Sparse multivariate polynomial factorization: A high-performance design and implementation.

Michael Monagan

Centre for Experimental and Constructive Mathematics
Simon Fraser University
British Columbia

This is joint work with Baris Tuncer.

Factoring polynomials using Wang's Hensel lifting

```
> e1 := (a = f*g);
```

$$e1 := x^5 + 20\,x^2y^6z + 5\,x^4y^3z + 30\,xy^4z^3 + 12\,xy^4z^2 + 4\,x^3y^3$$
$$+3\,x^3yz^2 + 18\,y^2z^4 + 6\,x^2yz^2$$
$$= \left(x^2 + 5\,xy^3z + 3\,yz^2\right)\left(x^3 + 4\,xy^3 + 6\,yz^2\right)$$

```
> e2 := eval(e1,z=3);
```

$$e2 := x^5 + 60\,x^2y^6 + 15\,x^4y^3 + 4\,x^3y^3 + 918\,xy^4 + 27\,x^3y + 54\,x^2y + 1458\,y^2$$
$$= \left(x^2 + 15\,xy^3 + 27\,y\right)\left(x^3 + 4\,xy^3 + 54\,y\right)$$

```
> e3 := eval(e2,y=-5);
```

$$e3 := x^5 - 1875\,x^4 - 635\,x^3 + 937230\,x^2 + 573750\,x + 36450$$
$$= \left(x^2 - 1875\,x - 135\right)\left(x^3 - 500\,x - 270\right)$$

Notes: Let h be any factor of a and let $B > \max(||h||_\infty, ||a||_\infty)$.
Multivariate Hensel Lifting (MHL) is done modulo $m = p^L > 2B$.

Wang's Multivariate Hensel Lifting (MHL)

Input $a \in \mathbb{Z}_p[x_1, ..., x_j]$, $\alpha = (\alpha_2, \ldots \alpha_j)$, $(f_0, g_0) \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ s.t.
(i) $a(x_1, ..., x_{j-1}, \alpha_j) = f_0 g_0$ and (ii) $\gcd(f_0(x_1, \alpha), g_0(x_1, \alpha)) = 1$.

Idea: $f = f_0 + f_1(x_j - \alpha_j) + f_2(x_j - \alpha_j)^2 + \cdots + f_{df}(x_j - \alpha_j)^{df}$

Wang's Multivariate Hensel Lifting (MHL)

Input $a \in \mathbb{Z}_p[x_1, ..., x_j]$, $\alpha = (\alpha_2, \ldots \alpha_j)$, $(f_0, g_0) \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ s.t.
(i) $a(x_1, ..., x_{j-1}, \alpha_j) = f_0 g_0$ and (ii) $\gcd(f_0(x_1, \alpha), g_0(x_1, \alpha)) = 1$.

Idea: $f = f_0 + f_1(x_j - \alpha_j) + f_2(x_j - \alpha_j)^2 + \cdots + f_{df}(x_j - \alpha_j)^{df}$

Initialize: $(f, g) := (f_0, g_0)$ and $error := a - f_0\, g_0$

For $k = 1, 2, \ldots,$ while $\deg(f, x_j) + \deg(g, x_j) < \deg(a, x_j)$ do

$c_k :=$ coeff($error$, $(x_j - \alpha_j)^k$)
If $c_k \neq 0$ then

Solve the MDP $f_k\ g_0 + g_k\ f_0 = c_k$ for $f_k, g_k \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$.

Set $f := f + f_k(x_j - \alpha_j)^k$ and $g := g + g_k(x_j - \alpha_j)^k$
Set $error := a - fg$

If $error = 0$ output (f, g) else output FAIL.

Implemented in Magma, Maple, Maxima, Mathematica and Singular.
Algorithms for Computer Algebra, Geddes, Czapor, and Labahn, 1992.
Wang's algorithm is highly sequential. $O(d^n)$ if the α_i's are non-zero.

The Taylor Coefficients : MT CASC 2016

$$f = x^3 - xyz^2 + y^3z^2 + z^4 - 2$$

Idea: $f = f_0 + \sigma_1(z-\alpha_3) + \sigma_2(z-\alpha_3)^2 + \sigma_3(z-\alpha_3)^3 + \sigma_4(z-\alpha_3)^4$.

If $\alpha_3 = 0$ then $f(z) = \underbrace{(x^3-2)}_{f_0} + \underbrace{(y^3-xy)}_{f_2} z^2 + \underbrace{1}_{f_4} z^4$.

