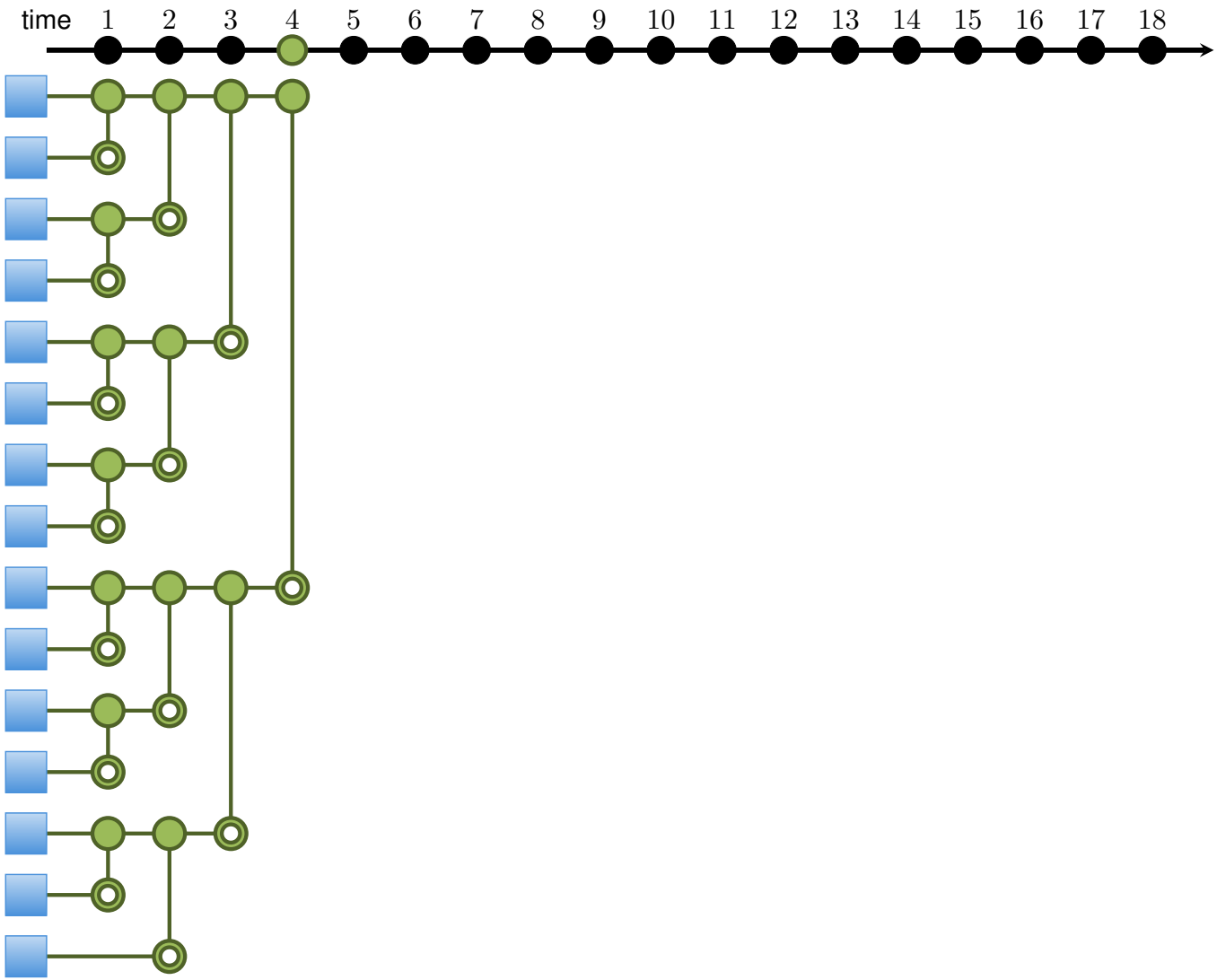


Reduction Trees for QR

- The QR and LQ steps are done using a tiled algorithm, therefore, for each step, we need to select a reduction tree.
  (Any tree will do the job; we want to choose an appropriate one given the context.)
- In previous work (2008-2014), we have studied reduction trees for QR factorization on tiled matrix.
- Reduction trees are selected in order to
  1. take into account the machine architecture, network topology, etc.
  2. take into account the matrix size and shape
  3. enable good pipelining of steps
  4. favor the use of TS kernels instead of TT kernels,

