SOME APPLICATIONS OF THE LLL ALGORITHM

KEVIN G. HARE[1]

ABSTRACT. In 1982, Lenstra, Lenstra and Lovász designed an algorithm, usually referred to as LLL, to find "short" vectors within a lattice of vectors. This algorithm, normally studied for vectors in $\mathbb{R}^n$, can be implemented in a more general setting - for vectors of real valued functions. This paper, by using various measures to define the inner product space, demonstrates how this more general setting for LLL can be used to explore interesting properties of real valued functions.

1) DEPARTMENT OF MATHEMATICS, SIMON FRASER UNIVERSITY, BURNABY, BRITISH COLUMBIA, CANADA V5A 1S6, KGHARE@CECM.MATH.SFU.CA

1. Introduction

The algorithm of Lenstra, Lenstra and Lovász, commonly referred to as LLL, is used to find "short" vectors within a lattice of vectors. This algorithm has numerous applications. The original application was for polynomial time factorization of polynommials in $\mathbb{Q}[x]$ [11]. Other applications include finding integer relations between real numbers, computing the integer kernel and image of a matrix, and finding minimal polynomials of real numbers [5]. These applications require an implementation of LLL that works on integer vectors over $\mathbb{R}^n$.

In contrast the applications that this paper looks at include:

1. Finding polynomials with integer coefficients with minimal 2 norm on $[0, 1]$.
2. Finding polynomials with coefficients $-1, 0, 1$ which are divisible by $(1-x)^N$ for some integer N.
3. Finding integer relationships between functions.

The applications in this paper differ from the traditional applications above, in that they use an implementation of LLL where the inner product is for real valued functions, as opposed to vectors in $\mathbb{R}^n$.

2. Background

Here, the definition from [13] is used:

Definition 1 (Lattice). *Let V be a vector space. A set $L \subset V$ is a lattice if there exist finitely many vectors $b_1, ..., b_n$ such that*

$$L = \left\{ \sum_{i=1}^{n} r_i b_i : r_i \in \mathbb{Z}, 1 \le i \le n \right\}.$$

If the set $\{b_1, ..., b_n\}$ is minimal, then this set is called a basis for L of rank n.

For example, $\{(x_1, x_2, x_3) : x_1 \in \mathbb{Z}, x_2 \in 2\mathbb{Z}, 3x_3 \in \mathbb{Z}\}$, is a lattice. One possible basis for this lattice would be $(1, 0, 0)$, $(0, 2, 0)$, $(0, 0, \frac{1}{3})$.

2.1. The Problem.

A given lattice can have many different bases. Some bases for a lattice L are "nicer" than others. For example $(2, 0, 1)$, $(0, 5, 0)$, $(2, 0, -6)$ is a "nicer" basis than $(2, 150, -48)$, $(0, 5, 98)$, $(0, 0, 7)$ though both lead to the same lattice structure. This example demonstrates that not all lattices need have an orthogonal basis. If $\{b_1, ..., b_n\}$ is a basis for a lattice L, and A is an integral matrix with determinant ± 1, then it is shown in [5] that $\{b'_1, ..., b'_n\}$ is a basis for the lattice, where

$$b'_i = \sum_{j=1}^{n} a_{i,j} b_j.$$

Further, it is known that all bases for the lattice L can be found in this way. Now, given the large number of possible bases a lattice could have, it is useful to have criteria for which basis is the most ideal. This paper uses the criteria of [5], thus a "nice" basis for a lattice is defined:

Definition 2 (Reduced). *The basis $b_1, ..., b_n$ for a lattice L is reduced if*

$$\prod_{i=1}^{n} ||b_i||$$

is minimal.

Now recall:

Theorem 1 (Gram Schmidt Orthogonalization). *If $b_1, ..., b_n$ is a basis for a vector space V, define $b_1^*, ..., b_n^*$ by:*

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*,$$

$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}.$$

Then the new basis $b_1^*, ..., b_n^*$ is an orthogonal basis.

The proof of this can be found in any introductory linear algebra text. See for instance [10].

