The filtering step of discrete logarithm and integer factorization algorithms

Cyril BOUVIER

Institut de Mathématiques de Bordeaux (IMB)
`Cyril.Bouvier@u-bordeaux.fr`

CATREL Workshop – October 2nd, 2015

Outline of the presentation

1. Introduction
2. Description of the filtering step
3. Weight functions for clique removal
4. Experiments

Outline of the presentation

1. Introduction
2. Description of the filtering step
3. Weight functions for clique removal
4. Experiments

The filtering step

- Filtering step: common step of integer factorization and discrete logarithms (DL) algorithms.
- In particular in NFS, NFS-DL and FFS algorithms. Also in other algorithms like MPQS for factorization and algorithms for DL based on index-calculus method.
- All these algorithms have a common structure:
  - first step (often, a kind of polynomial selection);
  - computations of relations;
  - filtering step;
  - linear algebra step;
  - last step (square root for factorization and individual logarithm for DL).

Common characteristics

- In these algorithms, a relation is the decomposition of an element in a factor base.
- The set of relations is seen as a matrix where
  - a row corresponds to a relation;
  - a column corresponds to an element of the factor base.
- The value of the coefficient in row i and column j is the valuation of the element of the factor base corresponding to the jth column in the ith relation.
- Excess: difference between the number of rows and the number of columns of the matrix.
- Goal of the filtering step: build a "good" matrix from the given relations.

Integer Factorization context

- Wanted: a subset of the relations that, when multiplied together, forms a square.
- Reformulation: a subset of the rows of the matrix that, when added, produces a vector with only even coefficients, *i.e.* the null vector over GF(2).
- The linear algebra step: computation of the left kernel of the matrix over GF(2).
- Excess is an lower bound on the dimension of the left kernel. Need around 150 vectors in the kernel, so final excess should be around 150.

Discrete Logarithm (DL) context

- Wanted: the logarithms of all the elements of the factor base.
- A relation is interpreted as an equality between the logarithms of the elements of the factor base
- The size of the group in which the DL computation is performed is denoted by ℓ. The size of ℓ is around a few hundred bits.
- The linear algebra step: computation of the right kernel of the matrix over $\mathrm{GF}(\ell)$.
- Need non-negative excess in order to have a kernel of dimension at most 1. Excess do not need to be positive, so the final matrix is often square.

Goal of the filtering step

- At the beginning of the filtering step, the matrix is
  - very large: up to a few billion rows and columns.
  - very sparse: around 20 to 30 non-zero coefficients per row.
- Goal of the filtering step: to produce a matrix as small and as sparse as possible from the given relations in order to decrease the time spent in the linear algebra step.
- Example: data from the factorization of RSA-768:
  - input: 48 billion rows and 35 billion columns.
  - output: 193 million rows and columns with 144 non-zero coefficients per row in average.
- Publicly available implementation: `GGNFS`, `Msieve`, `cado-nfs` for factorization (based on Cavallar's thesis); none for DL before this work.

Outline of the presentation

Stages of the filtering step

- Stages of the filtering step:
  - singleton removal: remove useless rows and columns;
  - clique removal: use the excess to reduce the size of the matrix;
  - merge: beginning of a Gaussian elimination.
- Assume no duplicate in relations (easy to spot and remove)
- Weight: the weight of a row (resp. column) is the number of non-zero coefficients in this row (resp. column). The total weight of the matrix is the total number of non-zero coefficients.
- The singleton removal and clique removal stages reduce the size and the total weight of the matrix.
- The merge stage reduces the size of the matrix but increases its total weight.

Singleton removal

- A singleton is a column of weight 1.
- Removing a singleton is the removal of the column and of the row corresponding to the non-zero coefficient.
- Rationale:
  - Factorization: relations containing singletons cannot be used to produce squares.
  - DL: the logarithm of the singleton can be computed from the others logarithms *after* the linear algebra step.

