Fast Straightening Algorithm for Bracket Polynomials Based on Tableau Manipulations

Changpeng Shao, Hongbo Li

KLMM, Chinese Academy of Sciences

July 18, 2018

Outline

- Background: Bracket Polynomials and Straightening
- Sros: New Straightening Algorithm and Tests

Bracket polynomials

Bracket: $[\mathbf{v}_1\mathbf{v}_2\ldots\mathbf{v}_n] := \det(x_{ij})_{i,j=1..n}$.

Bracket polynomials: polynomials with brackets as indeterminates.

Example. Bracket monomial of degree (height) 2 and dimension (length) 3 in vector variables $\mathbf{1}, \mathbf{2}, \ldots$

$$[\mathbf{123}][\mathbf{134}] := \begin{bmatrix} \mathbf{123} \\ \mathbf{134} \end{bmatrix}, \qquad [\mathbf{134}][\mathbf{132}] := -\begin{bmatrix} \mathbf{123} \\ \mathbf{134} \end{bmatrix}$$

History: grows out of classical invariant theory – generating all projective invariants.

Found important applications in

- representation theory
- projective geometry
- automated theorem proving
- robotics, mechanism design, etc.

Rectangular Young tableaux and straight tableaux

Young tableau (partition in combinatorics): the dimensions of the rows are non-increasing.

Example. $\begin{matrix}\mathbf{134}\\ \mathbf{15}\end{matrix}$, $\begin{matrix}\mathbf{134}\\ \mathbf{156}\end{matrix}$ (rectangular)

Straight tableau: along each row, the entries are increasing; along each column, the entries are non-decreasing.

Example. $\begin{matrix}\mathbf{123}\\ \mathbf{134}\end{matrix}$: straight; $\begin{matrix}\mathbf{125}\\ \mathbf{134}\end{matrix}$: non-straight.

Classical Theorem.

Any bracket polynomial equals a unique straight bracket polynomial (linear combination of brackets of straight tableaux), called the normal form.

Straightening: procedure of deriving the normal form.

Ordering tableau monomials

Total orders in monomials of the same dimension in vector variables $\mathbf{1} \prec \mathbf{2} \prec \ldots$:

First order by degree (the bigger the higher in order), then for monomials of the same degree, there are two typical orders:

Row order: scan each monomial row by row to get a sequence, then use lex order of the sequence.

Negative column order: scan each monomial column by column to get a sequence, then use negative of the lex order.

Example. In row order, $\begin{matrix}\mathbf{1}&\mathbf{3}&\mathbf{4}\\ \mathbf{1}&\mathbf{3}&\mathbf{5}\\ \mathbf{2}&\mathbf{4}&\mathbf{6}\end{matrix} \prec \begin{matrix}\mathbf{1}&\mathbf{4}&\mathbf{5}\\ \mathbf{1}&\mathbf{3}&\mathbf{6}\\ \mathbf{2}&\mathbf{3}&\mathbf{4}\end{matrix}$;

in negative column order, $\begin{matrix}\mathbf{1}&\mathbf{3}&\mathbf{4}\\ \mathbf{1}&\mathbf{3}&\mathbf{5}\\ \mathbf{2}&\mathbf{4}&\mathbf{6}\end{matrix} \succ \begin{matrix}\mathbf{1}&\mathbf{4}&\mathbf{5}\\ \mathbf{1}&\mathbf{3}&\mathbf{6}\\ \mathbf{2}&\mathbf{3}&\mathbf{4}\end{matrix}$.

Nature of straightening

Negative column order: the only order we use in this work.

Reason: much better property. Negative column order is an admissible order of straight bracket monomials; row order is not.

$\mathbb{K}$: field of char $\neq 2$.
$\mathcal{A}$: alphabet of letters $\mathbf{1} \prec \mathbf{2} \prec \ldots \prec \mathbf{m}$.
$Tab_n(\mathcal{A})$: free commutative algebra of tableaux of fixed dimension n whose content (multiset of the entries of a tableau) is letters of $\mathcal{A}$.