From Hadamard's inequality it follows that

$$\prod_{i=1}^{n} ||b_i|| \geq \prod_{i=1}^{n} ||b_i^*|| \tag{1}$$

with b_i^* defined as in the Gram Schmidt Orthogonalization Theorem. Moreover, $\prod_{i=1}^{n} ||b_i^*|| = \prod_{i=1}^{n} ||b_i'^*||$, where $\{b_i\}$ and $\{b_i'\}$ are different basis for the same lattice.

The problem of finding a reduced basis given an arbitrary basis is NP-hard [8, 12], (hence there are no known polynomial time algorithms).

A major step in solving this problem was made in 1982 by Lenstra, Lenstra and Lovász [11]. They weaken the requirements of what "reduced" meant sufficiently so that the problem could be solved in polynomial time. To this end we redefine what nice is:

Definition 3 (LLL Reduced). *Define $\mu_{i,j}$, b_i^* as in the Gram Schmidt Orthogonalization Theorem. A basis $b_1, ..., b_n$ is an LLL-reduced basis, if*

$$|\mu_{i,j}| \leq \frac{1}{2}, \quad 1 \leq j < i \leq n,$$

and

$$||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2, \quad 1 < i \leq n,$$

(or equivalently)

$$||b_i^*||^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2) ||b_{i-1}^*||^2, \quad 1 < i \leq n.$$

If $|\mu_{i,j}| = 0$, then the basis is an orthogonal basis, which is minimal as a consequence of equation (1). So if $|\mu_{i,j}| \leq \frac{1}{2}$ the basis is "close" to orthogonal, hence "close" to minimal. The second requirement is a bit more mysterious, for a good discussion of it, see [8].

2.2. **Some Properties of LLL.** The reason that LLL Reduced is defined in this way, despite being less restrictive than Definition 2 (Reduced), is that it still has some fairly restrictive properties. Namely, it is known that:

1. $\prod_{i=1}^{n} ||b_i^*|| \leq \prod_{i=1}^{n} ||b_i|| \leq 2^{\frac{n(n-1)}{4}} \prod_{i=1}^{n} ||b_i^*||$
2. $||b_j|| \leq 2^{\frac{i-1}{2}} ||b_i^*||, \ 1 \leq j \leq i \leq n$
3. $||b_1|| \leq 2^{\frac{n-1}{4}} \prod_{i=1}^{n} ||b_i^*||^{\frac{1}{n}}$
4. $\forall x \in L, x \neq 0, ||b_1|| \leq 2^{\frac{n-1}{2}} ||x||$

5. $||b_j|| \le 2^{\frac{n-1}{2}} \max(||x_1||, ..., ||x_t||)$ for $x_1, ..., x_t \in L$, $1 \le j \le t$, and x_j linearly independent.

For a proof of these properties see [5].

There does exist a polynomial time algorithm which finds an LLL Reduced basis, given an arbitrary basis. See [5, 11, 13] for detailed implementations. A detailed analysis would show that its running time is $\mathcal{O}(n^6 \ln^3 B)$ where $||b_i||^2 \le B \ \forall i$ [5]. Further, restricted to integer lattices, there exist variations of LLL which operate in $\mathcal{O}(n^{2.381} \ln B)$, [14].

There are some things to note about this algorithm. They are:

1. This method works over any inner product space
   (a) $\mathbb{R}^n$ with the standard inner product.
   (b) Set of functions on [0,1] with $\langle f, g\rangle = \int f(t)\, g(t)\, d\mu(t)$. (Useful for finding integer combinations of functions with small 2 norm.)
   (c) Set of polynomials on [0,1] with $\langle p(x), q(x)\rangle = p(\alpha)q(\alpha)$ for some point* $\alpha \in \mathbb{R}$. (Useful for finding minimal polynomials.)
2. The constant $\frac{3}{4}$ in Definition 3 (LLL Reduced) can be replaced by any c such that:
$$0 < c - \mu_{i,i-1}^2 < 1$$
$$\frac{1}{4} < c < 1$$
   The condition for LLL-reduced is then modified to:
$$||b_i^*||^2 \ge (c - \mu_{i,i-1}^2)||b_{i-1}^*||^2$$
   See [5] as to why this value can be altered. If c is close to 1, there are tighter restrictions to the properties 1 to 5 of the functions. The value 2 in the properties 1 to 5 should be replaced by $\frac{1}{c-\frac{1}{4}}$. If c is close to $\frac{1}{4}$ then the algorithm theoretically runs faster. (Although in practice the speedup is not that great.) Ideally running the algorithm with $c = 1$ would give the tightest restrictions on the basis, but in this case, the proof that the algorithm will terminate does not work. (Although in practice it does terminate in a reasonable amount of time.)

