Complexity of matrix multiplication

(For Hierarchical matrix)

For „Usual“ matrix

- The naive multiplication algorithm for nxn matrix needs n^3 multiplications (and n^3 additions)
- Is it Optimal ?
- No! [Strassen] do better (n^log2 7) using a trick akind to Karatsuba Multiplication (for reals), best known algorithms ~n^2.35.

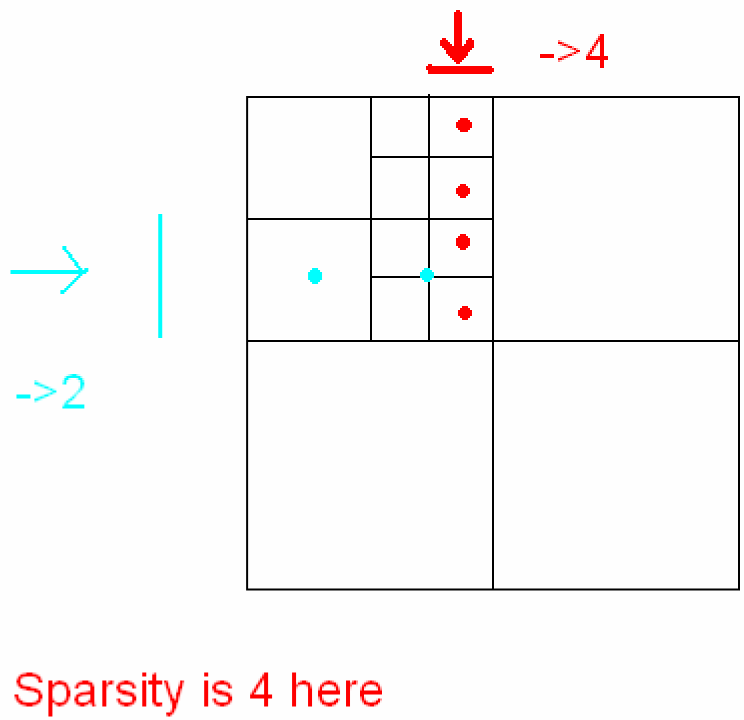
Complexity of HM

- Since the HM representation of a matrix is so flexible, we need ways to measure its complexity, to get meaningful complexity estimates of operations we want to perform.

Measures of complexity

- Nb Of Levels of the block cluster tree: p
- Rang of the admissible leaves (rkmatrix): k
- Size of the cluster tree: #I
- Max nb of nodes of some size on a row or a column: Sparsity

Ex. Of sparsity

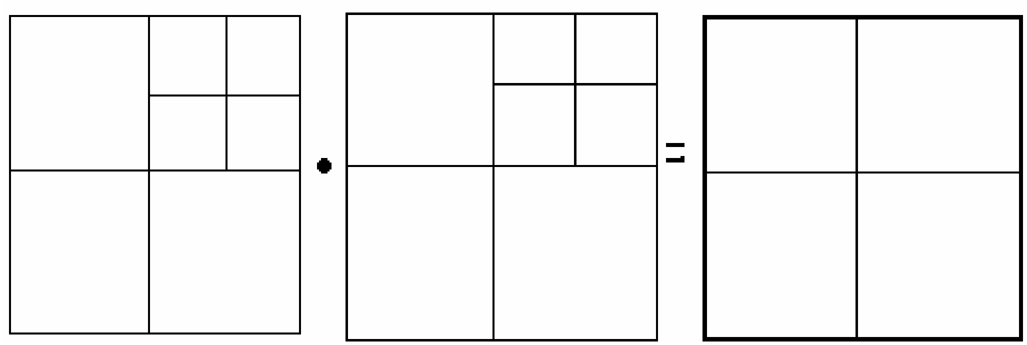


To business now!

