Masarykova univerzita v Brně

Přírodovědecká fakulta

Jiřina Forbelská

LLL algoritmus a jeho aplikace

Bakalářská práce

Vedoucí práce: Mgr. Michal Bulant, Ph.D.

Brno, 2006

Jméno a příjmení autora:	Jiřina Forbelská
Název bakalářské práce:	LLL algoritmus a jeho aplikace
Název v angličtině:	LLL algorithm and its applications
Studijní program:	Matematika
Studijní obor:	Matematika se zaměřením na vzdělávání, Informatika a druhý obor
Vedoucí bakalářské práce:	Mgr. Michal Bulant, Ph.D.
Rok obhajoby:	2006
Klíčová slova v češtině:	Gram-Schmidtův ortogonalizační proces, mřížka, LLL algoritmus, Merkle-Hellman šifrovací systém
Klíčová slova v angličtině:	Gram-Schmidt Orthogonalization, lattice, LLL algorithm, Merkle-Hellman Knapsack Cryptosystem

LLL algoritmus publikovali v roce 1982 A. K. Lenstra, H. W. Lenstra a L. Lovász jako algoritmus pro faktorizaci polynomů s racionálními koeficienty. LLL algoritmus hledá redukovanou bázi nebo nejkratší vektor mřížky, avšak mnoho zajímavých matematických i informatických problémů lze převést na některý z těchto problémů volbou vhodné mřížky. Cílem práce je představit LLL algoritmus, jeho teoretické aspekty a matematické pozadí, ale i četné aplikace. Na příkladu kryptoanalýzy Merkle-Hellman šifrovacího systému je pak ukázáno použití LLL algoritmu včetně konkrétních příkladů.

LLL algortihm was invented in 1982 by A. K. Lenstra, H. W. Lenstra and L. Lovász for factoring polynomials with rational coefficients. LLL algorithm is using for searching for reduced basis or the shortest vector of a lattice, but many interesting problems from mathematics and informatics can be transformed to these problems with a fitted lattice. The intention of this thesis is to introduce the LLL algorithm, its theoretical aspects, mathematical background and its multiple applications. There is shown an application of LLL algorithm including the illustrations in the thesis: the cryptoanalysis of Merkle-Hellman Knapsack Cryptosystem.

Prohlášení

Prohlašuji, že tuto práci jsem vypracovala samostatně. Veškerou literaturu a ostatní prameny, z nichž jsem při přípravě práce čerpala, řádně cituji a uvádím v seznamu použité literatury.

V Brně 9. 5. 2006 Jiřina Forbelská

Poděkování

Chtěla bych poděkovat Mgr. Michalu Bulantovi, Ph.D., vedoucímu práce, za cenné připomínky, podněty a motivaci v průběhu psaní práce. Jsem vděčná také všem učitelům a spolužákům, kteří mi pomohli získat potřebné znalosti a dovednosti, především Mgr. Martinu Dvořákovi, Ph.D. a RNDr. Romanu Plchovi, Ph.D., díky kterým jsem získala dovednosti klíčové pro napsaní této práce.
Zvláštní dík patří i mému manželovi a rodičům za jejich podporu při studiu.

Obsah

Úvod **6**

Často používané označení **8**

1 Matematický základ algoritmu **9**
1.1 Kvadratické a bilineární formy, prostory se skalárním součinem 9
1.2 Mřížky 13

2 LLL algoritmus **19**
2.1 LLL-redukovaná báze 19
2.2 Popis LLL algoritmu 22

3 Aplikace **29**
3.1 Problém batohu 31
3.2 Šifrovací systémy založené na problému batohu – prohraná sázka kryptografie? 32
3.2.1 Šifrovací systémy Merkle-Hellman založené na problému batohu 32
3.2.2 Kryptoanalýza Merkle-Hellman šifrovacího systému založeného na aditivním problému batohu 34
3.2.3 Řešení batohů s nízkou hustotou 35
3.2.4 Využití diofantických aproximací 37
3.2.5 Bezpečnost a budoucnost těchto šifrovacích systémů . . 42

Seznam použité literatury **42**

Přílohy **47**
A pseudokód LLL algoritmu 48
B Šifrovací systém Merkle-Hellman 50

Úvod

LLL algoritmus publikovali v roce 1982 A. K. Lenstra, H. W. Lenstra a L. Lovász jako algoritmus pro faktorizaci polynomů s racionálními koeficienty v článku [31], který spolu s [7, Chapter 2] tvoří hlavní zdroje, z kterých tato práce čerpá. Jedná se o poměrně jednoduchý algoritmus, který lze snadno naprogramovat a jehož pochopení nevyžaduje žádné speciální znalosti. Ihned po svém zveřejnění se však stal katalyzátorem dalšího vývoje a to v matematice, v informatice, ale také v kryptoanalýze a kryptografii. Velice rychle byla nacházena jeho další a další použití, algoritmus byl postupně specializován pro jednotlivé aplikace, ale také různě zefektivňován. Proto bude naším cílem představit tento algoritmus českému čtenáři. Rozsah práce bohužel nedovoluje věnovat se této problematice v celé její šíři, proto často odkazujeme na další zdroje, které mohou dotvořit představu o algoritmu a jeho aplikacích i o dalších souvislostech.

Na straně čtenáře předpokládáme základní znalosti z lineární algebry a geometrie (alespoň na úrovni bakalářského studia učitelství matematiky či elektronického učebního textu [50]). Algoritmus je postaven na zobecnění Gram-Schmidtova ortogonalizačního procesu, který můžeme provádět v prostorech se skalárním součinem. Teorii těchto prostorů stručně připomínáme v 1. části 1. kapitoly, přičemž vybíráme pouze základní věty a definice, které jsou klíčové pro porozumění LLL algoritmu. Pomíjíme zde důkazy a často i některé souvislosti mezi definovanými pojmy – tohoto zjednodušení se dopouštíme především proto, že k tématu existuje dostatek kvalitní literatury i v češtině a slovenštině, především [50, 21, 2]. Podrobně dokazujeme pouze samotný Gram-Schmidtův ortogonalizační proces, kde zavádíme i označení používané v celé další práci. Ve druhé části 1. kapitoly už se zaměřujeme podrobněji na mřížky, tedy strukturu, nad kterou pracuje LLL algoritmus. Oproti [7, Definition 2.5.2] se omezujeme pouze na mřížky nad vektorovým prostorem $\mathbb{R}^n$, a to především proto, že z hlediska aplikací téměř o nic nepřicházíme a tato definice je jednodušší a nevyžaduje definování dalších pojmů.

Ve 2. kapitole se soustředíme na samotný LLL algoritmus – pro dobré pochopení této kapitoly předpokládáme u čtenáře základní znalosti z teorie

algoritmů (definice algoritmu, časová složitost, korektnost algoritmu), které jsou shrnuty např. v elektornickém učebním textu [47]. Samotný algoritmus tu uvádíme formálním zápisem používaným např. v [7], tvar vhodný pro zápis v některém z vyšších programovacích jazyků je uveden v příloze.

Ve 3. kapitole alespoň stručně zmiňujeme hlavní oblasti použití LLL algoritmu. Poté podrobněji rozebíráme jednu z nejstarších aplikací algoritmu: útok na jedno-iterativní verzi šifrovacího systému Merkle-Hellman. Důvodem, proč jsme zvolili takto „starou“ aplikaci, je to, že novější aplikace tohoto algoritmu jsou zpravidla mnohem komplikovanější a vyžadovaly by zavedení mnoha dalších pojmů. Navíc zde můžeme alespoň částečně ukázat to, jak mocnou zbraní v souboji kryptoanalýzy a kryptografie se stal LLL algoritmus, který vznikl jen několik let po zveřejnění myšlenky šifrovacích systémů s veřejným klíčem (přestože jeho autoři původně zřejmě vůbec nepomýšleli na využití v kryptoanalýze či kryptografii). Část 3.2.4 předpokládá alespoň základní znalosti teorie diofantických aproximací. K podrobnějšímu studiu tohoto tématu můžeme doporučit např. [5, 30, 22]. Pro hrubší představu by však měla být tato část srozumitelná i bez předchozích znalostí této problematiky. Uvádíme ji především proto, že použití diofantických aproximací výrazně rozšiřuje použití LLL algoritmu při kryptoanalýze zvoleného systému a učinilo tento systém prokazatelně napadnutelným v polynomiálním čase. Hlavní zdroje, z kterých celá 3. kapitola čerpá, jsou [26, 15, 39, 33], které můžeme doporučit k rozšiřujícímu studiu tématu šifrovacích systémů založených na problému batohu a jejich kryptoanalýzy.

Často používané označení

$\Longleftrightarrow$	právě tehdy když
$\wedge$	a zároveň
$\forall$	obecný kvantifikátor
$\exists$	existenční kvantifikátor
E_n	jednotková matice řádu n
$\det(A)$	determinant matice A
A^T	matice transponovaná k matici A
A^{-1}	inverzní matice k matici A
$GL_n(\mathbb{Z})$	grupa regulárních čtvercových matic řádu n s celočíselnými prvky
$\mathbb{N}$	množina přirozených čísel
$\mathbb{N}_0$	$\mathbb{N} \cup \{0\}$
$\mathbb{Z}$	množina celých čísel
$\mathbb{Q}$	množina racionálních čísel
$\mathbb{R}$	množina reálných čísel
$\mathbb{R}^+$	množina kladných reálných čísel
$\mathbb{R}^n$	množina uspořádaných n-tic reálných čísel
$\gcd(a, b)$	největší společný dělitel čísel a a b
$\mathbf{u}$	vektor
$\mathbf{o}$	nulový vektor
$[\mathbf{u_1}, \ldots, \mathbf{u_k}]$	vektorový podprostor generovaný vektory $\mathbf{u_1}, \ldots, \mathbf{u_k}$

1. Matematický základ algoritmu

1.1 Kvadratické a bilineární formy, prostory se skalárním součinem

Nechť dále T je těleso, jehož charakteristika je různá od 2 a V vektorový prostor konečné dimenze nad tělesem T.