To this end the code presented in the appendix will allow for an arbitrary inner product, as well as any value of c. In the next three sections, we give examples of how to use this more general inner product to solve some interesting problems in number theory.

3. Small 2 norm of polynomials

The goal of this problem is to find a polynomial, with integer coefficients, of degree 21, such that the integral of the square from 0 to 1 is minimized. This is applicable to the study of the Integer Chebyshev Problem, see [2].

To do this consider a basis $1, x, ..., x^{21}$, with the inner product of polynomals $v(x), w(x)$ defined by $\int_0^1 v(x)\, w(x)\, dx$. An LLL-Reduced basis for this has basis elements of very small norm. Also, these basis elements are "sorted" (roughly) so that the first basis vector has the smallest norm. So it suffices to look at the first

*Note: if $p(\alpha) = 0$ for some $p(x) \ne 0$, then this is only a semi-inner product, in which the algorithm does not work. If this is the case, simply add the semi-inner product to a standard inner product to give a valid inner product.

element of the LLL-reduced basis as the candidate for the polynomial with very small norm.

```
> read LLLPoly;
> Basis := [seq(x^i, i=0..21)];
```

$$Basis := [1,\, x,\, x^2,\, x^3,\, x^4,\, x^5,\, x^6,\, x^7,\, x^8,\, x^9,\, x^{10},\, x^{11},\, x^{12},\, x^{13},\, x^{14},\, x^{15},\, x^{16},\, x^{17},\, x^{18},\, x^{19},\, x^{20},\, x^{21}]$$

```
> Small := LLLPoly( Basis, x, c=0.99, iprod=(
>     (v,w)->int( v*w, x= 0..1))) [1];
```

$$Small := -x^7 + 23\,x^8 - 240\,x^9 + 1508\,x^{10} - 6383\,x^{11} + 71545\,x^{14} - 89938\,x^{15} + 19279\,x^{12} - 42903\,x^{13} + 29735\,x^{18} - 10130\,x^{19} + 2100\,x^{20} - 200\,x^{21} + 84898\,x^{16} - 59293\,x^{17}$$

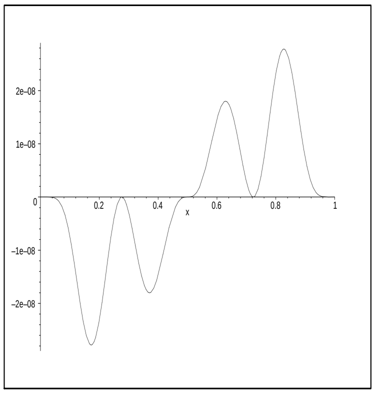
```
> int(Small^2,x=0..1);
```

$$\frac{1}{5843416971961800}$$

```
> factor(Small);
```

$$-x^7\,(1 - 5\,x + 5\,x^2)^2\,(2\,x - 1)^3\,(x - 1)^7$$

```
> plot(Small,x=0..1);
```



It can be seen in [3] the factors $x, (1-x), (2x-1)$ and $(5x^2-5x+1)$ are required factors for large degree Integer Chebyshev polynomials.

4. Small coefficient norm of polynomials

The goal of this problem is to find a polynomial of small degree with coefficients which are either $0, 1$ or -1, that is divisible by $(1-x)^N$ for some integer N. If in addition to this, the polynomial contains only $2N$ nonzero coefficients, then it is a polynomial as described in the Prouhet-Tarry-Escott problem. (The one type of problem is not provably a subset of the other.) See [2, 4].

