On computing arbitrary entries of the inverse of a matrix

François-Henry Rouet

Joint work with: Patrick Amestoy and Bora Uçar

Université de Toulouse, INPT(ENSEEIHT)-IRIT, France

SIAM Conference on CSC, 29-31 Oct 2009, Monterey, CA

Context of our study

Context

- Some applications require the partial computation of the inverse of a large, sparse matrix.
- Examples:
  - Computing the variances of the unknowns of a data fitting problem = computing the diagonal of a so-called variance-covariance matrix.
  - Computing short-circuit currents = computing blocks of a so-called impedance matrix.
  - Approximation of the condition number of a symmetric, positive definite matrix.
  - ...

Partial computation of A^{-1}

Central idea

Computing a set of entries in A^{-1} involves the solution of several linear systems. An efficient algorithm has to take advantage of the sparsity of A and the right-hand sides.

Partial computation of A^{-1}

Central idea

Computing a set of entries in A^{-1} involves the solution of several linear systems. An efficient algorithm has to take advantage of the sparsity of A and the right-hand sides.

Approach

1. Graph representation of the problem.

Partial computation of A^{-1}

Central idea

Computing a set of entries in A^{-1} involves the solution of several linear systems. An efficient algorithm has to take advantage of the sparsity of A and the right-hand sides.

Approach

1. Graph representation of the problem.
2. Computing a single entry: exploit sparsity.

Partial computation of A^{-1}

Central idea

Computing a set of entries in A^{-1} involves the solution of several linear systems. An efficient algorithm has to take advantage of the sparsity of A and the right-hand sides.

Approach

1. Graph representation of the problem.
2. Computing a single entry: exploit sparsity.
3. Computing a set of entries: a combinatorial problem.

Partial computation of A^{-1}

Central idea

Computing a set of entries in A^{-1} involves the solution of several linear systems. An efficient algorithm has to take advantage of the sparsity of A and the right-hand sides.

Approach

1. Graph representation of the problem.
2. Computing a single entry: exploit sparsity.
3. Computing a set of entries: a combinatorial problem.
4. Characterization of a solution: heuristics.

Partial computation of A^{-1}

Central idea

Computing a set of entries in A^{-1} involves the solution of several linear systems. An efficient algorithm has to take advantage of the sparsity of A and the right-hand sides.

Approach

1. Graph representation of the problem.
2. Computing a single entry: exploit sparsity.
3. Computing a set of entries: a combinatorial problem.
4. Characterization of a solution: heuristics.
5. An other approach: hypergraph model.

Graph representation

We consider a sparse matrix A, and a *factorization* $A = L\,U$.

1. We work on the *pattern* of A and its factors L and U.

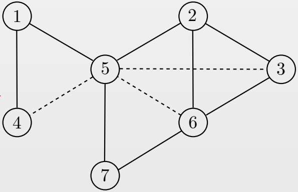
$$\begin{pmatrix}
1 & & & X & X & & \\
 & 2 & X & & X & X & \\
 & X & 3 & & F_2 & X & \\
X & & & 4 & F_1 & & \\
X & X & F_2 & F_1 & 5 & F_2 & X \\
 & X & X & & F_2 & 6 & X \\
 & & & & X & X & 7
\end{pmatrix}$$

Graph representation

We consider a sparse matrix A, and a *factorization* $A = L\,U$.

1. We work on the *pattern* of A and its factors L and U.
2. The pattern is represented by a graph.

$$\begin{pmatrix} 1 & & & X & X & & \\ & 2 & X & & X & X & \\ & X & 3 & & F_2 & X & \\ X & & & 4 & F_1 & & \\ X & X & F_2 & F_1 & 5 & F_2 & X \\ & X & X & & F_2 & 6 & X \\ & & & & X & X & 7 \end{pmatrix} \rightarrow$$

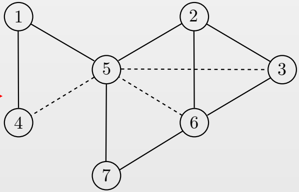


Graph representation

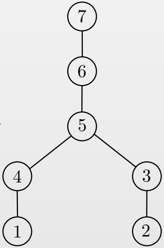
We consider a sparse matrix A, and a *factorization* $A = L\,U$.