Classical Result (1928) translated into modern terms.

There is a set of quadratic tableau polynomials in $Tab_n(\mathcal{A})$ called van der Waerden syzygies, that are a Gröbner basis of the generating ideal $\mathcal{I}$ of the bracket algebra in $Tab_n(\mathcal{A})$, such that the straightening of a bracket polynomial is the reduction wrt the GB, called Young's algorithm.

Performance of straightening algorithms

Young's algorithm: handle
$2 \times n$ brackets: efficient;
3×3 brackets: less efficiently;
4×3 and above: inefficient.

N. White (1991): supplement GB with another set of quadratic tableau polynomials of $\mathcal{I}$, called multiple syzygies. His method has 5 versions. In our tests, version C is the most efficient.

White's algorithm: handle
3×3 brackets: efficient;
4×3 brackets: less efficiently;
5×3 and above: inefficient.

J. Désarménien et al. (1978-80): a new straightening algorithm based on solving a linear triangular system obtained by Capelli operations, called Rota's algorithm. It is much more efficient.

Polarization: basics of Capelli operator

Polarization operator of a letter $\mathbf{a}$ by letter $\mathbf{d}$ in tableau T:

$$D_{\mathbf{d},\mathbf{a}}T := \lim_{\epsilon \to 0} \frac{T(\mathbf{a}+\epsilon\mathbf{d}) - T(\mathbf{a})}{\epsilon}.$$

Example.

$$D_{\mathbf{d},\mathbf{a}}\left(\begin{array}{cc} \mathbf{a} & \mathbf{c} \\ \mathbf{b} & \mathbf{a} \end{array} \right) = \begin{array}{cc} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{a} \end{array} + \begin{array}{cc} \mathbf{a} & \mathbf{c} \\ \mathbf{b} & \mathbf{d} \end{array}.$$

$\forall l \geq 0$, the *l-th order polarization operator* of letter $\mathbf{a}$ by letter $\mathbf{d}$ is denoted by $D^l_{\mathbf{d},\mathbf{a}}$:

$$D^l_{\mathbf{d},\mathbf{a}}T := \begin{cases} T, & \text{if} \quad l = 0; \\ D_{\mathbf{d},\mathbf{a}}(D^{l-1}_{\mathbf{d},\mathbf{a}}T), & \text{if} \quad l > 0. \end{cases}$$

When $D^l_{\mathbf{d},\mathbf{a}}$ acts on a tableau, the result is the sum of all possible replacements of letter $\mathbf{a}$ at l different positions in the tableau by letter $\mathbf{d}$.

Capelli operator

$\mathcal{A}$: alphabet of letters $\mathbf{a}_i$, $i = 1, \ldots, m$.
$\mathcal{U}$: alphabet of letters $\mathbf{u}_j$, $j = 1, \ldots, n$.
$T \in Tab_n(\mathcal{A})$, where $\mathbf{a}_i$ has multiplicity α_{ij} in column j.

C_T: Capelli operator associated with T:

$$C_T := \text{composition of } D^{\alpha_{ij}}_{\mathbf{u}_j,\mathbf{a}_i} \text{ for } i = 1..m,\ j = 1..n.$$

Example. For $T = \begin{array}{cc} \mathbf{a}_1 & \mathbf{a}_2 \\ \mathbf{a}_2 & \mathbf{a}_3 \end{array}$,

$$C_T T = D_{\mathbf{u}_1,\mathbf{a}_1} D_{\mathbf{u}_1,\mathbf{a}_2} D_{\mathbf{u}_2,\mathbf{a}_2} D_{\mathbf{u}_2,\mathbf{a}_3} T = \left(\begin{array}{cc} \mathbf{u}_1 & \mathbf{u}_2 \\ \mathbf{u}_1 & \mathbf{u}_2 \end{array}\right) + \left(\begin{array}{cc} \mathbf{u}_1 & \mathbf{u}_1 \\ \mathbf{u}_2 & \mathbf{u}_2 \end{array}\right).$$

Rota's Theorem:

Let $S \succ T$ be two tableaux of the same content, then $C_S S \neq 0$ but $C_S T = 0$.

