

Intra-Block Amalgamation in Sparse Hypermatrix Cholesky Factorization

José R. Herrero, Juan J. Navarro

{josepr,juanjo}@ac.upc.edu

Computer Architecture Department
Universitat Politècnica de Catalunya



Barcelona

Spain

Outline

- Introduction
 - Hypermatrices
 - Small Matrix Library (SML)
 - Windows within data submatrices
- Intra-Block Amalgamation
- Results
- Conclusions

Introduction

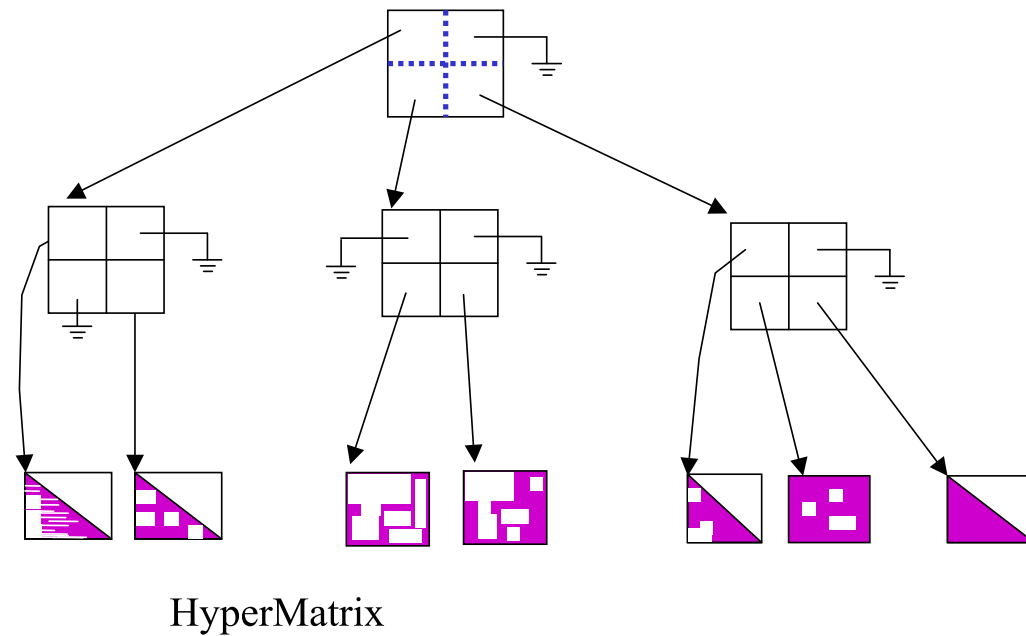
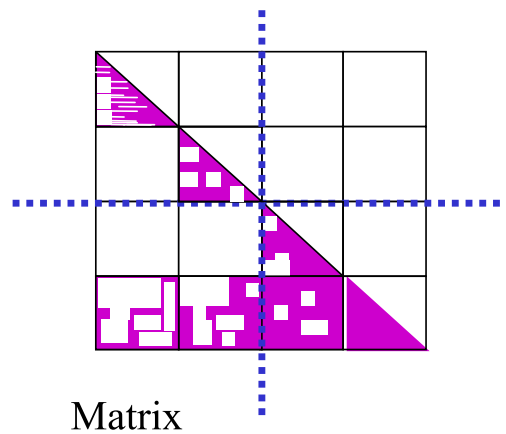
Cholesky factorization

- Symmetric Positive Definite (SPD) matrix

Sparse matrices

- Plenty of 0's
- Avoid storage and computation on 0's

A sparse matrix and an equivalent Hypermatrix



Overhead

Can store 0's within data submatrices

- Storage
- Computation

Trade-off in data submatrix size

- BLAS3 efficiency
- (Useless) operation on 0's

Goal

Efficient Implementation of a sparse Cholesky factorization

... using a hypermatrix

Important **fact**:

Matrix multiplication takes around **90%** of the total factorization time.

Reducing Overhead & Increasing Performance

Summary of (effective) previous work:

- Efficient kernels which operate on small data submatrices (SML)
- Windows within data submatrices

Efficient operation on small data submatrices

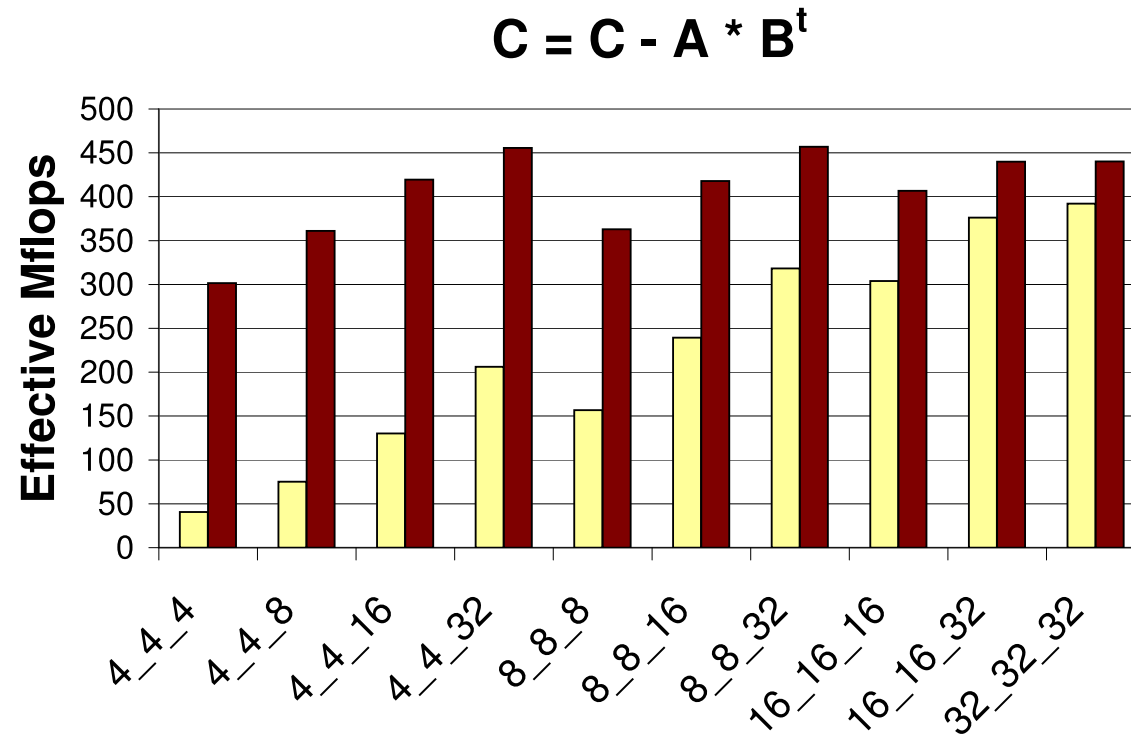
Goal:

- Reduce data submatrix size while keeping good BLAS3 performance

Idea [Euro-Par'03]:

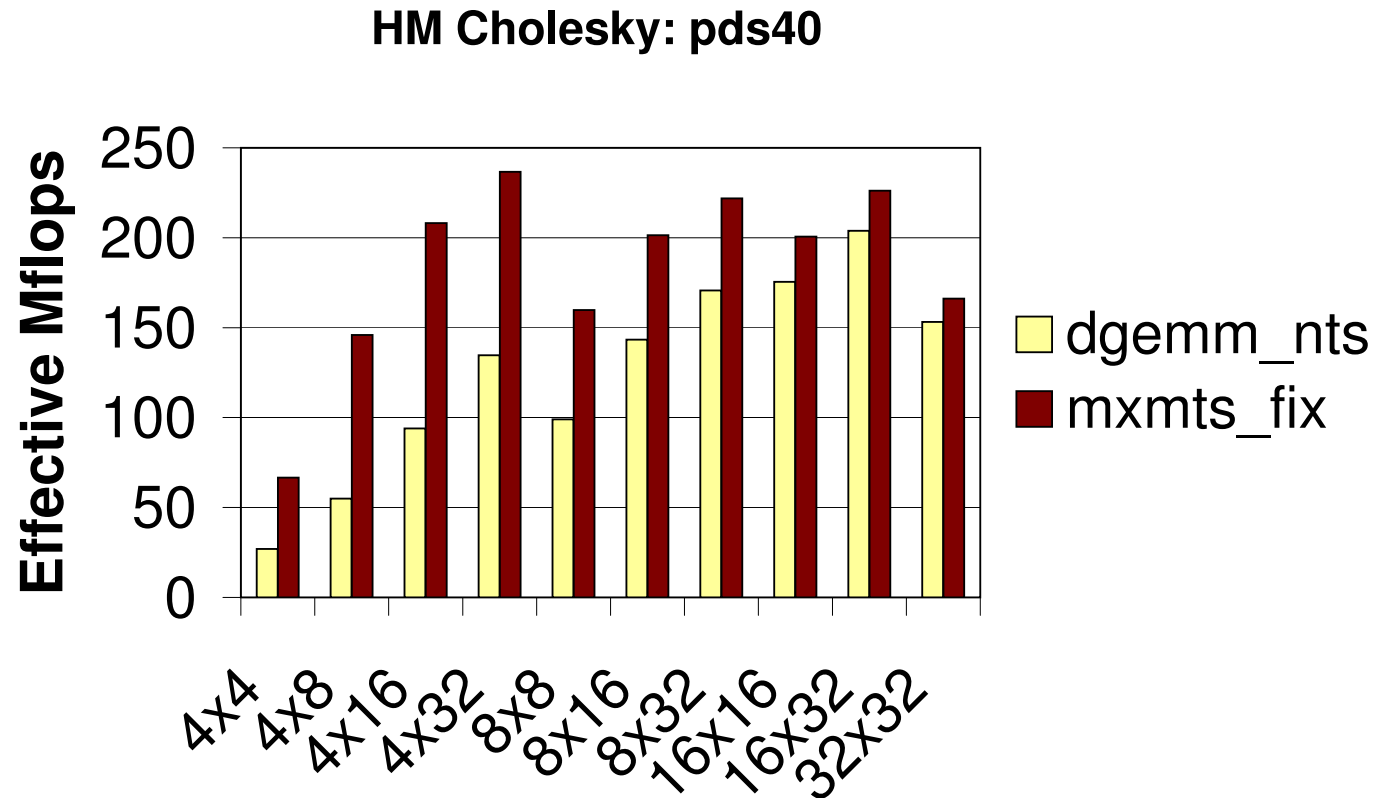
- Fix parameters at compilation time
- Choose best algorithm
 - Loop unrolling factors
 - Loop orders