1. We work on the *pattern* of A and its factors L and U.
2. The pattern is represented by a graph.
3. This graph is tidied of the redundant information $\Rightarrow$ elimination tree.

$$\begin{pmatrix} 1 & & & X & X & & \\ & 2 & X & & X & X & \\ & X & 3 & & F_2 & X & \\ X & & & 4 & F_1 & & \\ X & X & F_2 & F_1 & 5 & F_2 & X \\ & X & X & & F_2 & 6 & X \\ & & & & X & X & 7 \end{pmatrix} \rightarrow$$



$\rightarrow$



Computing a single entry in A^{-1}

We use the approach implemented in MUMPS during Tz. Slavova's PhD. It relies on:

- A traditional solution phase: $a_{i,j}^{-1} = (A^{-1}e_j)_i$

Computing a single entry in A^{-1}

We use the approach implemented in MUMPS during Tz. Slavova's PhD. It relies on:

- A traditional solution phase: $a_{i,j}^{-1} = (A^{-1} e_j)_i$
- The use of a direct solver: once one has factorized A (e.g. $A = L\,U$), $a_{i,j}^{-1}$ can be obtained by:

$$\begin{cases} y = L^{-1} e_j \\ a_{i,j}^{-1} = (U^{-1} y)_i \end{cases}$$

Computing a single entry in A^{-1}

We use the approach implemented in MUMPS during Tz. Slavova's PhD. It relies on:

- A traditional solution phase: $a_{i,j}^{-1} = (A^{-1}e_j)_i$
- The use of a direct solver: once one has factorized A (e.g. $A = L\,U$), $a_{i,j}^{-1}$ can be obtained by:

$$\begin{cases} y = L^{-1}e_j \\ a_{i,j}^{-1} = (U^{-1}y)_i \end{cases}$$

- Sparsity is exploited using a theorem by Gilbert.

Computing a single entry in A^{-1}

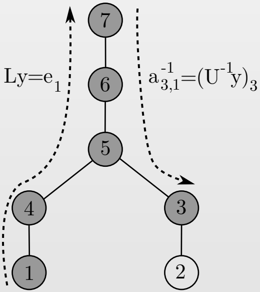
The following result takes advantage of the sparsity:

Theorem [derived from Gilbert, '86]

To compute a particular entry $a_{i,j}^{-1}$ in A^{-1}, one needs to follow:

- the path from j up to the root node (solution of $Ly = e_j$).
- the path going back from the root to node i ($a_{i,j}^{-1} = (U^{-1}y)_i$).

Example: traversal of the tree for the computation of $a_{3,1}^{-1}$.



Experiments: interest of exploiting sparsity

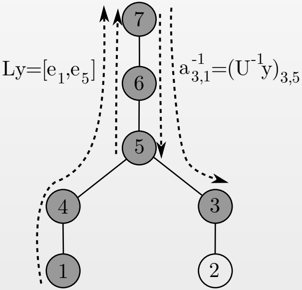
Experiments: computation of the diagonal of the inverse of matrices from data fitting in Astrophysics (CESR, Toulouse)

Matrix size	Time (s)	
	No ES	ES
46,799	6,944	472
72,358	27,728	408
148,286	>24h	1,391

Computing a set of entries of A^{-1}

- When computing several entries at the same time: nodes in common are loaded only once.

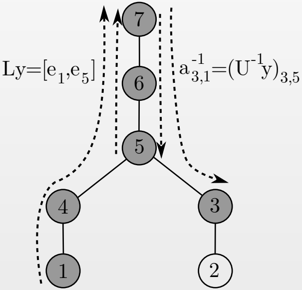
Example: when computing $a^{-1}_{3,1}$ and $a^{-1}_{5,5}$, second accesses to 5, 6, 7 are spared.



Computing a set of entries of A^{-1}

- When computing several entries at the same time: nodes in common are loaded only once.