Rota's algorithm

1. Given a tableau T, there are finitely many straight tableaux having the same content with it.

Find them all. Assume they are $S_1, S_2, \ldots, S_t$.

2. By the first main theorem of classical invariant theory,

$$[T] = \sum_{i=1}^{t} \lambda_i [S_i], \quad \text{for some } \lambda_i \in \mathbb{Z}.$$

Act each C_{S_i} on the two sides to get a linear equation in the λ's. There are t equations in t unknowns. The coefficient matrix is triangular and nondegenerate.

Solve the linear system to get the coefficients.

An example by Rota's algorithm

Input: $T = \begin{matrix}\mathbf{146}\\ \mathbf{235}\end{matrix}$.

Step 1. Find all straight tableaux having the same content with T and sort them decreasingly:

$$S_1 = \begin{matrix}\mathbf{135}\\ \mathbf{246}\end{matrix}, S_2 = \begin{matrix}\mathbf{134}\\ \mathbf{256}\end{matrix}, S_3 = \begin{matrix}\mathbf{125}\\ \mathbf{346}\end{matrix}, S_4 = \begin{matrix}\mathbf{124}\\ \mathbf{356}\end{matrix}, S_5 = \begin{matrix}\mathbf{123}\\ \mathbf{456}\end{matrix}.$$

Step 2. For equation $T = \sum_{i=1}^{5} \lambda_i S_i$, act C_{S_i} on it to get

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}.$$

Step 3. Solve the linear system to get

$$\lambda_1 = 1, \quad \lambda_2 = 0, \quad \lambda_3 = -1, \quad \lambda_4 = 0, \quad \lambda_5 = -1.$$

Performance of Rota's algorithm

- The GB polynomials have degree two, so Young's algorithm and White's algorithm deal with only two rows in each step.

  Rota's algorithm deals with all rows simultaneously. Solving triangular linear system is fast.

- However, the number of straight bracket monomials having the same content with the input can be big, so is the number of Capelli operations.

  The evaluation of a single Capelli operation is not time-consuming; the evaluation of all such Capelli operations is time-consuming.

- Rota's algorithm outperforms the other two in general, but can perform badly even for some 5×3 brackets.

Our work

- We propose a new operator on rectangular Young tableaux, called straight roll-and-sort (Sros), and prove its well-definedness in bracket algebra.
- This operator involves much fewer straight bracket monomials than in Rota's algorithm. We propose an efficient implementation of the operator.
- We further disclose the connection of this operator with Capelli operator. Then
- we propose an algorithm "*Sros*" to straighten bracket polynomials based on this operator, and test it with over 500 examples.
- The tests show that "*Sros*" outperforms Young's, White's, Rota's algorithms by achieving a speedup of one to three order of magnitude.

- Background: Bracket Polynomials and Straightening
- Sros: New Straightening Algorithm and Tests

Straight roll-and-sort operator: definition

Ros: roll-and-sort operator upon a $d \times n$ tableau T:

$$Ros(T) = \sum_{\sigma \in (S_n)^d} \text{sign}(\sigma)(\sigma T)^{\Downarrow},$$

where

1. $\sigma = (\sigma_1, \ldots, \sigma_d) \in (S_n)^d$, $\text{sign}(\sigma) = \prod_i \text{sign}(\sigma_i)$;
2. "roll operator" σ_i: permutes the entries of the i-th row of a tableau;
3. "sort operator" $\Downarrow$: sorts each column of a tableau into a non-decreasing sequence.