Definice 1.1.1. Zobrazení $b : V \times V \to T$ se nazývá *bilineární forma na vektorovém prostoru* V, jestliže pro každé tři vektory $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ a každé $r \in T$ platí:

(i) $b(\mathbf{u} + \mathbf{v}, \mathbf{w}) = b(\mathbf{u}, \mathbf{w}) + b(\mathbf{v}, \mathbf{w})$

(ii) $b(\mathbf{u}, \mathbf{v} + \mathbf{w}) = b(\mathbf{u}, \mathbf{v}) + b(\mathbf{u}, \mathbf{w})$

(iii) $b(r\mathbf{u}, \mathbf{v}) = b(\mathbf{u}, r\mathbf{v}) = rb(\mathbf{u}, \mathbf{v})$

Poznámka 1.1.2. Bilineární zobrazení b je tedy takové zobrazení, které je lineární v obou složkách.

Definice 1.1.3. Řekneme, že bilineární forma b na V je *symetrická*, resp. *antisymetrická*, jestliže pro každé dva vektory $\mathbf{u}, \mathbf{v} \in V$ platí $b(\mathbf{u}, \mathbf{v}) = b(\mathbf{v}, \mathbf{u})$, respektive $b(\mathbf{u}, \mathbf{v}) = -b(\mathbf{v}, \mathbf{u})$.

Věta 1.1.4. *Ke každé bilineární formě b na V existuje právě jedna symetrická bilineární forma b_s a právě jedna antisymetrická bilineární forma b_a na V tak, že:*

$$b(\mathbf{u}, \mathbf{v}) = b_s(\mathbf{u}, \mathbf{v}) + b_a(\mathbf{u}, \mathbf{v}).$$

Poznámka 1.1.5. Hledané formy b_s a b_a jsou tvaru:

$$b_s(\mathbf{u}, \mathbf{v}) = \frac{1}{2}(b(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, \mathbf{u})),$$
$$b_a(\mathbf{u}, \mathbf{v}) = \frac{1}{2}(b(\mathbf{u}, \mathbf{v}) - b(\mathbf{v}, \mathbf{u})),$$

Poznámka 1.1.6. Nechť $M = \{\mathbf{u_1}, \dots \mathbf{u_n}\}$ je libovolná báze ve V. Potom $\mathbf{u} = x_1\mathbf{u_1} + \cdots + x_n\mathbf{u_n}, \mathbf{v} = y_1\mathbf{u_1} + \cdots + y_n\mathbf{u_n}$, kde $x_i, y_i \in T$. Dosazením do bilineární formy dostaneme:

$$b(\mathbf{u}, \mathbf{v}) = b\left(\sum_{i=1}^{n} x_i\mathbf{u_i}, \sum_{j=1}^{n} y_j\mathbf{u_j}\right) = \sum_{i=1}^{n} x_i \sum_{j=1}^{n} y_j b(\mathbf{u_i}, \mathbf{u_j}).$$

Označme $a_{ij} = b(\mathbf{u_i}, \mathbf{u_j})$ a uvažujme matici $A_b = (a_{ij})$. Potom můžeme psát:

$$b(\mathbf{u}, \mathbf{v}) = \sum_{i,j=1}^{n} a_{ij}x_iy_j. \tag{1.1}$$

Definice 1.1.7. (1.1) nazýváme *analytickým vyjádřením formy b v bázi M* a matice $A_b = (b(\mathbf{u_i}, \mathbf{u_j}))$ se nazývá *matice bilineární formy b v bázi M.*

Poznámka 1.1.8. (1) Pro libovolnou bázi $M = \{\mathbf{u_1}, \dots \mathbf{u_n}\}$ vektorového prostoru V_n a libovolnou čtvercovou matici $A = (a_{ij})$ řádu n nad tělesem T existuje právě jedna bilineární forma b na prostoru V_n, jejíž matice vzhledem k bázi M je právě matice A.

(2) Bilineární forma b je symetrická (antisymetrická) právě tehdy, když A_b je symetrická (antisymetrická) matice.

Věta 1.1.9. *Nechť M a M' jsou dvě báze vektorového prostoru V_n a b je bilineární forma na V_n. Je-li A matice formy b vzhledem k bázi M a C matice přechodu od báze M k bázi M', pak $C^T AC$ je matice formy b vzhledem k bázi M'.*

Definice 1.1.10. *Hodností bilineární formy b* rozumíme hodnost matice formy A_b v libovolné bázi. Je-li A_b regulární matice, nazýváme bilineární formu b *regulární*, je-li A_b singulární, nazýváme bilineární formu b *singulární.*

Definice 1.1.11. Zobrazení $q\colon V \to T$ se nazývá *kvadratická forma na vektorovém prostoru V*, jestliže existuje bilineární forma b taková, že pro každý vektor $\mathbf{u} \in V$ je $q(\mathbf{u}) = b(\mathbf{u}, \mathbf{u})$. Říkáme, že bilineární forma b určuje kvadratickou formu q.

Věta 1.1.12. *Ke každé kvadratické formě existuje právě jedna symetrická bilineární forma, která ji určuje.*

Poznámka 1.1.13. Příslušná bilinární forma má tvar:

$$b(\mathbf{u}, \mathbf{v}) = \frac{1}{2}(q(\mathbf{u} + \mathbf{v}) - q(\mathbf{u}) - q(\mathbf{v})),$$

Definice 1.1.14. *Hodností kvadratické formy q* rozumíme hodnost příslušné symetrické bilineární formy ke q. Říkáme, že kvadratická forma je *regulární (singulární)*, je-li regulární (singulární) příslušná symetrická bilineární forma.

Definice 1.1.15. Bázi $M = \{\mathbf{u_1}, \dots \mathbf{u_n}\}$ vektorového prostoru V nazýváme *polární bází kvadratické formy q*, jestliže pro každé $i, j = 1, \dots, n, i \neq j$, platí $b(\mathbf{u_i}, \mathbf{u_j}) = 0$, kde b je bilineární forma, která určuje kvadratickou formu q.

Věta 1.1.16. *Ke každé kvadratické formě existuje polární báze.*

Věta 1.1.17. *Ke každé kvadratické formě*

$$q(\mathbf{u}) = \sum_{i,j=1}^{n} a_{ij} x_i x_j$$

existuje taková lineární transformace souřadnic

$$x_i = \sum_{j=1}^{n} q_{ij} y_j, \qquad i = 1, \ldots, n,$$

že souřadnicové vyjádření q v souřadnicích $y_1, \ldots, y_n$ je tvaru:

$$q(x) = \sum_{i=1}^{n} b_{ii} y_i^2,$$

tj. matice B_q kvadratické formy q je diagonální.

Definice 1.1.18. Nechť q je kvadratická forma na vektorovém prostoru V nad T. Říkáme, že q je *pozitivně (resp. negativně) definitní*, jestliže pro každý vektor $\mathbf{u} \neq \mathbf{o}$ platí $q(\mathbf{u}) > 0$ (resp. $q(\mathbf{u}) < 0$).

Poznámka 1.1.19. Je-li kvadratická forma q vyjádřena vzhledem k polární bázi (tj. její matice vzhledem k této bázi je diagonální), pak je forma pozitivně definitní, právě když $a_{ii} > 0 \; \forall i$ a negativně definitní, je-li $a_{ii} < 0 \; \forall i$.

Definice 1.1.20. *Prostorem se skalárním součinem* nazýváme dvojici (V, b), kde V je vektorový prostor a b je pozitivně definitní symetrická bilineární forma na V. Forma b se nazývá *skalární součin* a číslo $b(\mathbf{u}, \mathbf{v})$ se nazývá *skalární součin vektorů* $\mathbf{u}, \mathbf{v} \in V$. Jestliže $b(\mathbf{u}, \mathbf{v}) = 0$, pak říkáme, že vektory $\mathbf{u}, \mathbf{v}$ jsou navzájem *kolmé (ortogonální)* a zapisujeme $\mathbf{u} \perp \mathbf{v}$. Pokud je skalární součin b předem jednoznačně dán, budeme jej zjednodušeně označovat $(\mathbf{u}, \mathbf{v}) = b(\mathbf{u}, \mathbf{v})$.

Příklad 1.1.21. Ve vektorovém prostoru $\mathbb{R}^n$ je skalárním součinem např. bilineární forma zadaná vztahem:

$$(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} u_i v_i,$$

$$\text{pro } \mathbf{u} = (u_1, \ldots, u_n) \in \mathbb{R}^n, \mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n.$$

Takovýto skalární součin nazýváme *standartním skalárním součinem* na $\mathbb{R}^n$. Standartní skalární součin vektorů $\mathbf{u}$ a $\mathbf{v}$ označíme $\mathbf{u} \cdot \mathbf{v}$.

Definice 1.1.22. Buď (V_n, b) prostor se skalárním součinem. Pro každý vektor $\mathbf{u} \in V$ označíme $\|\mathbf{u}\|_b = \sqrt{q(\mathbf{u})}$. Číslo $\|\mathbf{u}\|_b$ nazýváme *normou vektoru* $\mathbf{u}$ (vzhledem ke skalárnímu součinu b). Pokud je skalární součin b předem jednoznačně dán, budeme normu zjednodušeně označovat $\|\mathbf{u}\| = \|\mathbf{u}\|_b$.