If $\alpha_3 = 2$ then

$$\begin{aligned} f(z) \;=\; & \underbrace{(x^3 + 4\,y^3 - 4\,xy + 14)}_{f_0} + \underbrace{(4\,y^3 - 4\,xy + 32)}_{f_1}(z-2) + \\ & \underbrace{(y^3 - xy + 24)}_{f_2}(z-2)^2 + \underbrace{8}_{f_3}(z-2)^3 + \underbrace{1}_{f_4}(z-2)^4 \end{aligned}$$

The Taylor Coefficients : MT CASC 2016

$$f = x^3 - xyz^2 + y^3z^2 + z^4 - 2$$

Idea: $f = f_0 + \sigma_1(z-\alpha_3) + \sigma_2(z-\alpha_3)^2 + \sigma_3(z-\alpha_3)^3 + \sigma_4(z-\alpha_3)^4$.

If $\alpha_3 = 0$ then $f(z) = \underbrace{(x^3-2)}_{f_0} + \underbrace{(y^3-xy)}_{f_2} z^2 + \underbrace{1}_{f_4} z^4$.

If $\alpha_3 = 2$ then

$$\begin{aligned} f(z) \;=\; & \underbrace{(x^3 + 4\,y^3 - 4\,xy + 14)}_{f_0} + \underbrace{(4\,y^3 - 4\,xy + 32)}_{f_1}(z-2) + \\ & \underbrace{(y^3 - xy + 24)}_{f_2}(z-2)^2 + \underbrace{8}_{f_3}(z-2)^3 + \underbrace{1}_{f_4}(z-2)^4 \end{aligned}$$

Lemma 1 [MT 2016] If α_j is chosen at random from a sufficiently large set then

$$\mathrm{Prob}[\,\mathrm{supp}(f_0) \supseteq \mathrm{supp}(f_1) \supseteq \cdots \supseteq \mathrm{supp}(f_k)\,] \text{ is high}$$

MTSHL : Our Sparse Hensel Lifting from CASC 2016

Solve the MDP $f_k\ g_0 + g_k\ f_0 = c_k$ for $f_k, g_k \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$.

1: Let $f_{k-1} = \sum_{i=1}^{df} a_i(x_2, \ldots, x_{j-1}) x_1^i$.
Set $f_k = \sum_{i=0}^{df} \left(\sum_{m \in \mathrm{supp}(a_i)} a_{im} m \right) x_1^i$. assumes $\mathrm{supp}(f_k) \subseteq \mathrm{supp}(f_{k-1})$

2: Set $s = \max(|\mathrm{supp}(a_i)|$. Set $\beta = (\beta_2, \ldots, \beta_{j-1}) \in \mathbb{Z}_p^{j-2}$.

3: For $1 \le j \le s$ solve $\sigma_j\ g_0(\beta^j) + \tau_j\ f_0(\beta^j) = c_k(\beta^j)$ for $\sigma_j, \tau_j \in \mathbb{Z}_p[x_1]$.

Needs $\gcd(g_0(\beta^j), f_0(\beta^j)) = 1\ \forall 1 \le j \le s$.

4: Solve the shifted Vandermonde systems

$$\left\{ coeff(f_k(\beta^j), x_1^i) = coeff(\sigma_j, x_1^i) \text{ for } 1 \le j \le \#a_i \right\} \quad \text{for } 0 \le i < df,$$

For each i needs $M_{ij}(\beta) \neq M_{ik}(\beta) \forall j \neq k$

5: Repeat this for g_k.

New Idea for ICMS 2018

Initialize: $(f, g) := (f_0, g_0)$ and $error := a - f_0\, g_0$

For $k = 1, 2, \ldots,$ while $\deg(f, x_j) + \deg(g, x_j) < \deg(a, x_j)$ do

$c_k :=$ coeff($error$, $(x_j - \alpha_j)^k$)

If $c_k \neq 0$ then

Solve the MDP $f_k\ g_0 + g_k\ f_0 = c_k$ for $f_k, g_k \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$.

Set $f := f + f_k (x_j - \alpha_j)^k$ and $g := g + g_k (x_j - \alpha_j)^k$

Set $error := a - fg$

If $error = 0$ output (f, g) else output FAIL.