Singleton removal — Example

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Singleton removal — Example

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Singleton removal — Example

$$
\begin{pmatrix}
\cancel{0} & 1 & 1 & 0 & 1 & 1 \\
\cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{1} \\
\cancel{0} & 0 & 1 & 1 & 0 & 0 \\
\cancel{0} & 0 & 1 & 0 & 0 & 1 \\
\cancel{0} & 0 & 0 & 1 & 0 & 1 \\
\cancel{0} & 0 & 0 & 0 & 1 & 0 \\
\cancel{0} & 0 & 0 & 0 & 1 & 1 \\
\cancel{0} & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$

Singleton removal — Example

$$\begin{pmatrix}
\cancel{0} & 1 & 1 & 0 & 1 & 1 \\
\cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{1} \\
\cancel{0} & 0 & 1 & 1 & 0 & 0 \\
\cancel{0} & 0 & 1 & 0 & 0 & 1 \\
\cancel{0} & 0 & 0 & 1 & 0 & 1 \\
\cancel{0} & 0 & 0 & 0 & 1 & 0 \\
\cancel{0} & 0 & 0 & 0 & 1 & 1 \\
\cancel{0} & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Singleton removal — Example

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Singleton removal — Remarks

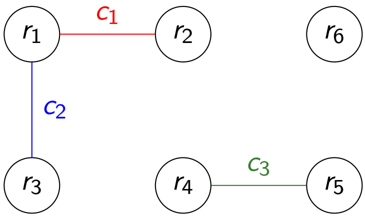
- During the singleton removal stage, rows and columns are removed, so the size and the total weight of the matrix decrease.
- Removing a singleton can create other singletons.
- Excess can only increase.
- Implementation:
  - Only need to know if coefficients are non-zero or not, not the actual values. Same code can be used for factorization and DL.
  - In the DL case, the deleted rows must be saved.

Clique removal

- While the excess is larger that what is needed, it is possible to remove some rows.
- Rationale:
  - Factorization: too much information, can loose some to reduce the size of the matrix.
  - DL: more equations than unknows, can remove some equations.
- If a row containing a column of weight 2 is removed, this column becomes a singleton and can be removed.
- A clique is a connected component of the graph where the nodes are the rows and the edges are the columns of weight 2.
- Not a clique in the sense of graph theory...

Clique removal — Example

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Clique removal — Remarks

- During the clique removal stage, rows and columns are removed, so the size and the total weight of the matrix decrease.
- Each deleted clique reduces the excess by 1.
- Removing a clique can connect other remaining cliques.
- Implementation:
  - Only need to know if coefficients are non-zero or not, not the actual values. Same code can be used for factorization and DL.
  - In the DL case, the deleted rows must be saved.

Merge

- Merge: combinations of rows to create singletons that are then removed.
- Rationale:
  - Factorization: pre-combination of rows to help the linear algebra step.
  - DL: beginning of a Gaussian elimination.
- Let $k \geq 2$ be a positive integer, C be a column of weight k and $r_1, \ldots, r_k$ be the k rows corresponding to these k non-zero coefficients.
- A k-merge is a way of performing successive rows additions of the form $r_i \leftarrow c_i r_i + c_j r_j$, with distinct i, j, such that the column C becomes a singleton that is deleted.

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

The last column have weight 2.

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

The last column have weight 2.

Merge — Examples

$$\begin{pmatrix} 1 & 1 & 0 & 1 & \color{red}{0} \\ \color{green}{0} & \color{green}{1} & \color{green}{1} & \color{green}{1} & \color{green}{0} \\ 1 & 1 & 1 & 0 & \color{red}{0} \\ 0 & 1 & 1 & 1 & \color{red}{0} \\ \color{red}{1} & \color{red}{1} & \color{red}{0} & \color{red}{1} & \color{red}{1} \\ 1 & 1 & 1 & 1 & \color{red}{0} \end{pmatrix}$$

The two corresponding rows are added, creating a singleton.

Merge — Examples

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Then the singleton is deleted. This is a 2-merge.

Merge — Examples

$$\begin{pmatrix} 1 & 1 & 0 & 1 & \cancel{0} \\ 0 & 1 & 1 & 1 & \cancel{0} \\ 1 & 1 & 1 & 0 & \cancel{0} \\ 0 & 1 & 1 & 1 & \cancel{0} \\ \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} \\ 1 & 1 & 1 & 1 & \cancel{0} \end{pmatrix}$$

The first column have weight 3. It exists 3 ways of combining these 3 rows.