Matrix multiplication performance on small matrices



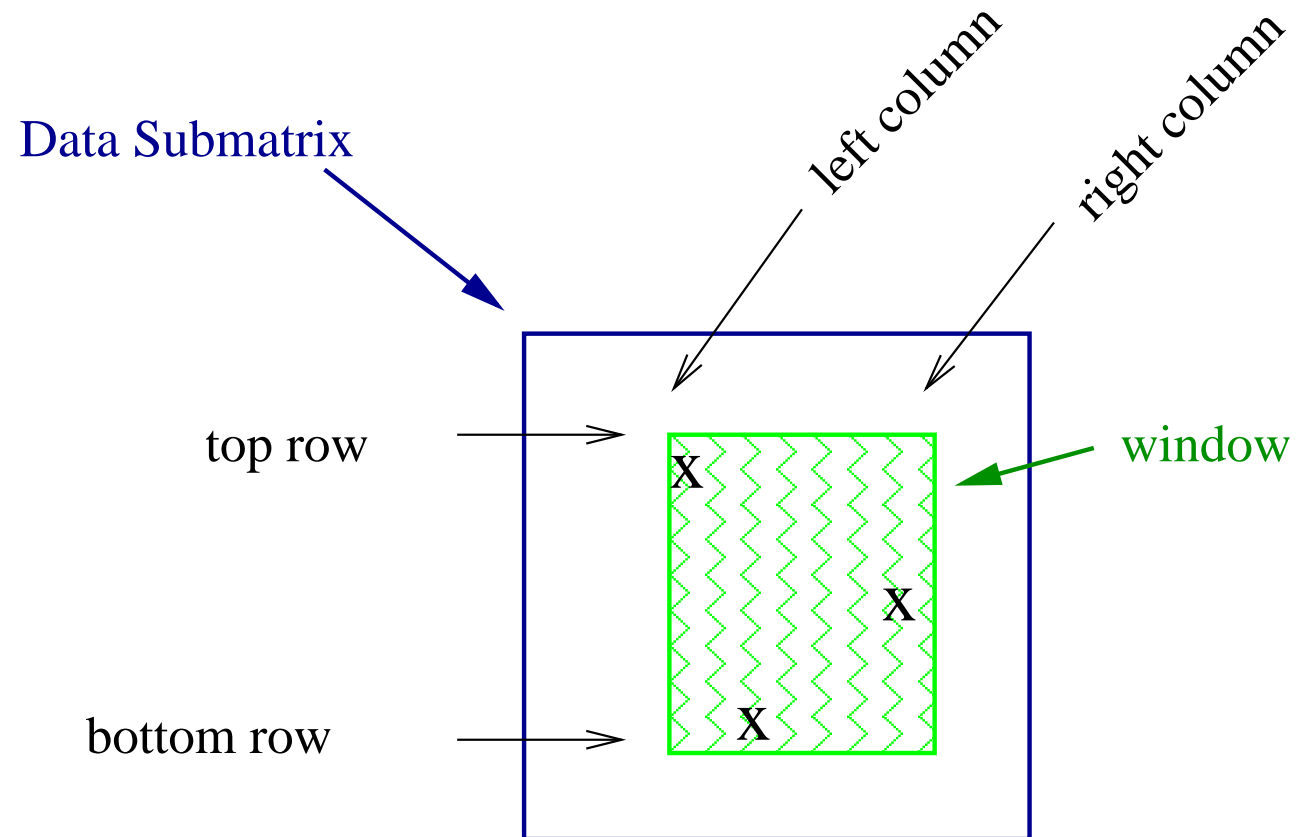
R10000 250 MHz (500 Mflops peak)

Hypermatrix Cholesky on problem PDS40



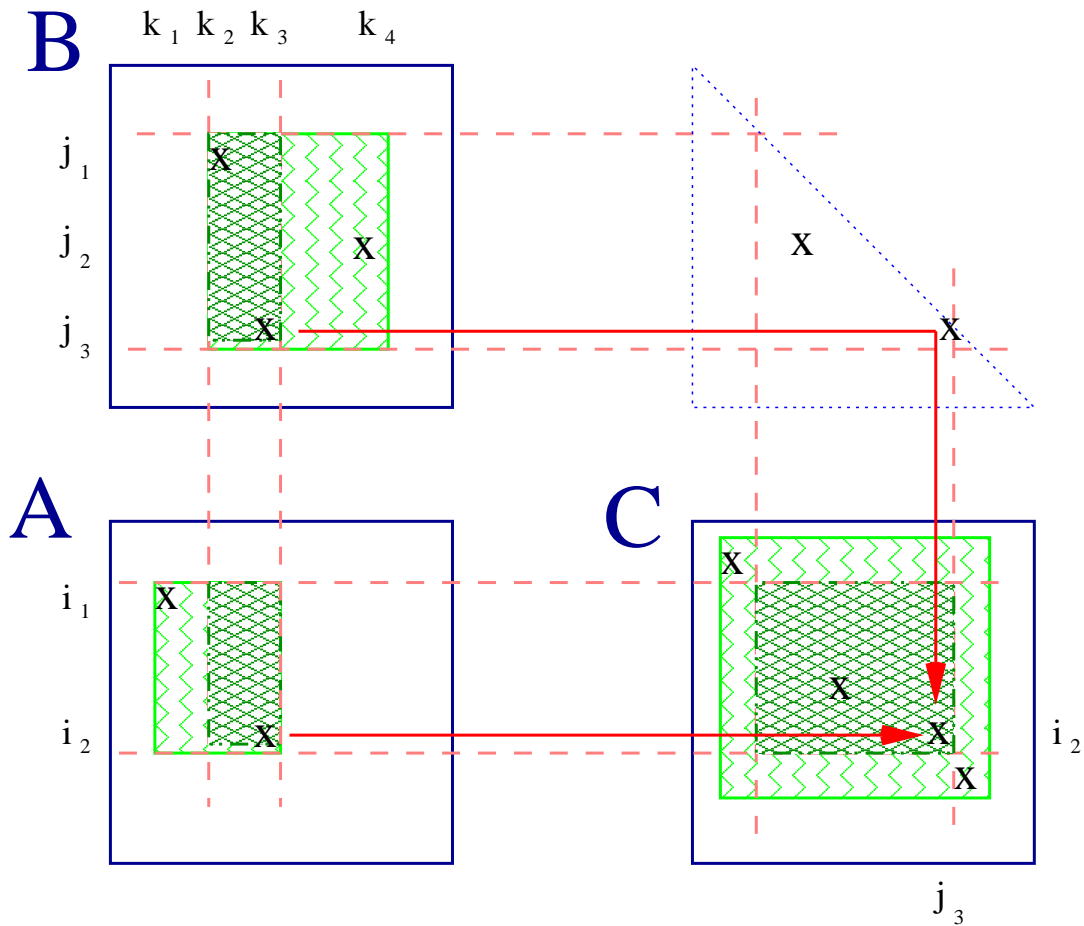
LP problem: Patient Distribution System (40 days)

Window within a data submatrix

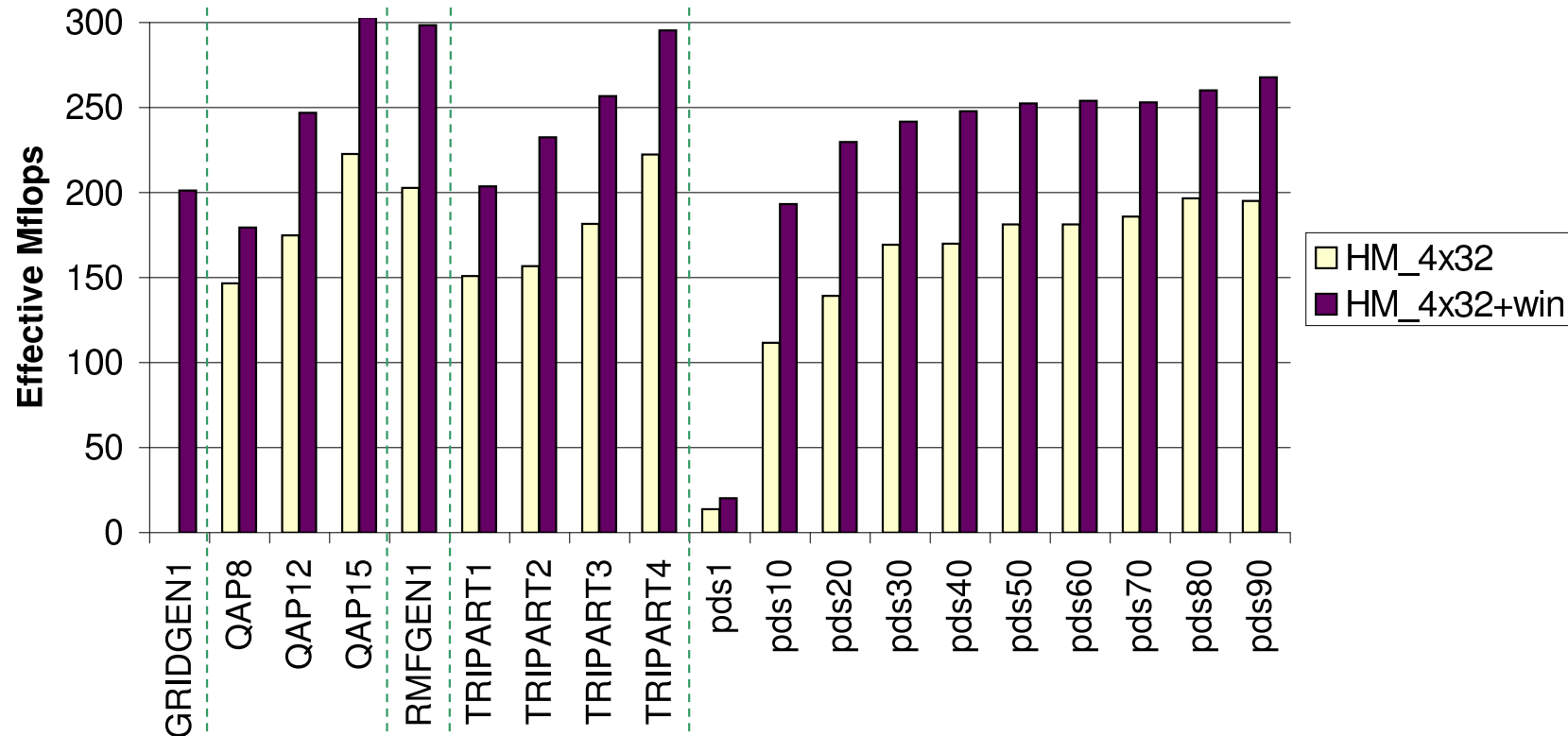


See [HPCSE'04, PARA'04]

