Reducing the number and size of linear programs in a dynamic Gröbner basis algorithm

John Perry

**ABSTRACT**

The dynamic algorithm to compute a Gröbner basis is nearly twenty years old, yet it seems to have arrived stillborn; aside from two initial publications, there have been no published followups. One reason for this may be that, at first glance, the added overhead seems to outweigh the benefit; the algorithm must solve many linear programs with many constraints. This paper describes two methods of reducing the cost substantially.

**Dynamic Algorithm**

**Idea**


- seek “optimal” ordering while computing basis
- measure “optimality” using Hilbert function

**Pseudocode**

inputs \( F \), generators of polynomial ideal \( I \)

outputs

\( \sigma \), monomial ordering

\( G \), Gröbner basis of \( I \) with respect to \( \sigma \)

1. Let \( G = \{ \} \), \( P = \{ (f, 0) : f \in F \} \), \( \sigma \) any ordering
2. repeat while \( P \neq \emptyset \)
   (a) Select \((p, q) \in P \) and remove it
   (b) Let \( r \) be some \( \sigma \)-normal form of spoly\((p, q)\) modulo \( G \)
   (c) If \( r \neq 0 \)
      i. Add \((g, r) \) to \( P \) for each \( g \in G \)
      ii. Add \( r \) to \( G \)
   (d) Select an ordering \( \tau \)
   (e) Add to \( P \) any \((p, q) \) such that \( p, q \in G \wedge \text{lm}_\tau(p) \neq \text{lm}_\tau(p) \)
   (f) Let \( \sigma = \tau \)
3. return \( G, \sigma \)

**Better Living Through Geometry**

**Geometry of monomial orderings**

- Orderings \( \leftrightarrow \) cones in positive orthant (Gröbner fan) [5]
- Add polynomials? split some cones

\[ g_1 = x^2 + y^2 - 4 \]
\[ g_2 = xy - 1 \]
\[ g_3 = \text{spoly}(g_1, g_2) = y^3 + x - 4y \]
\[ \text{lm}_\sigma(G) = (x^2, xy) \]
\[ \text{lm}_\tau(G) = (xy^2, x, y) \]
\[ \text{lm}_\mu(G) = (y^3, xy, y^3) \]

**Corner vectors**

**Theorem ([2]):** If we know corner vectors \Omega of cone, we need constraints only for monomials \( u \) such that \( \omega(t - u) > 0 \) for all \( \omega \in \Omega \).

- BUT! hard to find all corners
  - find corners that maximize, minimize each \( x_i \)
- BUT! might miss some potential leading monomials
  - \( \text{lm}(G) \) might change later, which is bad
  - add constraints to revert changes

Example: (cross-section of 3d-cone)

**Experimental Results**

<table>
<thead>
<tr>
<th>System</th>
<th>linear programs prevented by...</th>
<th>nmbr cmptd</th>
<th>max size</th>
<th>size of GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caboara 1</td>
<td>24</td>
<td>0</td>
<td>11</td>
<td>22</td>
</tr>
<tr>
<td>Caboara 2</td>
<td>88</td>
<td>0</td>
<td>17</td>
<td>9</td>
</tr>
<tr>
<td>Caboara 4</td>
<td>14</td>
<td>0</td>
<td>22</td>
<td>9</td>
</tr>
<tr>
<td>Caboara 6</td>
<td>3</td>
<td>0</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>Caboara 8</td>
<td>2</td>
<td>0</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Cyc-6</td>
<td>3,942</td>
<td>762</td>
<td>1988</td>
<td>89</td>
</tr>
<tr>
<td>Cyc-7</td>
<td>3,917,437</td>
<td>4,165</td>
<td>8,106</td>
<td>250</td>
</tr>
<tr>
<td>Cyc-6 hom.</td>
<td>2,042</td>
<td>6</td>
<td>83</td>
<td>54</td>
</tr>
<tr>
<td>Cyc-7 hom.</td>
<td>88,774</td>
<td>0</td>
<td>143</td>
<td>104</td>
</tr>
<tr>
<td>Kat-6</td>
<td>751</td>
<td>2</td>
<td>43</td>
<td>57</td>
</tr>
<tr>
<td>Kat-7</td>
<td>3,979</td>
<td>7</td>
<td>85</td>
<td>27</td>
</tr>
<tr>
<td>Kat-6 hom.</td>
<td>533</td>
<td>0</td>
<td>23</td>
<td>77</td>
</tr>
<tr>
<td>Kat-7 hom.</td>
<td>16,556</td>
<td>8</td>
<td>132</td>
<td>222</td>
</tr>
</tbody>
</table>

**Observations and comments**

- "cor vec’s" + "trck" + "nmbr cmptd = #lp’s by divisibility
- substantial reduction in number and size of linear programs
- determining feasibility, ordering no longer bottleneck
- Applied divisibility criterion \( O(n^2) \) comparisons before corner vectors \( O(n) \). Reversing increases "cor vec’s" efficiency.

**The Fine Print**

- Normal strategy. Results sensitive to strategy, first polynomial.
- Sage-5.0 w/Cython (patched). C++ implementation planned.
- "trck" counts programs not computed by/c already rejected.
- Cyc-7 used min. degree strategy, corner vectors first.

**Citations and Acknowledgments**


Thanks to Nathann Cohen for help with linear programming in Sage, and to Massimo Caboara for insight and encouragement.