To do this, define the inner product of the two polynomials to be the sum of the product of the coefficients of the same degree. For example $\langle ax^2+bx+c, dx^2+ex+f\rangle = ad+be+cf$. Thus a small basis vector has few non-zero coefficients, each of small modulus. To ensure that the polynomial has the desired divisibility requirement, simply start with a basis of M vectors where $(1-x)^N$ divides each basis element. Then run the LLL algorithm for increasing sized starting basis (altering M), until the desired polynomial is found.

```
> read LLLPoly;
> innerprod := proc(v,w) RETURN(add((coeff(v,x,i)*coeff(w,x,i))^1,
>         i=0..max(degree(v,x),degree(w,x)))); end:
> N := 3: M := 3: Basis := [seq(x^i*(1-x)^N,i=0..M)];
```

$$Basis := [(1-x)^3,\ x\,(1-x)^3,\ x^2\,(1-x)^3,\ x^3\,(1-x)^3]$$

```
>Basis := expand(Basis):
> P := LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];
```

$$P := x^4+x^5-x^6+1-x-x^2$$

```
> factor(P);
```

$$-(x+1)\,(x^2+x+1)\,(x-1)^3$$

```
> N := 4: M := 7: Basis := [seq(expand(x^(i)*(1-x)^N),i=0..M)]:
> P := LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];
```

$$P := -x^9-x^{10}+x^{11}-x-x^2+x^4+x^7+1$$

```
> factor(P);
```

$$(x+1)\,(x^2+x+1)\,(x^4+x^3+x^2+x+1)\,(x-1)^4$$

```
> N := 5: M := 10: Basis := [seq(expand(x^(i)*(1-x)^N),i=0..M)]:
> P := LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];}{
```

$$P := 1-x-x^2+x^5+x^6+x^7-x^8-x^9-x^{10}+x^{14}-x^{15}+x^{13}$$

```
> factor(P);
```

$$-(x^2+x+1)\,(x^4+x^3+x^2+x+1)\,(x^2+1)\,(x+1)^2\,(x-1)^5$$

```
> N := 6: M := 16: Basis := [seq(expand(x^(i)*(1-x)^N),i=0..M)]:
> P := LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];
```

$$P := 1-x-x^2+x^5+x^6-x^{10}-x^{12}-x^{20}-x^{21}+x^{22}+x^{16}+x^{17}$$

```
> factor(P);
```

$$(x^2+x+1)\,(x^4+x^3+x^2+x+1)\,(x^6+x^5+x^4+x^3+x^2+x+1)\,(x^2+1)\,(x+1)^2\,(x-1)^6$$

5. Relationships between functions

The goal of this problem is to find a relationship between some know functions. As the functions are analytic, it suffices to look at minimizing the 2 norm between 1 and 2 (to avoid the singularity). First, put the known functions into the basis, as well as a very small constant. The constants will be worked out afterwards.

```
> read LLLPoly;
> Basis := [lnGAMMA(3*z), 10^(-9),ln(3)*z,
>     lnGAMMA(z), lnGAMMA(z+1/3), lnGAMMA(z+2/3)];
```

$$Basis := [\mathrm{lnGAMMA}(3\,z),\ \frac{1}{1000000000},\ \ln(3)\,z,\ \mathrm{lnGAMMA}(z),\ \mathrm{lnGAMMA}(z+\frac{1}{3}),\ \mathrm{lnGAMMA}(z+\frac{2}{3})]$$

```
> Formula := LLLPoly(Basis, x, iprod=((v,w)->
>     evalf( int( v*w, z=1..2))), c=0.99)[2];
```

$$Formula := \mathrm{lnGAMMA}(z+\frac{2}{3}) + \mathrm{lnGAMMA}(z) - \frac{2387183211}{1000000000} - \mathrm{lnGAMMA}(3\,z) + 3\ln(3)\,z + \mathrm{lnGAMMA}(z+\frac{1}{3})$$