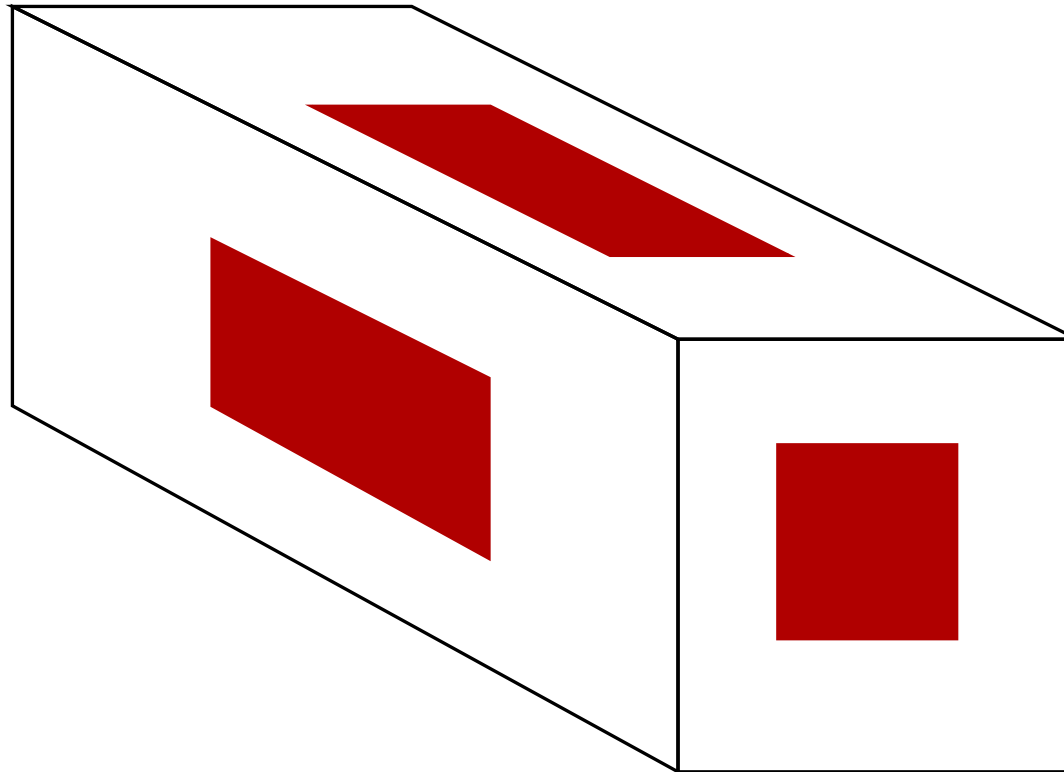
Using windows to avoid (some) useless operations



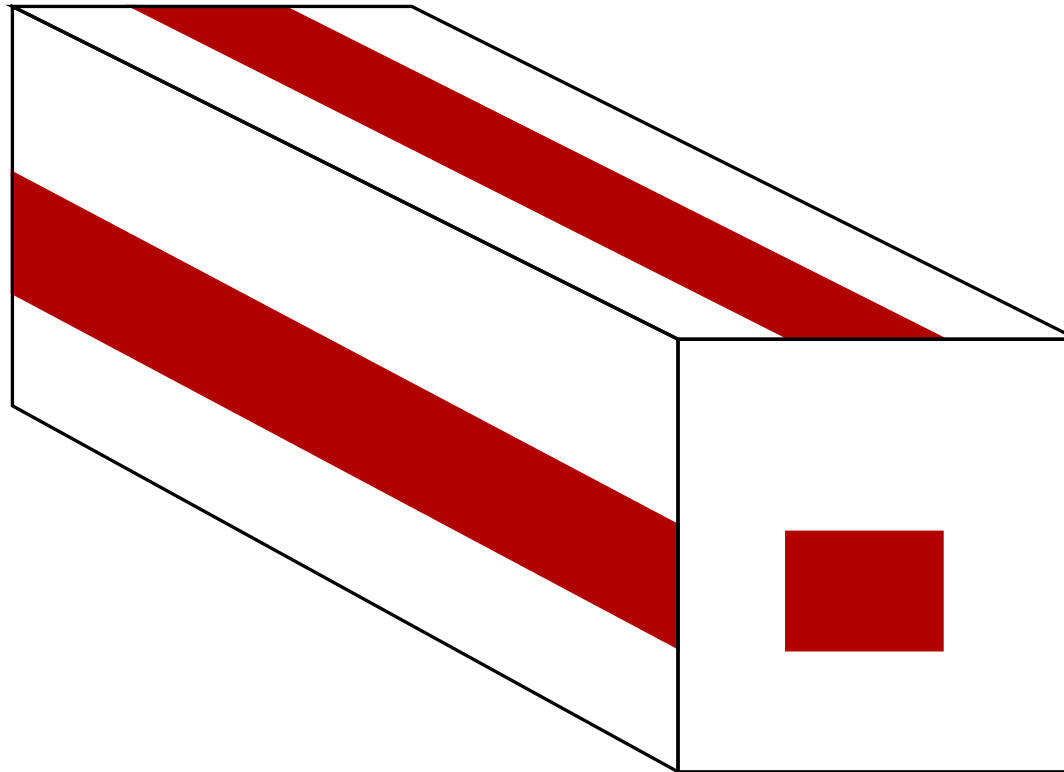
HM performance with and without windows



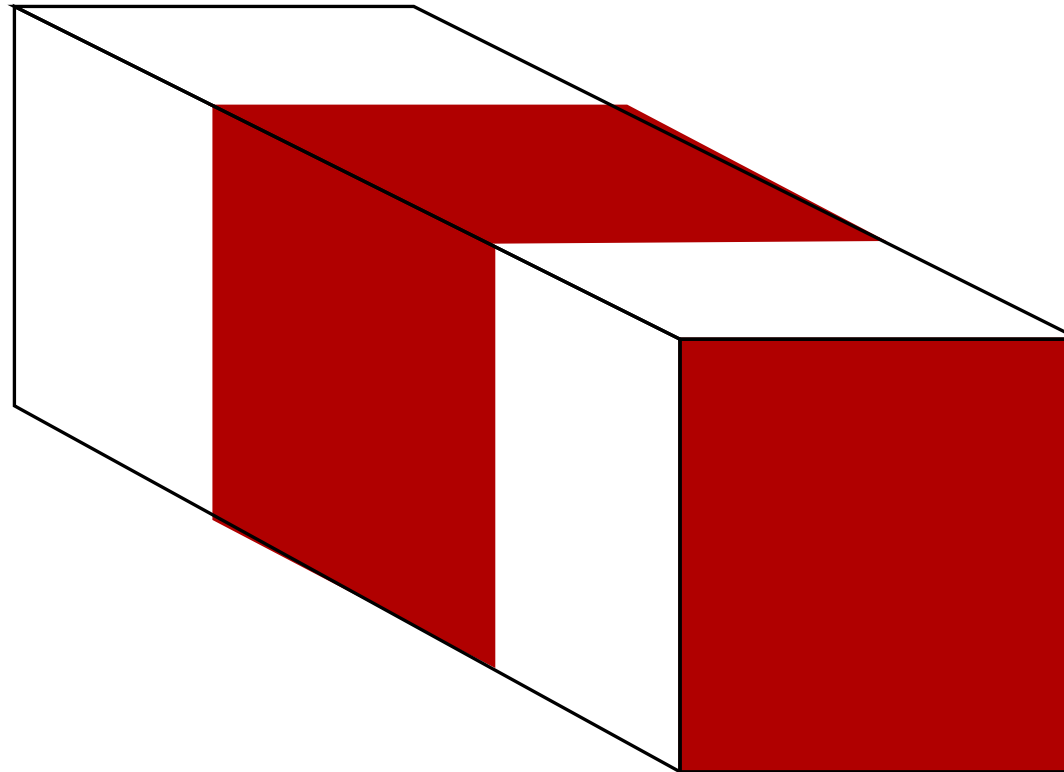
Matrix multiplication using windows in 2 dimensions



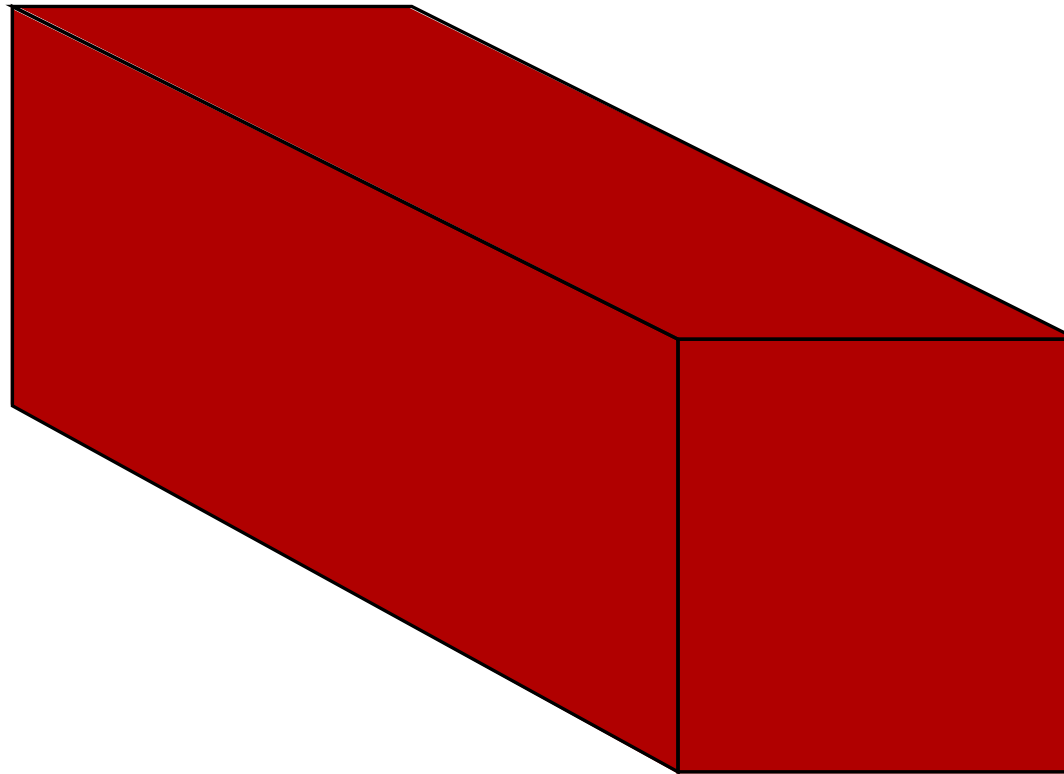
Matrix multiplication using windows in rows