Example: when computing $a^{-1}_{3,1}$ and $a^{-1}_{5,5}$, second accesses to 5, 6, 7 are spared.

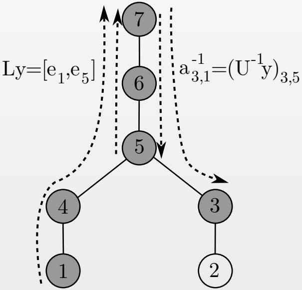


- In an out-of-core context, the solution time is dominated by the I/O, and an access to a node = an access to the hard disk.

Computing a set of entries of A^{-1}

- When computing several entries at the same time: nodes in common are loaded only once.

Example: when computing $a^{-1}_{3,1}$ and $a^{-1}_{5,5}$, second accesses to 5, 6, 7 are spared.



- In an out-of-core context, the solution time is dominated by the I/O, and an access to a node = an access to the hard disk.
- When one wants to compute a large number of entries of the inverse, the set of associated right-hand sides is divided into several blocks. ⇒ is there a way to form the blocks such that the number of accesses is minimized ?

Shape of an optimal solution

First, we have studied some properties of the problem; we have proposed:

- A lower bound of the minimum number of accesses.

Shape of an optimal solution

First, we have studied some properties of the problem; we have proposed:

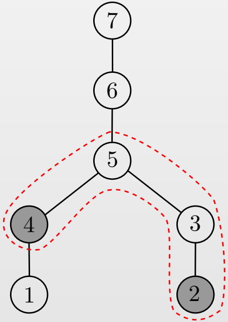
- A lower bound of the minimum number of accesses.
- A necessary and sufficient condition. Here we provide only a (weaker) sufficient condition.

Shape of an optimal solution

First, we have studied some properties of the problem; we have proposed:

- A lower bound of the minimum number of accesses.
- A necessary and sufficient condition. Here we provide only a (weaker) sufficient condition.
- We use the notion of *encompassing tree* of a block of entries: smallest tree containing these entries.

Example: with $a^{-1}_{4,4}$ and $a^{-1}_{2,2}$, the encompassing tree is $\{5, 4, 3, 2\}$.

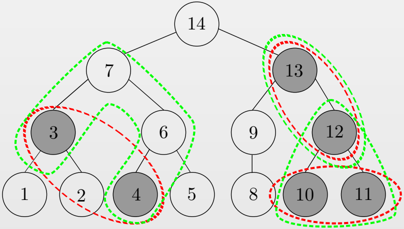


Shape of an optimal solution

Theorem: sufficient condition for reaching the lower-bound

The encompassing trees of the blocks of requested entries do not intersect, or intersect only in one node.

Example: nodes 3, 4, 10, 11, 12 and 13 are requested, and the block size is 2.



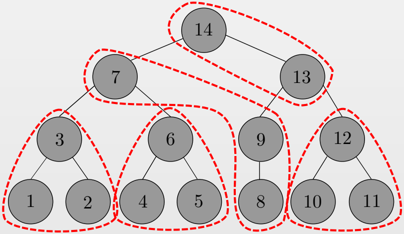
This partitioning reaches the lower bound.

Shape of an optimal solution

A first intuitive attempt to satisfy the previous condition is to use a topological order of the elimination tree.

Idea: in a post-order traversal of the tree, all nodes in a subtree have consecutive numbers.

Example: the block size is 3 and all the nodes are requested.



Experiments

Experiments of the same set of matrices from Astrophysics:

Matrix size	Lower bound	Factors loaded [MB]		
		No ES	Nat	Po
46,799	11,105	137,407	12,165	11,628
72,358	1,621	433,533	5,800	1,912
148,286	9,227	1,677,479	18,143	9,450

The post-order provides good result for this set of experiments, but is it always the case ?

More experiments...

Experiments on a set a various matrices: the ratio of number of accesses over the lower bound is measured:

Matrix	10% diagonal	10% off-diag
CESR(46799)	1.01	1.28
af2356	1.02	2.09
boyd1	1.03	1.92
ecl32	1.01	2.31
gre1107	1.17	1.89
saylr4	1.06	1.92
sherman3	1.04	2.51
grund/bayer07	1.05	1.96
mathworks/pd	1.09	2.10
stokes64	1.05	2.35

$\Rightarrow$ topological orders provide good results for the diagonal case, but are not efficient enough for the general case.