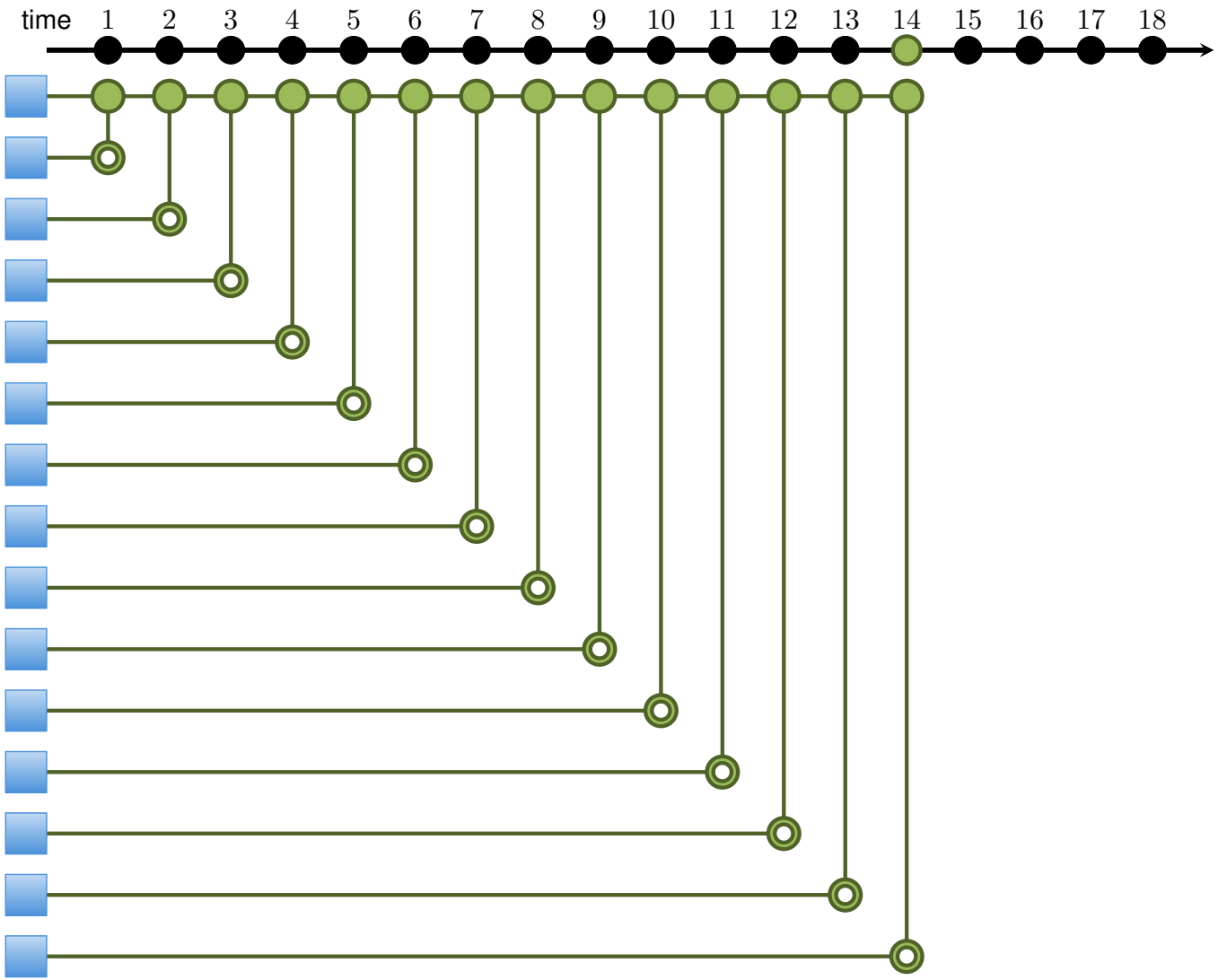
Reduction Trees for QR: binomial tree

$p = 15, q = 1$



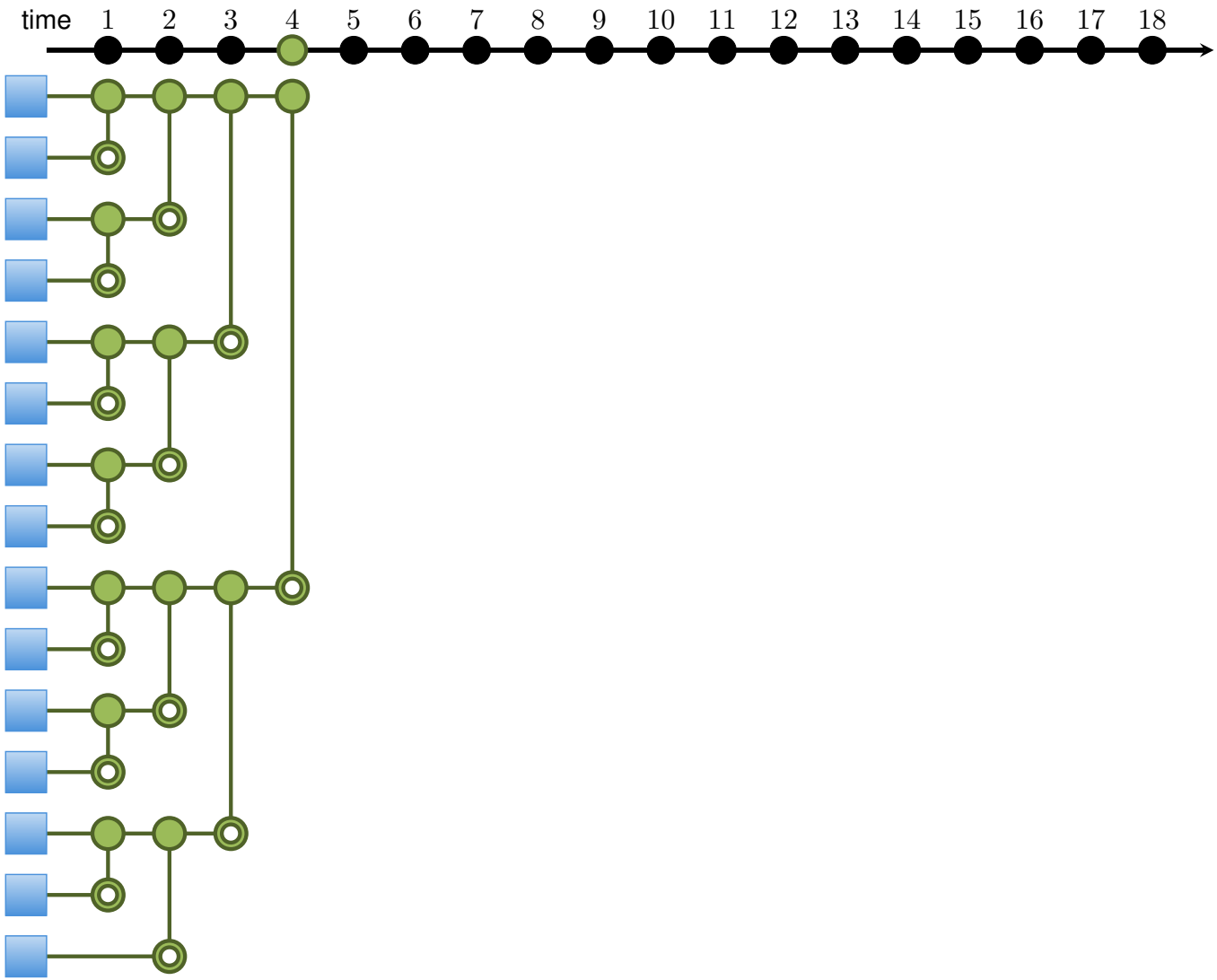
Reduction Trees for QR: flat tree

$p = 15, q = 1$



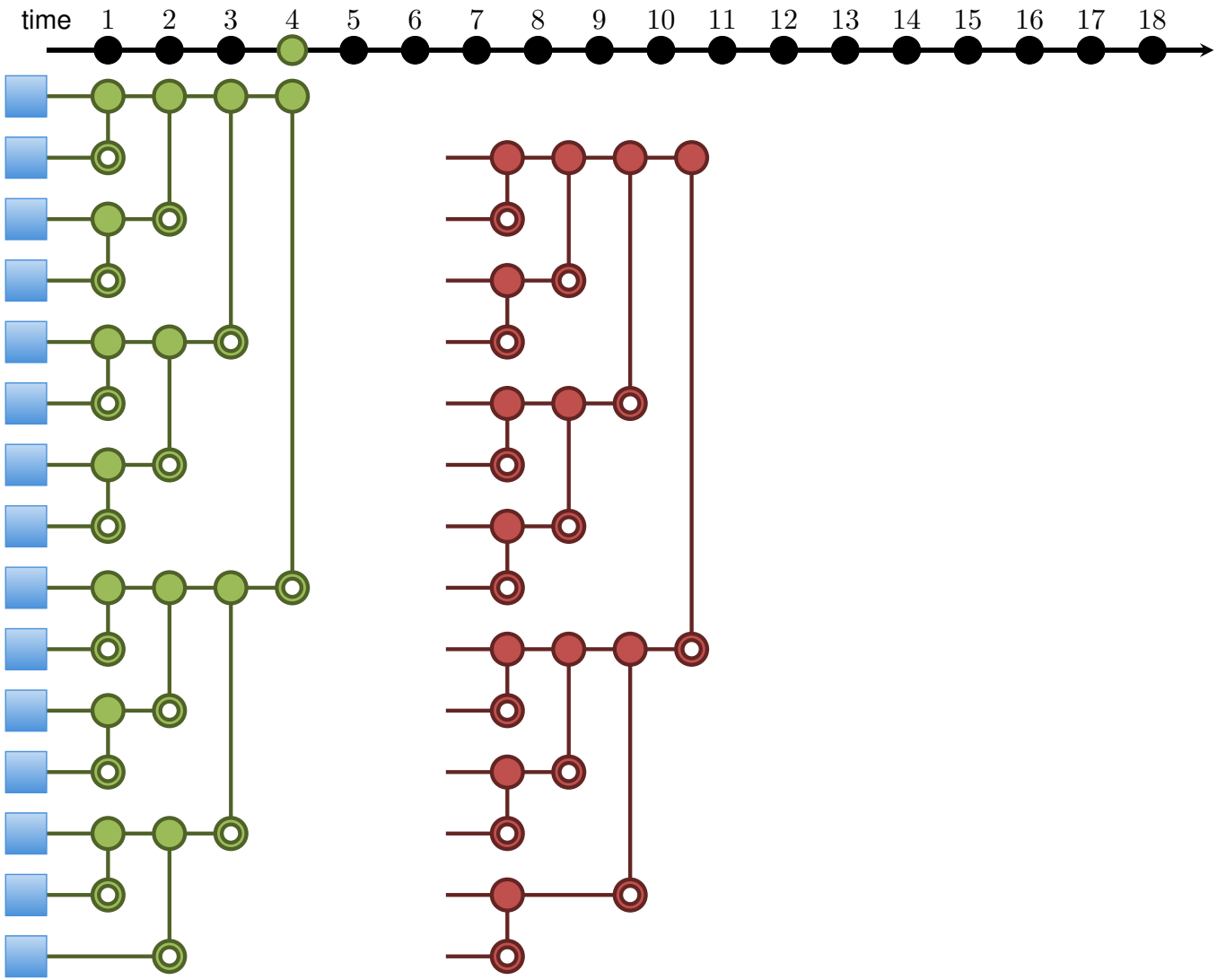
Reduction Trees for QR: binomial tree

$p = 15, q = 1$



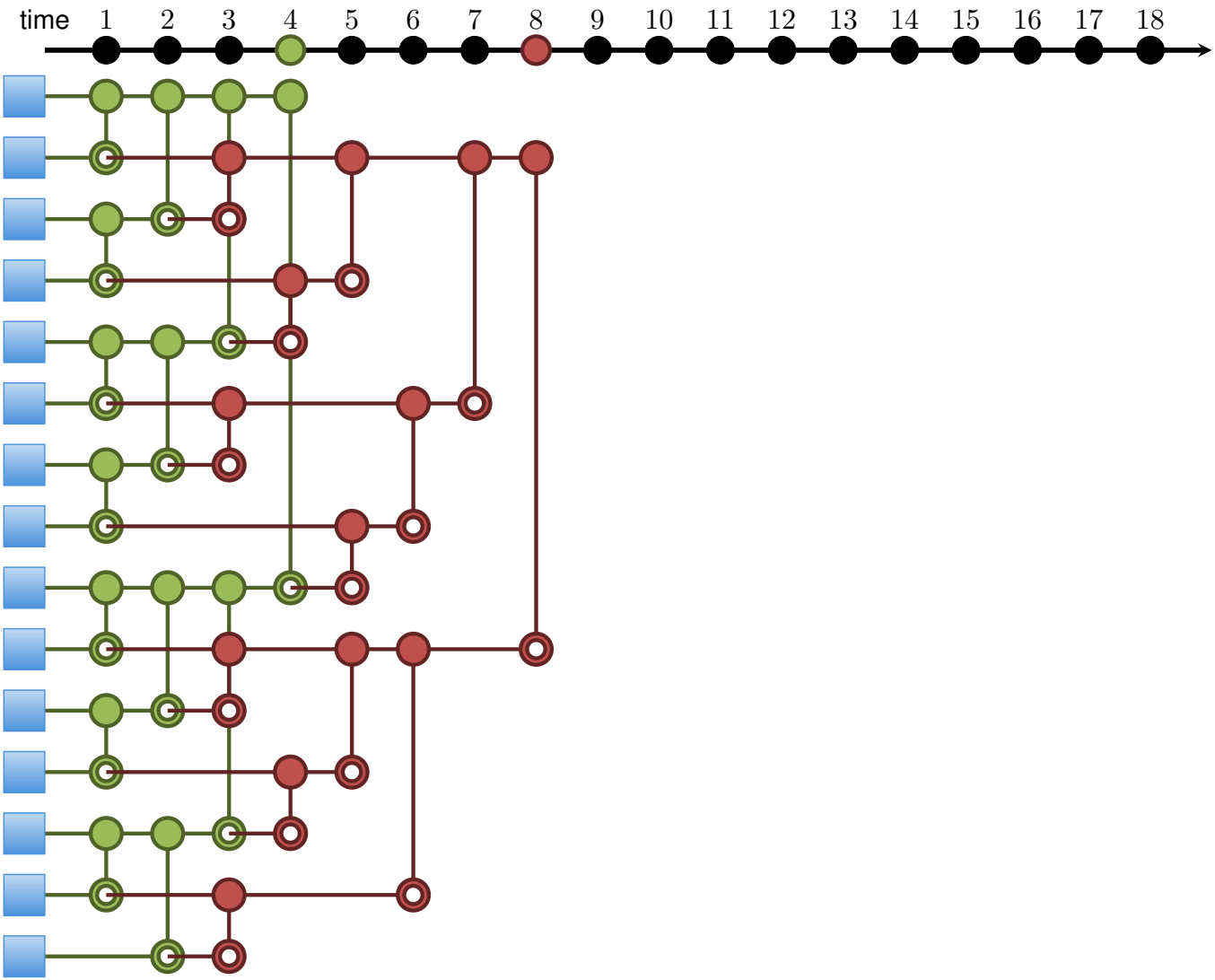
Reduction Trees for QR: binomial tree

$p = 15, q = 2$



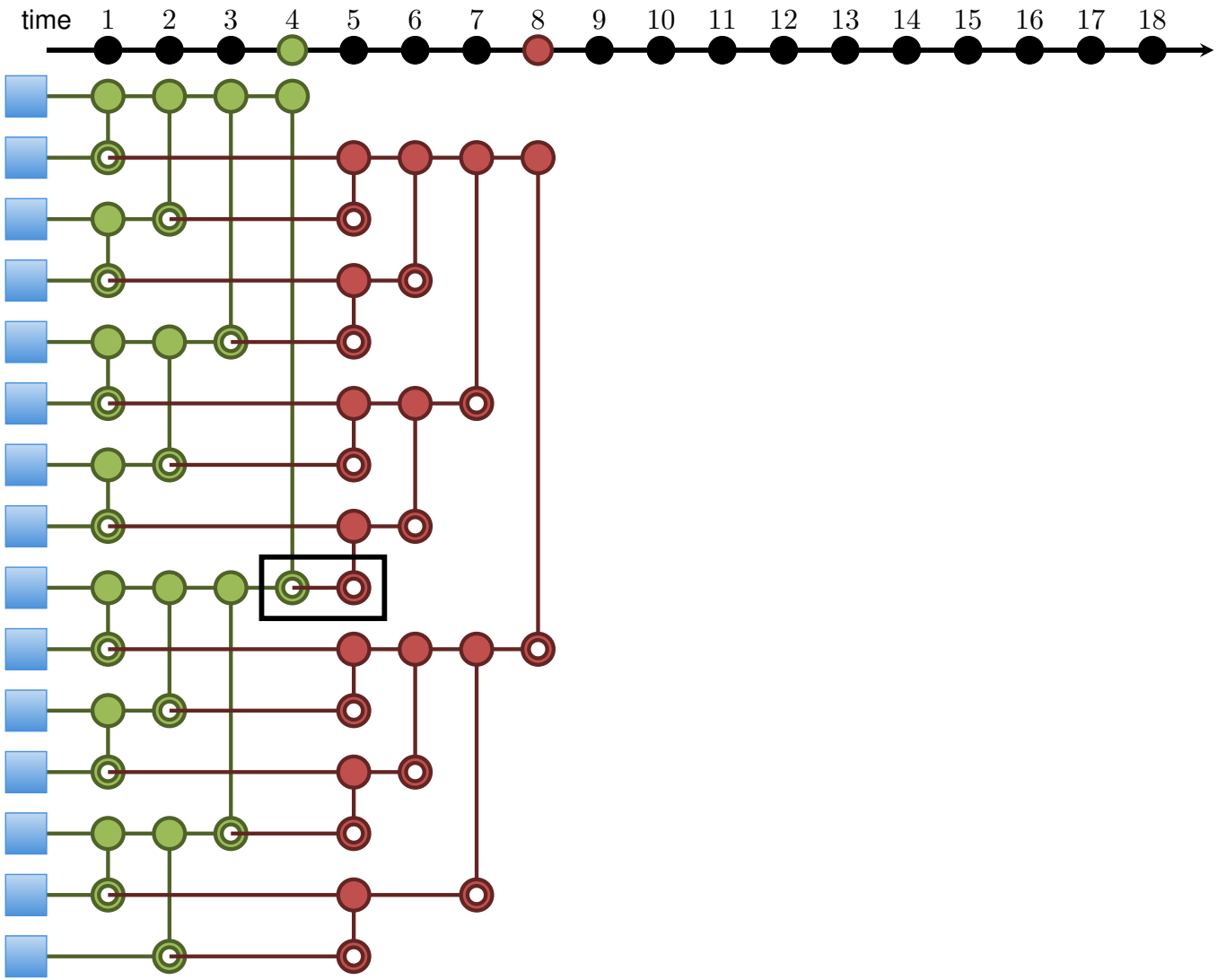
Reduction Trees for QR: binomial tree

$p = 15, q = 2$



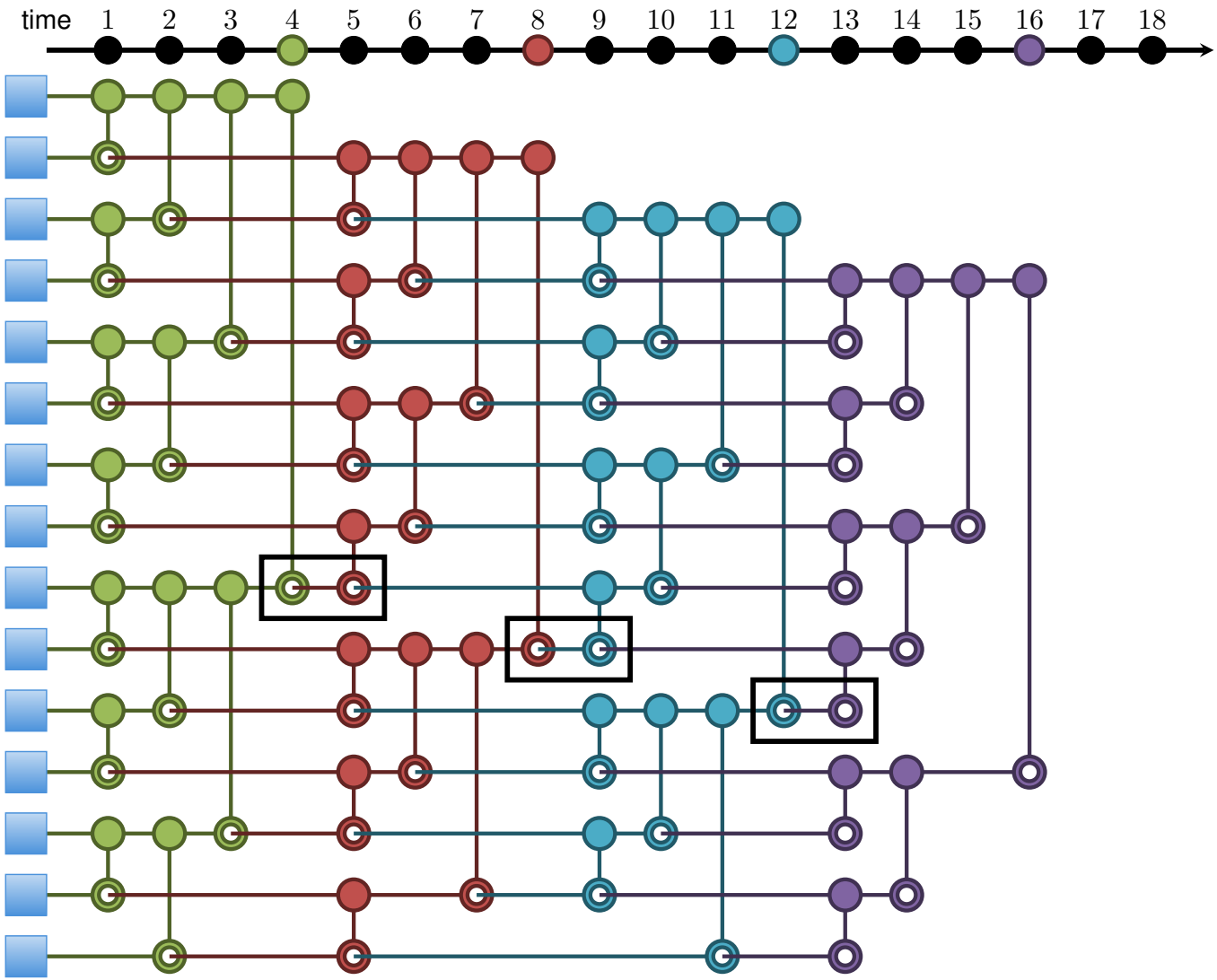
Reduction Trees for QR: binomial tree

$p = 15, q = 2$



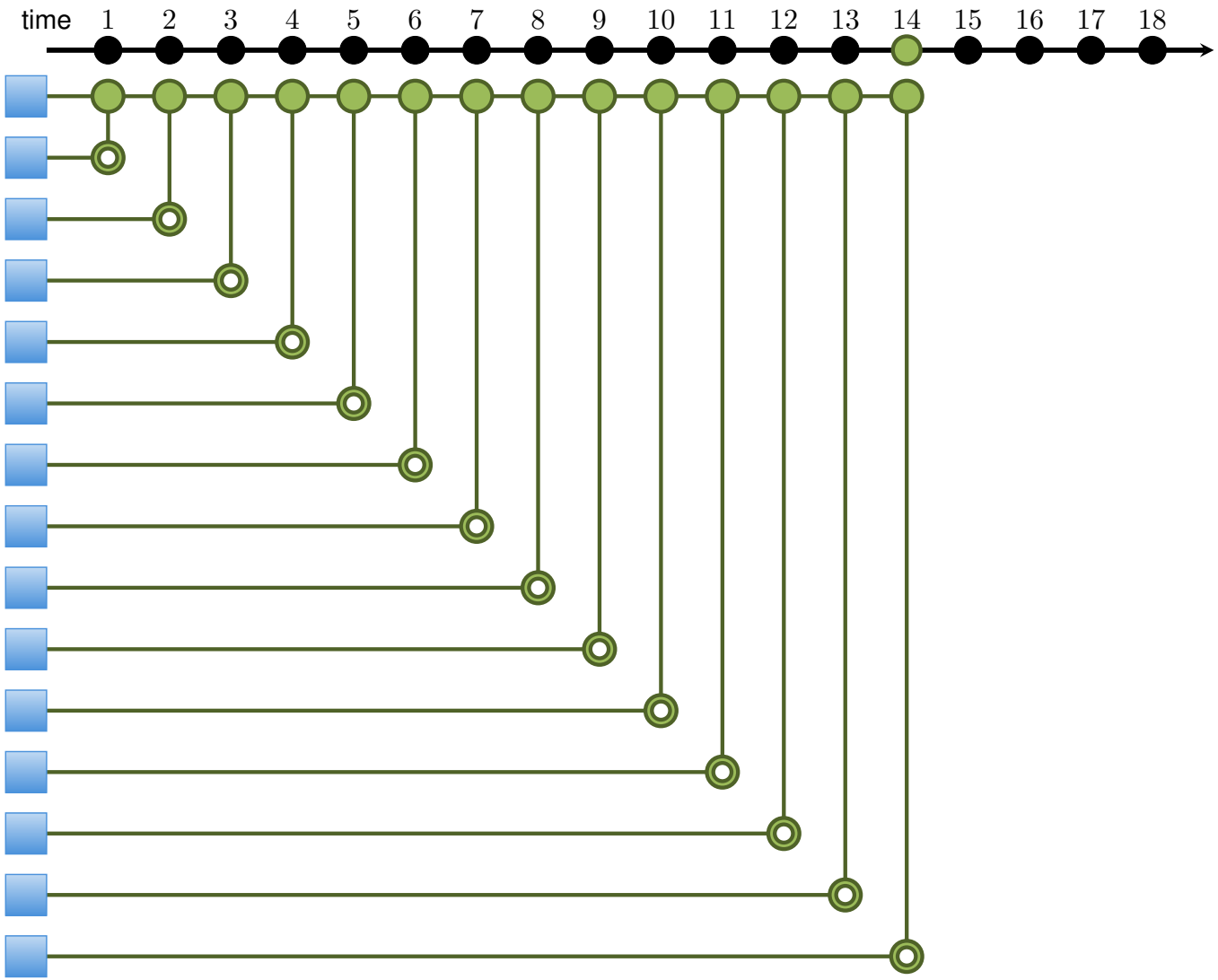
Reduction Trees for QR: binomial tree

$p = 15, q = 4$



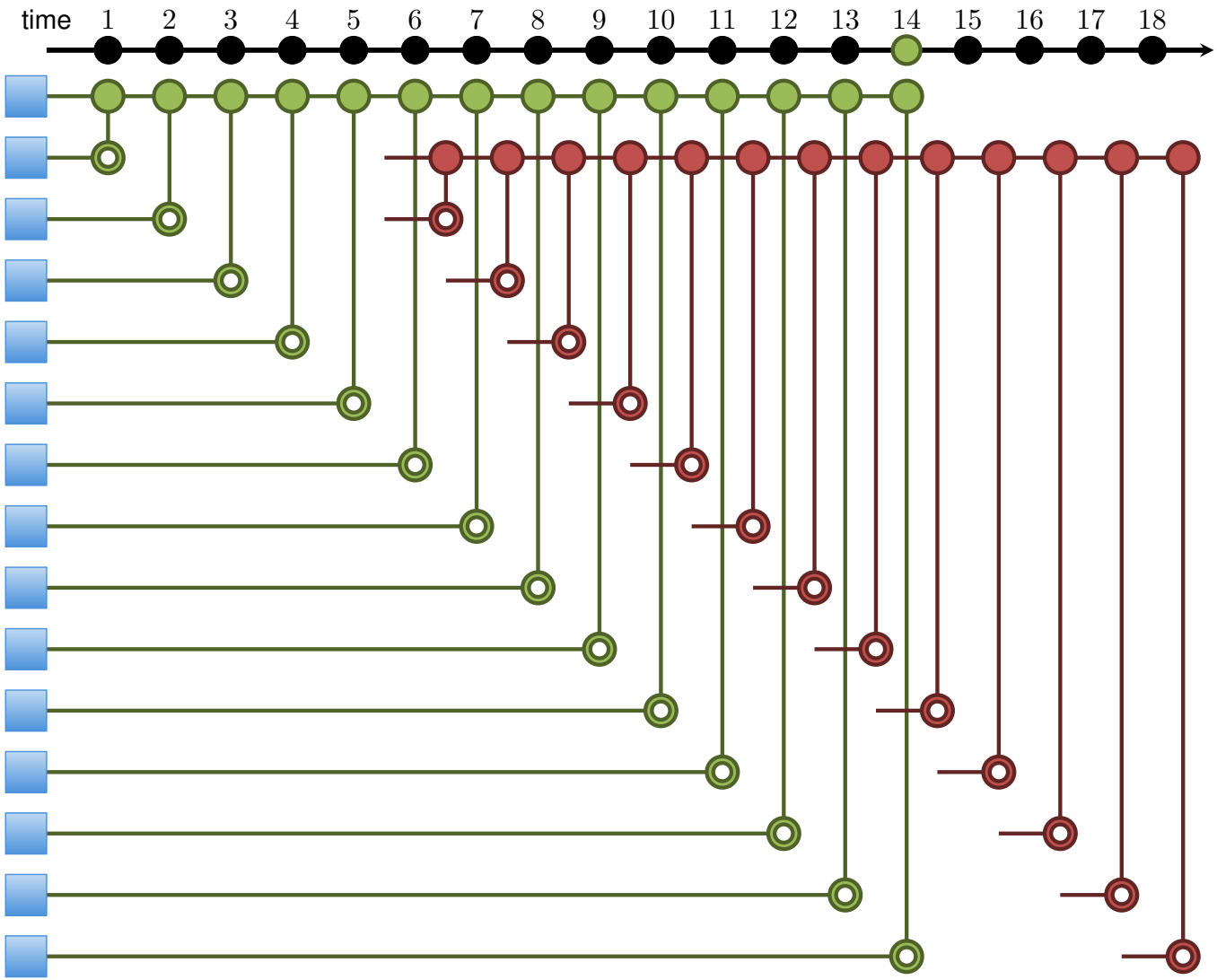
Reduction Trees for QR: flat tree

$p = 15, q = 1$



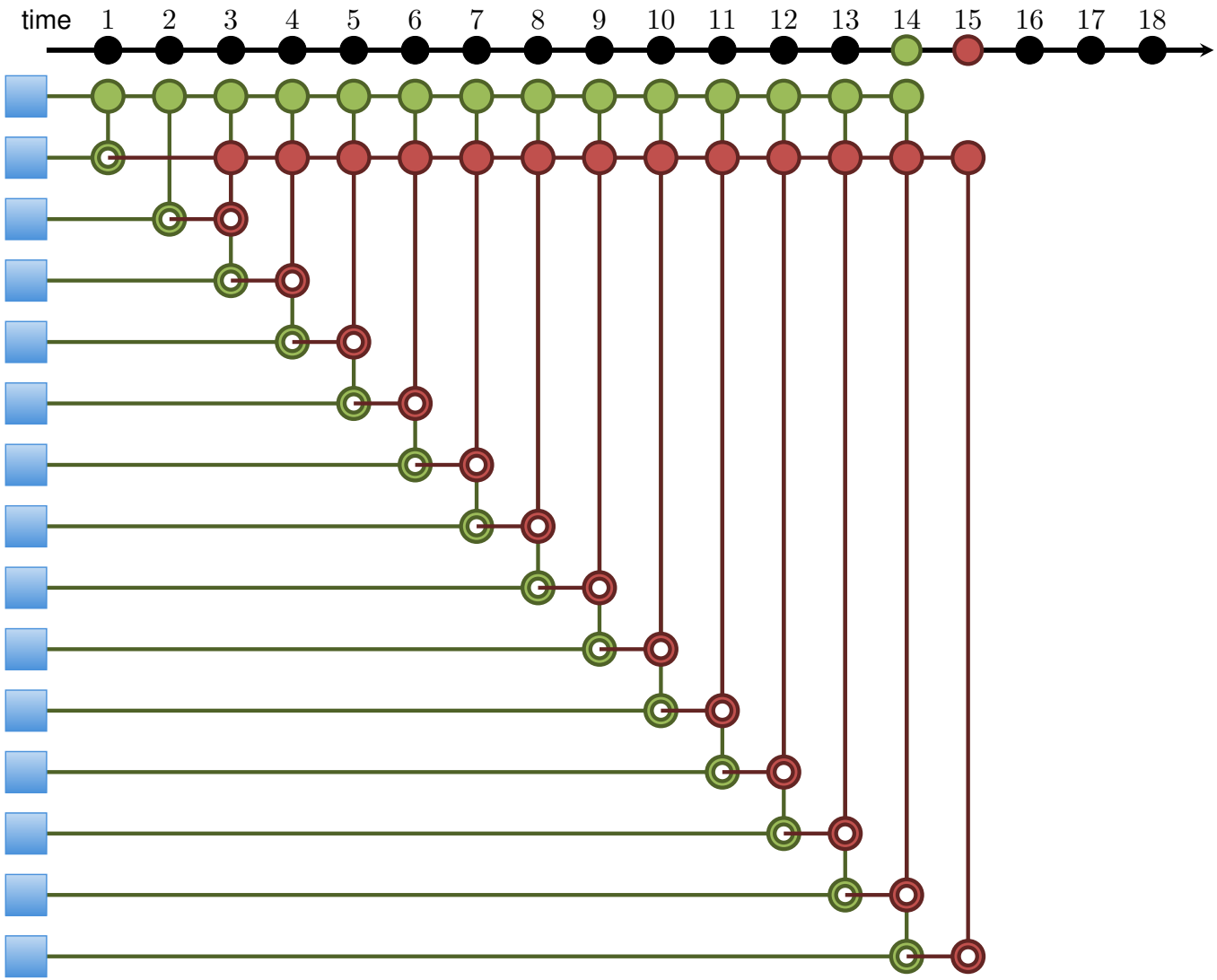
Reduction Trees for QR: flat tree

$p = 15, q = 2$



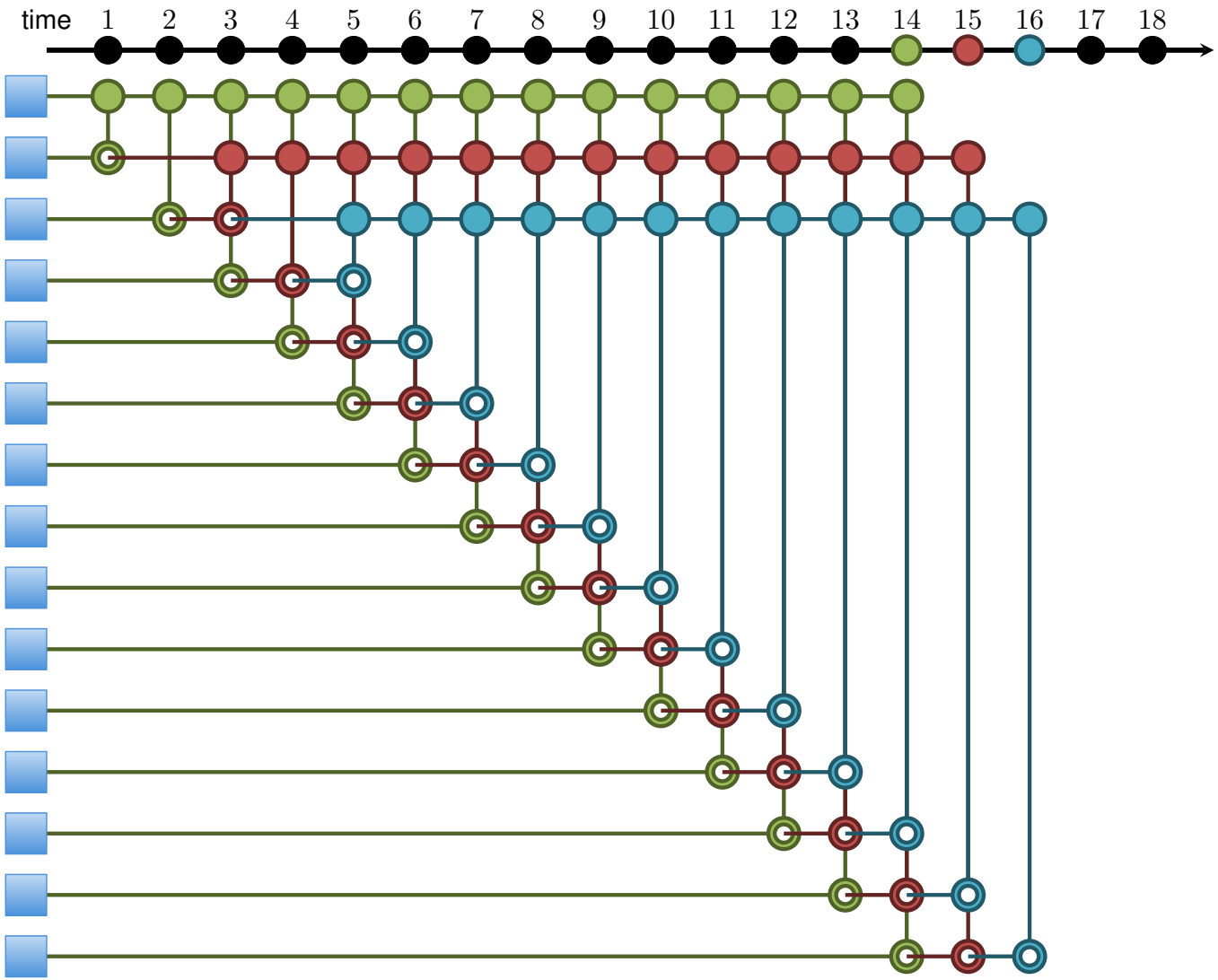
Reduction Trees for QR: flat tree

$p = 15, q = 2$



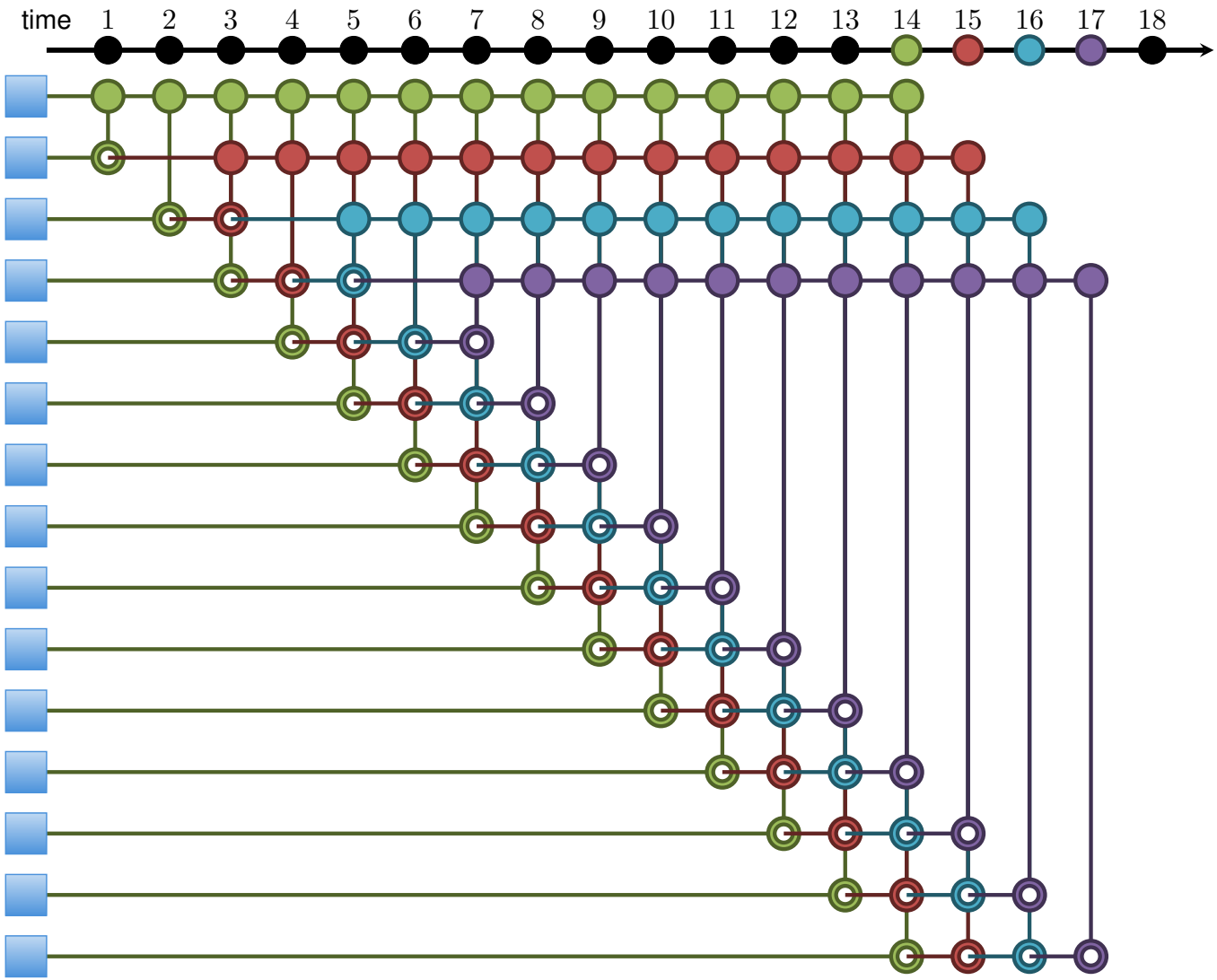
Reduction Trees for QR: flat tree

$p = 15, q = 3$



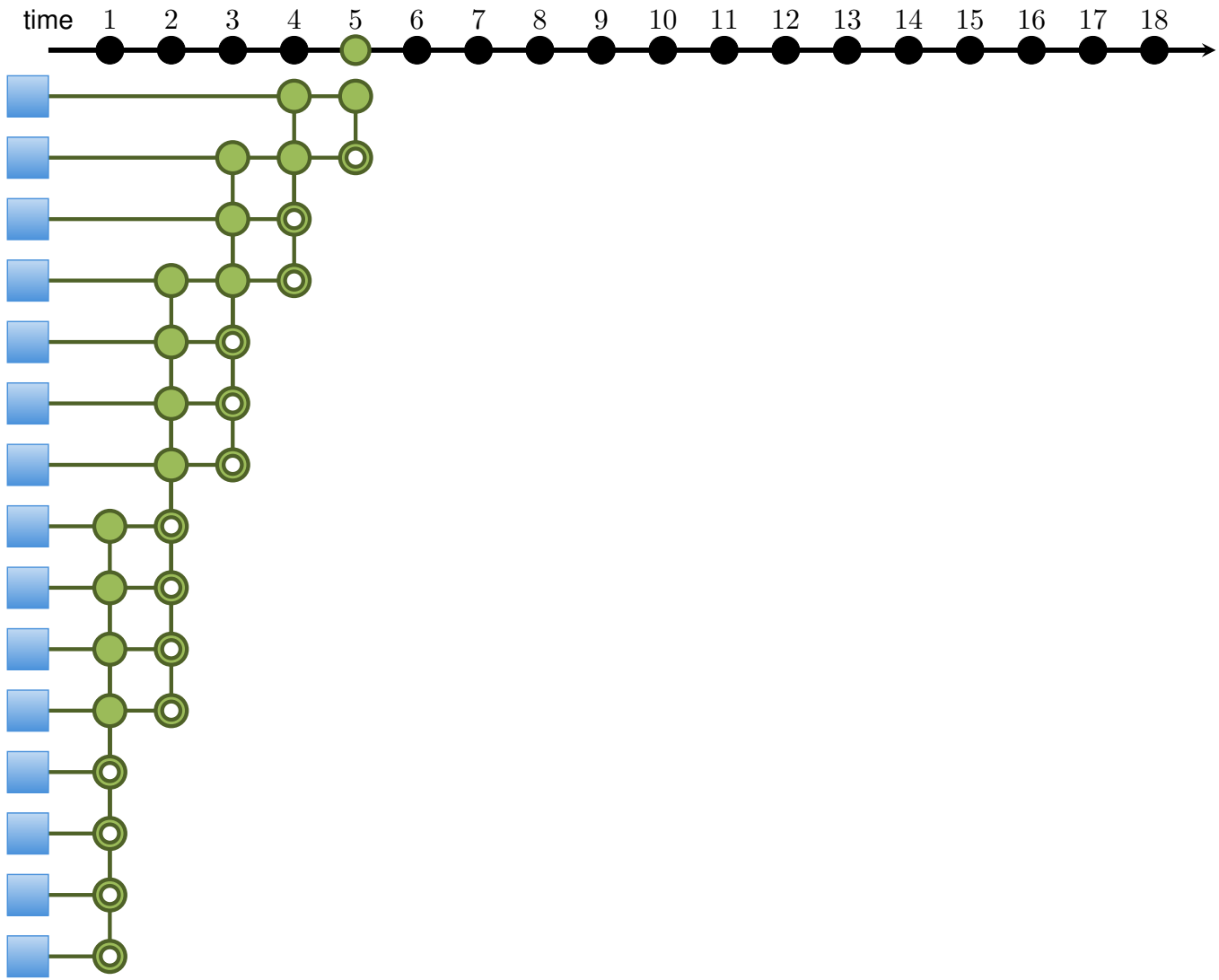
Reduction Trees for QR: flat tree

$p = 15, q = 4$



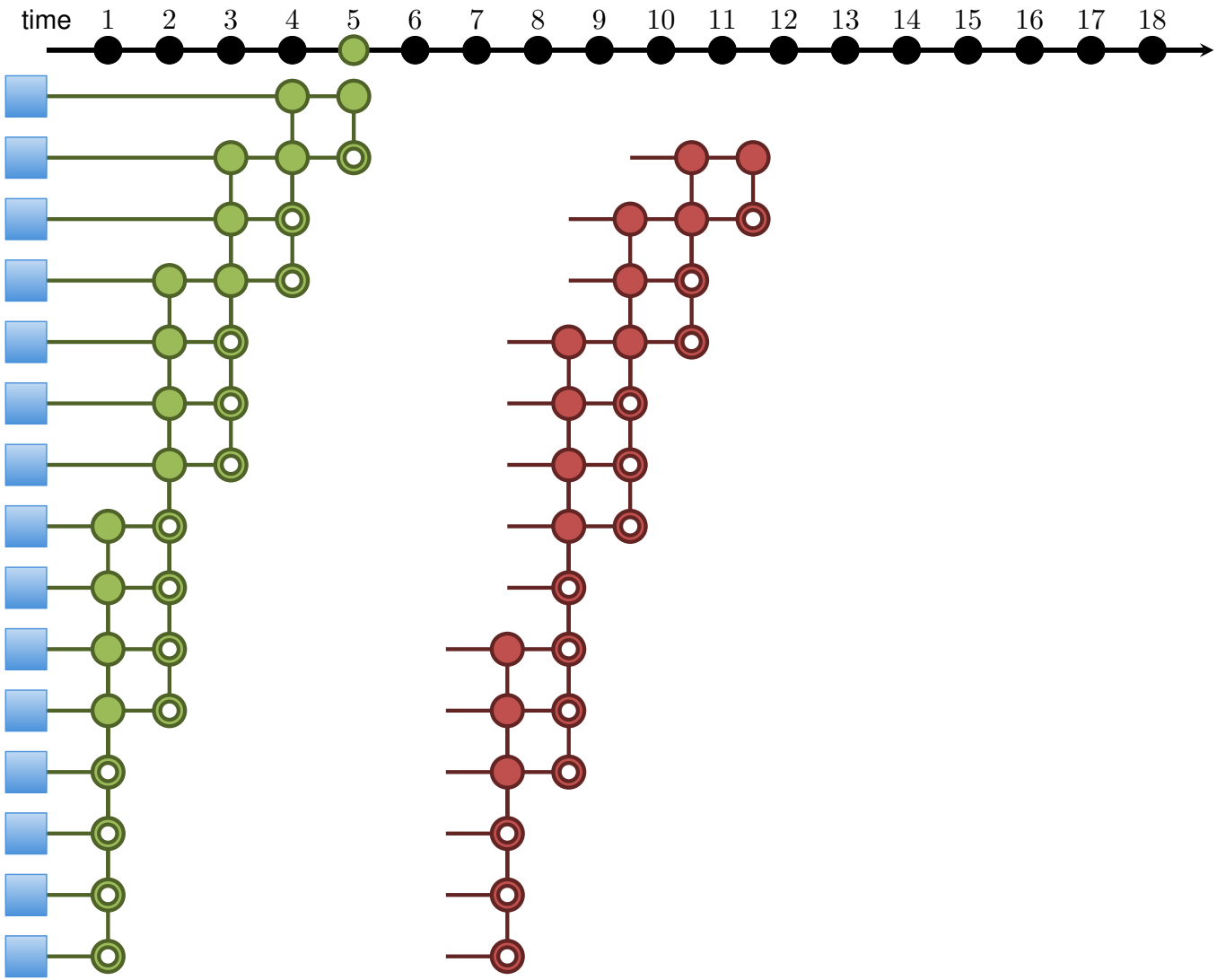
Reduction Trees for QR: Fibonacci tree

$p = 15, q = 1$



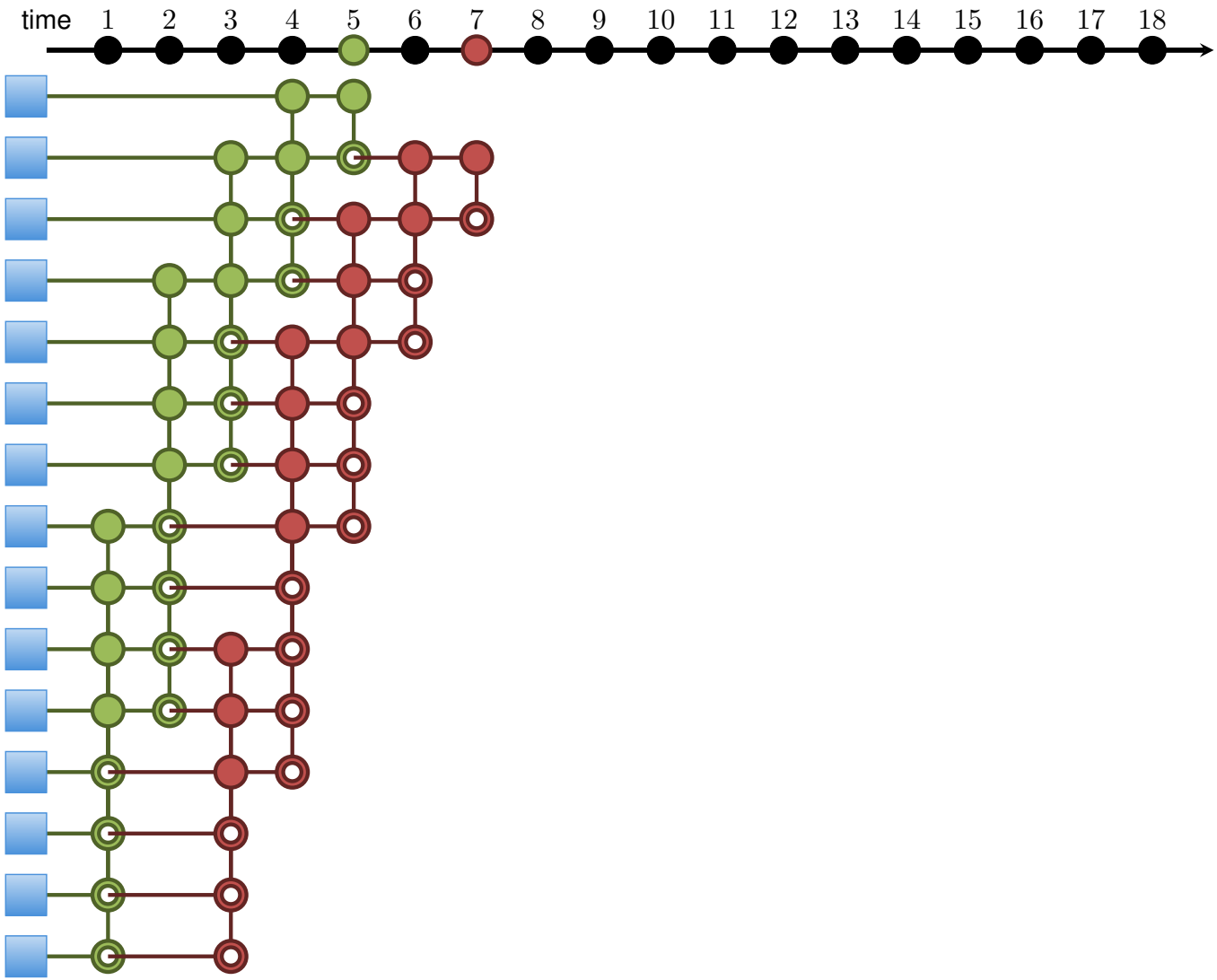
Reduction Trees for QR: Fibonacci tree

$p = 15, q = 2$



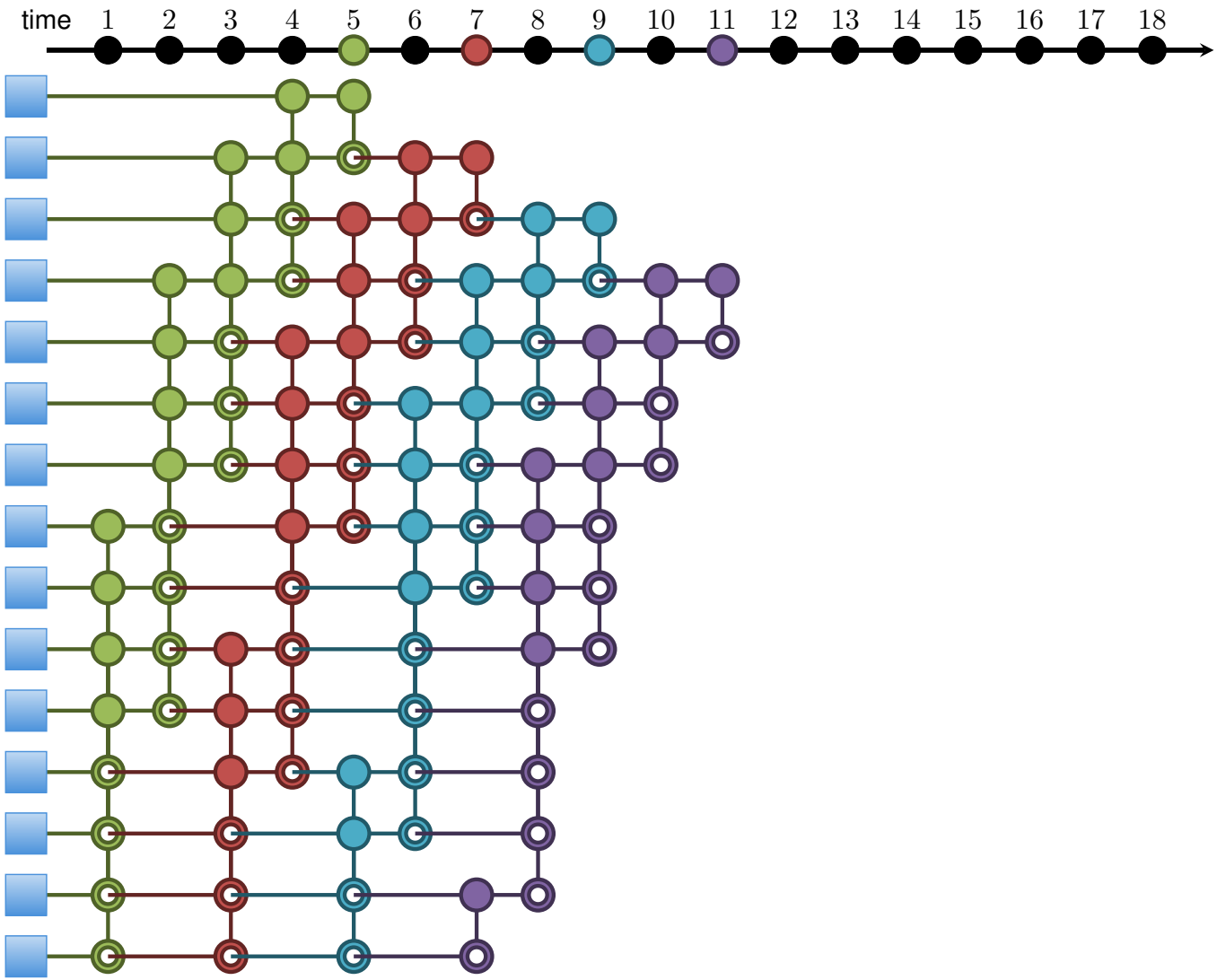
Reduction Trees for QR: Fibonacci tree

$p = 15, q = 2$



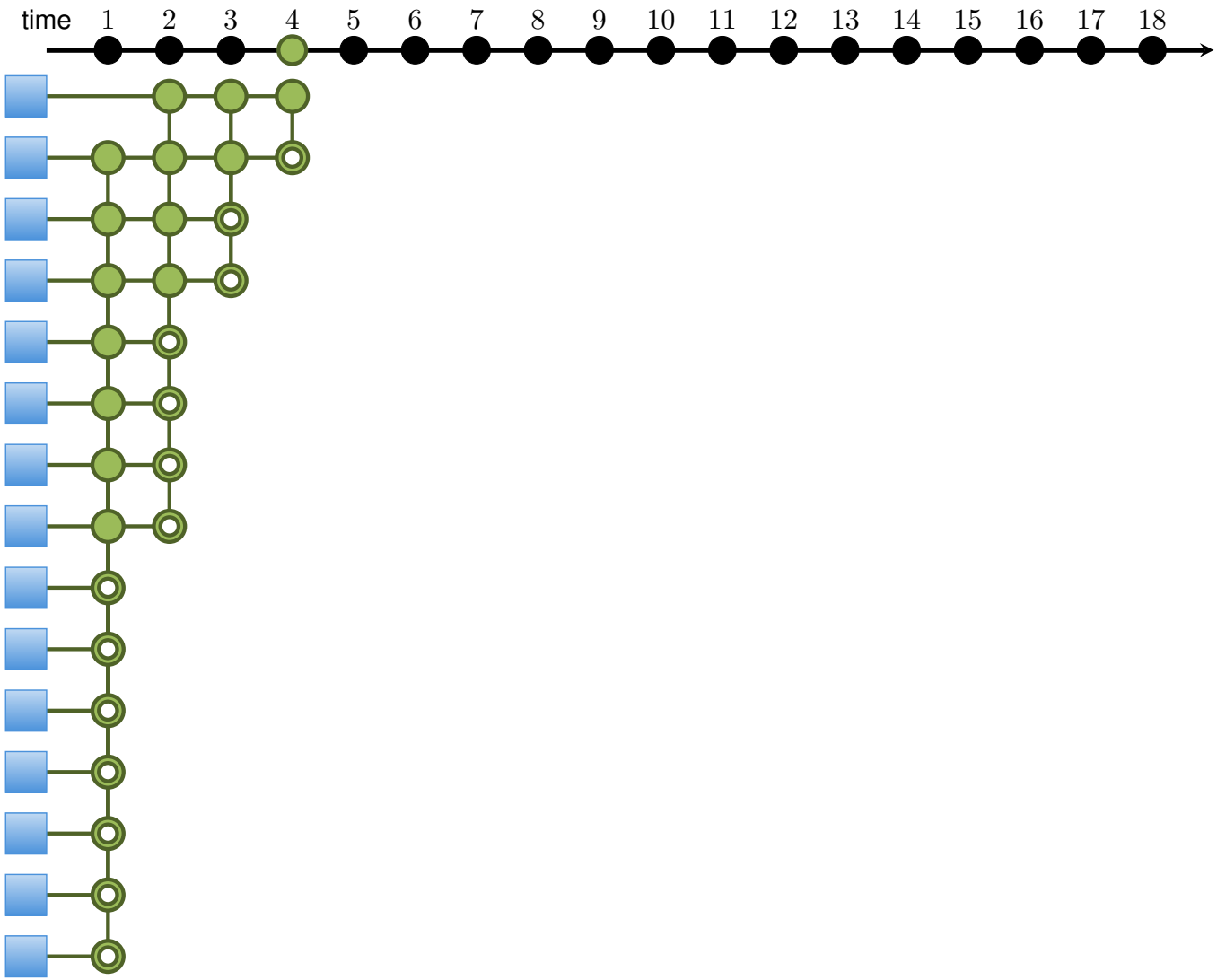
Reduction Trees for QR: Fibonacci tree

$p = 15, q = 4$



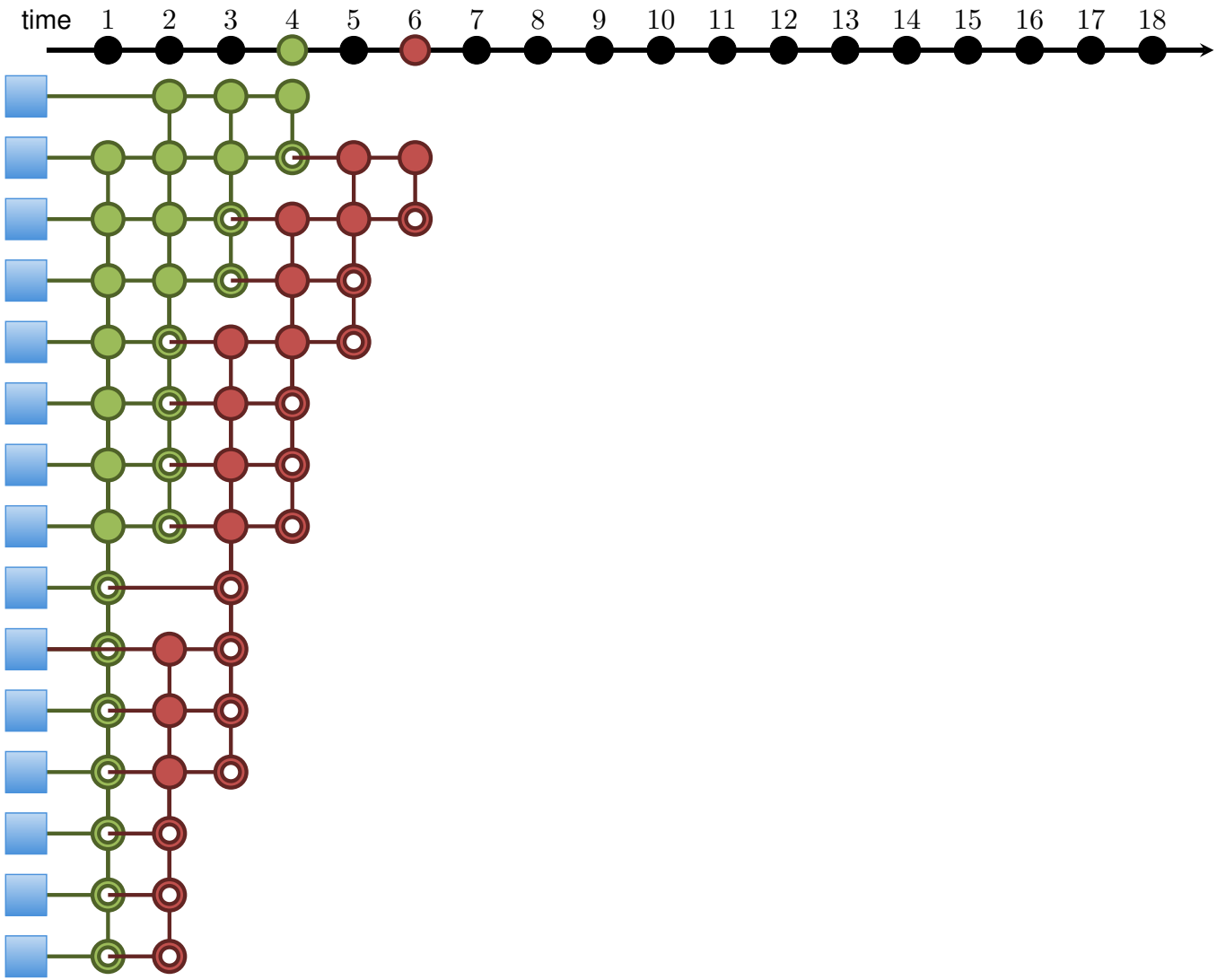
Reduction Trees for QR: Greedy tree

$p = 15, q = 1$



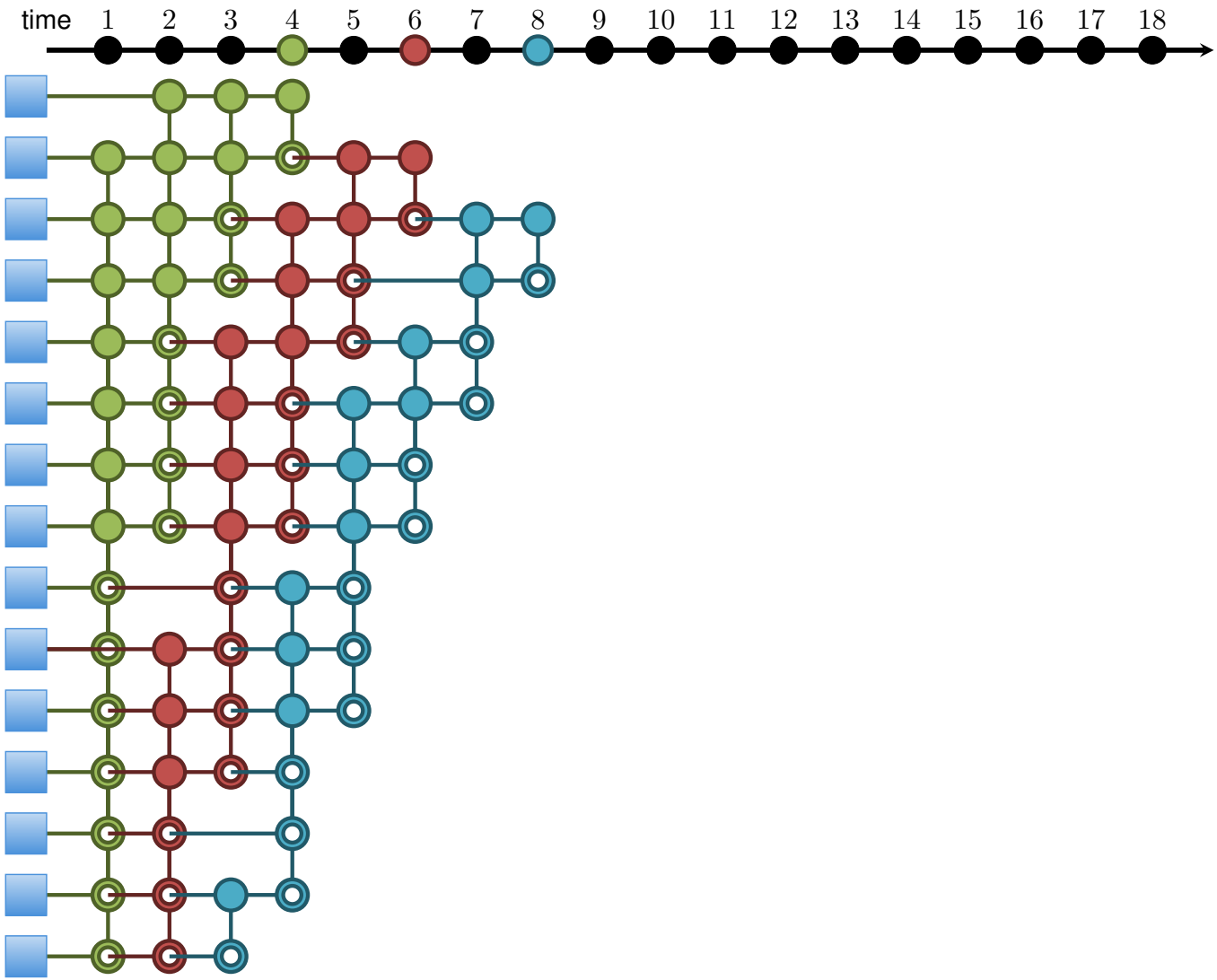
Reduction Trees for QR: Greedy tree

$p = 15, q = 2$



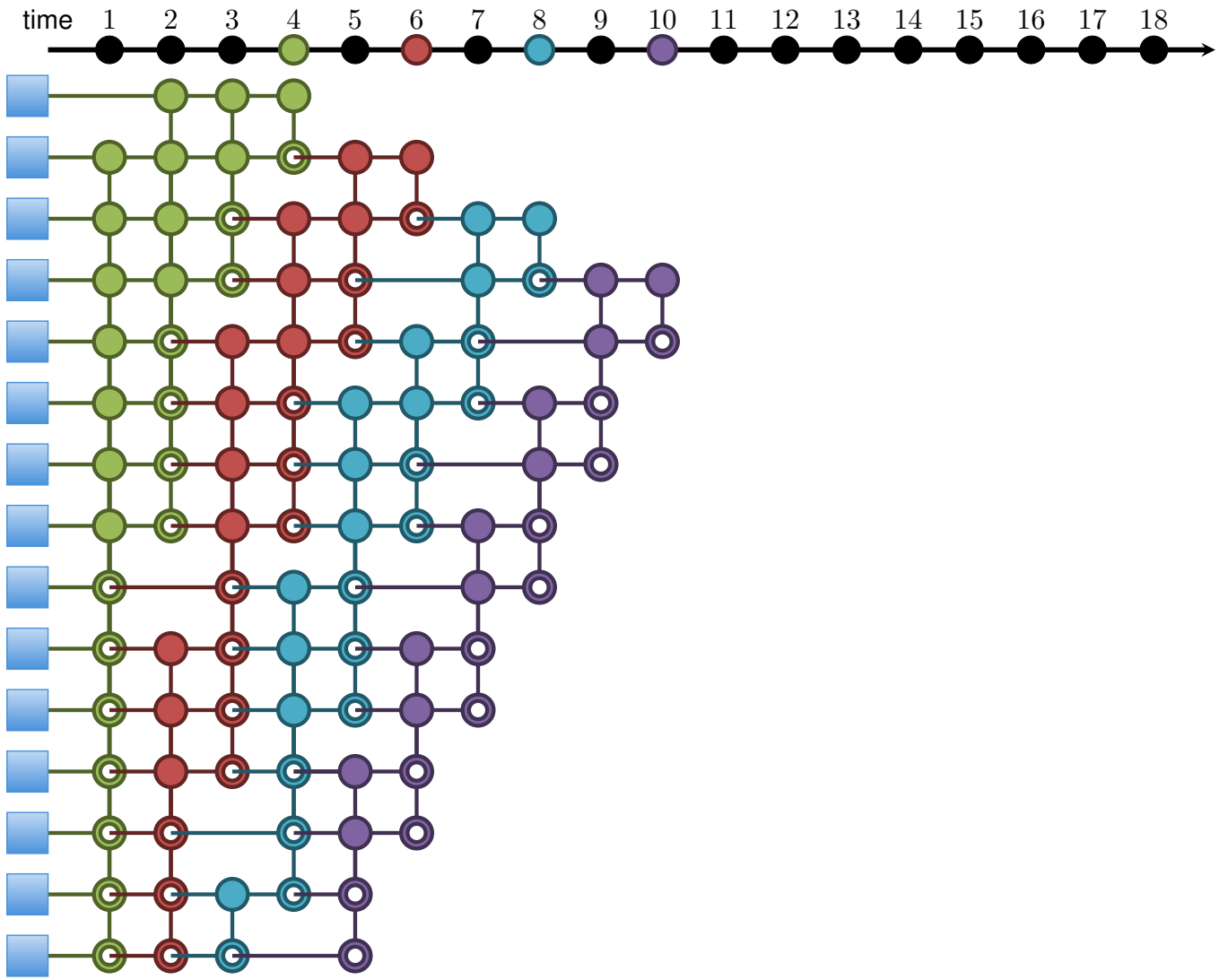
Reduction Trees for QR: Greedy tree

$p = 15, q = 3$



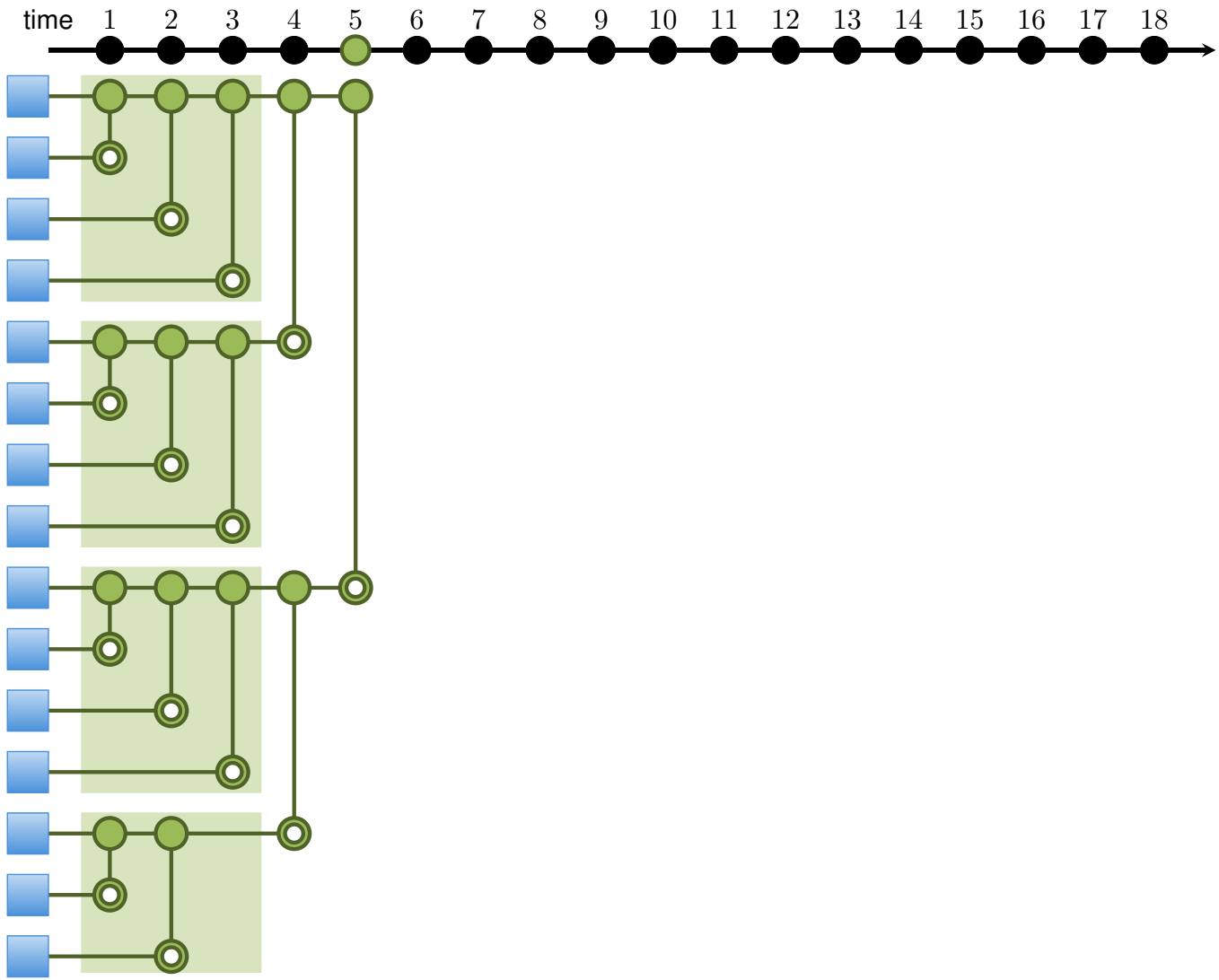
Reduction Trees for QR: Greedy tree

$p = 15, q = 4$



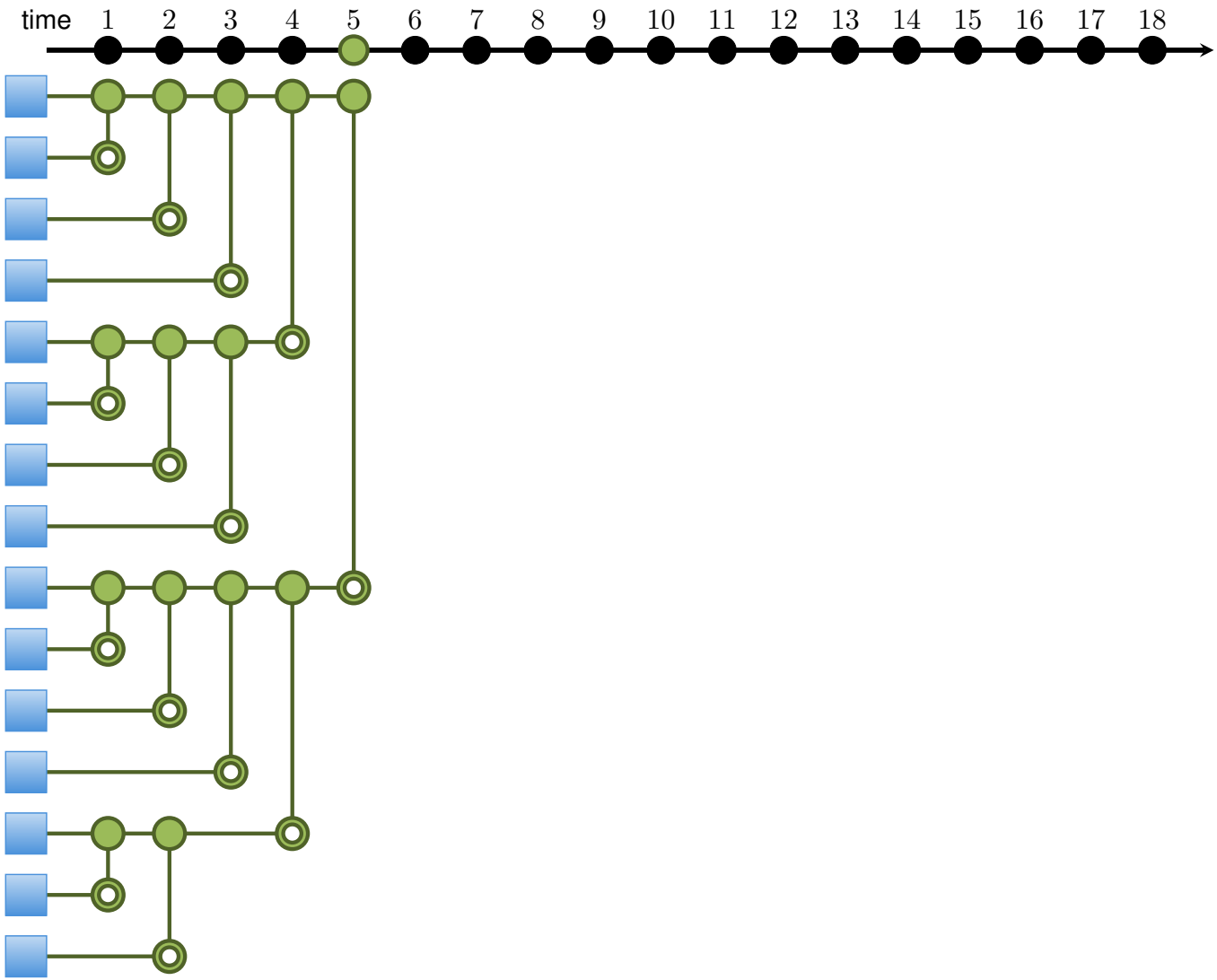
Reduction Trees for QR: PLASMA tree

$p = 15, q = 1$



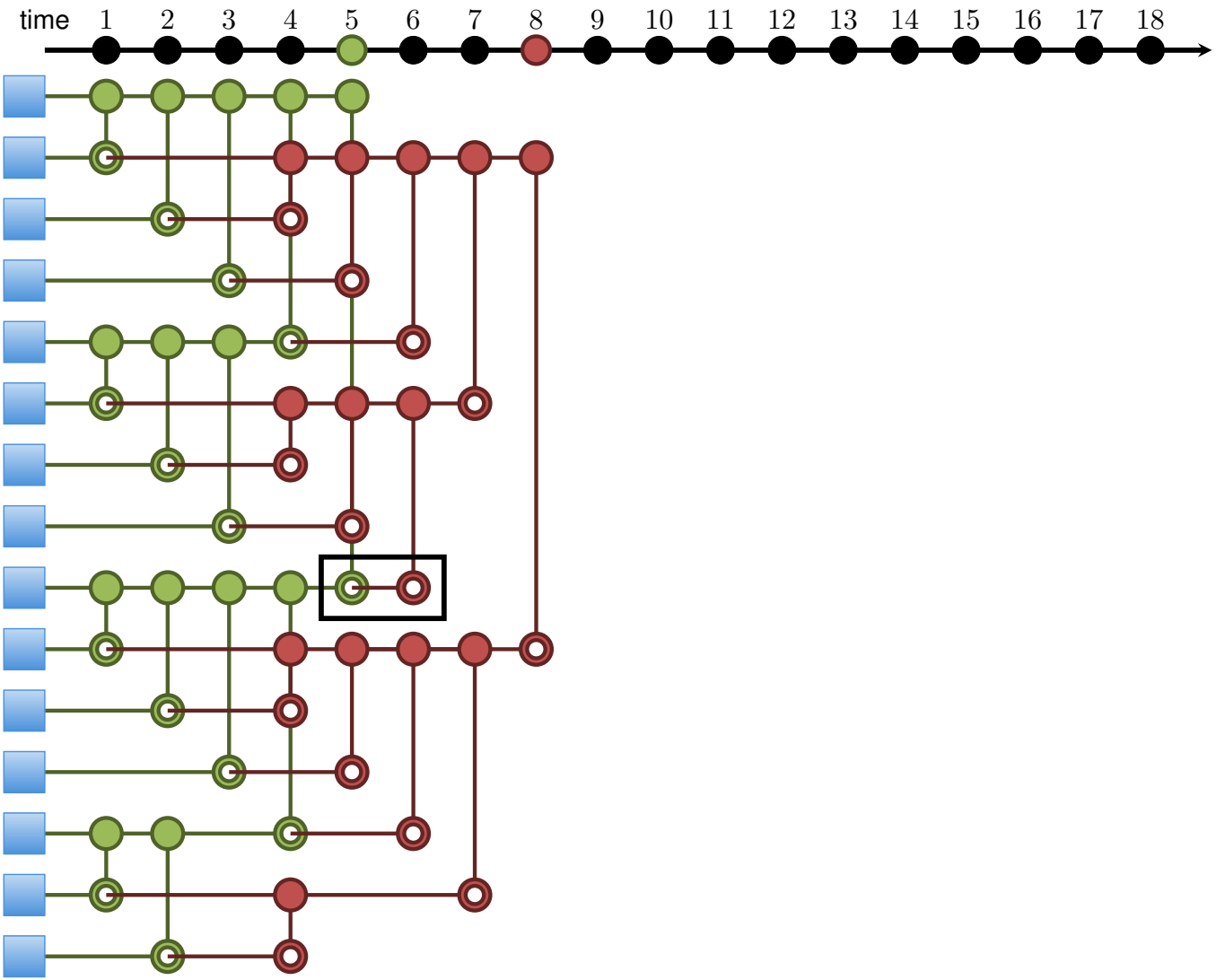
Reduction Trees for QR: PLASMA tree

$p = 15, q = 1$



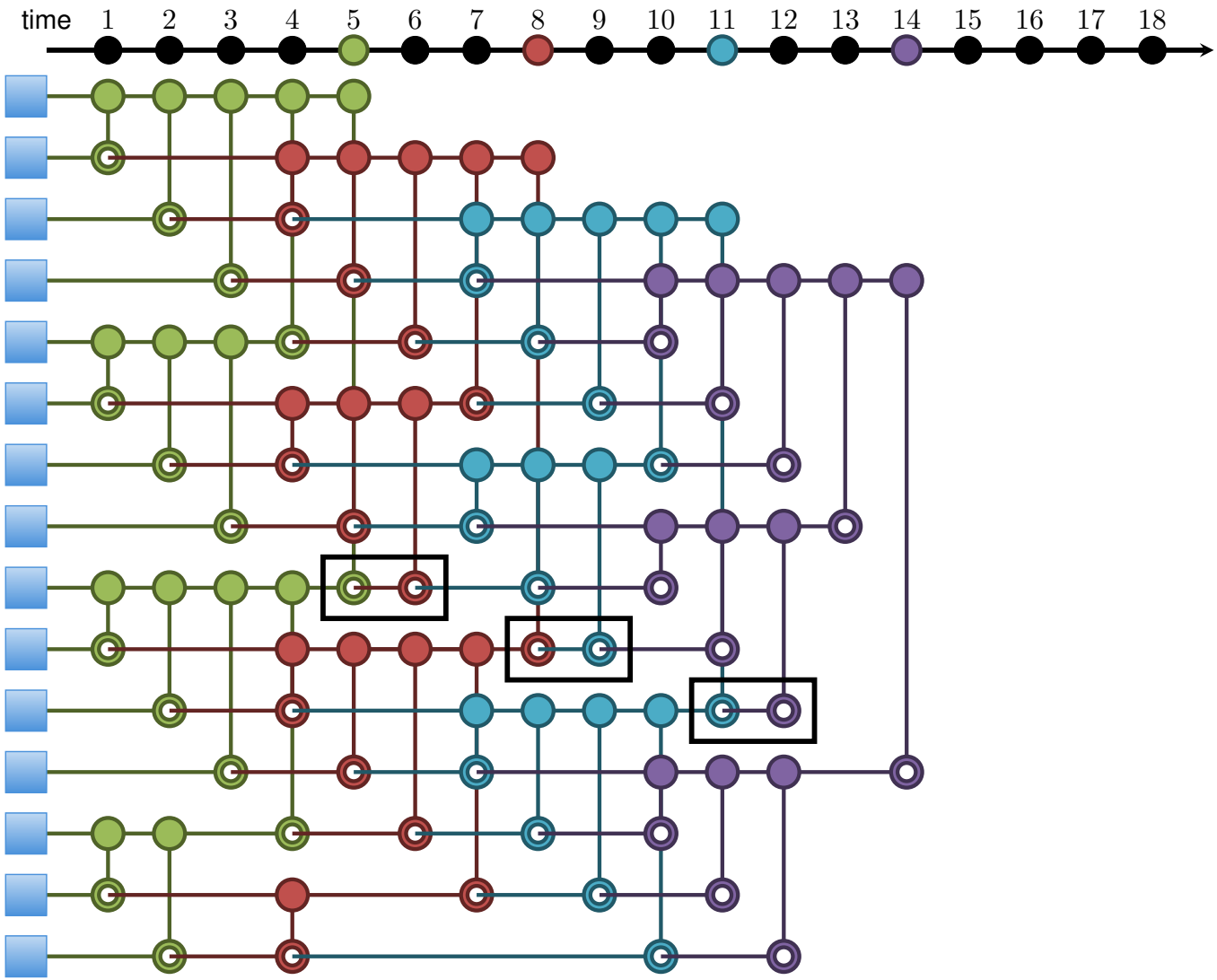
Reduction Trees for QR: PLASMA tree

$p = 15, q = 2$



Reduction Trees for QR: PLASMA tree

$p = 15, q = 4$



Reduction Trees for QR

In previous work (2008-2014),

1. we wrote optimized kernels for performing the "sequential" operations involved in these reduction trees;

Reduction Trees for QR

In previous work (2008-2014),

1. we wrote optimized kernels for performing the "sequential" operations involved in these reduction trees;

2. we provided critical path lengths for many of these trees with various weights;

Reduction Trees for QR

In previous work (2008-2014),

1. we wrote optimized kernels for performing the "sequential" operations involved in these reduction trees;

2. we provided critical path lengths for many of these trees with various weights;