Definice 1.1.23. Báze $M = \{\mathbf{u_1}, \dots \mathbf{u_n}\}$ prostoru se skalárním součinem (V_n, b) se nazývá *ortogonální báze*, jestliže je polární bází bilineární formy b (tj. je-li $b(\mathbf{u_i}, \mathbf{u_j}) = 0$ pro všechna $i, j = 1, \dots, n, i \neq j$). Nechť q je kvadratická forma určená bilineární formou b. Ortogonální báze, pro kterou navíc platí $q(\mathbf{u_i}) = 1, i = 1, \dots, n$, se nazývá *ortonormální báze prostoru* V_n.

Poznámka 1.1.24. (1) Předchozí věta jinými slovy slovy říká, že matice skalárního součinu b vzhledem k ortogonální bázi je diagonální a vzhledem k ortonormální bázi dokonce jednotková.

(2) Nechť M je ortonormální báze prostoru se skalárním součinem (V, b), $\mathbf{u}, \mathbf{v} \in V$ vektory, jejichž souřadnice vzhledem k M jsou $\mathbf{u} = (u_1, \dots, v_n)$, $\mathbf{v} = (v_1, \dots, v_n)$, pak skalární součin vektorů $\mathbf{u}, \mathbf{v}$ je tvaru: $(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} u_i v_i$.

(3) Je-li $\mathbf{v} \in V_n, \mathbf{v} \neq \mathbf{o}$ libovolný, pak položíme-li $\mathbf{u} = \frac{\mathbf{v}}{\|\mathbf{v}\|_q} = \frac{\mathbf{v}}{\sqrt{q(\mathbf{v})}}$, dostaneme:

$$q(\mathbf{u}) = b\left(\frac{\mathbf{v}}{\sqrt{q(\mathbf{v})}}, \frac{\mathbf{v}}{\sqrt{q(\mathbf{v})}}\right) = \frac{1}{q(\mathbf{v})} \cdot q(\mathbf{v}) = 1.$$

Tímto „znormováním“ každého vektoru ortogonální báze $\{\mathbf{v_1}, \dots, \mathbf{v_n}\}$ prostoru se skalárním součinem V_n získáme vektory $\mathbf{u_i} = \frac{\mathbf{v_i}}{\|\mathbf{v_i}\|_q}, i = 1, \dots, n$, které tvoří ortonormální bázi prostoru V_n.

Věta 1.1.25. *V každém prostoru se skalárním součinem (V_n, b) existuje ortonormální báze.*

Důkaz. Podle věty 1.1.16 existuje ortogonální báze prostoru (V_n, b) a z ní už ortonormální bázi získáme postupem popsaným v poznámce. □

Věta 1.1.26. *Nechť $\{\mathbf{u_1}, \dots, \mathbf{u_k}\}$ je libovolná báze podprostoru W prostoru se skalárním součinem (V_n, b). Pak existuje ortogonální báze $\{\mathbf{u_1^*}, \dots, \mathbf{u_k^*}\}$ prostoru W taková, že $\mathbf{u_j^*} \in [\mathbf{u_1}, \dots, \mathbf{u_j}]$ pro každé $j = 1, \dots, k$.*

Důkaz. (Gram-Schmidtův ortogonalizační proces.) Budeme postupovat indukcí podle k:

(1) Položme $\mathbf{u_1^*} = \mathbf{u_1}$, pak zřejmě $\{\mathbf{u_1^*}\}$ je ortogonální bází prostoru W.

(2) Nechť $k > 1$ a předpokládejme, že existuje $\{\mathbf{u_1^*}, \dots, \mathbf{u_{k-1}^*}\}$ ortogonální báze podprostoru $[\mathbf{u_1}, \dots, \mathbf{u_{k-1}}]$ taková, že $\mathbf{u_j^*} \in [\mathbf{u_1}, \dots, \mathbf{u_j}]$ $\forall j : 1 \leq j < k$. Pak položíme:

$$\mathbf{u_k^*} = -\sum_{j=1}^{k-1} \mu_{k,j} \mathbf{u_j^*} + \mathbf{u_k}, \quad \mu_{k,j} \in \mathbb{R}, j = 1, \dots, k-1 \tag{1.2}$$

Zřejmě pro libovolnou volbu koeficientů $\mu_{k,j}$ platí $\mathbf{u_j^*} \in [\mathbf{u_1}, \dots, \mathbf{u_j}]$ $\forall j = 1, \dots, k$. Koeficienty $\mu_{k,j}$ zvolíme tak, aby vektory $\mathbf{u_1^*}, \dots, \mathbf{u_k^*}$ byly navzájem ortogonální. Vzhledem k indukčnímu předpokladu stačí $b(\mathbf{u_k^*}, \mathbf{u_j^*}) = 0$ pro $j = 1, \dots, k-1$. Z (1.2) pak dostáváme: $b(\mathbf{u_k^*}, \mathbf{u_j^*}) = 0 = -\mu_{k,j} q(\mathbf{u_j^*}) + b(\mathbf{u_k}, \mathbf{u_j^*})$. Platí: $\mathbf{u_j^*} \neq \mathbf{o}$ (protože $\{\mathbf{u_1}, \dots, \mathbf{u_k}\}$ je báze podprostoru W ve (V_n, b)), a tedy $\mu_{k,j} = \frac{b(\mathbf{u_k}, \mathbf{u_j^*})}{q(\mathbf{u_j^*})}$. Tím jsme získali všechny hledané koeficienty $\mu_{k,j}$:

$$\mu_{k,j} = \frac{b(\mathbf{u_k}, \mathbf{u_j^*})}{q(\mathbf{u_j^*})} \quad \forall j, k : 1 \leq j < k \tag{1.3}$$

a z rovnosti (1.2) pak získáme hledaný vektor:

$$\mathbf{u_k^*} = \mathbf{u_k} - \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{u_j^*}. \tag{1.4}$$

□

Lemma 1.1.27. *Nechť $M = \{\mathbf{u_1}, \dots, \mathbf{u_n}\}$ je libovolná báze prostoru se skalárním součinem V_n, $M^* = \{\mathbf{u_1^*}, \dots, \mathbf{u_n^*}\}$ ortogonální báze vzniklá z M Gram-Schmidtovým ortogonalizačním procesem, koeficienty $\mu_{k,j}$ označují koeficienty zavedené v (1.3) v předchozí větě. Pak platí: $(\mathbf{u_k}, \mathbf{u_k^*}) = (\mathbf{u_k^*}, \mathbf{u_k^*})$ $\forall k = 1, \dots, n$.*

Důkaz. Z (1.4) si můžeme vyjádřit $\mathbf{u_k} = \mathbf{u_k^*} + \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{u_j^*}$. Pak díky vlastnostem skalárního součinu (symetrické pozitivně definitní bilineární formy) dostáváme:

$$(\mathbf{u_k}, \mathbf{u_k^*}) = (\mathbf{u_k^*} + \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{u_j^*}, \mathbf{u_k^*}) = (\mathbf{u_k^*}, \mathbf{u_k^*}) + \sum_{j=1}^{k-1} \mu_{k,j} (\mathbf{u_j^*}, \mathbf{u_k^*}) = (\mathbf{u_k^*}, \mathbf{u_k^*}),$$

kde $(\mathbf{u_j^*}, \mathbf{u_k^*}) = 0$ pro $j \neq k$, protože $\mathbf{u_j^*} \perp \mathbf{u_k^*}$. □

1.2 Mřížky

Definice 1.2.1. Nechť $m, n \in \mathbb{N}$. $L \subset \mathbb{R}^n$ nazveme *mřížka*, pokud existuje množina lineárně nezávislých vektorů $M = \{\mathbf{b_1}, \dots \mathbf{b_m}\}, \mathbf{b_i} \in \mathbb{R}^n$ pro $i = 1, \dots m$ tak, že:

$$L = \left\{ \sum_{i=1}^{m} r_i \mathbf{b_i} \;\middle|\; r_i \in \mathbb{Z}, i = 1, \dots, m \right\}.$$

Množinu $M = \{\mathbf{b_1}, \dots \mathbf{b_m}\}$ nazýváme *bází mřížky L*, m nazýváme *dimenzí mřížky L*.

Poznámka 1.2.2. (1) Tato definice mřížky už je poněkud zjednodušená – abychom mohli definovat LLL-redukovanou bázi mřížky, budeme potřebovat ještě pozitivně definitní kvadratickou formu q a příslušnou symetrickou bilineární formu b na $\mathbb{R}^n$ a správně bychom tedy měli jako mřížku označovat uspořádanou dvojici (L, q). Vzhledem k tomu, že kvadratická forma (a tím i příslušná symetrická bilineární forma) je většinou předem dána, budeme dále mřížku označovat pouze L.

(2) Pokud není v textu výslovně uvedeno jinak, mřížkou dimenze n máme na mysli mřížku dimenze n v $\mathbb{R}^n$.

(3) V [7, Definition 2.5.2] lze najít obecnější definici mřížky, která používá pojmy volný $\mathbb{Z}$-modul (anglicky free $\mathbb{Z}$-module) a tenzorový součin. Vysvětlení těchto pojmů však přesahuje rozsah této práce, proto jsme se v definici omezili pouze na prostory se skalárním součinem nad $\mathbb{R}^n$. Pro bližší studium výše zmíněných pojmů a pochopení obecnější definice však lze doporučit např. [12, Chapter 10] a [29, Chapter III, Chapter XVI] nebo pro příznivce slovensky psané literatury např. [28, kapitoly VI a IX].