Improving topological orders

- Some local strategies aiming at improving topological orders have been studied:
  - Slight improvements in the diagonal case...
  - ...but they could not be extended to the general case.

Improving topological orders

- Some local strategies aiming at improving topological orders have been studied:
  - Slight improvements in the diagonal case...
  - ...but they could not be extended to the general case.
- The general case is difficult because:
  - The lower bound seems to be a bad criterion...
  - ...hence the previous condition might not be relevant.

Hypergraph partitioning

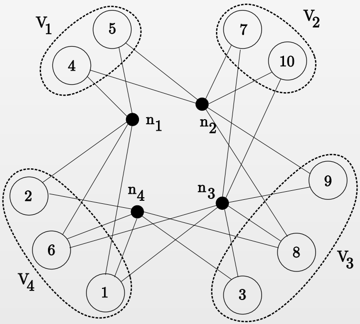
Now we present a completely different approach, based on hypergraph partitioning.

- Hypergraph: $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is defined as a set of vertices $\mathcal{V}$, and a set of nets $\mathcal{N}$. Every net is a subset of vertices.
  Weights associated with vertices.
  Cost $c(n_i)$ is associated with net n_i.
- Vertex partition: $\Pi = \{\mathcal{V}_1, \ldots, \mathcal{V}_K\}$.
- Connectivity: $\lambda(n_i)$ is the number of partitions of Π in which n_i has vertices.
- Objective: Minimize

$$cutsize(\Pi) = \sum_{n_i \in \mathcal{N}} (\lambda(n_i) - 1) c(n_i) \; .$$

- Constraint: Satisfy a balance on the partition weights (sum of the weights of the vertices in each partition).

Hypergraph partitioning: an example



10 vertices and 4 nets.

Partitioned into 4 parts:
$\{4, 5\}, \{7, 10\}, \{3, 8, 9\}, \{1, 2, 6\}$.

$\lambda(n_1) = 2, \quad \lambda(n_2) = 3$
$\lambda(n_3) = 3, \quad \lambda(n_4) = 2$

$cutsize(\Pi) =$
$c(n_1) + 2c(n_2) + 2c(n_3) + c(n_4)$

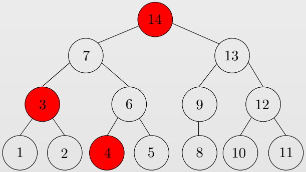
Hypergraph model for the diagonal case

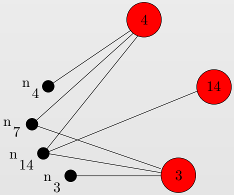
Model for the diagonal case

Vertices: a vertex for each requested entry.

Nets:
- There is a net for each node corresponding to a requested entry, initially containing that node.
- There is a net for each intersection node (e.g. node 7).
- A net is a super set of all the nets associated with nodes that are descendants of its defining node.

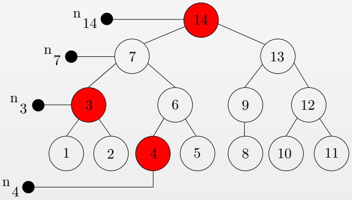
Costs: the cost of a net is the sum of the sizes of the factors from its defining node v to the first significant ancestor of v, e.g., $c(n_4) = w(4) + w(6)$.

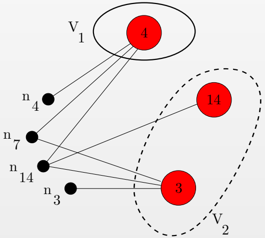




Hypergraph model: an example

We show that the cutsize is the extra cost induced by the partition:





Epochs: $\left\{a^{-1}_{3,3}, a^{-1}_{14,14}\right\}$ and $\left\{a^{-1}_{4,4}\right\}$.