3. when non-unit weights are considered, we proved that GREEDY "almost" has the shortest critical path length over all sequences of reduction trees, and that the so-called GRASAP algorithm was optimal;

Reduction Trees for QR

In previous work (2008-2014),

1. we wrote optimized kernels for performing the "sequential" operations involved in these reduction trees;

2. we provided critical path lengths for many of these trees with various weights;
3. when non-unit weights are considered, we proved that GREEDY "almost" has the shortest critical path length over all sequences of reduction trees, and that the so-called GRASAP algorithm was optimal;
4. we produced implementation of these algorithms over various runtimes (QUARK, PARSEC)

Reduction Trees for QR

In previous work (2008-2014),

1. we wrote optimized kernels for performing the "sequential" operations involved in these reduction trees;

2. we provided critical path lengths for many of these trees with various weights;
3. when non-unit weights are considered, we proved that GREEDY "almost" has the shortest critical path length over all sequences of reduction trees, and that the so-called GRASAP algorithm was optimal;
4. we produced implementation of these algorithms over various runtimes (QUARK, PARSEC)
5. we performed numerical experiments in shared memory and parallel distributed

Reduction Trees for QR

In previous work (2008-2014),

1. we wrote optimized kernels for performing the "sequential" operations involved in these reduction trees;

2. we provided critical path lengths for many of these trees with various weights;
3. when non-unit weights are considered, we proved that GREEDY "almost" has the shortest critical path length over all sequences of reduction trees, and that the so-called GRASAP algorithm was optimal;
4. we produced implementation of these algorithms over various runtimes (QUARK, PARSEC)
5. we performed numerical experiments in shared memory and parallel distributed
6. we produced open-source freely-available libraries with these algorithms (PLASMA, DPLASMA)

Reduction Trees for QR

In previous work (2008-2014),

1. we wrote optimized kernels for performing the "sequential" operations involved in these reduction trees;

2. we provided critical path lengths for many of these trees with various weights;
3. when non-unit weights are considered, we proved that GREEDY "almost" has the shortest critical path length over all sequences of reduction trees, and that the so-called GRASAP algorithm was optimal;
4. we produced implementation of these algorithms over various runtimes (QUARK, PARSEC)
5. we performed numerical experiments in shared memory and parallel distributed
6. we produced open-source freely-available libraries with these algorithms (PLASMA, DPLASMA)

Goal: extend this work from QR factorization to Bidiagonalization

Related work

H. Ltaief, J. Kurzak, and J. Dongarra.
Parallel two-sided matrix reduction to band bidiagonal form on multicore architectures.