Poznámka 1.2.3. Vzhledem k tomu, že v mřížce uvažujeme pouze celočíselné souřadnice vzhledem k bázovým vektorům, nemusí (ale mohou) být dvě mřížky určené dvěma různými bázemi v $\mathbb{R}^n$ stejné. Mřížky nad dvěma bázemi budeme považovat za stejné, pokud jsou tvořeny stejnými množinami vektorů. Zřejmě například platí: pokud báze $M = \{\mathbf{u_1}, \ldots, \mathbf{u_n}\}$ generuje mřížku L, pak báze $M' = \{\mathbf{v_1}, \ldots, \mathbf{v_n}\}$, kde $\mathbf{v_i} = -\mathbf{u_i}$ pro $i = 1, \ldots, n$ generuje mřížku L', která je ekvivalentní s mřížkou L. Dále podáme přesnou definici ekvivalence mřížek a budeme zjišťovat, které mřížky v $\mathbb{R}^n$ jsou navzájem ekvivalentní.

Definice 1.2.4. Bijektivní zobrazení $f\colon (L, q) \to (L', q')$ nazveme *izomorfismus mřížek L a L'*, pokud pro každý vektor $\mathbf{u} \in L$ platí $f(q(\mathbf{u})) = q'(f(\mathbf{u}))$.

Definice 1.2.5. Řekneme, že mřížky (L, q) a (L', q') jsou *ekvivalentní*, pokud existuje izomorfismus $f\colon (L, q) \to (L', q')$.

Poznámka 1.2.6. Nechť $M = \{\mathbf{u_1}, \ldots, \mathbf{u_n}\}$ a $M' = \{\mathbf{v_1}, \ldots, \mathbf{v_n}\}$ jsou dvě báze $\mathbb{R}^n$, $P = (p_{ij})$ matice přechodu od M' k M, L mřížka generovaná bází M, L' mřížka generovaná bází M', $\mathbf{u} \in L$ libovolný vektor, $X = (x_1, \ldots, x_n)$,resp. $Y = (y_1, \ldots, y_n)$ sloupcové vektory souřadnic $\mathbf{u}$ vzhledem k bázi M, resp. M'. Pak vhodným zobrazením zřejmě bude f určené předpisem: $f(\mathbf{u}) = \mathbf{u}$, tedy souřadnicově: $f(X) = PX$. Aby se jednalo o zobrazení L do L', musí platit $p_{ii} \in \mathbb{Z}$, aby $f(\mathbf{u}) \in L'$. Tedy $\det(P) \in \mathbb{Z}$ a zároveň $P \in GL_n(\mathbb{Z})$ (vzhledem k tomu, že P je matice přechodu). Vzorem Y je prvek $X = P^{-1}Y$

a aby bylo f surjektivní, musí platit $X \in \mathbb{Z}^n$, tedy $\det(P^{-1}) \in \mathbb{Z}$. Zároveň ale platí: $\det(P)\det(P^{-1}) = \det(E_n) = 1$. Pro matice P a P^{-1}, které splňují tuto rovnost a zároveň $\det(P) \in \mathbb{Z} \wedge \det(P^{-1}) \in \mathbb{Z}$, musí platit: $\det(P) = \det(P^{-1}) = \pm 1$. V tomto případě je f surjektivní zobrazení L do L' a vzhledem k tomu, že P je regulární, je f i prosté.
Dvě mřížky L a L' prostoru $\mathbb{R}^n$ jsou tedy ekvivalentní, právě tehdy když pro matici P přechodu od báze mřížky L k bázi mřížky L' platí: $P \in GL_n(\mathbb{Z}) \wedge \det(P) = \pm 1$.

Poznámka 1.2.7. Mřížku můžeme jednoznačně zadat různými způsoby – podívejme se alespoň na dva, které jsou dobře použitelné v praxi:

(1) Můžeme vybrat libovolnou bázi $M = \{\mathbf{b_1}, \ldots, \mathbf{b_m}\}$ mřížky L. Potom vektor $\mathbf{x} \in L$ můžeme vyjádřit jeho (celočíselnými) souřadnicemi vzhledem k bázi M. Skalární součin je pak vzhledem k bázi M vyjádřen pozitivně definitní symetrickou maticí $Q = (q_{ij}), q_{ij} = b(\mathbf{b_i}, \mathbf{b_j})$. Matice Q je čtercová matice řádu m a nazývá se *Gramova matice*. Poznamenejme, že Gramova matice je zároveň maticí symetrické pozitivně definitní bilineární formy b vzhledem k bázi M. Pak $b(\mathbf{x}, \mathbf{y}) = Y^T QX$ a speciálně, pokud q je kvadratická forma určená bilineární formou b, $q(x) = X^T QX$, kde X, resp. Y jsou sloupcové vektory určující souřadnice $\mathbf{x}$, resp. $\mathbf{y}$ v bázi M.
Pokud bychom místo báze M vybrali bázi M' a P by byla matice přechodu od M k M', pak podle 1.1.9 dostáváme $q(\mathbf{x}) = (PX)^T Q(PX) = X^T Q'X$, kde $Q' = P^T QP$. Tedy třídy ekvivalence mřížek odpovídají třídám ekvivalence pozitivně definitních symetrických matic, kde dvě matice jsou v ekvivalenci $Q \sim Q'$ právě tehdy když existuje matice $P \in GL_m(\mathbb{Z})$ tak, že $Q' = P^T QP$. Determinant P je podle předchozí poznámky roven ± 1, takže determinant Q je nezávislý na volbě báze mřížky. Q je navíc pozitivně definitní, takže $\det(Q) > 0$ a můžeme položit $d(L) = +\sqrt{\det(Q)}$. Číslo $d(L)$ nazýváme *determinant mřížky L*.

(2) Nechť $M = \{\mathbf{b_1}, \ldots, \mathbf{b_m}\}$ je libovolná báze L ve vektorovém prostoru $\mathbb{R}^n$. Můžeme vybrat ortonormální bázi $E = \{\mathbf{e_1}, \ldots, \mathbf{e_m}\}$ vektorového prostoru $[\mathbf{b_1}, \ldots, \mathbf{b_m}]$. Nechť B je matice přechodu od E k M typu $n \times m$. Pak podle 1.1.9 $Q = B^T E_n B = B^T B$ a tedy pro mřížky dimenze n v $\mathbb{R}^n$ platí $d(L) = |\det(B)|$.
Pokud bychom vybrali jinou ortonormální bázi E', pak matice přechodu B' od E' k M by měla tvar $B' = KB$, kde K je matice přechodu od E' k E. K je matice přechodu od ortonormální báze k ortonormální bázi, proto K je ortogonální matice – tedy matice, pro kterou platí $K^T K = KK^T = E_n$.

Těmito úvahami jsme dokázali následující tvrzení:

Věta 1.2.8. *(i) Nechť Q je maticí pozitivně definitní kvadratické formy. Pak Q je Gramovou maticí některé báze mřížky, tzn. existuje matice $B \in GL_n(\mathbb{R})$ tak, že $Q = B^T B$.*

(ii) Gramova matice vzhledem k bázi mřížky $M = \{\mathbf{b_1}, \dots, \mathbf{b_n}\}$ určuje tuto bázi jednoznačně až na ortogonální transformaci, přesněji: Pokud M a M' mají stejnou Gramovu matici, pak M' vznikla z M ortogonální transformací. Maticově: $M' = KM$, kde K je ortogonální matice.

Věta 1.2.9. *Podle předchozího tedy jsou izomorfní následující množiny:*

$$\boxed{\{\textit{Třídy ekvivalence mřížek dimenze } n\}}$$

$$\cong$$

$$\boxed{\begin{array}{c}\{\textit{Pozitivně definitní symetrické matice }\}/_{\sim}\ , \\ \textit{kde } Q \sim Q' \iff Q' = P^T Q P \textit{ pro nějakou matici } P \in GL_n(\mathbb{Z})\end{array}}$$

$$\cong$$

$$\boxed{\begin{array}{c}\{GL_n(\mathbb{R})\}/_{\sim}\ , \\ \textit{kde } B \sim B' \iff B' = KBP \textit{ pro nějakou matici } P \in GL_n(\mathbb{Z}) \textit{ a} \\ \textit{nějakou ortogonální matici } K\end{array}}$$

Poznámka 1.2.10. Dále budeme používat označení zavedené ve větě 1.1.26 a v jejím důkazu (Gram-Schmidtův ortogonalizační proces) a odvodíme některé její důsledky. Především z této věty plyne následující vztah pro determinant mřížky L, jejíž báze je $M = \{\mathbf{b_1}, \dots, \mathbf{b_n}\}$:

$$d(L)^2 = \prod_{i=1}^{n} \|\mathbf{b_i^*}\|^2 .$$

Označení. Dále budeme symbolem B_i označovat $\|\mathbf{b_i^*}\|^2$.

Důsledek 1.2.11 (Hadamardova nerovnost). *Nechť (L, q) je mřížka, $d(L)$ její determinant, $M = \{\mathbf{b_1}, \dots, \mathbf{b_n}\}$ báze L. Pak platí:*

(i)

$$d(L) \leq \prod_{i=1}^{n} \|\mathbf{b_i}\|$$

(ii) Pro matici B typu $n \times n$ platí:

$$|det(B)| \leq \prod_{i=1}^{n} \sqrt{\sum_{j=1}^{n} |b_{ij}|^2}$$

Důkaz. (i) Z ortogonality vektorů $\mathbf{b_i^*}$ dostáváme:

$$q(\mathbf{b_i}) = \|\mathbf{b_i}\|^2 = B_i + \sum_{j=1}^{i-1} \mu_{ij}^2 B_j \geq B_i.$$

$$\text{Pak:} \quad d(L)^2 = \prod_{i=1}^{n} B_i \leq \prod_{i=1}^{n} \|\mathbf{b_i}\|^2 .$$

(ii) Jedná se o vyjádření (i) pro případ, že q je určena standartním skalárním součinem.