1: Evaluate $a(x_1, \beta^r, x_j), f_0(x_1, \beta^r), g_0(x_1, \beta^r)$ for $1 \leq r \leq s$.

2: Hensel lift each image (in parallel) obtaining

$f(x_1, \beta^r, x_j) = \sum_{k=0}^{df} f_k(x_1, \beta^r)(x_j - \alpha_j)^k$ and

$g(x_1, \beta^r, x_j) = \sum_{k=0}^{dg} g_k(x_1, \beta^r)(x_j - \alpha_j)^i$.

Gain: $error := a - fg$ is now computed in $\mathbb{Z}_p[x_1, x_j]$ and not $\mathbb{Z}_p[x_1, x_2, \ldots, x_j]$.

3: Interpolate $coeff(f_k, x_1^i)$ and $coeff(g_k, x_1^i)$ from these images.

Monagan and Tuncer ICMS 2018

$$
\begin{array}{c}
a_j(x_1, x_2, \ldots, x_j) \\
f_{j-1}(x_1, x_2, \ldots, x_{j-1}) \\
\text{evaluate } x_3, \ldots, x_{j-1} \text{ for } 1 \le k \le s \\
\downarrow \\
a_j(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
f_{j-1}(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k) \\
\text{evaluate } x_2 \text{ for } 1 \le l \le \deg(a_j, x_2) \\
\downarrow \\
a_j(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
f_{j-1}(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k)
\end{array}
\quad
\begin{array}{c}
\\ \\ \\ \\ \text{Hensel} \\ \text{lift } x_j \\ \longrightarrow \\ \\ \\ \\ \text{Hensel} \\ \text{lift } x_j \\ \longrightarrow
\end{array}
\quad
\begin{array}{c}
f_j(x_1, x_2, x_3, \ldots, x_j) \\
g_j(x_1, x_2, x_3, \ldots, x_j) \\
\uparrow \\
\text{interpolate } x_3, \ldots, x_{j-1} \\
f_j(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
g_j(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
\uparrow \\
\text{dense interpolate } x_2 \\
f_j(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
g_j(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k, x_j)
\end{array}
$$

Monagan and Tuncer ICMS 2018

$$
\begin{array}{ccc}
a_j(x_1, x_2, \ldots, x_j) & & f_j(x_1, x_2, x_3, \ldots, x_j) \\
f_{j-1}(x_1, x_2, \ldots, x_{j-1}) & & g_j(x_1, x_2, x_3, \ldots, x_j) \\
\text{evaluate } x_3, \ldots, x_{j-1} \text{ for } 1 \le k \le s & & \uparrow \\
\downarrow & & \text{interpolate } x_3, \ldots, x_{j-1} \\
a_j(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) & \text{Hensel} & f_j(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
 & \text{lift } x_j & \\
f_{j-1}(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k) & \longrightarrow & g_j(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
\text{evaluate } x_2 \text{ for } 1 \le l \le \deg(a_j, x_2) & & \uparrow \\
\downarrow & & \text{dense interpolate } x_2 \\
a_j(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) & \text{Hensel} & f_j(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k, x_j) \\
 & \text{lift } x_j & \\
f_{j-1}(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k) & \longrightarrow & g_j(x_1, \gamma_{\mathbf{m}}, \beta_3^k, \ldots, \beta_{j-1}^k, x_j)
\end{array}
$$

Parallelized evaluations $a(x_1, x_2, \beta_3^k, \ldots, \beta_{j-1}^k, x_j)$ using Cilk C
Do evaluations $x_2 = \gamma_{\mathbf{m}}$ and **bivariate Hensel lifts** in parallel.
Optimize the bivariate Hensel lifts in $\mathbb{Z}_p[x_1, x_j]$ mod $(x_j - \alpha_j)^k$.