IEEE Transactions on Parallel and Distributed Systems, 21(4):417–423, Apr. 2010.

- Contribution: First paper presenting tiled band bidiagonalization over runtime.
- Only GE2BND step.
- Shared memory.
- Use Flat TS QR trees and Flat TS LQ trees.
- Algorithm currently in PLASMA.
- Square matrices.

Related work

H. Ltaief, P. Luszczek, and J. Dongarra.
High-performance bidiagonal reduction using tile algorithms on homogeneous multicore architectures.
ACM Trans. Math. Softw., 39(3):16:1–16:22, May 2013.

- (Technical Report in 2011.)
- Contribution: Added a shared memory Band to Bidiagonal step (BND2BD) so as to have GE2VAL=GE2BND+BND2BD+BD2VAL.
- Shared memory.
- Use Flat TS QR trees and Flat TS LQ trees for GE2BND.
- Algorithm currently in PLASMA.
- Contribution: Study trade-off of the bandwidth/tile size: small vs large.
- Square matrices.

Related work

H. Ltaief, P. Luszczek, and J. Dongarra.
Enhancing parallelism of tile bidiagonal transformation on multicore architectures using tree reduction.
In R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, editors, *Parallel Processing and Applied Mathematics: 9th International Conference, PPAM 2011, Torun, Poland, September 11-14, 2011. Revised Selected Papers, Part I*, pages 661–670. Springer Berlin Heidelberg, 2012.

- Only GE2BND step.
- Shared memory.
- Contribution: Use Binomial TS QR trees and Flat TS LQ trees for GE2BND.
- Rectangular matrices.

Related work

A. Haidar, H. Ltaief, P. Luszczek, and J. Dongarra.
A comprehensive study of task coalescing for selecting parallelism granularity in a two-stage bidiagonal reduction.
In *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 25–35, May 2012.

- Contribution: Improve the BND2BD step of Ltaief et al. (2013)
- Shared memory.
- Use Flat TS QR trees and Flat TS LQ trees for GE2BND.
- Square matrices.

Related work

A. Haidar, J. Kurzak, and P. Luszczek.
An improved parallel singular value algorithm and its implementation for multicore hardware.
In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 90:1–90:12, New York, NY, USA, 2013. ACM.

- Contribution: Finally, in 2013, Haidar, Kurzak, and Luszczek [1] consider the problem of computing singular vectors (GESVD) by performing GE2BND+BND2BD+BD2VAL+VAL2BD+BD2BND+BND2GE. They show that the two-step approach (from full to band, then band to bidiagonal) can be successfully used not only for computing singular values, but also for computing singular vectors.
- Shared memory.
- Use Flat TS QR trees and Flat TS LQ trees for GE2BND.
- Square matrices.

Our work in context (1/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

Our work in context (1/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- Only GE2BND step.
  (We rely on previous work for BND2BD and BD2VAL.)

Our work in context (1/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- Only GE2BND step.
  (We rely on previous work for BND2BD and BD2VAL.)
- Rectangular and square matrices. (Not just square.)