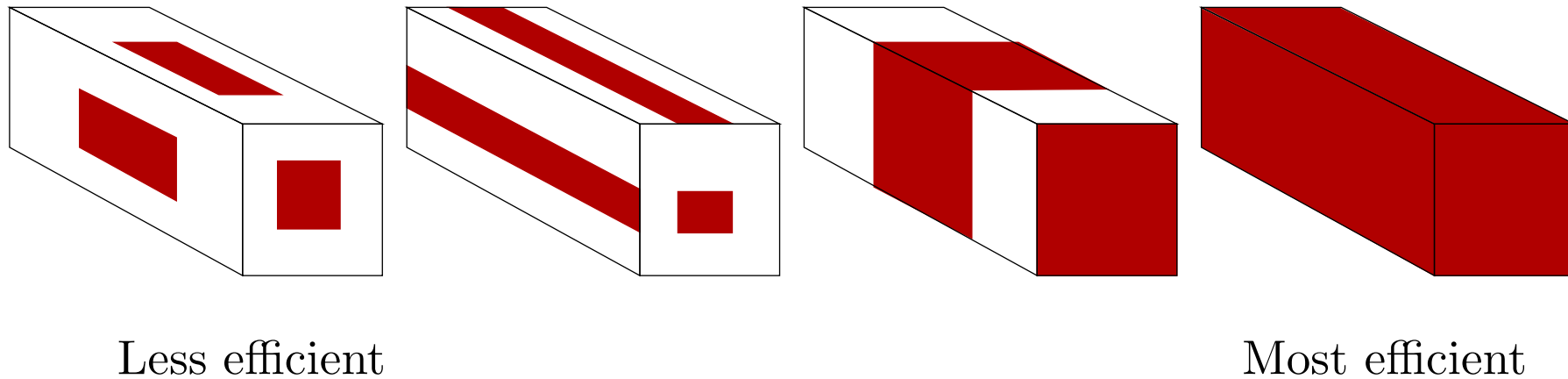
Matrix multiplication using windows in columns



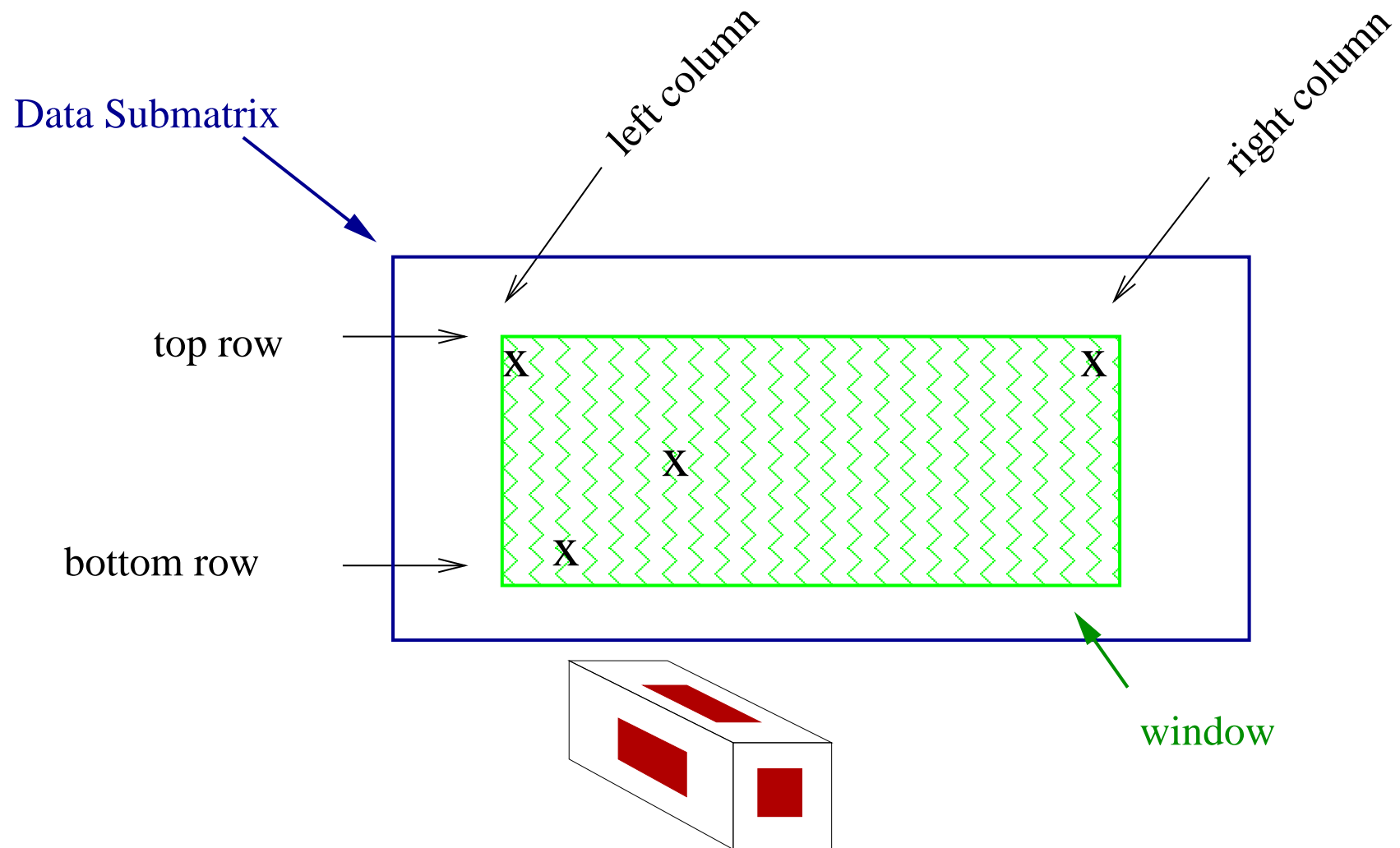
Matrix multiplication of full data submatrices



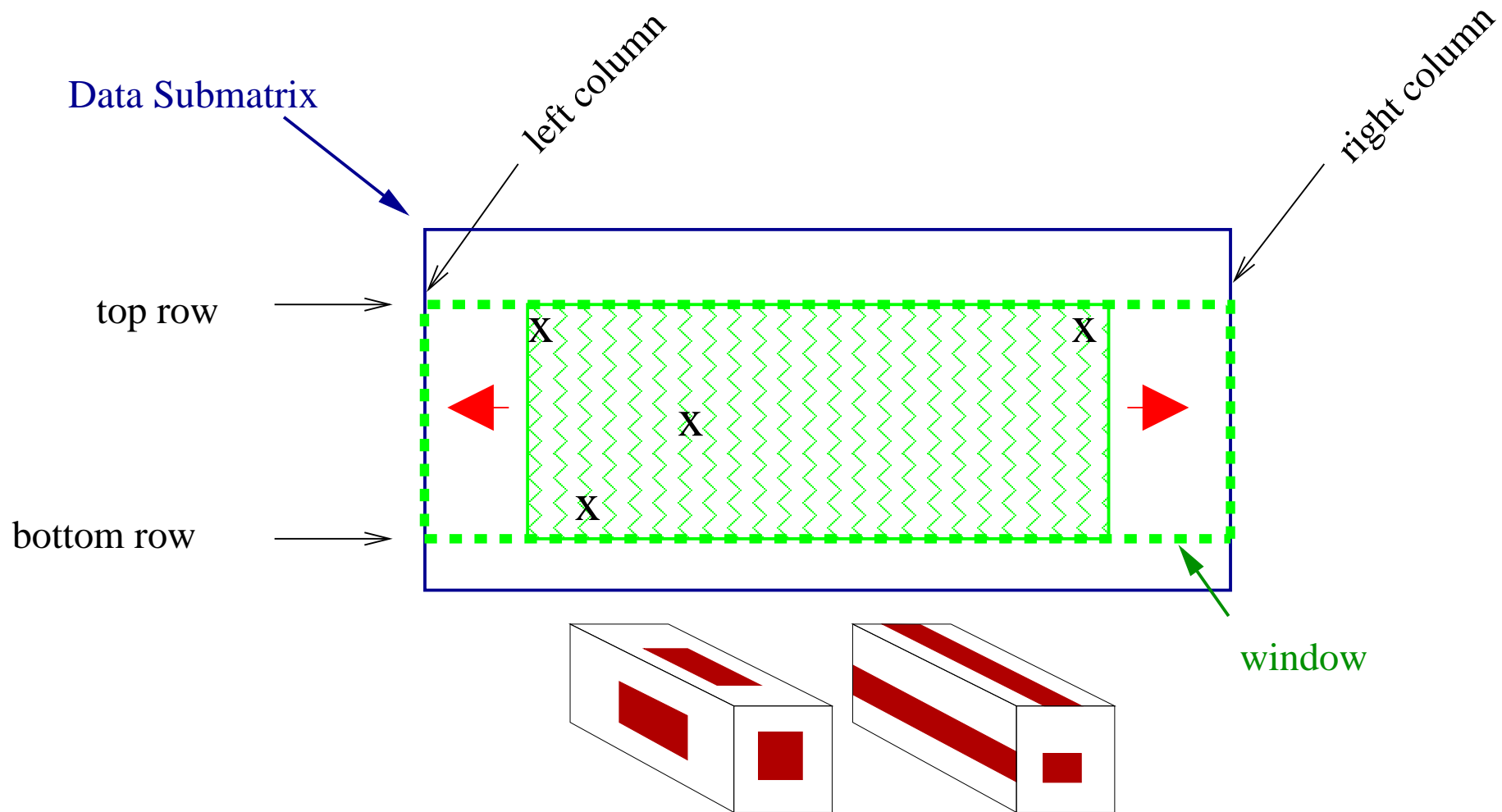
Matrix multiplication: efficiency of codes