□

Důsledek 1.2.12. *Nechť $B \in GL_n(\mathbb{R})$. Pak existují jednoznačně určené matice K, A a N tak, že:*

(i) $B = KAN$.

(ii) K je ortogonální matice (tzn. $K^T = K^{-1}$).

(iii) A je diagonální matice s kladnými prvky.

(iv) N je horní trojúhelníková matice a platí: $n_{ii} = 1 \; \forall i = 1, \ldots, n$

Důkaz. Existence: Nechť $\mathbf{b_1}, \ldots, \mathbf{b_n}$ jsou sloupcové vektory matice B. Na tyto vektory uplatníme Gram-Schmidtův ortogonalizační proces, při kterém získáme vektory $\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}$, které budou tvořit sloupce matice B^*. Pak $B^* = BN$, kde $N = (n_{ij})$ je horní trojúhelníková matice, která má na diagonále 1 (a nad diagonálou na pozici n_{ij} je prvek $-\mu_{i,j}$). Protože $\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}$ je ortogonální bází $\mathbb{R}^n$, Gramova matice těchto vektorů je diagonální s kladnými koeficienty, označme ji D. Nechť A je matice, která z matice D vznikne takto: $a_{ij} = +\sqrt{d_{ij}}$. Pak Gramovou maticí sloupcových vektorů A bude opět matice D, z čehož podle 1.2.7 a 1.2.8 plyne: ${B^*}^T B^* = D \iff B^* = KA$ pro nějakou ortogonální matici K. Celkem tedy: $BN = KA \iff B = KAN^{-1}$, přičemž N^{-1} je horní trojúhelníková matice (neboť N byla horní trojúhelníková).
Jednoznačnost: $B^* = BN = KA$, což znamená, že $\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}$ tvoří ortogonální bázi, kterou lze získat z $\mathbf{b_1}, \ldots, \mathbf{b_n}$ Gram-Schmidtovým ortogonalizačním procesem, který je jednoznačný. □

Poznámka 1.2.13. (1) Předpoklad, že prvky na diagonále matice A jsou kladné, není podstatný a je ve větě pouze kvůli jednoznačnosti vyjádření.

(2) Stejného výsledku lze dosáhnout, pokud matice N bude dolní trojúhelníková matice. Pak bude platit $B = NAK$.

(3) $S = AN$ je horní trojúhelníková matice s kladnými prvky na diagonále. Každá taková matice může být jednoznačně zapsána jako součin právě těch matic A a N, uvedených v 1.2.12.

Věta 1.2.14. *Nechť Q je matice pozitivně definitní kvadratické formy. Pak existuje právě jedna horní trojúhelníková matice S, která má na diagonále kladné prvky tak, že $Q = S^T S$ (nebo ekvivalentně: $Q = N^T D N$, kde N je horní trojúhelníková matice s prvky na diagonále rovnými* 1 *a D je diagonální matice s kladnými prvky na diagonále).*

Důkaz. Existence. Podle 1.2.8 i existuje $B \in GL_n(\mathbb{R})$ tak, že $Q = B^T B$. Z 1.2.12 a 1.2.13 víme, že existují matice K a S tak, že $B = KS$, kde K je ortogonální a S je horní trojúhelníková matice s kladnými prvky na diagonále. Celkem tedy $Q = (KS)^T KS = S^T K^T KS = S^T S$.
Jednoznačnost. Sporem: Nechť existuje S_1 různá od S tak, že

$$Q = S^T S = S_1^T S_1 \tag{1.5}$$

a S i S_1 jsou horní trojúhelníkové matice s kladnými prvky na diagonále. Pak platí:

$$\left({S_1}^T\right)^{-1} S^T = S_1 S^{-1}, \tag{1.6}$$

přičemž S^{-1} a $\left({S_1}^T\right)^{-1}$ existují, protože matice Q je pozitivně definitní. Na levé straně rovnosti (1.6) je dolní trojúhelníková matice, na pravé straně horní trojúhelníková matice. Tedy obě strany této rovnosti musí být rovny diagonální matici D. Z rovnosti (1.5) (dalším přenásobením inverzními prvky) dostáváme $D^2 = E_n$. Protože prvky na diagonále matice $D = S_1 S^{-1}$ jsou kladné, musí platit $D = E_n$, z čehož plyne $S = S_1$, což je spor. □

2. LLL algoritmus

2.1 LLL-redukovaná báze

Báze mřížky není určena jednoznačně, dokonce každá mřížka v prostoru $\mathbb{R}^n$, $n \geq 2$ má nekonečně mnoho bází – některé z nich jsou však významější než jiné. Ty, které obsahují „nejkratší“ vektory (tedy vektory s co nejmenší normou vzhledem ke skalárnímu součinu b), se nazývají *redukované*. Gramovy matice všech bází jedné mřížky mají stejný determinant, a proto „redukovanost“ znamená, že příslušná báze se blíží k tomu být ortogonální.

Definice 2.1.1. Nechť L je mřížka dimenze n. Pro $i \in \mathbb{N}, 1 \leq i \leq n$, *i-té postupné minimum* L značíme $\lambda_i(L)$ a definujeme ho takto:

$$\lambda_i(L) = \min_{\substack{\mathbf{v_1},\ldots\mathbf{v_n} \in L \\ \text{lineárně nezávislé}}} \{\max_{1 \leq j \leq i} \|\mathbf{v_j}\|\}.$$

Poznámka 2.1.2. (1) Ideální redukovaná báze by tedy byla taková báze $R = \{\mathbf{r_1}, \ldots, \mathbf{r_n}\}$, pro kterou by platilo: $\|\mathbf{r_i}\| = \lambda_i(L)$. Pokud bychom uměli takovou bázi najít (v rozumném čase), dokázali bychom vyřešit mnoho matematických a informatických problémů, které lze převést na hledání redukované báze mřížky či nejkratšího vektoru v mřížce. O těchto problémech se podrobněji zmiňujeme ve 3. kapitole. Dodnes však takovouto bázi rychle najít neumíme, a proto za redukovanou bázi považujeme takovou bázi, která se této ideální alespoň blíží.

(2) Z právě uvedené definice i-tého postupného minima je zřejmé, že vektory redukované báze jsou uspořádané vzestupně podle normy. Záleží nám tedy i na pořadí vektorů v bázi – první vektor je nejmenší, který se nám podařilo najít.

(3) Představa redukované báze je poměrně stará, ale dlouho nebyl znám žádný algoritmus, který by našel redukovanou bázi v rozumném čase pro dimenzi $n > 2$ (pro dimenzi 2 popsal algoritmus s kvadratickou časovou složitostí na počátku 19. století Gauss, mnohem později ještě objevil kubický algoritmus pro dimenzi 3 Vallée). Se skutečným převratem přišli v roce 1982 A. K. Lenstra, H. W. Lenstra a L. Lovász, kteří zavedli novou definici redukované báze a zároveň předložili algoritmus, který pro libovolnou dimenzi najde takto definovanou redukovanou bázi v polynomiálním čase.

Definice 2.1.3. Nechť $M = \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ je báze mřížky L a $\{\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}\} = M^*$ je ortogonální báze získaná z M Gram-Schmidtovým ortogonalizačním procesem 1.1.26, $\mu_{i,j} = \frac{(\mathbf{u_i}, \mathbf{u_j^*})}{(\mathbf{u_j^*}, \mathbf{u_j^*})}$ pro $\forall i, j : 1 \leq j < i$. Pak báze M se nazývá *LLL-redukovaná* právě tehdy, když splňuje následující podmínky:

(i)
$$|\mu_{i,j}| \leq \frac{1}{2} \qquad \text{pro } 1 \leq j < i \leq n$$

(ii) tzv. *Lovászova podmínka*:
$$\left\|\mathbf{b_i^*} + \mu_{i,i-1}\mathbf{b_{i-1}^*}\right\|^2 \geq \frac{3}{4}\left\|\mathbf{b_{i-1}^*}\right\|^2 \qquad \text{pro } 1 < i \leq n$$

Poznámka 2.1.4. (1) Podmínka (*ii*) z předchozí definice je zřejmě ekvivalentní následující podmínce:
$$\|\mathbf{b_i}^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right)\left\|\mathbf{b_{i-1}^*}\right\|^2$$

(2) Místo konstanty $\frac{3}{4}$ lze v (ii) zvolit libovolné pevné $c \in \mathbb{R} : \frac{1}{4} < c < 1$. Pak se ale také změní odhady a důkaz následující věty – 2 bychom museli nahradit $\alpha = \frac{1}{c-\frac{1}{4}}$ a použít slabší nerovnost $\mu_{k,l}^2 \leq \frac{\alpha-1}{\alpha}$.