Sros: straight roll-and-sort operator:

$$Sros(T) = \text{sum of straight tableaux of } Ros(T).$$

An illustrative example

$T = \begin{matrix}\mathbf{146}\\ \mathbf{235}\end{matrix}$. $Ros(T)$ contains 36 terms:

$$Ros(T) = \left(\begin{matrix}\mathbf{146}\\ \mathbf{235}\end{matrix} - \begin{matrix}\mathbf{146}\\ \mathbf{253}\end{matrix} - \begin{matrix}\mathbf{146}\\ \mathbf{325}\end{matrix} + \begin{matrix}\mathbf{146}\\ \mathbf{352}\end{matrix} + \cdots \right)^{\Downarrow}$$

$$= \begin{matrix}\mathbf{135}\\ \mathbf{246}\end{matrix} - \begin{matrix}\mathbf{143}\\ \mathbf{256}\end{matrix} - \begin{matrix}\mathbf{125}\\ \mathbf{346}\end{matrix} + \begin{matrix}\mathbf{142}\\ \mathbf{356}\end{matrix} + \cdots$$

In contrast, $Sros(T)$ contains 2 terms:

$$Sros(T) = \begin{matrix}\mathbf{135}\\ \mathbf{246}\end{matrix} - \begin{matrix}\mathbf{125}\\ \mathbf{346}\end{matrix} .$$

Related operators in combinatorics

The row-column action by the two groups $(S_n)^d$ and $(S_d)^n$ on a tableau T of $d \times n$ is as follows:

For $\sigma \in (S_n)^d$ and $\tau \in (S_d)^n$, first σ acts on T from the left as the row action within each row, then τ acts on σT from the right as the column action within each column.

This is not a group action.

In Désarménien (1980), the action is denoted by $(\sigma T)\tau$.
In Doubilet, Rota and Stein (1974), the action is written as $\sigma \leftrightarrow T \updownarrow \tau$.

Properties of Ros and Sros

1. Ros and $Sros$ are linear maps from the bracket ring to the tableau ring.
2. Let T be a tableau in pre-normal form, then in both $Ros(T)$ and $Sros(T)$, the leading term is $T^{\Downarrow}$.

   Pre-normal form of a tableau of $d \times n$: along each row the entries are increasing, and the d rows are non-decreasing in the lex order of length-n sequences.
3. Let f be a homogeneous bracket polynomial in pre-normal form, then $Sros(f)$ and the straight tableau form of f have the same leading term.

Algorithm 1 "`Sros`": straightening homogeneous bracket polynomial by successive straight roll-and-sort operations

Input: A homogeneous tableau polynomial F

Output: The straight tableau form of $[F]$

1: Replace F by its pre-normal form.
If F is straight then return F and exit.

2: Set $g :=$ leading term of $F^{\Downarrow}$.
(get first leading term by column sorting)
If $g \prec T^{\Downarrow}$ for some term T of F, then set $g := 0$.

3: Compute $q := Sros(F - g)$.

4: While $q \neq 0$ do
    Set $h :=$ leading term of q.
    Set $g := g + h$. (extract new leading term to the result)
    Set $q := q - Sros(h)$. (subtract $Sros$ of leading term)
End do.

5: **return** g.

Example.

Input: $f = \begin{matrix}\mathbf{146}\\ \mathbf{235}\end{matrix}$.

Round 1. $g = f^{\Downarrow} = \begin{matrix}\mathbf{135}\\ \mathbf{246}\end{matrix}$ is the leading term of the result;

$$q = Sros(f - f^{\Downarrow}) = -\begin{matrix}\mathbf{125}\\ \mathbf{346}\end{matrix} - \begin{matrix}\mathbf{123}\\ \mathbf{456}\end{matrix}.$$

Round 2. The leading term of q is $h_2 = -\begin{matrix}\mathbf{125}\\ \mathbf{346}\end{matrix}$.

$g = g + h_2$, and

$$q = q - Sros(h_2) = -\begin{matrix}\mathbf{123}\\ \mathbf{456}\end{matrix}.$$

The leading term of q is q.

Round 3. $g = g + q$, and $q = 0$.

Result. $f = \begin{bmatrix}\mathbf{135}\\ \mathbf{246}\end{bmatrix} - \begin{bmatrix}\mathbf{125}\\ \mathbf{346}\end{bmatrix} - \begin{bmatrix}\mathbf{123}\\ \mathbf{456}\end{bmatrix}$.