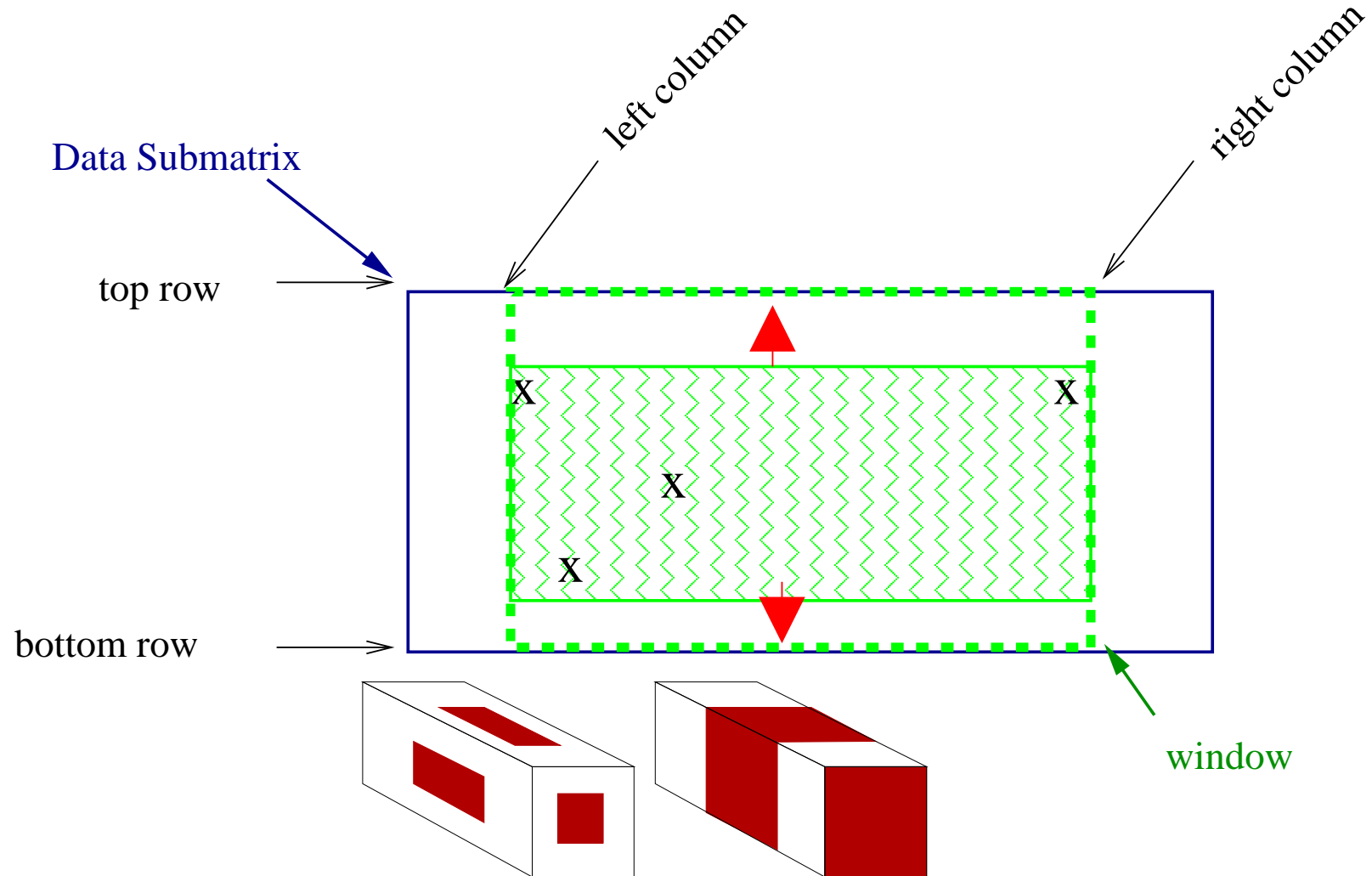
Intra-Block Amalgamation: Original window