Benchmark 1 – many variables, modest degree

On a server with 2 Intel Xeon E2660 8 core CPUs – 2.2 GHz(base) – 3.0 GHz(turbo)

			Maple 2018		New times (1 core)			New times (16 cores)		
n	d	t	best	worst	total	(hensel)	(eval)	total	(hensel)	(eval)
6	7	500	0.411	28.84	0.098	(0.015)	(0.042)	0.074	(0.019)	(0.008 – 5.2x)
6	7	1000	1.140	58.46	0.414	(0.025)	(0.247)	0.180	(0.027)	(0.030 – 8.2x)
6	7	2000	3.066	99.88	1.593	(0.041)	**(1.132)**	0.285	(0.042)	**(0.121 – 9.4x)**
6	7	4000	7.173	162.49	5.072	(0.069)	(4.070)	0.814	(0.074)	(0.380 – 10.7x)
6	7	8000	15.61	NA	12.75	(0.122)	(10.95)	1.896	(0.130)	(0.939 – 11.7x)
9	7	500	1.171	7564.9	0.105	(0.013)	(0.040)	0.101	(0.024)	(0.010 – 4.0x)
9	7	1000	3.704	10010.4	0.524	(0.025)	(0.297)	0.233	(0.026)	(0.030 – 11.4x)
9	7	2000	13.43	NA	2.838	(0.042)	**(1.973)**	0.483	(0.045)	**(0.193 – 10.2x)**
9	7	4000	51.77	NA	18.35	(0.078)	(14.84)	2.325	(0.083)	(1.350 – 11.0x)
9	7	8000	NA	NA	116.6	(0.139)	(102.5)	11.50	(0.145)	(7.947 – 12.9x)

Table 1: Timings (real time in seconds) for Hensel lift of x_n.
Legend: $n = \#\text{variables}$, $d = \deg(f, x_j) = \deg(g, x_j)$, $t = \#f = \#g$.

Benchmark 2: higher degree

			Maple 2018		New time (1 core)			New time (16 cores)		
n	d	t	best	worst	total	(hensel)	(eval)	total	(hensel)	(eval)
6	10	500	0.571	92.49	0.099	(0.029)	(0.025)	0.081	(0.024 – 1.2x)	(0.006)
6	15	500	0.751	7956.5	0.134	(0.070)	(0.016)	0.093	(0.034 – 2.1x)	(0.006)
6	20	500	0.919	48610.1	0.238	(0.168)	(0.017)	0.130	(0.065 – 2.6x)	(0.005)
6	40	500	1.615	NA	1.207	**(1.128)**	(0.015)	0.282	**(0.203 – 5.6x)**	
6	80	500	4.485	NA	13.76	(13.65)	(0.016)	1.674	(1.554 – 8.8x)	(0.012)
6	10	2000	5.237	976.94	1.775	(0.089)	(1.067)	0.374	(0.055 – 1.6x)	(0.121)
6	15	2000	7.166	23128.5	1.616	(0.221)	(0.706)	0.413	(0.107 – 2.1x)	(0.061)
6	20	2000	9.195	NA	1.635	(0.451)	(0.480)	0.431	(0.150 – 3.0x)	(0.040)
6	40	2000	15.98	NA	4.008	**(2.993)**	(0.260)	0.854	**(0.505 – 5.9x)**	(0.038)
6	80	2000	57.33	NA	26.34	(25.25)	(0.217)	3.340	(2.839 – 8.9x)	(0.050)

Table 2: Timings (real time in seconds) for increasing degree d.
Legend: $n = \#\text{variables}$, $d = \deg(f, x_j) = \deg(g, x_j)$, $t = \#f = \#g$.

Software Design Issues

For high performance we prefer arrays of machine integers

Data structure for $f = 3x^4 + 5x^2y^3z^4 - 2yz^6 + 7$ in $\mathbb{Z}_p[x, y, z]$.

array of coefficients (64 bit integers):

3	5	-2	7

array of monomials (64 bit integers):

4|0|0	2|3|4	0|1|6	0|0|9

For polynomials in more variables use `__int128_t` and `__uint128_t` in gcc.

Software Design Issues

For multi-core performance using Cilk C we like "in-place" algorithms which reduce memory management.

Solve $f_k g_0 + g_k f_0 = c_k$ given for $f_k, g_k \in \mathbb{Z}_p[x_1]$ given $sg_0 + tf_0 = 1$.

```
for( k=1; k<da; k++ ) {
    ...
    polcopy64s( ck, dck, T );                    // T = ck
    dT  = poldiv64s( T, f0, dt, df0, p );        // T = Tk mod f0
    dfk = polmul64s( T, s, dT, ds, fk, p );      // fk = T * s
    dfk = poldiv64s( fk, f0, dfk, df0, p );      // fk = fk mod f0
    ...
}
```

All routines for $\mathbb{Z}_p[x_1]$ and $\mathbb{Z}_p[x_1, \ldots, x_n]$ do not allocate memory. The array T is reused in the next iteration.

Thank you for attending!