Merge — Examples

$$\begin{pmatrix} 1 & 1 & 0 & 1 & \cancel{0} \\ 0 & 1 & 1 & 1 & \cancel{0} \\ 0 & 0 & 1 & 1 & \cancel{0} \\ 0 & 1 & 1 & 1 & \cancel{0} \\ \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} \\ 0 & 0 & 1 & 0 & \cancel{0} \end{pmatrix}$$

For example, adding the first row to the other two. It creates a singleton.

Merge — Examples

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Then the singleton is deleted. This is a 3-merge.

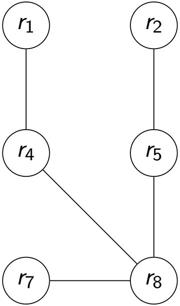
Merge — Examples

- For a k-merge with $k \geq 3$, there is a choice on how to combine the k rows. Use minimal spanning tree to minimize the increase of total weight.

- Example of a 6-merge:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

	r_2	r_4	r_5	r_7	r_8
r_1	4	2	4	2	3
r_2		4	2	4	3
r_4			2	2	1
r_5				2	1
r_7					1



Merge — Remarks

- A k-merge removes 1 row and 1 column, but increases the total weight of the matrix (except for a 2-merge).
- Merge is performed until a given average weight per row is reached.
- In merge, the values of the non-zero coefficients matter:
  - Factorization: coefficients are in GF(2), easy.
  - DL: coefficients are small (in practice, 99 % in $[-10, 10]$), so no modular reduction, consider them in $\mathbb{Z}$.
- Merge is the last stage of the filtering step. The matrix returned by merge should be as sparse and as small as possible.

Outline of the presentation

Choice of cliques in clique removal

- During clique removal, one can choose which cliques are removed.
- How to choose? What choice of cliques, done during clique removal, produces the smallest and sparsest matrix at the end of merge?
- Used weight function to compare the cliques and removed the heaviest ones.
- The number of rows of the matrix is denoted by N, its total weight by W. The weight of a column C is denoted by $w(C)$.
- To a first approximation, the time spent in the linear algebra step is proportional to the product $N \times W$ for the final matrix.

What do we want in a weight function ?

- Remove cliques containing lots of rows:
  - It does not cost more to remove large cliques: removing 1 clique reduces the excess by 1 whatever the number of rows in the cliques.
  - The size of the matrix is reduced by the number of rows in the deleted clique.
  - So the weight functions should have a term taking into account the number of rows in the cliques.
- Reduce the weight of columns to have more columns of weight $2, 3, 4, \ldots$
  - New columns of weight 2 will create larger cliques.
  - New columns of weight $3, 4, \ldots$ will reduce the fill-in in the following merge stage.
  - So the weight functions should have a term taking into account the weight of the columns appearing in the rows of a clique.

Weight functions used in software

- Msieve uses Cavallar's weight function:

$$\sum_{\text{row}\in\text{clique}} \left(1 + \sum_{\text{col}\in\text{row}, w(\text{col})\geq 3} \frac{1}{2^{w(\text{col})-2}}\right)$$

- GGNFS uses the following weight function:

$$\sum_{\text{row}\in\text{clique}} \left(1 + \sum_{\text{col}\in\text{row}, w(\text{col})\geq 3} 1\right)$$

- The default weight function of cado-nfs 1.1 was

$$\sum_{\text{row}\in\text{clique}} 1$$

- cado-nfs 2.0 uses a new weight function identified during this work.

New weight functions

- Proposed 27 new weight functions. Most of these new weight function have the following form:

$$\sum_{\text{row}\in\text{clique}} \left(? + \sum_{\text{col}\in\text{row}, w(\text{col})\geq 3} f(w(\text{col})) \right)$$

- In total, 31 weight functions were compared (27 new ones, 1 from `Msieve`, 1 from `GGNFS` and 2 from `cado-nfs` 1.1)

Outline of the presentation

Experiments

- To have a fair comparison between the 31 weight functions, they were all implemented in `cado-nfs` .
- All the weight functions were benchmarked on 8 data sets:
  - 3 from factorization computations with NFS: RSA-155, B200 and RSA-704;
  - 2 from DL computations with NFS-DL: in two prime fields of size 155 digits and 180 digits;
  - 3 from DL computations with FFS: in $GF(2^{619})$, $GF(2^{809})$ and $GF(2^{1039})$.
- Input: set of unique relations, the target excess and the target average number of non-zero coefficients per row.
- Output: the matrix after merge.
- How to compare the final matrices? Compare the product $N \times W$ of the final matrices.