Věta 2.1.5. *Nechť $M = \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ je LLL-redukovaná báze mřížky L. Pak platí:*

(i)
$$\|\mathbf{b_j}\| \leq 2^{\frac{i-1}{2}}\left\|\mathbf{b_i^*}\right\|, \textit{ pokud } 1 \leq j \leq i \leq n,$$

(ii)
$$d(L) \leq \prod_{i=1}^{n}\|\mathbf{b_i}\| \leq 2^{\frac{n(n-1)}{4}} d(L),$$

(iii)
$$\|\mathbf{b_1}\| \leq 2^{\frac{n-1}{4}}\sqrt[n]{d(L)},$$

(iv) Pro každý vektor $\mathbf{x} \in L, \mathbf{x} \neq \mathbf{o}$ platí:
$$\|\mathbf{b_1}\| \leq 2^{\frac{n-1}{2}}\|\mathbf{x}\|,$$

(v) Obecněji, pro $\mathbf{x_1}, \ldots \mathbf{x_t} \in L$ lineárně nezávislé platí:
$$\|\mathbf{b_j}\| \leq 2^{\frac{n-1}{2}}\max\{\|\mathbf{x_1}\|, \ldots, \|\mathbf{x_t}\|\} \quad \textit{pro } 1 \leq j \leq t.$$

Důkaz. (i) Je-li báze $M = \{\mathbf{b_1}, \dots \mathbf{b_n}\}$ LLL-redukovaná, pak z definice platí:

$$B_i \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) B_{i-1} \geq \frac{B_{i-1}}{2}, \quad \text{neboť} \quad |\mu_{i,i-1}| \leq \frac{1}{2}.$$

Z toho matematickou indukcí dostáváme $B_j \leq 2^{i-j} B_i$ pro $1 \leq j \leq i \leq n$. Dále z 1.1.26 a 2.1.3 (i) plyne:

$$\begin{aligned} \|\mathbf{b_i}\|^2 &= B_i + \sum_{j=1}^{i-1} \mu_{i,j}^2 B_j \leq B_i + \sum_{j=1}^{i-1} \tfrac{1}{4} 2^{i-j} B_i = \\ &= B_i(1 + \tfrac{1}{4}(2^i - 2)) \leq 2^{i-1} B_i, \end{aligned}$$

z čehož dostáváme:

$$\|\mathbf{b_j}\|^2 \leq 2^{j-1} B_j \leq 2^{i-1} B_i \text{ pro } 1 \leq j \leq i \leq n.$$

(ii) První nerovnost platí z 1.2.11. Podle (i) platí $\|\mathbf{b_i}\| \leq 2^{\frac{(i-1)}{2}} \|\mathbf{b_i^*}\|$ a pak už dostáváme

$$d(L) \leq \prod_{i=1}^{n} \|\mathbf{b_i}\| \leq 2^{\frac{n(n-1)}{4}} d(L),$$

(iii) Volbou $j = 1$ v (i), provedením součinu na obou stranách rovnosti pro $i = 1, \dots, n$ a dále z 1.2.10 dostáváme:

$$\|\mathbf{b_1}\|^{2n} \leq 2^{n(n-1)/2} \prod_{i=1}^{n} B_i = 2^{n(n-1)/2} d(L)^2,$$

odkud už plyne tvrzení.

(iv) Nechť $\mathbf{x} = \sum_{i=1}^{n} r_i \mathbf{b_i} = \sum_{i=1}^{n} s_j \mathbf{b_i^*}$, kde $r_i \in \mathbb{Z}, s_i \in \mathbb{R}$ pro $i = 1, \dots, n$. Nechť i je největší index takový, že $r_i \neq 0$, pak $r_i = s_i$ a platí $\|\mathbf{x}\|^2 \geq s_i^2 B_i \geq B_i$. Podle (i) platí: $\|\mathbf{b_1}\|^2 \leq 2^{i-1} B_i \leq 2^{n-1} B_i$ a odtud už $\|\mathbf{b_1}\|^2 \leq 2^{n-1} B_i \leq 2^{n-1} \|\mathbf{x}\|^2$.

(v) Nechť $\mathbf{x_j} = \sum_{i=1}^{n} r_{ij} b_i$, kde $r_{ij} \in \mathbb{Z}$. Pro pevné j nechť $i(j)$ je největší index i tak, že $r_{ij} \neq 0$. Pak podle důkazu (iv) platí:

$$\|\mathbf{x_j}\|^2 \geq B_{i(j)} \text{ pro } j = 1, \dots, t \tag{2.1}$$

Přečíslujeme vektory $\mathbf{x_j}$ tak, aby $i(1) \leq i(2) \leq \dots \leq i(t)$ a ukážeme, že $i(j) \geq j$ pro $j = 1, \dots, t$. Sporem: Pokud by platilo $i(j) < j$, pak platí

$\mathbf{x_s} \in [\mathbf{b_1}, \dots \mathbf{b_{j-1}}]$ pro $s = 1, \dots, j$, což je spor s lineární nezávislostí $\mathbf{x_1}, \dots \mathbf{x_t}$. Pak z (2.1) dostáváme:

$$\|\mathbf{b_j}\|^2 \leq 2^{i(j)-1} B_{i(j)} \leq 2^{n-1} \|\mathbf{x_j}\|^2 \text{ pro } j = 1, \dots, t,$$

z čehož plyne dokazované tvrzení.

□

Poznámka 2.1.6. (1) Nechť $M' = \lambda_1(L), \dots, \lambda_n(L)$ jsou i-tá postupná minima mřížky L a $M = \{\mathbf{b_1}, \dots, \mathbf{b_n}\}$ je LLL-redukovaná báze L. Pak podle 2.1.5 (i) a (iv) a platí:

$$2^{1-i}\lambda_i(L) \leq \|\mathbf{b_i}\|^2 \leq 2^{n-1}\lambda_i(L) \quad \text{pro } i = 1, \dots, n,$$

tedy $\|\mathbf{b_i}\|^2$ je poměrně dobrým přiblížením $\lambda_i(L)$.

(2) Z 2.1.5 (iii) zase vidíme, že vektor $\mathbf{b_1}$ LLL-redukované báze je jedním z nejkratších nenulových vektorů mřížky L. Ve skutečnosti je to často opravdu ten nejkratší nebo s ním lze pracovat místo nejkratšího vektoru mřížky.

2.2 Popis LLL algoritmu

Základní myšlenka algoritmu

Představme si, že vektory $\mathbf{b_1}, \dots, \mathbf{b_{k-1}}$ už jsou LLL-redukované (přesněji jsou LLL-redukovanou bází mřížky, kterou generují) – to je triviálně splněno pro $k = 2$.Vektor $\mathbf{b_k}$ chceme změnit tak, aby vektory $\mathbf{b_1}, \dots, \mathbf{b_k}$ byly LLL-redukované.

Nejdříve tedy změníme vektor $\mathbf{b_k}$ tak, aby byla splněna podmínka (i) definice LLL-redukované báze 2.1.3, tedy: $|\mu_{k,j}| \leq \frac{1}{2}$ pro všechna $j < k$. Splnění této podmínky dosáhneme přiřazením $\mathbf{b_k} := \mathbf{b_k} - \sum_{j=1}^{k} a_j \mathbf{b_j}$ pro vhodné $a_j \in \mathbb{Z}$ takto: a_j budeme hledat postupně pro $l = k, \dots, 1$. Pro každé takové l budeme předpokládat platnost podmínky pro $l < j < k$ a budeme chtít dosáhnout platnosti podmínky pro $l - 1 < j < k$. Předpokládejme, že $|\mu_{k,j}| \leq \frac{1}{2}$ pro $l < j < k$. Pak pokud $q = \lfloor \mu_{k,l} \rceil$ je nejbližší celé číslo k $\mu_{k,l}$, provedeme přiřazení $\mathbf{b_k} := \mathbf{b_k} - q\mathbf{b_l}$. Tímto přiřazením se koeficienty $\mu_{k,j}$ změní takto:

$$\mu_{k,j} = \frac{(\mathbf{b_k}, \mathbf{b_j^*})}{(\mathbf{b_j^*}, \mathbf{b_j^*})} - \frac{q(\mathbf{b_l}, \mathbf{b_j^*})}{(\mathbf{b_j^*}, \mathbf{b_j^*})}$$

Tedy pro $j > l$ se koeficienty $\mu_{k,j}$ nezmění, neboť $\mathbf{b_j^*} \perp \mathbf{b_l}$, tzn. že druhý člen výrazu je roven 0. $\mu_{k,l}$ se změní na $\mu_{k,l} - q$, protože podle 1.1.27 je druhý

člen výrazu roven q. Touto změnou $\mathbf{b_k}$ jsme však především dosáhli toho, že $|\mu_{k,l} - q| \leq \frac{1}{2}$, a proto nové koeficienty $\mu_{k,j}$ splňují podmínku (i) definice 2.1.3: $|\mu_{k,j}| \leq \frac{1}{2}$ pro $l - 1 < j < k$.

Teď, když vektor $\mathbf{b_k}$ splňuje podmínku (i) definice 2.1.3, potřebujeme, aby splnil ještě podmínku (ii): $B_k \geq (\frac{3}{4} - \mu_{k,k-1}^2)B_{k-1}$. Otestujeme tedy, jestli vektor $\mathbf{b_k}$ tuto podmínku splňuje. Pokud ano, zvýšíme k o 1 a pokračujeme s dalším vektorem, pokud ještě nějaký zbývá. Pokud ne, vyměníme vektory $\mathbf{b_k}$ a $\mathbf{b_{k-1}}$ – pak musíme snížit k o 1 a víme, že vektory $\mathbf{b_1}, \ldots, \mathbf{b_{k-2}}$ jsou LLL-redukované a pokračujeme znovu od začátku.

Z výše popsaného postupu není hned vidět, jestli postupné zvyšování a snižování k vždycky skončí, ale později ukážeme, že tomu tak opravdu je.