Our work in context (1/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- Only GE2BND step.
  (We rely on previous work for BND2BD and BD2VAL.)
- Rectangular and square matrices. (Not just square.)
- Presentation and study of BiDiag with many types of reduction QR and LQ trees. (Not just FlatTS.)

Our work in context (1/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- Only GE2BND step.
  (We rely on previous work for BND2BD and BD2VAL.)
- Rectangular and square matrices. (Not just square.)
- Presentation and study of BiDiag with many types of reduction QR and LQ trees. (Not just FlatTS.)
- Proof that, in the context of BiDiag, Binomial QR and LQ trees provide the shortest critical path.

Our work in context (1/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- Only GE2BND step.
  (We rely on previous work for BND2BD and BD2VAL.)
- Rectangular and square matrices. (Not just square.)
- Presentation and study of BiDiag with many types of reduction QR and LQ trees. (Not just FlatTS.)
- Proof that, in the context of BiDiag, Binomial QR and LQ trees provide the shortest critical path.
- Introduction and study of R-BiDiag in the context of tiled algorithms.

Our work in context (2/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- A detailed study of critical path lengths for few sequences of reduction QR and LQ trees: FlatTS, FlatTT, Greedy with BiDiag and R-BiDiag (so six different algorithms in total), which shows that:

Our work in context (2/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- A detailed study of critical path lengths for few sequences of reduction QR and LQ trees: FlatTS, FlatTT, Greedy with BiDiag and R-BiDiag (so six different algorithms in total), which shows that:
  - The Greedy based schemes (BiDiag and R-BiDiag) are much better than earlier proposed variants.

Our work in context (2/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- A detailed study of critical path lengths for few sequences of reduction QR and LQ trees: FlatTS, FlatTT, Greedy with BiDiag and R-BiDiag (so six different algorithms in total), which shows that:
  - The Greedy based schemes (BiDiag and R-BiDiag) are much better than earlier proposed variants.
  - On the one hand, BiDiagGreedy has a shorter critical path length than R-BiDiagGreedy for square matrices; on the other hand, R-BiDiagGreedy has a shorter critical path length than BiDiagGreedy for tall and skinny matrices.

Our work in context (3/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- A practical auto-adaptative tree (AUTO) that self-tunes for increased performance.
  (Useful (1) when critical path length is not a major consideration due to limited number of resources or (2) when intra-node "communication" is expensive or (3) both.)

Our work in context (3/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- A practical auto-adaptative tree (AUTO) that self-tunes for increased performance.
  (Useful (1) when critical path length is not a major consideration due to limited number of resources or (2) when intra-node "communication" is expensive or (3) both.)