Settings for two experiments

		GF(p_{180})	GF(2^{1039})
Beginning	Number of rows	175M	1306M
	Number of columns	78M	986M
After singleton removal	Number of rows	171M	1080M
	Number of columns	78M	746M
	Excess	93M	334M
	Relative excess	119 %	44.7 %
After clique removal	Excess	5	0
After merge	k-merge for k from	2 to 30	2 to 30
	W/N	150	100

Partial results for GF(p_{180}) and GF(2^{1039})

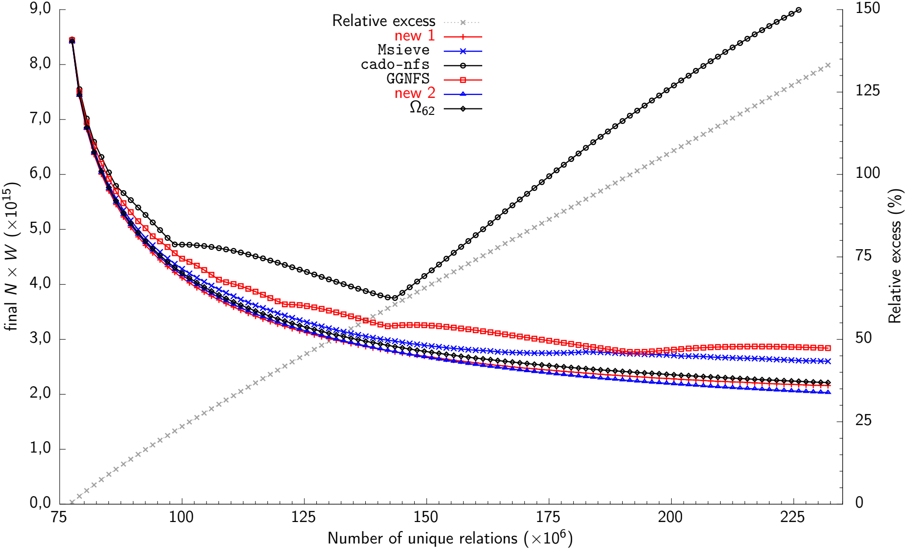
GF(p_{180})	After clique removal	At the end of the filtering step		
	N	N	$N \times W$	
new 1	21 468 306	7 288 100	7.97×10^{15}	
new 2	21 546 475	7 400 557	8.22×10^{15}	+3.11 %
Msieve	20 395 070	7 866 604	9.28×10^{15}	+16.51 %
GGNFS	25 676 095	9 163 369	1.26×10^{16}	+58.08 %
cado-nfs 1.1	28 940 807	10 769 526	1.74×10^{16}	+118.36 %

GF(2^{1039})	After clique removal	At the end of the filtering step		
	N	N	$N \times W$	
new 2	188 580 425	65 138 845	4.24×10^{17}	
new 1	188 302 437	65 800 281	4.33×10^{17}	+2.04 %
Msieve	182 939 672	67 603 362	4.57×10^{17}	+7.71 %
GGNFS	197 703 703	74 570 015	5.56×10^{17}	+31.05 %
cado-nfs 1.1	203 255 785	78 239 129	6.12×10^{17}	+44.27 %

Some remarks

- Found two new weight functions that outperformed the others in all experiments.
- The best weight functions after clique removal are not the best at the end of the filtering step.
- The best weight functions are the ones that have few or no contribution from the number of rows in the clique.
- The larger the initial excess, the larger the differences between the weight functions.

Excess — RSA-155



Conclusion

- Unified description of the filtering step for integer factorization and discrete logarithm computation.
- First publicly available implementation (in `cado-nfs`) of the filtering step for discrete logarithm.
- Proposed new weight functions for the clique removal stage.
- Compared them on data sets coming from actual computations and found two new weight functions that perform better.

Thanks you for your attention.

Questions ?