On some matrix approach to constructing Gröbner bases[1]

I. A. Borisov

Abstract. We describe matrix approach to constructing Gröbner bases of polynomial ideals and data structures needed to efficient execution in computer algebra system Mathpar.

1. Introduction

Gröbner bases of polynomial ideals have a wide application. There are different algorithms for Gröbner bases [1]-[4]. Matrix algorithms and ways to parallelize it are described in [5]-[7].

We review the new data structures which has been developed special for matrix approach.

2. The matrix algorithm

In 1999 J. C. Faugere has introduced efficient matrix algorithm for Gröebner basis construction called F_4 [5].

The main differenced from the Buchberger algorithm:
1) Polynomials representation: instead of set polynomials the matrix of its coefficients is used.
2) Constructing S-polynomials: Faugere's algorithm allows not to compute full S-polynomial, but to add in matrix its parts for further making of linear combinations.
3) S-polynomials reduction: reduction is performed in one step with computing row echelon form of the matrix with the coefficients. To make this operation completely equivalent to reduction we should add the matrix rows which correspond to some polynomial of basis with appropriate multipliers.

[1]This work was partially supported by the Russian Foundation for Basic Research (grant No. 12-07-00755, 12-01-06020).

For constructoin of matrix we should specify an monomial ordering and associate each monomial of each polynomial to matrix element.

Example of mapping the set of polynomials to the matrix is shown in the figure 1.

$$\begin{array}{c|cccc} & z^3 & z & y^2 & yx^2 \\ \hline p_1 & 0 & 2 & 0 & 1 \\ p_2 & 1 & 0 & -3 & 0 \\ p_3 & -10 & 0 & 0 & 0 \end{array}$$

Figure 1. Mapping the set of polynomial $\{p_1 = 2z + yx^2,\ p_2 = z^3 - 3y^2,\ p_3 = -10z\}$ to the matrix

3. Data structures for matrix algorithm

Our previous implementatioin of matrix algorithm contained transformation of polynomials to matrix of coefficients and recovery set of polynomials and terms from matrix on each iteration of Groenber basis computation. This approach is inefficient because on every operation of obtaining matrix of coefficients from set of polynomials we allocate and fill large arrays in memory, and then during recovery set of polynomials from matrix we find terms from list of all polynomials of current basis needed to obtaining each polynomial.

We propose a special data structure for speedup of matrix algorithm, which allows to don't perform transormations described above, but it performs all needed operations only on matrix of coefficients. We called it `PolynomialList`.

Class `PolynomialList` contains methods needed on each step of matrix algorithm execution. The main operations are: create object `PolynomialList` from list of polynomials, construct and select critical pairs, multiplication term to polynomial and add polynomial to this object.

The way of how to store list of polynomial is as follows. All terms of all polynomials are stored in special sorted list `TermList` named `AllTerms`. Polynomials are placed in rows of two two-dimensional arrays. First array contains indexes of terms from `AllTerms`. It's an array of integer numbers and is called `ColIndexes`. Second array stores appropriate coefficients of terms. It's an array of `Element` objects and is called `Rows`.

Terms are placed in special object of class `TermList`. On insertion of new terms it provides permanent descending ordering of terms and updating indexes of terms that already added.

There are three `PolynomialList` objects during execution of matrix algorithm. First is used to store current basis state. Second contains left and right parts of S-polynomials of critical pairs. Third is used to prepare matrix for reduction. It stores critical pairs selected on each step and polynomials obtained as the result of preprocessing.

4. Conclusion and future work

We plan to continue development of matrix algorithm and parallelize it's matrix operations to execute it on high performance computers with distributed memory.

References

[1] Buchberger B., Collins G. E., Loos R. Computer Algebra: Symolic and Algebraic Computation, 2nd Edition, Springer-Verlag, New York, 1983.

[2] Becker T., Wespfenning V. Groebner bases. A computational approach to commutative arlgebra. New York: Springer, 1993. V. 141.

[3] Cox D., Little J., O'Shea D. Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra. Springer New York, 2007.

[4] Borisov I.A. Some approaches to Gröbner bases computation. Tambov University Reports. Research projects of students. Supplement to journal. 2011. P. 191-194. (Russian)

[5] Faugere J.-C. A new efficient algorithm for computing Groebner bases (F4). J. Pure Appl. Algebra. 1999. V. 139. No 1–3. P. 61–88.

[6] Malashonok G.I., Starov M.V., Borisov I.A. To parallel Gröbner bases computation. Tambov University Reports. Series: Natural and Technical Sciences. V. 15. Issue 6. 2010. P. 1755-1760. (Russian)

[7] Alexandrov D.E., Galkin V.V., Zobnin A.I., Levin M.V. Parallelization of matrix algorithms for Gröbner basis computation. Fundamental and applied mathematics. Moscow. V. 14. Issue 4. 2008. P. 35-64.

I. A. Borisov
e-mail: `ivan@iborisov.ru`