Epoch	Cost
1^{st}	$w(3) + w(7) + w(14)$
2^{nd}	$w(4) + w(6) + w(7) + w(14)$

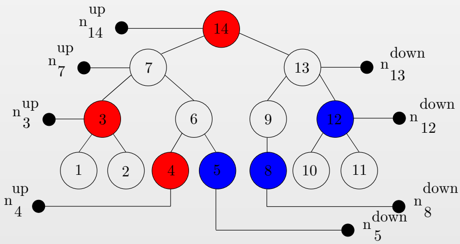
Nets 7 and 14 are cut.

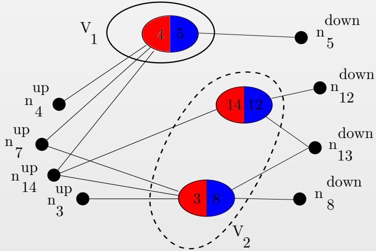
The cutsize is $c(n_7) + c(n_{14}) = w(7) + w(14)$.

In any solution, we have to load 3, 4, 6, 7, and 14; having a bare minimum cost: $w(3) + w(4) + w(6) + w(7) + w(14)$.

Hypergraph model for the general case

The model is obtained by *vertex amalgamation*: consider the hypergraph defined by the row subscripts and the hypergraph defined by the column subscripts, and simply "sew" them:





Experiments: hypergraph model

We use PaToH [Çatalyürek and Aykanat, '99] for the tests. Here we measure the ratio hypergraph / post-order:

Matrix	10% diagonal	10% off-diag
CESR(46799)	1.01	0.75
af2356	1.03	0.69
boyd1	1.03	0.54
ecl32	1,05	0.56
gre1107	0.86	0.80
saylr4	0.98	0.80
sherman3	0.97	0.65
grund/bayer07	0.97	0.72
mathworks/pd	0.94	0.60
stokes64	0.99	0.80

- Diagonal case: no gain, except for "tough" problems.
- General case: on average, a gain of 30%.

Remarks on the hypergraph model

Flexibility of the epochs sizes

In the previous experiments, no unbalance of the block sizes was allowed. In pratice, some sloppiness in the number of RHS per epoch, and hypergraph partitioning tools naturally exploit this leeway.

Structure of the model

- Our hypergraphs are peculiar because the nets are nested; this could be exploited.
- The hypergraph becomes rapidly dense. If the number of requested entries is large, partitioning can be really expensive (in terms of memory and running time) $\Rightarrow$ develop algorithms that work directly on the tree itself, "hiding" the underlying hypergraph.

Conclusion

Conclusions

- This combinatorial problem is interesting and significant gains can be expected.
- Several approaches have been considered.
- The methods presented here show promising results.

Perspectives

Several extensions and improvements can be studied:

- In-core case.
- Multiple entries per RHS.
- Parallel environment.
- Compressed representations of the problem.
- ...

Thank you for your attention !

Any questions ?

References

Y. E. Campbell and T. A. Davis.
Computing the sparse inverse subset: an inverse multifrontal approach.
Technical Report TR-95-021, CIS Dept., Univ. of Florida, 1995.

UV. Çatalyurek and C. Aykanat.
PaToH: partitioning tool for hypergraphs.
User's guide, 1999.

A. M. Erisman and W. F. Tinney.
On computing certain elements of the inverse of a sparse matrix.
Comm. ACM, 18:177–179, 1975.

J. R. Gilbert and J. W. H. Liu.
Elimination structures for unsymmetric sparse LU factors.
SIAM J. Matrix Analysis and Applications, 1993.

Tz. Slavova.
Parallel triangular solution in an out-of-core multifrontal approach for solving large sparse linear systems.
PhD thesis, Institut National Polytechnique de Toulouse, Toulouse, France, 2009.

K. Takahashi, J. Fagan, and M. S. Chen.
Formation of a sparse bus impedance matrix and its application to short circuit study.
In *Power Industry Computer Applications Conference*, pages 63–69, 1973.

B. Uçar and C. Aykanat.
Revisiting hypergraph models for sparse matrix partitioning.
SIAM Review, 49:595–603, 2007.