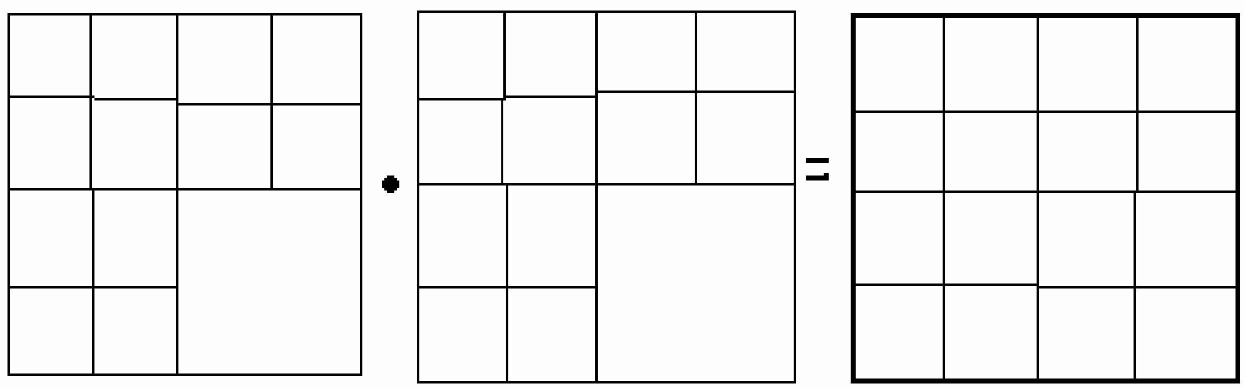
Exact multiplication of hierarchical matrices.

Structure of the product

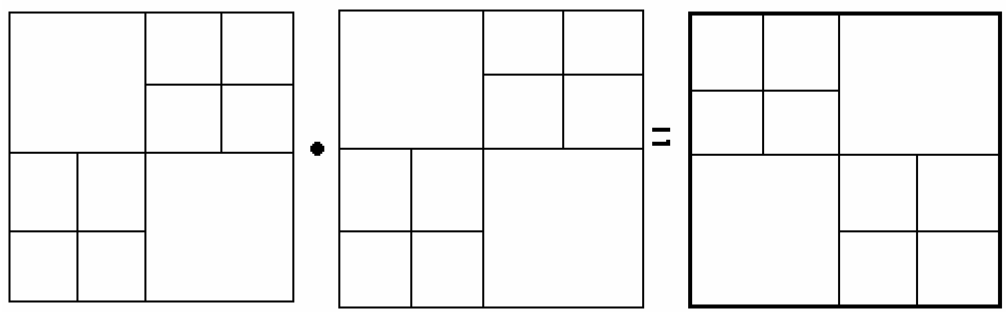
- Remember that multiplicating by a matrix of rank k you always get a matrix of rang k!!



The product can also become more complexe



Or simply different

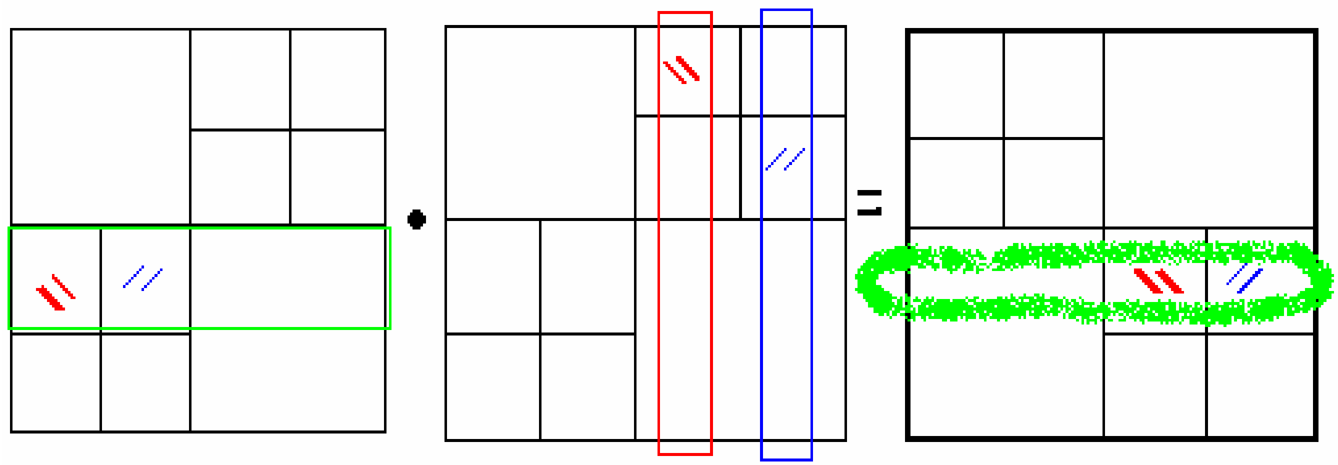


A product of tree

- We define the product of tree(s) in order to represent the tree of the product.
- T x T is based on the same cluster tree than T.
- If r x t is a node of T x T, the sons of r x t, are r‘ x t‘ with s,s‘ so that r‘ x s‘ is a son of r x s and s‘ x t‘ is a son of s x t.