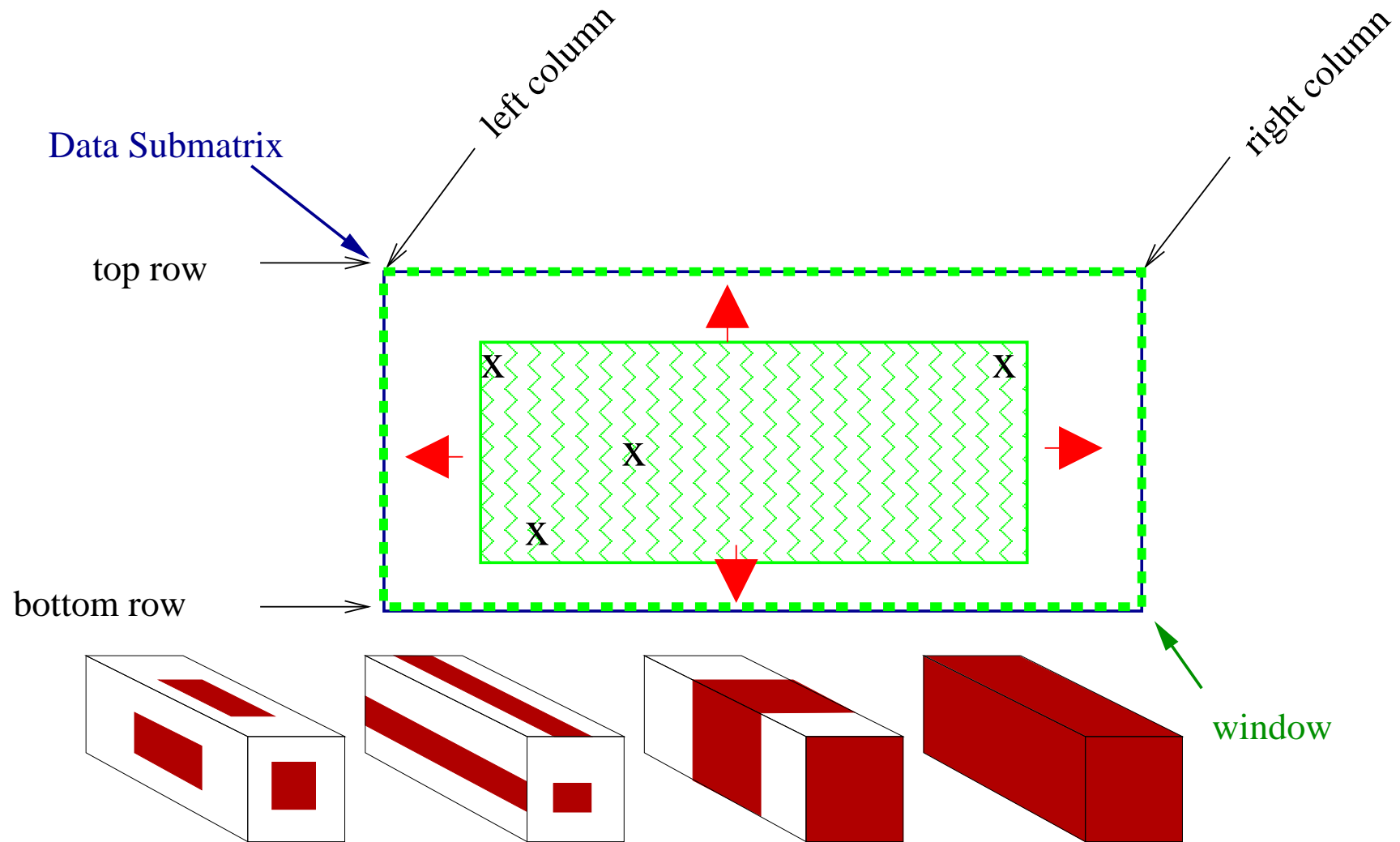
Intra-Block Amalgamation: column-wise



Intra-Block Amalgamation: row-wise



Intra-Block Amalgamation: row and column-wise



Results: Context information

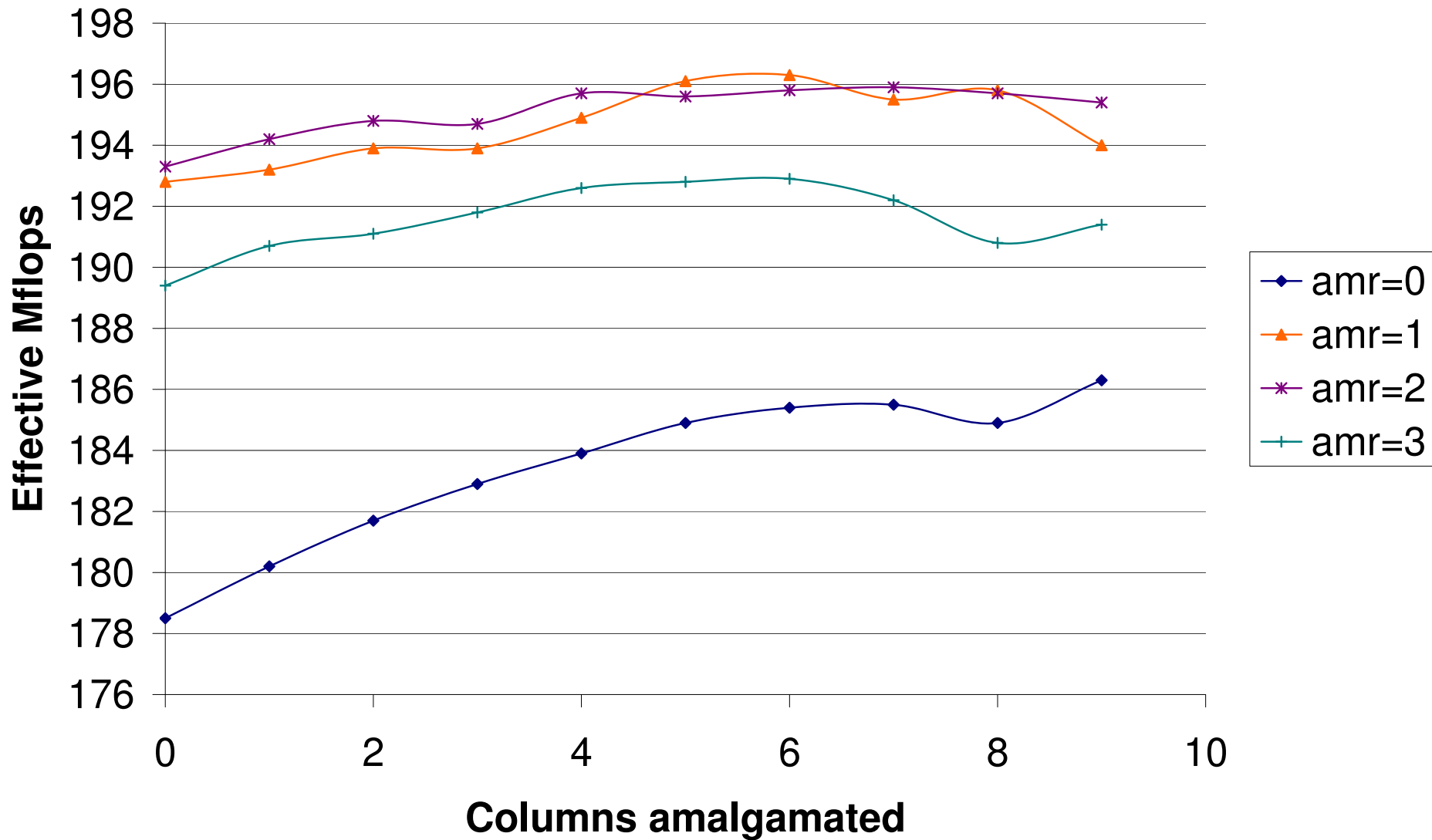
- MIPS R10000 @ 250 MHz (500 Mflops peak)
- Sequential code
- Data submatrices of fixed size
- Large problems solved In-Core
- Ordered using METIS
- Linear Programming problems
 - NetLib
 - Multicommodity Network Flow generators

Results: Matrix Characteristics

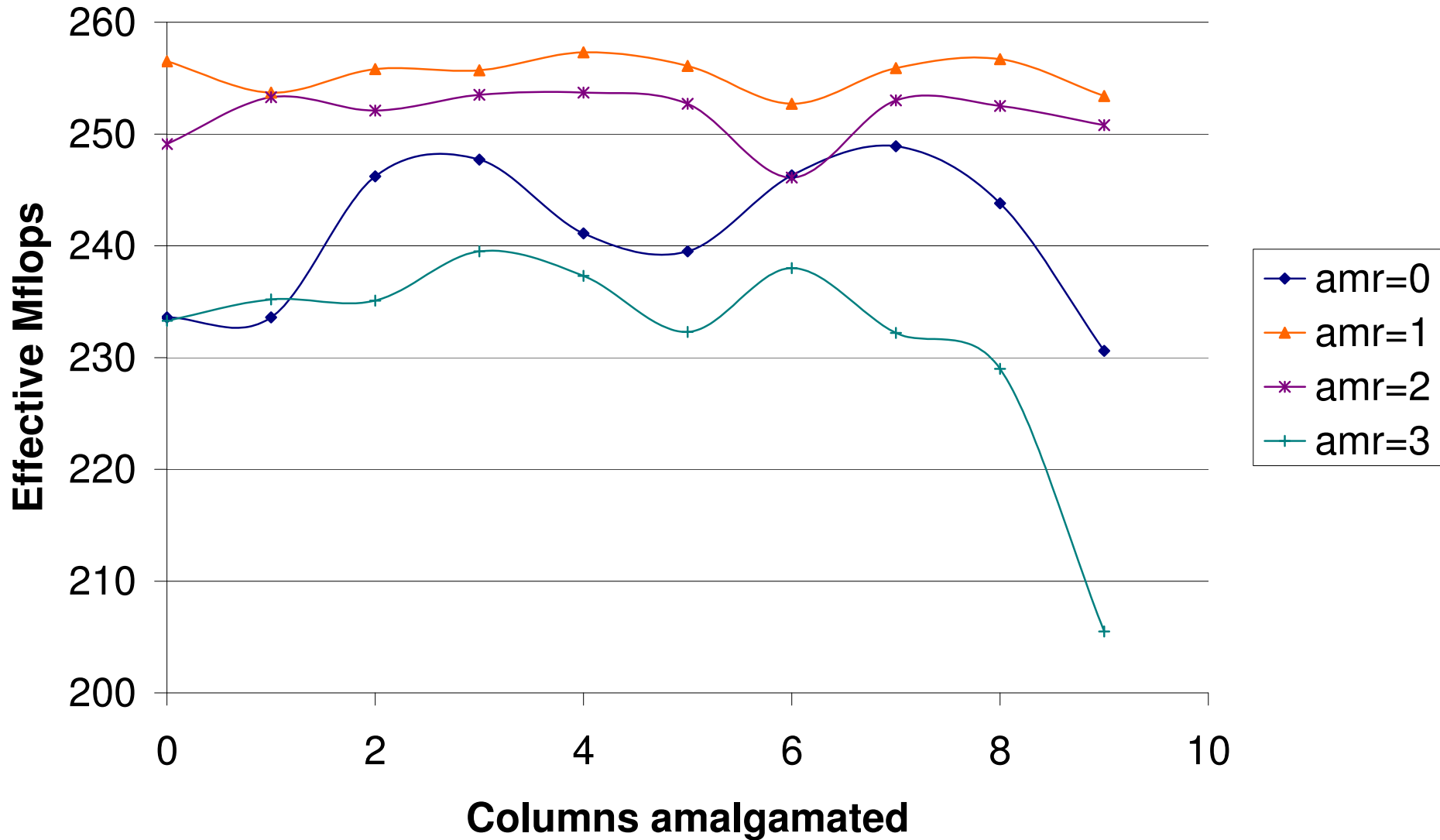
Matrix	Dimension	NZs	NZs in L ^a	Density	Flops to factor ^b
GRIDGEN1	330430	3162757	130586943	0.002	278891
QAP8	912	14864	193228	0.463	63
QAP12	3192	77784	2091706	0.410	2228
QAP15	6330	192405	8755465	0.436	20454
RMFGEN1	28077	151557	6469394	0.016	6323
TRIPART1	4238	80846	1147857	0.127	511
TRIPART2	19781	400229	5917820	0.030	2926
TRIPART3	38881	973881	17806642	0.023	14058
TRIPART4	56869	2407504	76805463	0.047	187168
pds1	1561	12165	37339	0.030	1
pds10	18612	148038	3384640	0.019	2519
pds20	38726	319041	10739539	0.014	13128
pds30	57193	463732	18216426	0.011	26262
pds40	76771	629851	27672127	0.009	43807
pds50	95936	791087	36321636	0.007	61180
pds60	115312	956906	46377926	0.006	81447
pds70	133326	1100254	54795729	0.006	100023
pds80	149558	1216223	64148298	0.005	125002
pds90	164944	1320298	70140993	0.005	138765

- ^aNumber of non-zeros in factor L (matrix ordered using METIS).
- ^bNumber of floating point operations (in Millions) necessary to obtain L from the original matrix (ordered with METIS).