Next the constant needs to be determined. Guessing that it is a sum of logarithms leads to the following result.

```
> Const[1] := (2387183211/1000000000);
```

$$Const_1 := \frac{2387183211}{1000000000}$$

```
> Const[2] := ln(2);
```

$$Const_2 := \ln(2)$$

```
> Const[3] := ln(3);
```

$$Const_3 := \ln(3)$$

```
> Const[4] := ln(5);
```

$$Const_4 := \ln(5)$$

```
> Const[5] := ln(Pi);
```

$$Const_5 := \ln(\pi)$$

```
> M := 10^5;
```

$$M := 100000$$

```
> Basis := [seq(x^i-evalf(M*Const[i]),i=1..5)];
```

$$Basis := [x - 238718.3211,\ x^2 - 69314.71806,\ x^3 - 109861.2289,\ x^4 - 160943.7912,\ x^5 - 114472.9886]$$

```
> innerprod := proc(v,w)
>     RETURN(add((coeff(v,x,i)*coeff(w,x,i))^1,
>         i=0..max(degree(v,x),degree(w,x)))); end:
>
> LLLPoly(Basis,x,iprod=innerprod);
```

$$[x^3 + .00002 + 2\,x^2 - 2\,x + 2\,x^5,\ 26\,x^2 + .56940 + 5\,x - 9\,x^4 - 12\,x^3 - 2\,x^5,\\ x^5 + 19.26418 - 4\,x^3 - x^2 + 3\,x - 2\,x^4,\ -x^4 + 8.44414 + 19\,x^3 - 26\,x^2 - x + x^5,\\ -2\,x + 13.05674 + 22\,x^2 + 3\,x^3 - 5\,x^4 - 5\,x^5]$$

From this it is seen that 2 times the mystery constant is $2\ln(2)+\ln(3)+2\ln(\pi)$, which gives

```
> C := Const[3]/2+Const[2]+Const[5];}{
```

$$C := \frac{1}{2}\ln(3) + \ln(2) + \ln(\pi)$$

So the final formula

```
> Formula + Const[1] - C;
```

$$0 = \mathrm{lnGAMMA}(\mathrm{z} + \frac{2}{3}) + \mathrm{lnGAMMA}(\mathrm{z}) - \mathrm{lnGAMMA}(3\,\mathrm{z}) + 3\ln(3)\,\mathrm{z}\\ + \mathrm{lnGAMMA}(\mathrm{z} + \frac{1}{3}) - \frac{1}{2}\ln(3) - \ln(2) - \ln(\pi)$$

This could have also been derived by taking the logarithm of the triplication formula for GAMMA, as given in [1].

References

1. Milton Abramowitz and Irene A. Stegun (eds.), *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, Dover Publications Inc., New York, 1992, Reprint of the 1972 edition.
2. Peter Borwein, *Some old problems on polynomials with integer coefficients*, Approximation theory IX, Vol. I. (Nashville, TN, 1998), Vanderbilt Univ. Press, Nashville, TN, 1998, pp. 31–50.
3. Peter Borwein and Tamás Erdélyi, *The integer Chebyshev problem*, Math. Comp. **65** (1996), no. 214, 661–681.
4. Peter Borwein and Michael J. Mossinghoff, *Polynomials with height 1 and prescribed vanishing at 1*, Experiment. Math. **9** (2000), no. 3, 425–433.
5. Henri Cohen, *A course in computational algebraic number theory*, Springer-Verlag, Berlin, 1993.
6. K.O. Geddes, S.R. Czapor, and G. Labahn, *Algorithms for computer algebra*, Kluwer Academic Publishers, 1996.
7. K.O. Geddes, G. Labahn, M. B. Monagan, and S. Vorketter, *The Maple programming guide*, Springer-Verlag, New York, 1996.
8. Martin Grötschel, László Lovász, and Alexander Schrijver, *Geometric algorithms and combinatorial optimization, (algorithms and combinatorics; 2)*, Springer-Verlag, New York, 1980.
9. K. G. Hare, *Home page*, http://www.cecm.sfu.ca/~kghare, 1999.
10. Kenneth Hoffman and Ray Kunze, *Linear algebra*, second ed., Prentice-Hall, Inc, Toronto, 1971.
11. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), no. 4, 515–534.
12. Lovász, (unpublished).