The sparsity of the product

- Is smaller than the product of the sparsity.



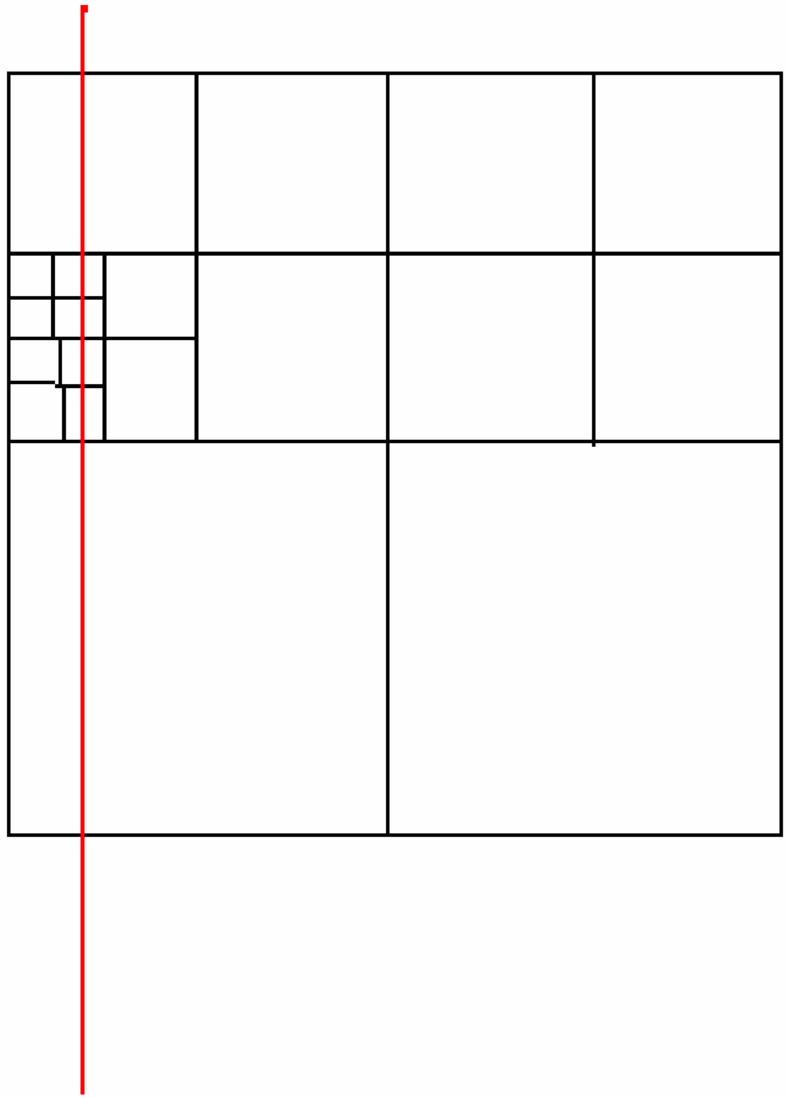
Rank of the product

- k‘ < (p+1) x sparsity x k
- The (exact) product of two hierarchical matrices of rank k on block cl. tree T is a HM on a bl. cl. tree T*T of rank k‘.
- In fact, instead of k i should perhaps write max(k, rank of full matrix).
- H(T,k) x H(T,k) -> H(T*T,k‘)

Why?

- The sum of matrix of rank a and b has rank a+b.
- To calculate the content of a leaf of T*T, we must sum at most (p+1)*sparsity products of leaves (or rather (block)minors)

I can meet at most (p+1)*sparsity leaves.



