Fast algorithms for computing Gröbner bases

(1) Gröbner bases: What they are and why we like them

Let $S = k[x_1, \ldots, x_n]$.  

Definition 1. A **Gröbner basis** with respect to an order $>$ on $S$ is a set $\{g_1, \ldots, g_t\}$ such that if $I = (g_1, \ldots, g_t)$, then $\text{in}(I) = (\text{in}(g_1), \ldots, \text{in}(g_t))$. We call $g_1, \ldots, g_t$ a **Gröbner basis for** $I$.

Why Gröbner bases are useful:

- Testing for ideal membership.
- Elimination theory/solving systems of polynomial equations.
- Homogenization/finding projective closures.
- Finding the initial ideal (which gives us lots of information).

(2) Buchberger’s Algorithm

Let $g_1, \ldots, g_t \in S$. Are they a Gröbner basis? Compute $s$-pairs for $g_i$ and $g_j$:

$$s_{ij} = m_{ij}g_i - m_{ij}g_j,$$

where $m_{ij} = \frac{\text{in}(g_i)}{\gcd(\text{in}(g_i), \text{in}(g_j))}$. Reduce these differences by $g_k$’s:

$$s_{ij} = \sum f_k^{(ij)}g_k + h_{ij}.$$

Theorem 1 (Buchberger’s Criterion). The set $g_1, \ldots, g_t$ is a Gröbner basis if and only if all the $h_{ij}$’s are 0.

Algorithm 1 (Buchberger’s Algorithm). Input: a set $F = \{g_1, \ldots, g_t\}$. 
Output: A Gröbner basis $G$ for $I = (g_1, \ldots, g_t)$.

1. Let $G := F$. 
2. Select an s-pair from $G$ that has not yet been reduced. (If none, go to (5).)

3. Reduce this s-pair.

4. If reduction is 0, go to 2. If reduction is $h \neq 0$, add $h$ to $G$ and go to 2.

5. Output $G$.

**Remark 1.** We often avoid considering useless s-pairs (ones that will “obviously” reduce to 0) to speed up the run time.

**Remark 2.**

(3) Faugere’s $F4$ Algorithm (barebones)

**Big idea:** Buchberger reduced one s-pair at a time. $F4$ reduces many at once.

**Algorithm 2** ($F4$ Algorithm). Input: a set $F = \{g_1, \ldots, g_t\}$.

Output: A Gröbner basis $G$ for $I = (g_1, \ldots, g_t)$.

1. Let $G := F$.

2. Select a collection of s-pairs from $G$ that has not yet been reduced. (If none, go to (5).)

3. Reduce all s-pairs simultaneously. (Actually, we’re reducing all the $m_j, g_i$’s.)

4. Add any nonzero reductions to $G$, and go to 2.

5. Output $G$.

(4) Details/Improvements for $F4$

(i) We need to select some subset of the s-pairs. If we selected sets of size 1, $F4$ would reduce to Buchberger. Best method tried by Faugere: first all s-pairs of degree 1, then of degree 2, then of degree 3, and so on with each iteration.

(ii) To reduce the s-pairs simultaneously, we will encode them into a matrix, and then reduce to a row-echelon form. This will model polynomial reduction.

Here’s a baby example for modeling polynomial reduction in a matrix. Say we want to reduce $f = X^2 - Y$ by $r = X + 2$. We’ll need to use $Xr = X^2 + 2X$. $f$ and $Xr$ are the only relevant polynomials, so we put them in a matrix as

$$
\begin{pmatrix}
X^2 & X & Y & 1 \\
Xr & \begin{pmatrix} 1 & 2 & 0 & 0 \\
\end{pmatrix} \\
\begin{pmatrix} f & \begin{pmatrix} 1 & 0 & -1 & 0 \end{pmatrix} \end{pmatrix}
\end{pmatrix}.
$$
This matrix reduces to
\[
\begin{pmatrix}
1 & 2 & 0 & 0 \\
0 & -2 & -1 & 0
\end{pmatrix},
\]
which corresponds to the reduction of \( f = X^2 - Y \) to \(-2X - Y\). We could have done this with \emph{lots} of \( f \)'s (the \( s \)-pairs) and \emph{lots} of \( r \)'s (the elements of the tentative Gröbner basis).

Row reduction can be computed quickly since the matrix will be very sparse. Some matrix specs that show up: for solving Cyclic 7, a \( 475 \times 786 \) matrix showed up, with only 13.8% of entries nonzero.

(iii) As with Buchberger, we can eliminate some useless \( s \)-pairs from the beginning. Faugere calls the algorithm with this modification \emph{the improved \( F4 \) algorithm}.

(iv) Faugere improves upon (iii) with his \( F5 \) algorithm.

(5) Actual Computations

Faugere tests improved \( F4 \) versus Buchberger. See [Fau2].

Kim was working on ideals that arise from the min-rank problem yesterday in a coffee shop. He used M2 (Buchberger) and Maple (\( F5 \)). Trying to solve the min-rank problem for five \( 6 \times 6 \) matrices and rank deficiency at least 2, the corresponding Gröbner basis calculation took 53 seconds on M2 and 1.3 seconds on Maple. Upping the matrices to \( 7 \times 7 \), M2 took more then 1000 seconds and Maple took 6.4 seconds. For \( 8 \times 8 \), Maple took 73 seconds.

It’s worth noting that all these computations were for graded reverse lexicographic ordering. An important step is to then convert this to a lexicographic ordering. For the \( 8 \times 8 \) case, this conversion took 725 seconds. (For grevlex, the Gröbner basis had 771 elements. For lex, the Gröbner basis had 84 elements. The ideal was degree 331.)

References