Příklad 2.2.1. Mějme v $\mathbb{R}^3$ mřížku (L, q) určenou bází $M = \{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\}$, kde $\mathbf{b_1} = (-1, 0, 1)$, $\mathbf{b_2} = (1, 2, 3)$, $\mathbf{b_3} = (0, 1, 1)$, q je kvadratická forma určená standartním skalárním součinem. Snadno se přesvědčíme, že báze M není LLL-redukovaná, protože např. $\mu_{2,1} = 1$, tedy není splněna podmínka (i) definice LLL-redukované báze 2.1.3. (Bázi M zde zapisujeme jako množinu, v LLL algoritmu s ní však zacházíme jako s uspořádanou trojicí. Pokud bychom zadali vektory báze M v jiném pořadí, získali bychom jinou LLL-redukovanou bázi.)

Z této báze získáme LLL-redukovanou bázi postupem uvedeným v základní myšlence takto:
$k \leftarrow 2, l \leftarrow 1$, Gram-Schmidtovým ortogonalizačním procesem spočítáme všechny $\mu_{i,j}, \mathbf{b_i^*}$ pro $i = 1, 2, 3, j = 1, \ldots, i - 1$;

$[k = 2]$	$[l = 1]$	$\mu_{2,1} = 1, q = 1, \mathbf{b_2} \leftarrow \mathbf{b_2} - \mathbf{b_1} = (2, 2, 2)$. Tím se změní $\mu_{2,1}$, ale nezmění se $\mathbf{b_i^*}$ pro $i = 1, 2, 3$. $\mu_{2,1} \leftarrow \mu_{2,1} - q$. Ověříme $B_2 \geq \left(\frac{3}{4} - \mu_{2,1}^2\right) B_1$, což platí. $k \leftarrow k + 1 = 3, l \leftarrow k - 1 = 2$
$[k = 3]$	$[l = 2]$	$q \leftarrow \lfloor \mu_{3,2} + \frac{1}{2} \rfloor = 0$, tedy vektory $\mathbf{b_i}$ se nezmění a $l \leftarrow l - 1 = 1$
	$[l = 1]$	$q \leftarrow \lfloor \mu_{3,1} + \frac{1}{2} \rfloor = 1$, tedy $\mathbf{b_3} \leftarrow \mathbf{b_3} - \mathbf{b_1} = (1, 1, 0)$, $\mu_{3,1} \leftarrow \mu_{3,1} - q$. $B_3 < \left(\frac{3}{4} - \mu_{3,2}^2\right) B_2$ (Lovászova podmínka není splněna) $\mathbf{b_3} \leftrightarrow \mathbf{b_2}$; Gram-Schmidtovým ortogonalizačním procesem přepočítáme všechny $\mu_{i,j}, \mathbf{b_i^*}$ pro $i = 1, \ldots, k, j = 1, \ldots, i-1$; $k \leftarrow k-1 = 2, l \leftarrow k-1 = 1$
$[k = 2]$	$[l = 1]$	$q \leftarrow \lfloor \mu_{2,1} + \frac{1}{2} \rfloor = 0$, tedy $\mathbf{b_2}$ se nezmění, $B_2 \geq \left(\frac{3}{4} - \mu_{2,1}^2\right) B_1$, tedy $k \leftarrow k + 1 = 3$, $l \leftarrow k - 1 = 2$
$[k = 3]$	$[l = 2]$	$q \leftarrow \lfloor \mu_{3,2} + \frac{1}{2} \rfloor = 3$, $\mathbf{b_3} \leftarrow \mathbf{b_3} - 3\mathbf{b_2} = (-1, -1, 2)$, $\mu_{3,2} \leftarrow \mu_{3,2} - q$, $l \leftarrow l - 1 = 1$.
	$[l = 1]$	$q \leftarrow \lfloor \mu_{3,1} + \frac{1}{2} \rfloor = 2$, $\mathbf{b_3} \leftarrow \mathbf{b_3} - 2\mathbf{b_1} = (1, -1, 0)$, $\mu_{3,1} \leftarrow \mu_{3,1} - q$. $B_3 \geq \left(\frac{3}{4} - \mu_{3,2}^2\right) B_2$, takže jsme hotovi.

Výsledná LLL-redukovaná báze tedy je: $\{(-1,0,1),(1,1,0),(1,-1,0)\}$. Jako výstup LLL algoritmu, který uvedeme na str. 25, bychom dostali jinou bázi $\{(-1,0,1),(0,1,1),(0,-1,1)\}$. Je to způsobeno tím, že tento algoritmus používá některá zefektivnění, která nemají vliv LLL-redukovanost báze, konkrétní výsledná báze je však odlišná.

Jak vylepšit základní myšlenku?

Gram-Schmidtovy koeficienty $\mu_{i,j}$ můžeme aktualizovat pouze, až s nimi budeme potřebovat dále počítat. Ve výše popsaném algoritmu totiž vždy musí být změněny koeficienty $\mu_{i,k}$ a $\mu_{i,k-1}$ pro $i > k$, které se pravděpodobně ještě několikrát změní před tím, než budou použity. Proto budeme koeficienty $\mu_{i,j}$ dopočítávat, až když je budeme potřebovat, a v proměnné k_{max} si budeme udržovat maximální aktuální hodnotu.

Dalším zlepšením základní myšlenky může být zkontrolovat v prvním kroku pouze koeficient $\mu_{k,k-1}$ a ne všechny $\mu_{k,m}$ pro $m < k$, protože koeficient $\mu_{k,k-1}$ je jediný, který musí být v absolutní hodnotě menší než $\frac{1}{2}$ před ověřením Lovászovy podmínky (a pokud tato podmínka nebude splněna, pak jsme všechny $\mu_{k,m}$ pro $m < k-1$ měnili zbytečně).

Ověřování Lovászovy podmínky

Všechna tato zlepšení použijeme a v této situaci si ještě podrobněji rozeberme ověřování Lovászovy podmínky. Předpokládejme tedy, že:

$$|\mu_{k,k-1}| \leq \frac{1}{2}, \text{ pokud } k > 1. \tag{2.2}$$

(Toho dosáhneme zjednodušením postupu popsaného v základní myšlence algoritmu.) Při ověřování Lovászovy podmínky pak mohou nastat dva případy:

(1) $k \geq 2$ a zároveň podmínka není splněna, tedy

$$B_k < \left(\frac{3}{4} - \mu_{k,k-1}^2\right) B_{k-1} \tag{2.3}$$

Vyměníme vektory $\mathbf{b_{k-1}}$ a $\mathbf{b_k}$, ostatní vektory $\mathbf{b_i}$ ponecháme beze změny. Musíme tedy změnit i vektory $\mathbf{b^*_{k-1}}$ a $\mathbf{b^*_k}$ a čísla $\mu_{k,k-1}$, $\mu_{k-1,j}$, $\mu_{k,j}$, $\mu_{i,k-1}$, $\mu_{i,k}$ pro $j < k-1$ a pro $i > k$. Nejdůležitější z těchto změn je: $\mathbf{b^*_{k-1}} \leftarrow \mathbf{b^*_k} + \mu_{k,k-1}\mathbf{b^*_{k-1}}$, tedy nová hodnota B_{k-1} je rovna nejvýš $\frac{3}{4}$ staré hodnoty tohoto výrazu. Po provedení těchto změn snížíme k o 1 a pokračujeme dalším průchodem algoritmem.

(2)

$$k = 1 \text{ nebo } \left(k \geq 2 \wedge B_k \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right) B_{k-1}\right) \tag{2.4}$$

V tomto případě už má smysl, abychom zajistili splnění následující podmínky:

$$|\mu_{k,j}| \leq \frac{1}{2} \quad \text{pro } 1 \leq j \leq k-1. \tag{2.5}$$

(Pro $j = k-1$ už tato podmínka platí podle (2.2).) Pro ostatní indexy opakujeme postup popsaný výše pro splnění podmínky (i) definice LLL-redukované báze tak dlouho, dokud neplatí (2.5). Pak k zvýšíme o 1 a pokračujeme dalším průchodem algoritmem.

Poznamenejme, že v případě, kdy $k = 1$ neudělá celý algoritmus nic jiného, než že k zvýší o 1.

Na základě těchto úvah můžeme uvést kompletní algoritmus. V následujícím algoritmu $c \leftrightarrow d$ označuje „prohození“ hodnot v proměnných c a d.

Algoritmus (LLL algoritmus). Mějme danou mřížku (L, q) a její libovolnou bázi $M = \{\mathbf{b_i}, \ldots, \mathbf{b_m}\}$ (zadanou souřadnicemi vzhledem ke kanonické bázi $\mathbb{R}^n$ pro $n \geq m$). Výstupem algoritmu budou nové vektory $\mathbf{b_1}, \ldots, \mathbf{b_m}$, které budou tvořit LLL-redukovanou bázi L a matice H, která bude udávat souřadnice této LLL-redukované báze vzhledem k původně zadané bázi, sloupcové vektory matice H označíme $\mathbf{h_i}$.

1. [Inicializace] Polož $k \leftarrow 2, k_{\max} \leftarrow 1, \mathbf{b_1^*} \leftarrow \mathbf{b_1}, B_1 \leftarrow (\mathbf{b_1}, \mathbf{b_1}), H \leftarrow E_m$.
2. [Gram-Schmidt] Je-li $k \leq k_{\max}$, jdi na 3. Jinak polož $k_{\max} \leftarrow k, \mathbf{b_k^*} \leftarrow \mathbf{b_k}$. Pro $j = 1, \ldots, k-1$ polož $\mu_{k,j} \leftarrow \frac{(\mathbf{b_k}, \mathbf{b_j^*})}{B_j}, \mathbf{b_k^*} \leftarrow \mathbf{b_k^*} - \mu_{k,j}\mathbf{b_j^*}$. Poté polož $B_k \leftarrow (\mathbf{b_k^*}, \mathbf{b_k^*})$. Je-li $B_k = 0$, vypiš chybové hlášení, že vektory $\mathbf{b_i}$ netvoří bázi mřížky a ukonči algoritmus.
3. [Test Lovászovy podmínky] Proveď RED$(k, k-1)$. Je-li splněno $B_k < \left(\frac{3}{4} - \mu_{k,k-1}^2\right) B_{k-1}$, proveď SWAP$(k)$, polož $k \leftarrow \max\{2, k-1\}$ a jdi na 3. Jinak pro $l = k-2, k-3, \ldots, 1$ proveď RED(k, l), poté polož $k \leftarrow k+1$.
4. [Hotovo?] Je-li $k \leq m$, jdi na 2. Jinak vypiš LLL-redukovanou bázi $\mathbf{b_i}$ a transformační matici $H \in GL_m(\mathbb{Z})$ a skonči.