13. Maurice Mignotte, *Mathematics for computer algebra*, Springer-Verlag, New York, 1992, Translated from the French by Catherine Mignotte.
14. Arne Storjohann, *Algorithms for integer lattice basis reduction*, Tech. Report 249, Departement Informatik, ETH Zurich, July 1996.

6. Appendix: Code

This follows the algorithm given in [13], and is written in Maple [7, 6]. This code is also available at [9].

```
# This follows the algorithm given in Maurice Mignotte's
# "Mathematics for Computer Algebra"

LLLPoly := proc()
    local dim, basis, new_basis, local_iprod, local_c, i,
            j, u, B, k, h, l, param;

    basis := args[1];
    param := args[2];

    dim := nops(basis);
    userinfo(2, 'LLLPoly', "Working on dimension "||dim||" problem.");

    local_iprod := subs(x=param,proc(v,w)
            RETURN(int(v*w,x=0..1)):
        end):

    local_c := 0.75;

    for i from 3 to nargs do
        if op(1,args[i]) = iprod then
            local_iprod := op(2,args[i]);
        elif op(1,args[i]) = c then
            local_c := op(2,args[i]);
        fi;
    od;

    for i from 1 to dim do
        userinfo(2, 'LLLPoly', "Using Gram-Schmitd on row "||i||" of "||dim);

        new_basis[i] := basis[i];
        for j from 1 to i - 1 do
            u[i][j] := local_iprod(basis[i], new_basis[j])/B[j];
            new_basis[i] := new_basis[i] -u[i][j]*new_basis[j];
        od;
        B[i] := local_iprod(new_basis[i],new_basis[i]);
    od;

    for j from 1 to dim do
        new_basis[j] := basis[j];
    od;
```

```
    k := 2;

    while k <= dim do
        userinfo(3, 'LLLPoly', "Working on row "||k|| " of "||dim);
        `LLLPoly/trans`(u, k-1, k, dim, [new_basis], local_iprod);

        if B[k] < (local_c-u[k][k-1]^2)*B[k-1] then
            `LLLPoly/modif`(k, u, B, dim, [new_basis], local_iprod);
            if k > 2 then k := k - 1; fi;
            userinfo(1, 'LLLPoly', "Modified "||k|| " of "||dim);
            next;
        fi;

        for h from k-2 by -1 to 1 do
            `LLLPoly/trans`(u, h, k, dim, [new_basis], local_iprod);
        od;

        k := k + 1;
    od;

    userinfo(3, 'LLLPoly', "Finished working on LLL");
    RETURN([seq(new_basis[i],i=1..dim)]);
end:

`LLLPoly/trans` := proc(u, h, k, dim, basis, iprod)
    local r, i, j, b, l;

    b := basis[1];
    if abs(u[k][h]) > 1/2 then
        r := floor(u[k][h] + 1/2);
        b[k] := b[k] - r*b[h];

        for j from 1 to h-1 do
            u[k][j] := u[k][j] - r*u[h][j];
        od;

        u[k][h] := u[k][h] - r;
    fi;
end:

`LLLPoly/modif` := proc(k, u, B, dim, b, iprod)
    local j, u_temp, B_temp, u_prime, b_temp, u_perm, i, basis;

    basis := b[1];
    u_temp := u[k][k-1];
    B_temp := B[k] + u_temp^2*B[k-1];
    u[k][k-1] := u_temp*B[k-1]/B_temp;
    B[k] := B[k-1]*B[k]/B_temp;
```

```
    B[k-1] := B_temp;
    b_temp := copy(basis[k]);
    basis[k] := copy(basis[k-1]);
    basis[k-1] := copy(b_temp);

    for j from 1 to k-2 do
        u_perm := u[k-1][j];
        u[k-1][j] := u[k][j];
        u[k][j] := u_perm;
    od;

    for i from k+1 to dim do
        u_prime := u[i][k-1] - u_temp*u[i][k];
        u[i][k-1] := u[i][k] + u[k][k-1]*u_prime;
        u[i][k] := u_prime;
    od;
end:

#savelib(''LLLPoly/trans'',''LLLPoly/modif'',''LLLPoly'',''LLLPoly.m''):
```