- Implementation of our algorithms within the DPLASMA framework, which runs on top of the PARSEC runtime system, and which enables parallel distributed experiments on multicore nodes.

Our work in context (3/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- A practical auto-adaptative tree (AUTO) that self-tunes for increased performance.
  (Useful (1) when critical path length is not a major consideration due to limited number of resources or (2) when intra-node "communication" is expensive or (3) both.)
- Implementation of our algorithms within the DPLASMA framework, which runs on top of the PARSEC runtime system, and which enables parallel distributed experiments on multicore nodes.
- Our final implementation for BIDIAG has three level of trees. High level for parallel distributed, middle level within a node, and low level for enabling TS kernels. Optionally pre-processed by a QR factorization.

Our work in context (3/3)

M. Faverge, J. Langou, Y. Robert, and J. Dongarra.
Bidiagonalization with parallel tiled algorithms.
Research Report 8969, INRIA, Oct. 2016.

- A practical auto-adaptative tree (AUTO) that self-tunes for increased performance.
  (Useful (1) when critical path length is not a major consideration due to limited number of resources or (2) when intra-node "communication" is expensive or (3) both.)
- Implementation of our algorithms within the DPLASMA framework, which runs on top of the PARSEC runtime system, and which enables parallel distributed experiments on multicore nodes.
- Our final implementation for BIDIAG has three level of trees. High level for parallel distributed, middle level within a node, and low level for enabling TS kernels. Optionally pre-processed by a QR factorization.
- Experiments on a single multicore node (1 node, 24 cores), and on a few multicore nodes of a parallel distributed shared-memory system (25 nodes, 600 cores), on a variety of matrix sizes, matrix shapes and core counts.

No overlap theorem

Theorem

For any sequence of QR and LQ steps (any reduction trees), no overlap between QR and LQ steps is possible

It says that the critical path length of BiDiag is the sum of the critical path lengths of the individual QR and LQ steps.

No overlap theorem

This theorem has theoretical and practical applications.
First, from a theoretical point of view, this makes critical path lengths of BiDiag algorithms "easy" to compute.
More importantly,

- In theory, this means that the shortest critical path length of BiDiag is obtained by minimizing the shortest critical path length of each individual step.
  $\Rightarrow$ This means that Binomial trees are optimal.
- In practice, this means that the smallest time of BiDiag is obtained by minimizing the time of each individual step. This means that we want to perform steps as far as possible (without consideration for previous or future steps).
  $\Rightarrow$ This leads to the Auto tree.