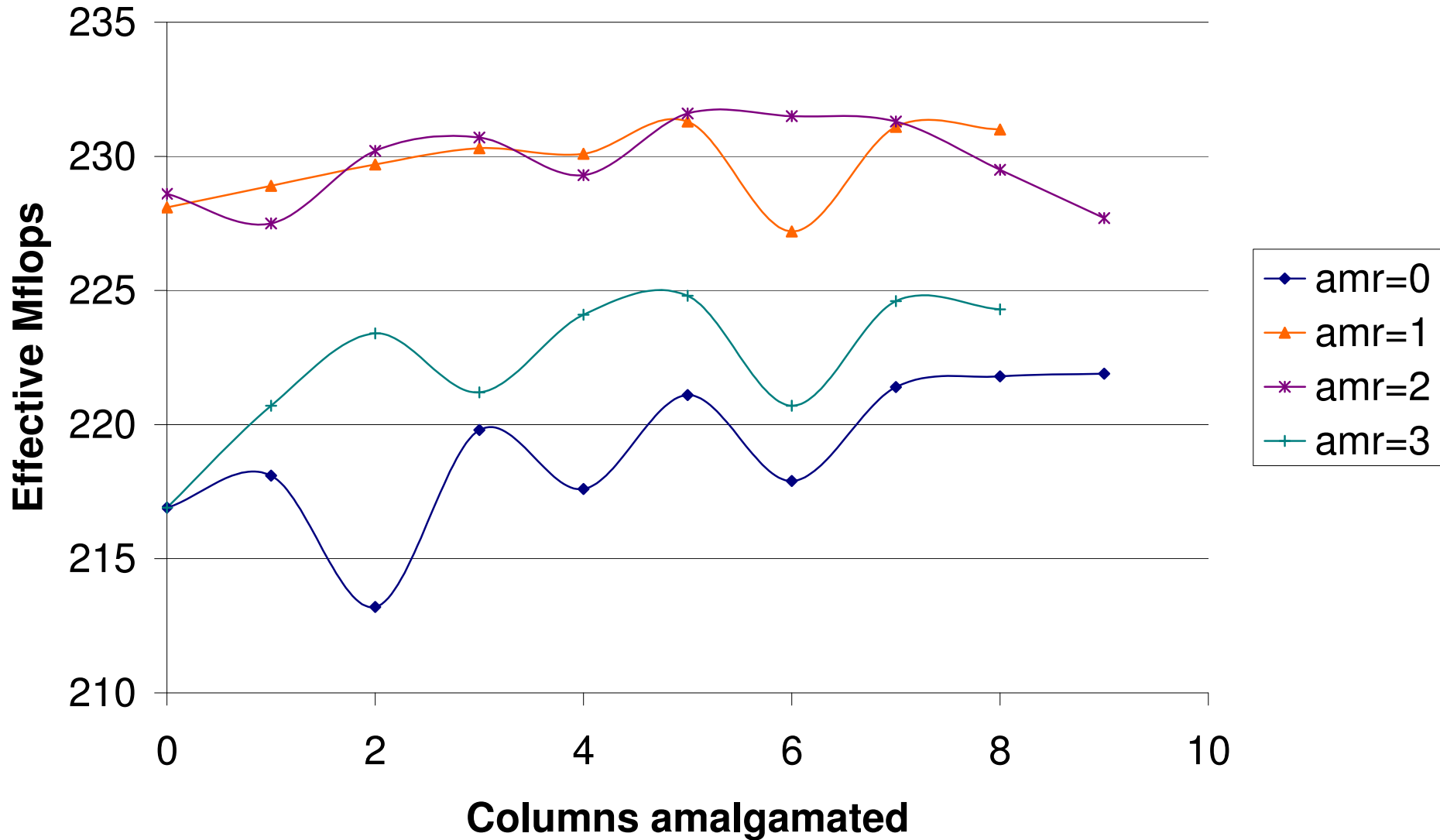
Results: QAP8



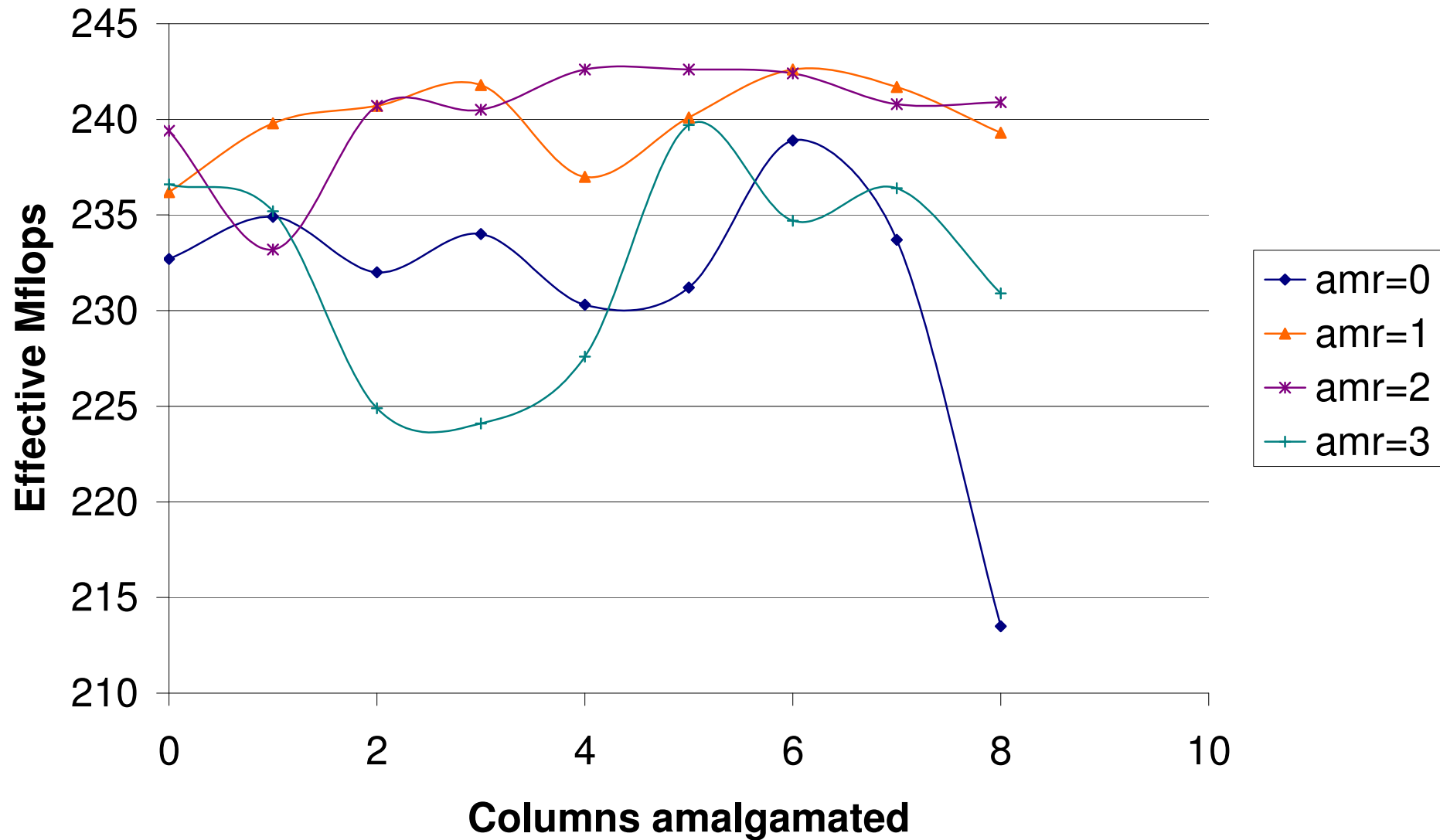
Results: QAP12



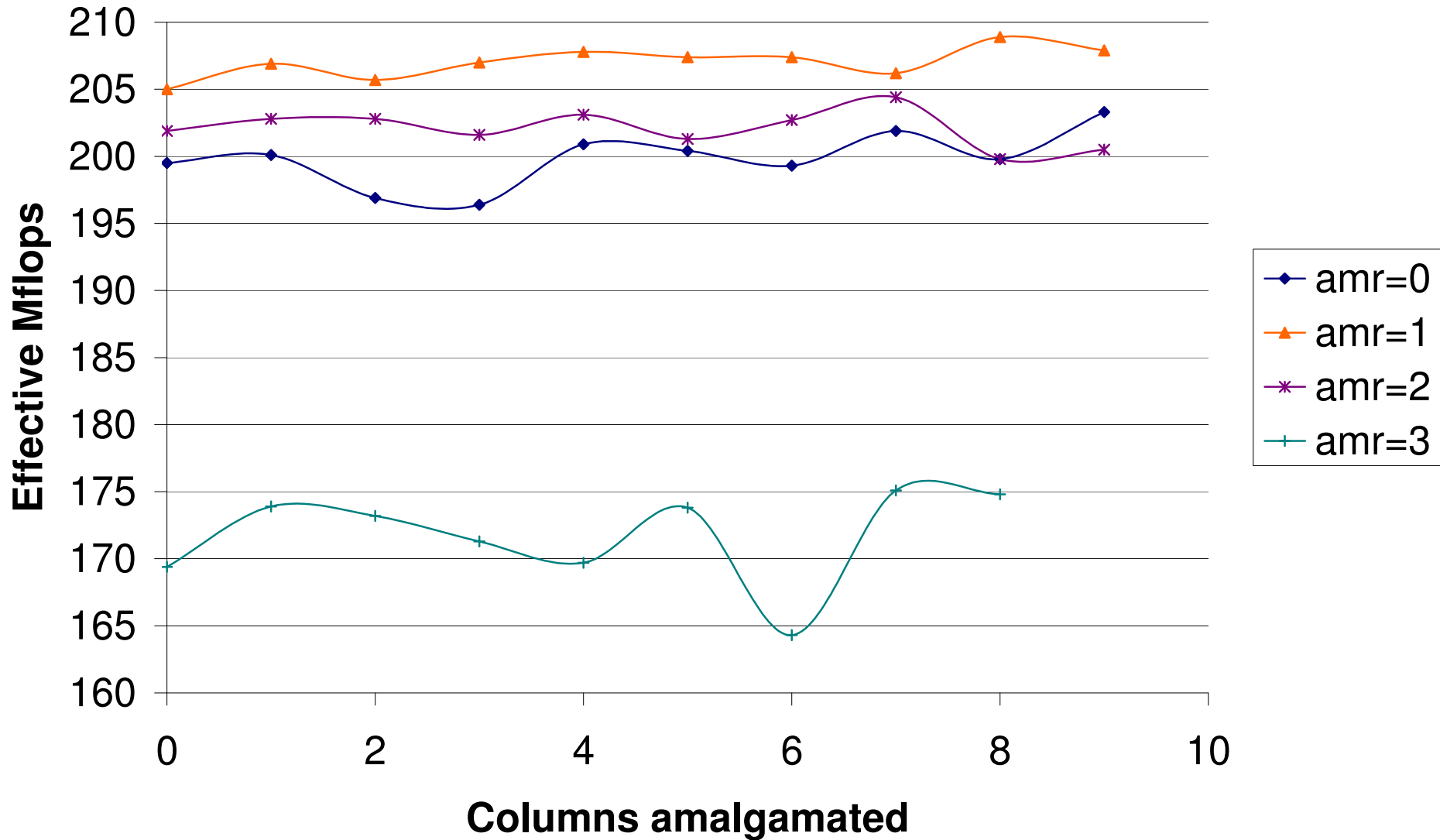
Results: TRIPART1



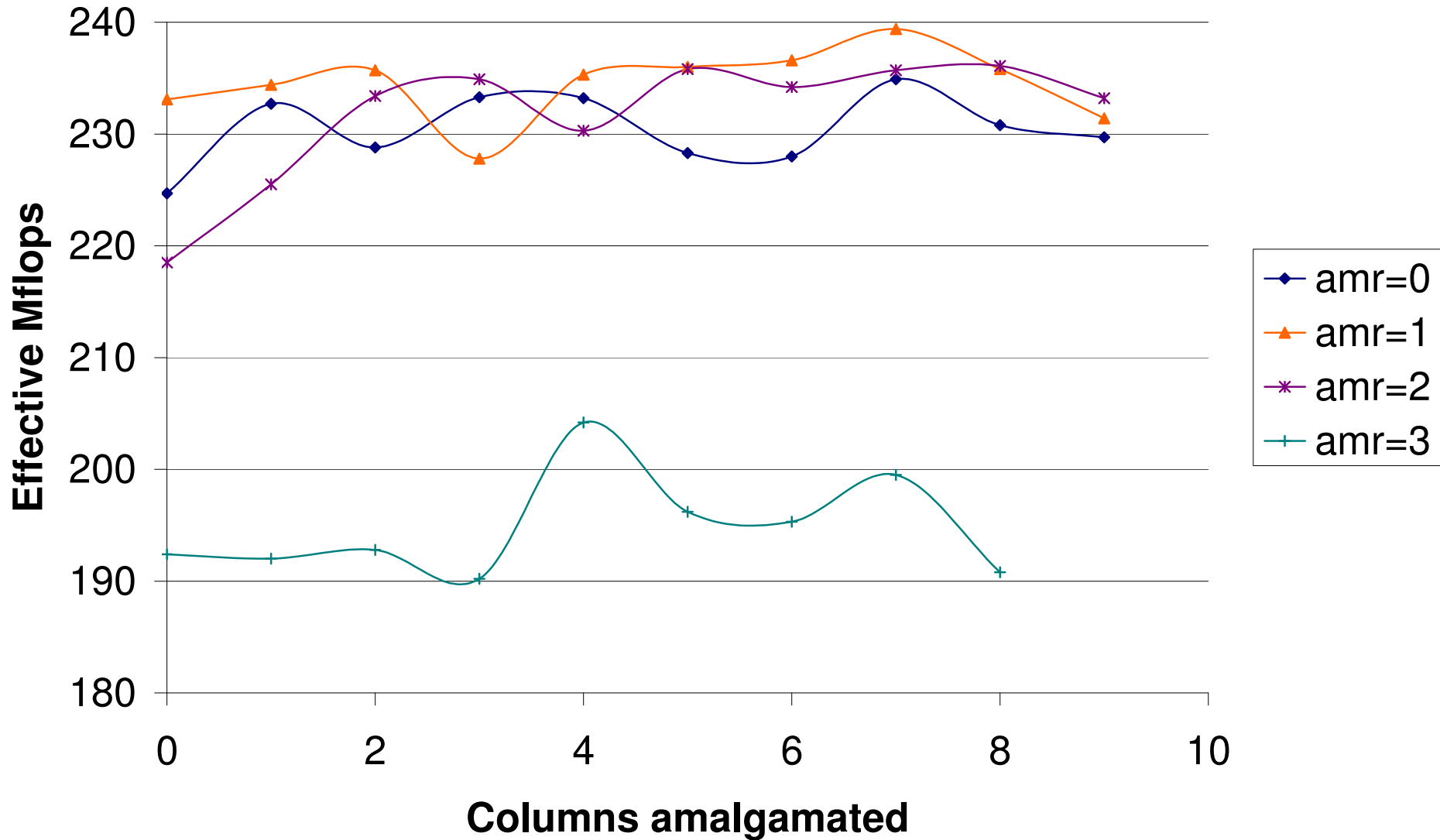
Results: TRIPART2



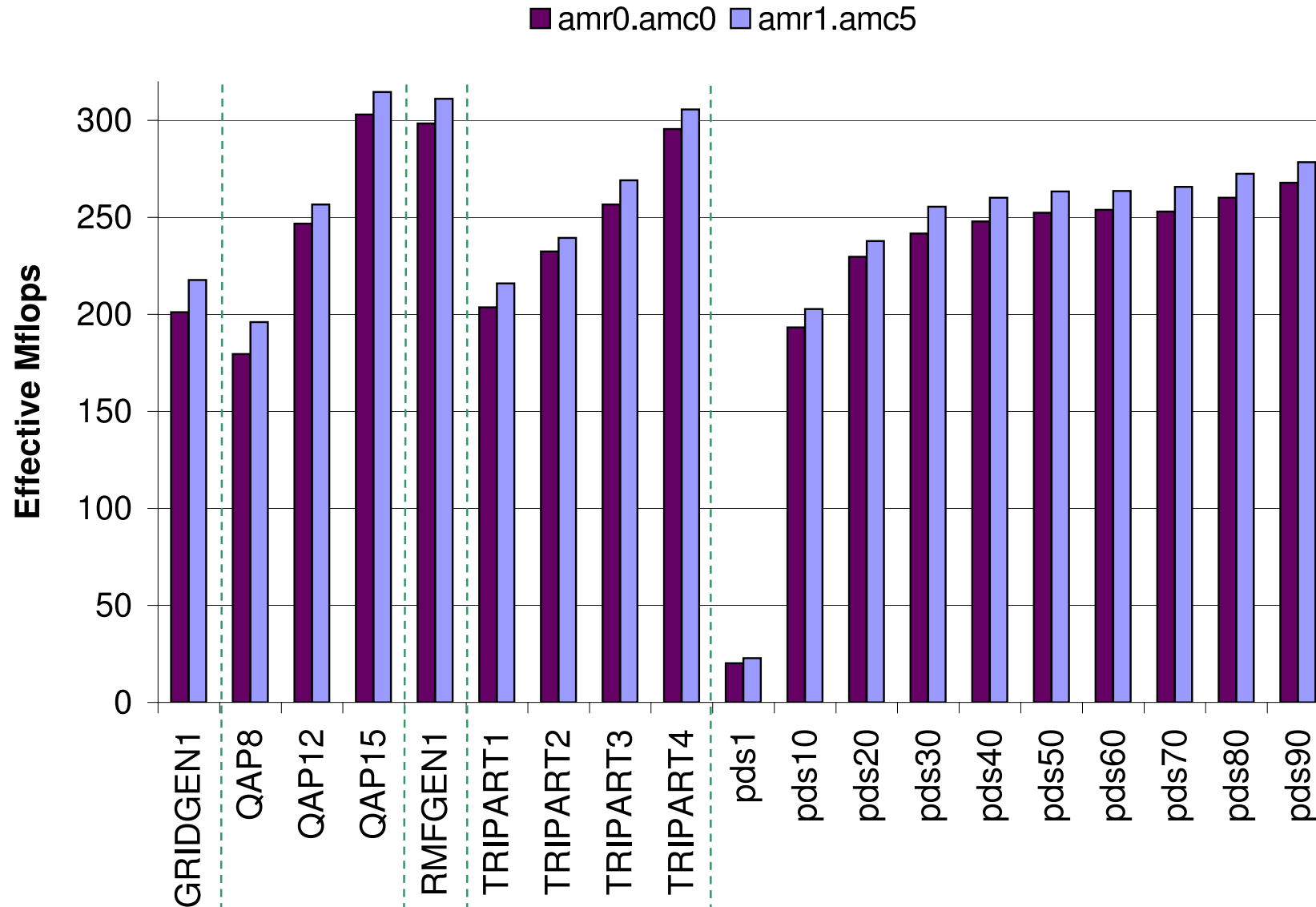
Results: pds10



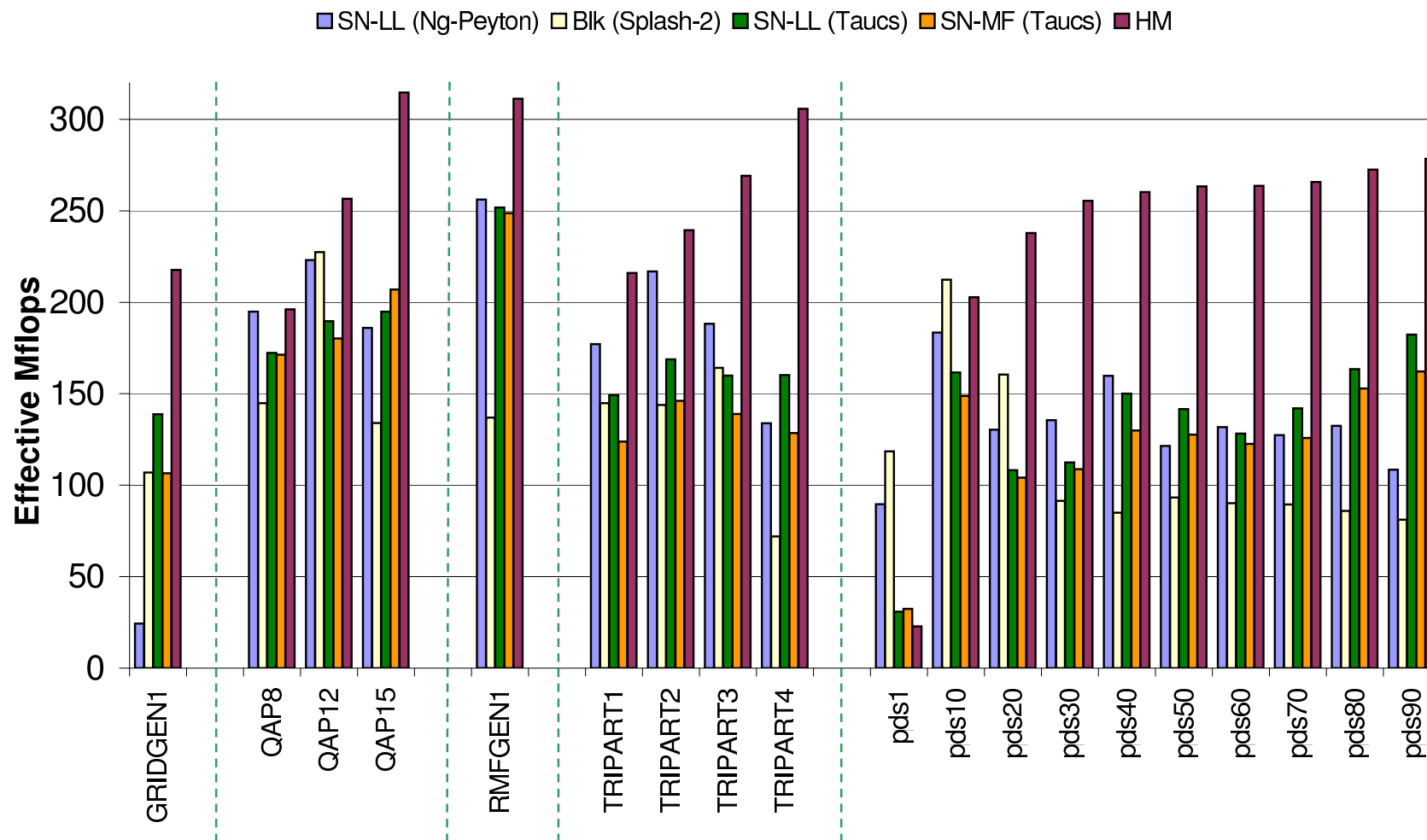
Results: pds20



Results: Original (without amalgamation) vs Intra-block amalgamation



Results: Performance of several sparse Cholesky codes



Conclusions and future work

- R10000: row amalgamation 1 + column amalgamation 5
 - 5.3% Average improvement on matrix test suite
- Current work (Paper in progress)
 - Use variable partitioning of hypermatrix using the Elimination Tree
 - Hypermatrix oriented supernode amalgamation
- Future work
 - Store data submatrices as supernodes

Overhead in number of operations in sparse HM Cholesky (4x32 + windows).