Complexity of exact multiplication

$$\leq 4(p+1) \bullet {C_{sp}}^{2} \bullet \max(k, n_{\min}) \bullet N_{st}(T, k)$$

the proof

- 1. Expressing the problem : Summing for all leaves of T*T its „cost“.

$$\sum_{r\times t\in L(T\cdot T)} \sum_{j=0}^{p} \sum_{s\in U(r\times t,j)} k\cdot \max(2\bullet N_{st}(T_{r\times s},k),2\bullet N_{st}(T_{s\times t},k))$$

The matrix by vector product

- Depends of the complexity for the storage

$$2 \cdot N_{st}(T,k)$$

- Therefore multiplying a k-matrix by something of such a storage complexity, gives a cost of:

$$k \cdot 2 \cdot N_{st}(T,k)$$

$$\leq \sum_{r\times t\in L(T\cdot T)} 2\bullet k\bullet \max(N_{st}(T_{s\times I},k),N_{st}(T_{I\times t},k))$$

$$\leq 2\cdot k\cdot \sum_{j=0}^{p}\left(\sum_{r\times t\in L(T\cdot T,j)} N_{st}(T_{r\times I},k)+\sum_{r\times t\in L(T\cdot T,j)} N_{st}(T_{I\times t},k)\right)$$

$$\leq 4(p+1)\bullet C_{sp}{}^{2}\bullet k\bullet N_{st}(T,k)$$

What is Idempotency

- The Idempotency complexity of a bl. cl. tree is the maximum on all leaves of nb of pair of descendant (r‘,t‘) (of the leaf) so that there is s‘ with: r‘ x s‘,s‘x t‘ are in T.
- Intuitively, it measures the number of summand you will need to calculate a node in the worst case.

Rank of product

- We calculated the rank of the product by putting the product in another tree, what is the rank k‘: H(T,k) x H(T,k) -> H(T,k‘)?
- Answer:k‘< sparsity x idempotency x p x k.
- Why? By forcing the data of T*T in T!

Complexity of formatted multiplication

- What is formatted muliplication?
- Truncation of rank k‘ of the product.
- The fast truncation of rank k‘.

Decomposition of the problem

- Complexity of the exacte product
- Complexity of rkmatrx->fullmatrix

(remember that small admissible m can meld to inadmisible matrices)

- Complexity of rkmatrix->rk‘matrix by truncation
- Complexity of rkmatrix-> rk‘matrix by fast truncation

Complexity of exact product

$$\leq 4(p+1) \bullet {C_{sp}}^{2} \bullet \max(k, n_{\min}) \bullet N_{st}(T,k)$$

$$N_{st(T,k)} \leq 2 \cdot C_{sp} \cdot (p+1) \cdot \max(k, n_{\min}) \# I$$

$$\Rightarrow N_{mul} \leq 4 \cdot {C_{sp}}^{3} \cdot (p+1)^{2} \cdot k^{2} \cdot \# I$$

Total rkmatrix->fullmatrix complexity

$$\leq 4 \cdot (p+1)^2 \cdot C_{sp}{}^2 \cdot C_{id} \cdot k \cdot n_{\min} \cdot \# I$$

- Why? You must use that for one rkmatrix of size a,b you need 2*rank*a*b operations.
- And the simply use estimate from seen previously.

Complexity of truncation and fast truncation

- …were derived previously:

$$N_{format.} \leq 35 \cdot (p+1)^3 \cdot C_{sp}{}^3 \cdot C_{id}{}^3 \cdot \max(\#I, \#L(t))$$

$$N_{fastformat.} \leq 48 \cdot (p+1)^2 \cdot C_{sp}{}^2 \cdot C_{id}{}^3 + 184 \cdot (p+1) \cdot k^3 \cdot C_{sp} \cdot C_{id} \cdot \#L(t))$$

Summary for complexity of multiplication

$$Truncated\_multiplication:$$

$$\leq 43 \cdot (p+1)^3 \cdot C_{sp}{}^3 \cdot C_{id}{}^3 \cdot \max(\# I, \# L(t))$$

$$Fast\_truncated\_multiplication:$$

$$\leq 48 \cdot (p+1)^2 \cdot C_{sp}{}^2 \cdot C_{id}{}^3 + 184 \cdot (p+1) \cdot k^3 \cdot C_{sp} \cdot C_{id} \cdot \# L(t))$$