Optimality of BiDiagGreedy

Theorem

Overall BiDiag tiled algorithms, using binomial trees for LQ and QR steps provides us with the shortest critical path.

Critical path analysis of BiDiag algorithms

Theorem

For a matrix of p-by-q tiles, the critical path lengths of BiDiagFlatTS, BiDiagFlatTT, and BiDiagGreedy are such that

$$\text{BiDiagFlatTS}(p,q) = 12pq + \mathcal{O}(\max(p,q))$$

$$\text{BiDiagFlatTT}(p,q) = 6pq + \mathcal{O}(\max(p,q))$$

$$\text{BiDiagGreedy}(p,q) = 6q\log_2(p) + 6q\log_2(q) + \mathcal{O}(\max(p,q))$$

Critical path analysis of R-BiDiagGreedy algorithm

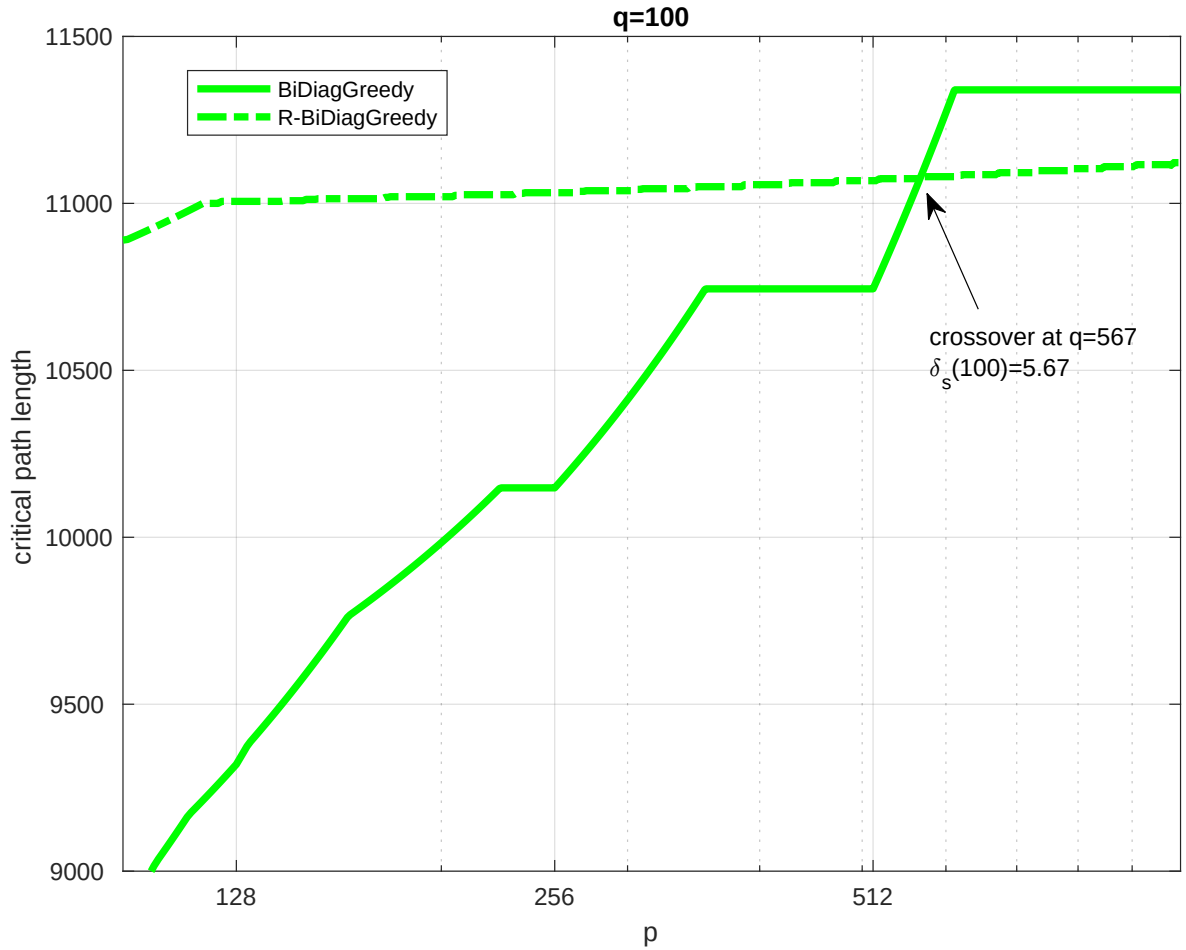
Theorem

For a matrix of p-by-q tiles, the critical path lengths of BiDiagGreedy and R-BiDiagGreedy are such that

$$\text{BiDiagGreedy}(p,q) = 6q\log_2(p) + 6q\log_2(q) + \mathcal{O}(\max(p,q))$$

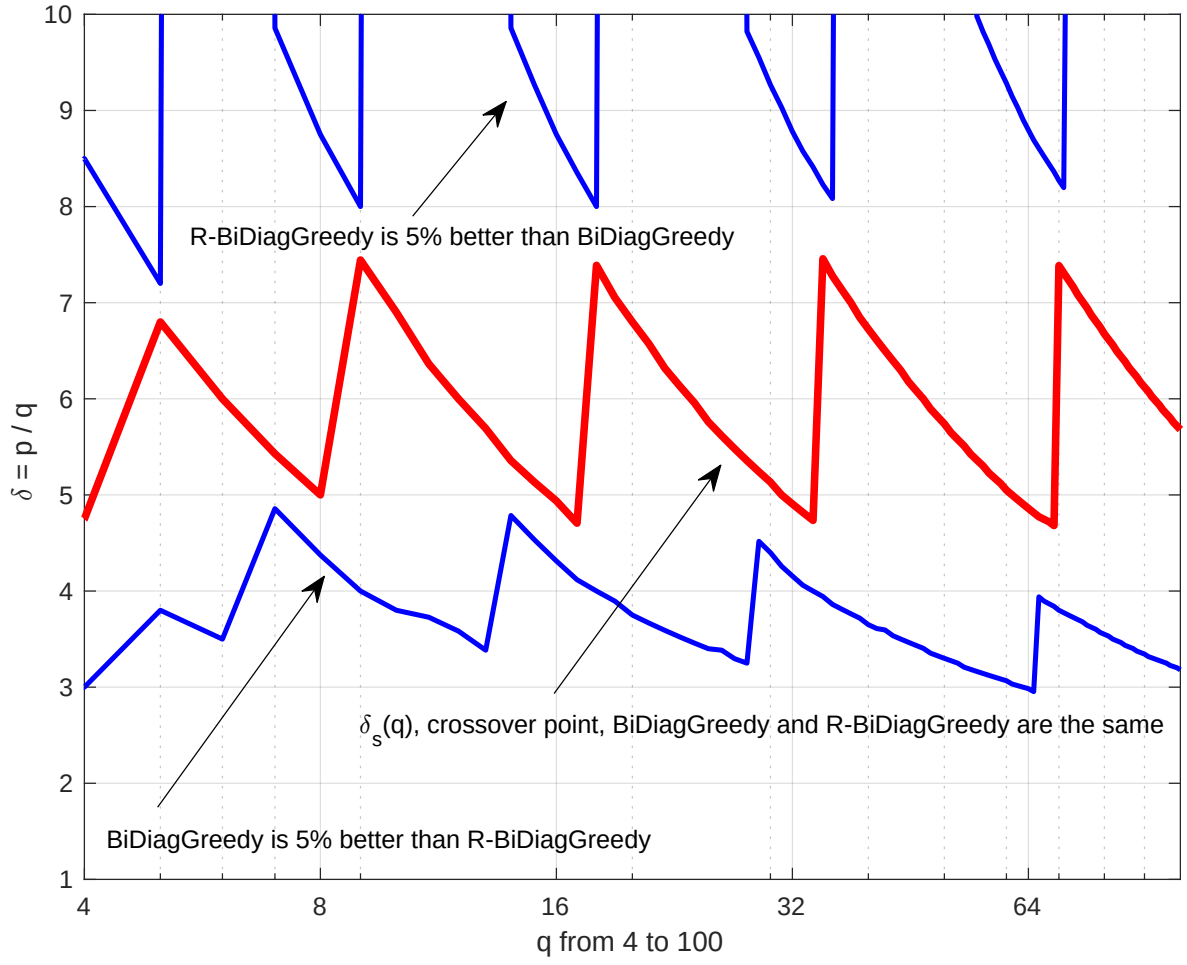
$$r\text{BiDiagGreedy}(p,q) \leq 6\lceil\sqrt{2p}\rceil + 12q\log_2(q) + \mathcal{O}(\max(p,q))$$

Switching between BiDiag and R-BiDiag



Critical path lengths for BiDiagGreedy and R-BiDiagGreedy for $q = 100$ as a function of p. We see that $\delta_s(100) = 5.67$.

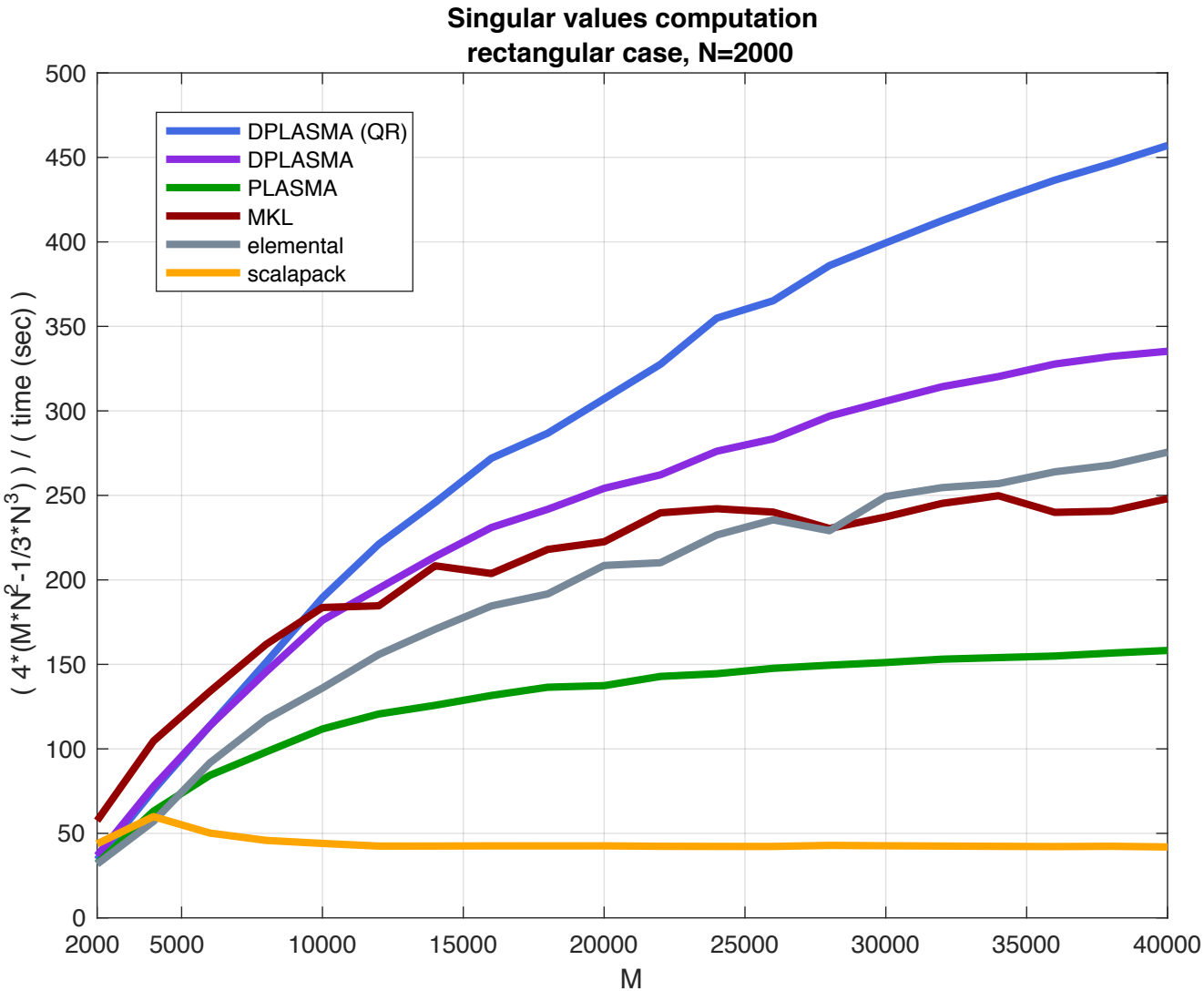
Switching between BiDiag and R-BiDiag



Values of δ_s where to switch as a function of q.

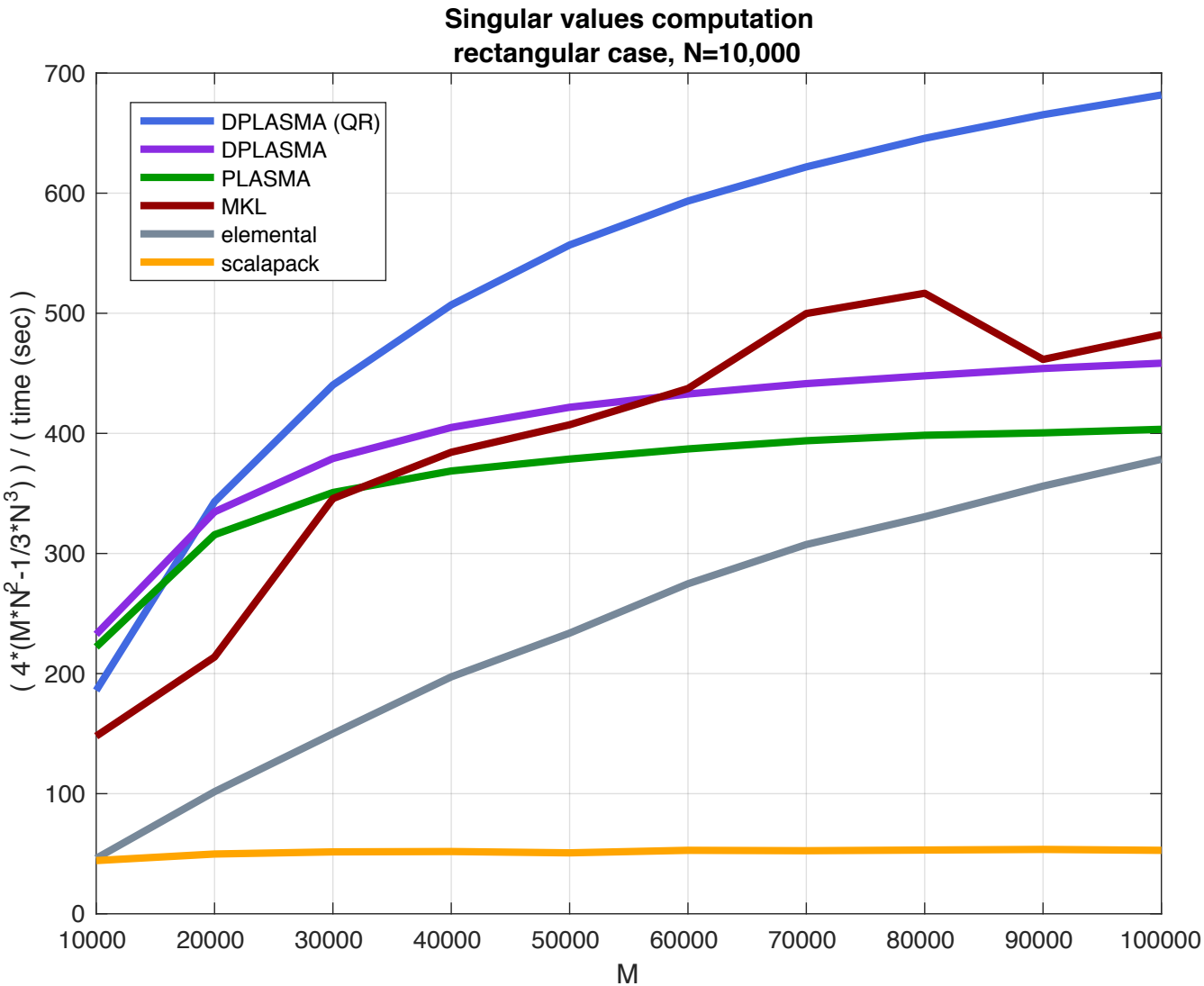
Shared Memory – GE2VAL – square – weak scalability experiment – overall performance