Compute Sros operator efficiently

By definition, very complicated:

$$Sros(T) = \sum_{\sigma=(\mathrm{id},\sigma_2,\ldots,\sigma_d)\in(S_n)^d} \mathrm{sign}(\sigma)\mathrm{sign}(\tau)^d \ \ {}^{\rightarrow}((\sigma T)^{\Downarrow})|_{STAB}.$$

Idea of efficient computing of Sros: control the number of useful elements of $(S_n)^d$ by constraints derived from properties of Sros.

Details of a recursive algorithm are omitted.

Tests

Example. (1)-(4): degree 5, (5)-(8): degree 6, (9)-(12): degree 7. "–" indicates **failure**: either test does not finish in 24 hours on our Dell workstation, or Maple 16 returns "Memory allocation failed".

Table 1: Time (seconds) consumed in straightening

NO.	Young	White	Rota	Rota+	Sros
(1)	–	30240.57	43116.46	16070.75	13.07
(2)	–	35326.12	43095.52	13495.11	9.66
(3)	–	–	–	–	128.83
(4)	–	–	–	–	3798.29
(5)	–	64245.62	–	–	43.95
(6)	–	–	–	–	16417.88
(7)	35624.27	13712.12	8684.90	5721.24	38.13
(8)	39241.46	15328.91	28705.62	8245.29	49.51
(9)	–	56019.29	–	67958.13	3028.11
(10)	–	33261.75	–	23134.19	911.67
(11)	–	–	–	–	7824.91
(12)	–	–	70785.71	36726.39	8167.86

Comparison of examples: degree 5

"Soft" example (1): $\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_{10} & \mathbf{a}_{15} \\ \mathbf{a}_2 & \mathbf{a}_9 & \mathbf{a}_{14} \\ \mathbf{a}_3 & \mathbf{a}_8 & \mathbf{a}_{13} \\ \mathbf{a}_4 & \mathbf{a}_7 & \mathbf{a}_{12} \\ \mathbf{a}_5 & \mathbf{a}_6 & \mathbf{a}_{11} \end{bmatrix}$, success for all but Young:

White (30240.57s),
Rota (43116.46s),
Rota+ (16070.75s),
Sros (13.07s).

"Hard" example (3): $\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_{18} & \mathbf{a}_{19} & \mathbf{a}_{20} \\ \mathbf{a}_2 & \mathbf{a}_{15} & \mathbf{a}_{16} & \mathbf{a}_{17} \\ \mathbf{a}_3 & \mathbf{a}_{12} & \mathbf{a}_{13} & \mathbf{a}_{14} \\ \mathbf{a}_4 & \mathbf{a}_9 & \mathbf{a}_{10} & \mathbf{a}_{11} \\ \mathbf{a}_5 & \mathbf{a}_6 & \mathbf{a}_7 & \mathbf{a}_8 \end{bmatrix}$, success only for

Sros (128.83s).

Comparison of examples: degree 6

"Soft" example (7): $\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_7 & \mathbf{a}_{10} \\ \mathbf{a}_1 & \mathbf{a}_6 & \mathbf{a}_9 \\ \mathbf{a}_1 & \mathbf{a}_5 & \mathbf{a}_8 \\ \mathbf{a}_2 & \mathbf{a}_5 & \mathbf{a}_8 \\ \mathbf{a}_2 & \mathbf{a}_4 & \mathbf{a}_7 \\ \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_7 \end{bmatrix}$, success for all:

Young (35624.27s),
White (13712.12s),
Rota (8684.90s),
Rota+ (5721.24s),
Sros (38.13s).