$(M = 2,000 \rightarrow 40,000) \times (N = 2,000)$, $(NC = 24)$

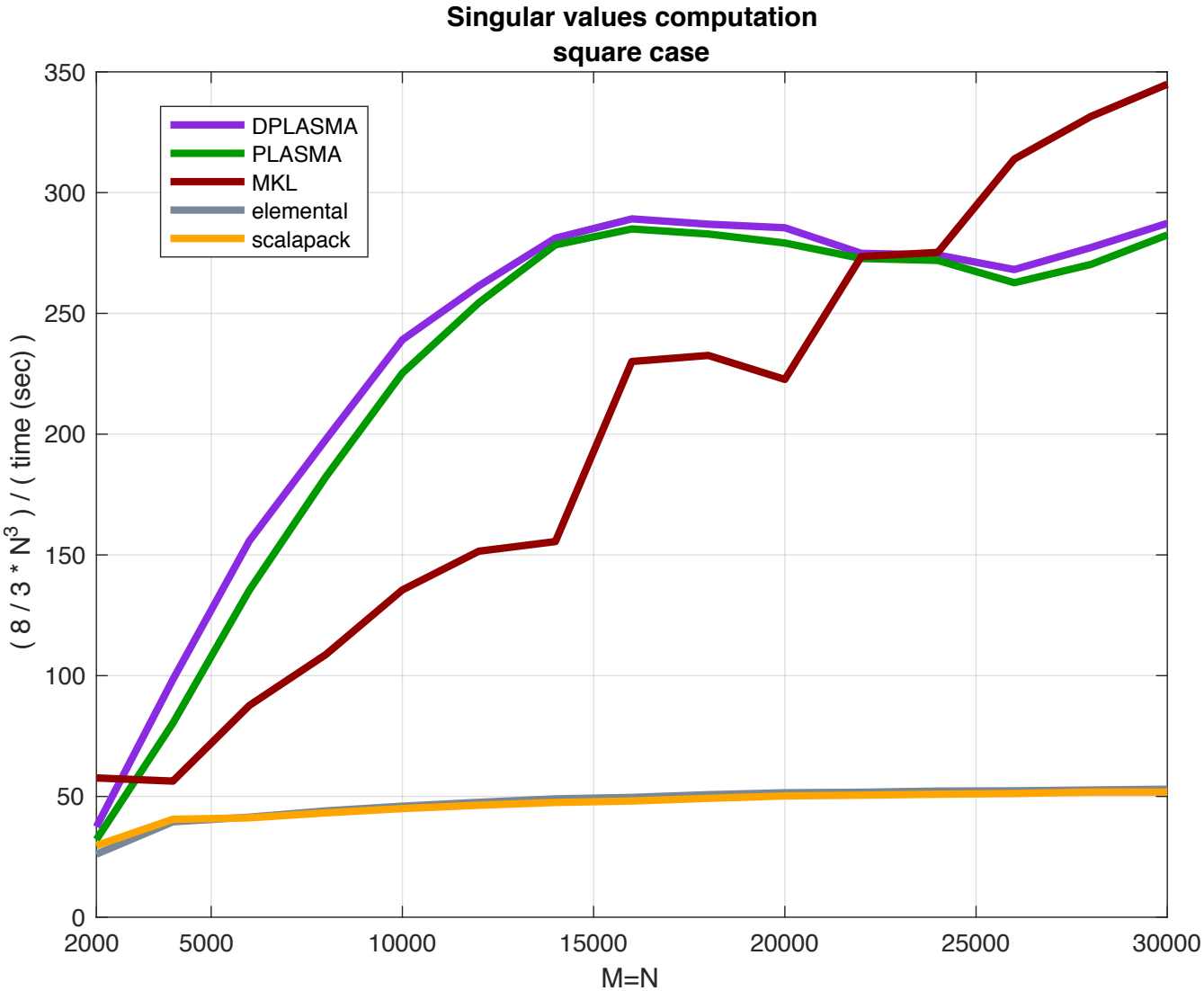


Shared Memory – GE2VAL – square – weak scalability experiment – overall performance

$(M = 10,000 \rightarrow 100,000) \times (N = 10,000)$, $(NC = 24)$

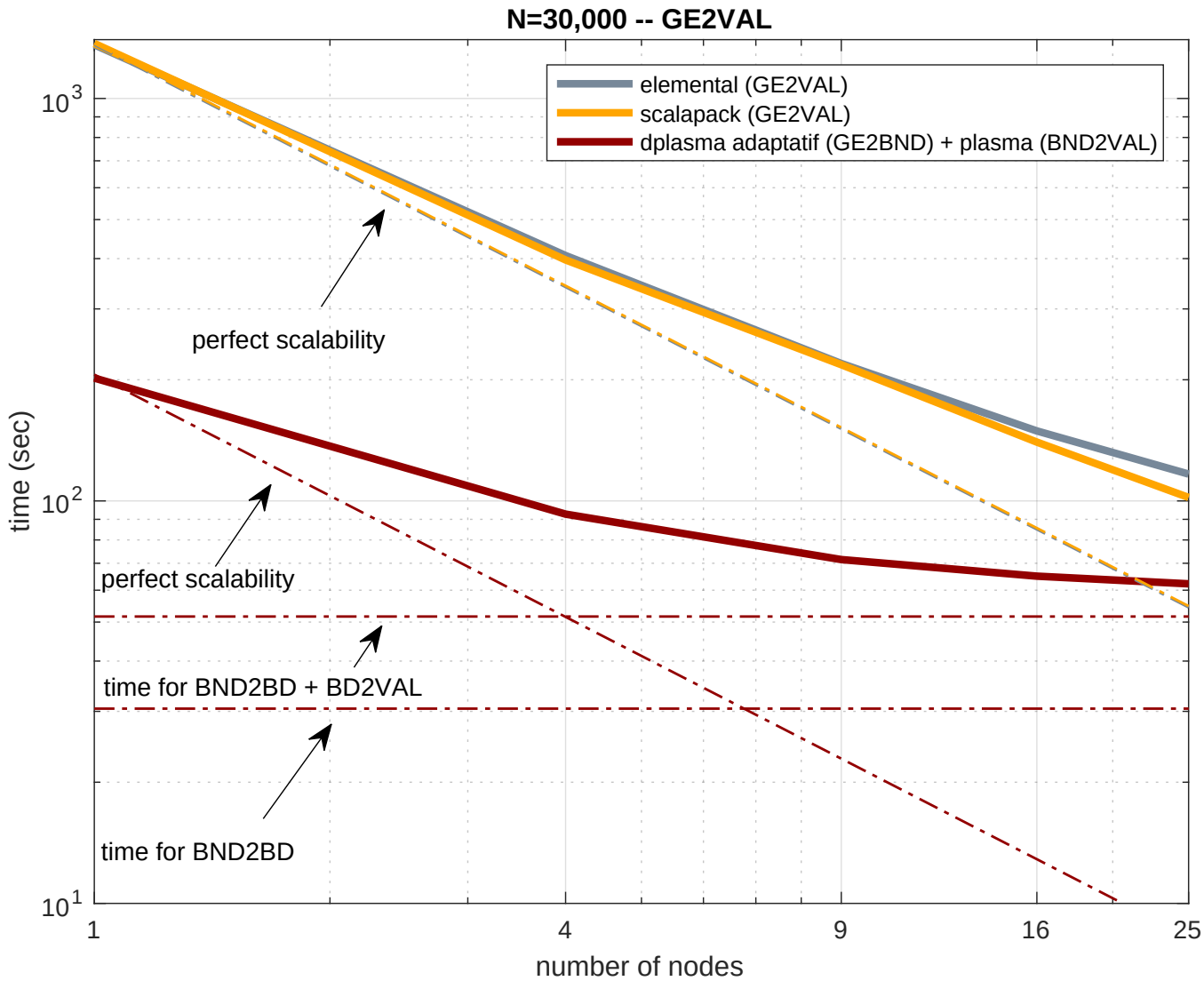


Shared Memory – GE2VAL – square – weak scalability experiment – overall performance ($M = N = 2,000 \rightarrow 30,000$), ($NC = 24$)



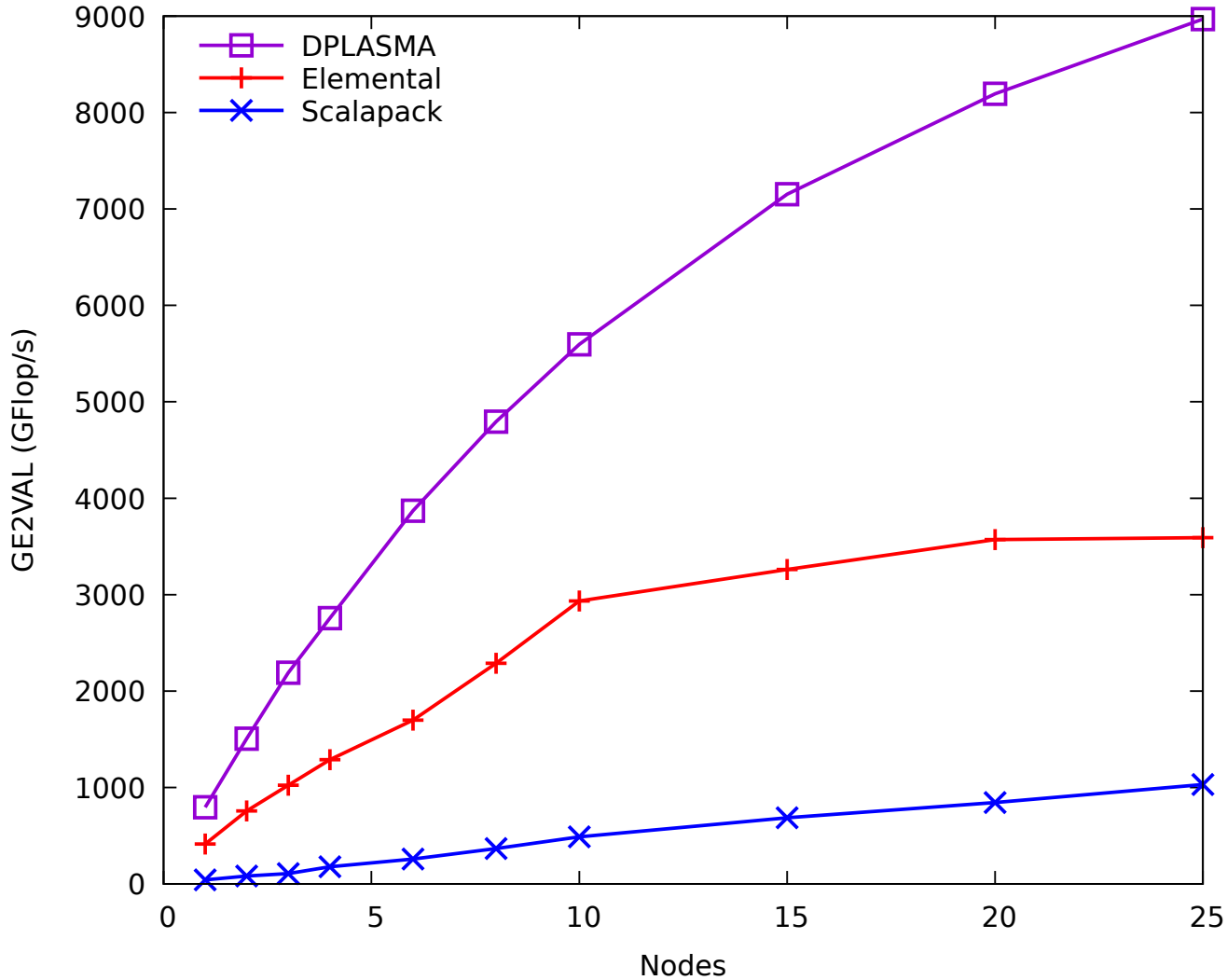
Parallel Distributed – GE2VAL – square – strong scalability experiment – time to solution

$M = N = 30,000$, $(NP = 1 \rightarrow 25)$



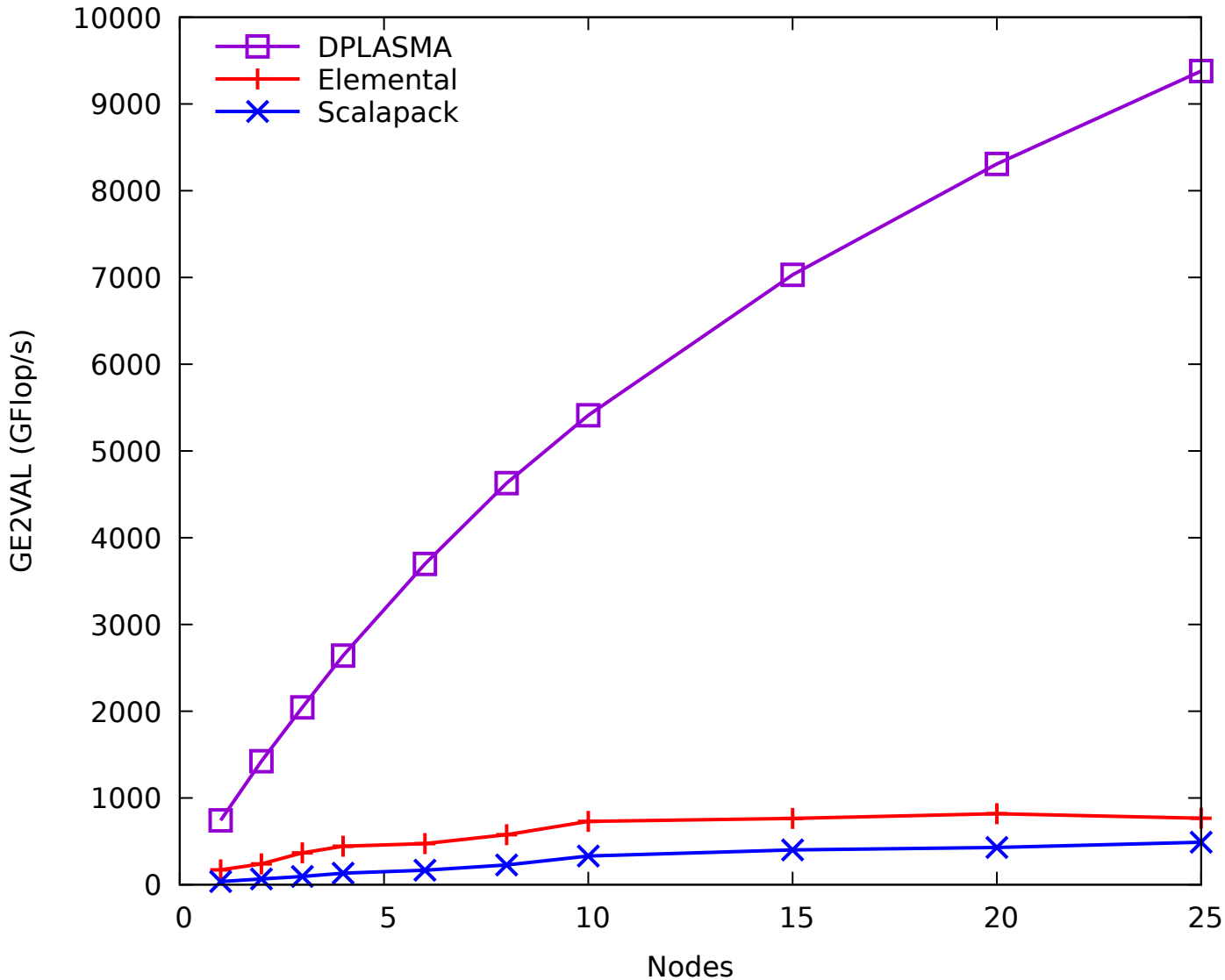
Parallel Distributed – GE2VAL – rectangular – strong scalability experiment – overall performance

$M = 1,000,000$ – $N = 10,000$ – $(NP = 1 \rightarrow 25)$



Parallel Distributed – GE2VAL – rectangular – strong scalability experiment – overall performance

$M = 2,000,000$ – $N = 2,000$ – $(NP = 1 \rightarrow 25)$



Future work

- Improve PLASMA and ScaLAPACK BiDiag by adding a QR factorization pre-processing when appropriate

Future work

- Improve PLASMA and ScaLAPACK BiDiag by adding a QR factorization pre-processing when appropriate
- Compare with "basic" Level 3 BLAS "à la" LAPACK Band Bidiagonalization algorithm (Großer and Lang, 1999)

Future work

- Improve PLASMA and ScaLAPACK BiDiag by adding a QR factorization pre-processing when appropriate
- Compare with "basic" Level 3 BLAS "à la" LAPACK Band Bidiagonalization algorithm (Großer and Lang, 1999)
- Obtain a scalable parallel distributed implementation of BND2BD (e.g., Großer and Lang, 1999) and BD2VAL (e.g., ScaLAPACK) in order to have a scalable parallel distributed GE2VAL solver

Future work

- Improve PLASMA and ScaLAPACK BiDiag by adding a QR factorization pre-processing when appropriate
- Compare with "basic" Level 3 BLAS "à la" LAPACK Band Bidiagonalization algorithm (Großer and Lang, 1999)
- Obtain a scalable parallel distributed implementation of BND2BD (e.g., Großer and Lang, 1999) and BD2VAL (e.g., ScaLAPACK) in order to have a scalable parallel distributed GE2VAL solver
- Study when singular vectors are requested. (In particular in this case, some BiDiag trees will be able to pipeline.)

Future work

- Improve PLASMA and ScaLAPACK BiDiag by adding a QR factorization pre-processing when appropriate
- Compare with "basic" Level 3 BLAS "à la" LAPACK Band Bidiagonalization algorithm (Großer and Lang, 1999)
- Obtain a scalable parallel distributed implementation of BND2BD (e.g., Großer and Lang, 1999) and BD2VAL (e.g., ScaLAPACK) in order to have a scalable parallel distributed GE2VAL solver
- Study when singular vectors are requested. (In particular in this case, some BiDiag trees will be able to pipeline.)
- Extend these ideas to Symmetric Tridiagonalization