"Hard" example (6): $\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_{12} & \mathbf{a}_{18} \\ \mathbf{a}_2 & \mathbf{a}_{11} & \mathbf{a}_{17} \\ \mathbf{a}_3 & \mathbf{a}_{10} & \mathbf{a}_{16} \\ \mathbf{a}_4 & \mathbf{a}_9 & \mathbf{a}_{15} \\ \mathbf{a}_5 & \mathbf{a}_8 & \mathbf{a}_{14} \\ \mathbf{a}_6 & \mathbf{a}_7 & \mathbf{a}_{13} \end{bmatrix}$, success only for Sros

(16417.88s).

Comparison of examples: degree 7

"Soft" example (10): $\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_{11} & \mathbf{a}_{12} \\ \mathbf{a}_2 & \mathbf{a}_{10} & \mathbf{a}_{11} \\ \mathbf{a}_2 & \mathbf{a}_9 & \mathbf{a}_{10} \\ \mathbf{a}_3 & \mathbf{a}_8 & \mathbf{a}_9 \\ \mathbf{a}_3 & \mathbf{a}_7 & \mathbf{a}_8 \\ \mathbf{a}_4 & \mathbf{a}_6 & \mathbf{a}_7 \\ \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_6 \end{bmatrix}$, success for

White (33261.75s),
Rota+ (23134.19s),
Sros(911.67s).

"Hard" example (11): $\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_{14} & \mathbf{a}_{15} \\ \mathbf{a}_1 & \mathbf{a}_{13} & \mathbf{a}_{14} \\ \mathbf{a}_2 & \mathbf{a}_{11} & \mathbf{a}_{12} \\ \mathbf{a}_2 & \mathbf{a}_{10} & \mathbf{a}_{11} \\ \mathbf{a}_3 & \mathbf{a}_9 & \mathbf{a}_{10} \\ \mathbf{a}_3 & \mathbf{a}_6 & \mathbf{a}_8 \\ \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_7 \end{bmatrix}$, success only for Sros

(7824.91s).

Table 2: Total number of steps

NO.	Young	White	Rota	Rota+	Sros
(1)	–	11020	18050489	1486066	1718
(2)	–	11581	18050489	982509	1396
(3)	–	–	–	–	3487
(4)	–	–	–	–	11338
(5)	–	19853	–	–	2037
(6)	–	–	–	–	26093
(7)	1682	935	306345	182041	77
(8)	18635	5173	7672345	472412	418
(9)	–	21566	–	403788272	1935
(10)	–	19886	–	89831247	1713
(11)	–	–	–	–	4111
(12)	–	–	46807984	20793721	4799

Table 3: Total number of terms

NO.	Young	White	Rota	Rota+	Sros
(1)	–	24345233	260206664	2133228	856398
(2)	–	35537137	259821978	2023092	634952
(3)	–	–	–	–	4913519
(4)	–	–	–	–	50265949
(5)	–	42898729	–	–	1802416
(6)	–	–	–	–	220261416
(7)	8361222	2108691	13238968	71341	3637
(8)	92634585	12046158	40411868	116173	78849
(9)	–	49019189	–	23695787	1565798
(10)	–	44229315	–	9408518	1568021
(11)	–	–	–	–	6480851
(12)	–	–	934535234	17634525	9007292

Table 4: Numbers of straight tableaux involved

NO.	`Rota` / `Rota+`	`Sros`	Final
(1)	6006	2384	1718
(2)	6006	2782	1396
(3)	1662804	10204	3487
(4)	1662804	21961	11338
(5)	87516	7092	2037
(6)	87516	62623	26093
(7)	782	233	77
(8)	3915	852	418
(9)	41688	4546	1935
(10)	15711	2832	1713
(11)	68151	7293	4111
(12)	9675	6295	4799

Table 5: Time (seconds) consumed at each stage of `Rota`

NO.	Straight tableaux enumeration	Capelli operation	Linear equation solving
(1)	16.72	42243.49	856.25
(2)	16.72	42243.49	835.31
(7)	27.13	8415.86	241.91
(8)	42.41	28211.92	451.29
(12)	504.98	69182.54	1098.19

Conclusion

This paper defines and implements a new operator in bracket algebra: straight roll-and-sort.

The algorithm Sros based on this operator is efficient in practice.

Thanks.