Funkce RED(k, l). Je-li $|\mu_{k,l}| \leq \frac{1}{2}$ skonči. Jinak polož $q \leftarrow \lfloor \mu_{k,l} \rceil = \lfloor \frac{1}{2} + \mu_{k,l} \rfloor, \mathbf{b_k} \leftarrow \mathbf{b_k} - q\mathbf{b_l}, H_k \leftarrow H_k - qH_l, \mu_{k,l} \leftarrow \mu_{k,l} - q$, pro všechna $i = 1, \ldots, l-1$ přiřaď $\mu_{k,i} \leftarrow \mu_{k,i} - q\mu_{l,i}$ a skonči.

Funkce SWAP(k). Proveď $\mathbf{b_k} \leftrightarrow \mathbf{b_{k-1}}, H_k \leftrightarrow H_{k-1}$. Je-li $k > 2$, pro každé $j = 1, \ldots, k-2$ proveď $\mu_{k,j} \leftrightarrow \mu_{k-1,j}$. Potom přiřaď $\mu \leftarrow \mu_{k,k-1}, B \leftarrow$

$B_k + \mu^2 B_{k-1}, \mu_{k,k-1} \leftarrow \frac{\mu B_k - 1}{B}, \mathbf{b} \leftarrow \mathbf{b}^*_{\mathbf{k-1}}, \mathbf{b}^*_{\mathbf{k-1}} \leftarrow \mathbf{b}^*_{\mathbf{k}} + \mu \mathbf{b}, \mathbf{b}^*_{\mathbf{k}} \leftarrow -\mu_{k,k-1}\mathbf{b}^*_{\mathbf{k}} + \frac{B_k}{B}\mathbf{b}, B_k \leftarrow \frac{B_{k-1}B_k}{B}, B_{k-1} \leftarrow B$. Pro $i = k+1, k+2, \ldots, k_{\max}$ přiřaď $t \leftarrow \mu_{i,k}, \mu_{i,k} \leftarrow \mu_{i,k-1} - \mu t, \mu_{i,k-1} \leftarrow t + \mu_{k,k-1}\mu_{i,k}$ a skonči.

Tvrzení 2.2.2. *Existuje konstanta $\gamma_n > 0$ tak, že platí: v každé mřížce (L, q) nad $\mathbb{R}^n$ $\exists \mathbf{x}, \mathbf{x} \neq \mathbf{o}$ tak, že $q(x) \leq \gamma_n^n d(L)^{\frac{2}{n}}$.*

Poznámka 2.2.3. (1) Důkaz tohoto tvrzení překračuje rámec této práce, lze ho najít v [23].

(2) Nejlepší možné konstanty γ_n se nazývají Hermitovy konstanty a jejich hodnoty jsou známé pouze pro $n \leq 8$:

$$\gamma_1 = 1, \gamma_2^2 = \frac{4}{3}, \gamma_3^3 = 2, \gamma_4^4 = 4, \gamma_5^5 = 8, \gamma_6^6 = \frac{64}{3}, \gamma_7^7 = 64, \gamma_8^8 = 256$$

Pro $n > 8$ máme rekurzivní odhad horní hranice γ_n^n: $\gamma_n^n \leq \gamma_{n-1}^{\frac{n(n-1)}{n-2}}$.

Důkaz. (Důkaz toho, že algoritmus skončí a je korektní.) Je jednoduché ukázat, že na začátku kroku 4 jsou LLL podmínky z definice 2.1.3 splněny pro $i \leq k-1$ (stačí použít úvahy, které jsme už uvedli v základní myšlence algoritmu a jejím vylepšení). Proto pro $k > m$ jsme skutečně obdrželi LLL-redukované vektory. Vzhledem k tomu, že na vstupní matici $M = \{\mathbf{b_i}\}$ jsme průběhu algoritmu prováděli pouze takové operace, které jsou ekvivalentní s přenásobením maticí s determinantem ± 1, tvoří tyto LLL-redukované vektory LLL-redukovanou bázi L.

Zbývá dokázat, že algoritmus vždy skončí. Zřejmě algoritmus skončí, pokud $k = m+1$. Na začátku má k hodnotu 2, ta se v průběhu algoritmu buď zvýší o 1 (pokud je splněna Lovászova podmínka pro aktuální B_k), nebo se sníží o 1 (pokud Lovászova podmínka pro B_k není splněna). To, jestli podmínka bude nebo nebude splněna, závisí pouze na hodnotách B_k a B_{k-1}. Přitom hodnoty B_i se mění pouze ve funkci SWAP(k), kterou voláme pouze v případě, že Lovászova podmínka pro příslušné k není splněna. Uvnitř této funkce se vždy zmenší B_{k-1} na maximálně $\frac{3}{4}$ své původní hodnoty. Pokud by se nám podařilo odhadnout $\prod_{i=1}^{n-1} B_i$ zdola vhodnou konstantou (závislou na mřížce L), znamenalo by to, že po určité době už určitě Lovászova podmínka bude vždy splněna, a tedy k už se bude pouze zvyšovat, až dosáhne hodnoty $m+1$ a a lgoritmus skončí.

Označme tedy pro $0 \leq i \leq m$:

$$d_i = \prod_{1 \leq j \leq i} B_j$$

Zavedené označení bude jasnější, uvědomíme-li si, že podle poznámky 1.2.10 $d_i = \det\left((\mathbf{b_r}, \mathbf{b_s})_{1\le r,s\le i}\right)$. Zřejmě $d_i > 0, d_0 = 1, d_n = d(L)^2$. Označme L_i mřížku dimenze i generovanou vektory $\mathbf{b_j}$ pro $j \le i$. Pak zřejmě také $d_i = d(L_i)^2$. Dále položíme $D = \prod_{i=1}^{n-1} d_i$. K dokončení důkazu nám nyní stačí odhadnout d_i zdola vhodnou konstantou závislou pouze na L a i – tím získáme dolní odhad pro D i pro $\prod_{i=1}^{n-1} B_i$ závislý pouze na L, a tedy algoritmus skončí.

Nechť s_i je nejmenší nenulová hodnota q v L_i. Z tvrzení 2.2.2 získáváme:

$$d_i = d(L_i)^2 \ge d(L_i)^{\frac{2}{n}} \ge s_i^i \gamma_i^{-i} \ge s_n^i \gamma_i^{-i}.$$

Protože s_n je nejmenší nenulová hodnota $q(x)$ na L, poslední výraz závisí pouze na i a nezávisí na $\mathbf{b_j}$, tedy d_i je zdola ohraničeno kladnou konstantou závislou pouze na i a L. □

Poznámka 2.2.4. (1) Časová složitost LLL algoritmu je $O(n^6 \ln^3 B)$, pokud $\|\mathbf{b_i}\|^2 \le B$ pro všechna i a použijeme klasické aritmetické operace. Pokud použijeme efektivnější algoritmy pro aritmetické operace, můžeme časovou složitost snížit na $O(n^4 \ln B)$. Důkaz této časové složitosti lze najít v [31, s. 522-524].

(2) Pokud bychom měli bázi M zadanou Gramovou maticí, vektory $\mathbf{b_i}$ a $\mathbf{b_i^*}$ existují pouze abstraktně. Algoritmus tedy musíme pro tuto situaci upravit a jediným výstupem algoritmu bude matice H. Příslušnou úpravu algoritmu uvádí [7, s. 89]. Takto upravený algoritmus je efektivnější než původní, ale ne zas o tolik, aby se nám vyplatilo Gramovu matici počítat, pokud máme bázi mřížky zadanou souřadnicemi vzhledem ke kanonické bázi.

(3) Jak už jsme uvedli v 2.1.4 (2), konstantu $\frac{3}{4}$ v kroku 3 můžeme nahradit libovolnou konstantou c, $c \in \mathbb{R}$, $\frac{1}{4} < c < 1$. Použití jiné konstanty samozřejmě ovlivní „kvalitu“ výsledné LLL-redukované báze i čas, který bude algoritmus potřebovat k jejímu spočítání. Ideální by bylo zvolit $c = 1$, ale v tomto případě už není zaručeno, že algoritmus bude mít polynomiální časovou složitost. Můžeme tedy například zvolit $c = 0{,}99$. Další možností je variace konstanty c v průběhu výpočtu: začneme s konstantou $c = \frac{1}{4}$, pro kterou je redukce nejrychlejší a postupně zvyšujeme c až na hodnotu $c = 0{,}99$ – pro tuto hodnotu zase dostaneme nejlepší redukovanou bázi. Více podrobností o této metodě je popsáno v [27].

(4) Také bychom mohli Lovászovu podmínku $B_k \ge \left(\frac{3}{4} - \mu_{k,k-1}^2\right) B_{k-1}$ nahradit tzv. Siegelovou podmínkou $B_k \ge \frac{B_{k-1}}{2}$. Protože $|\mu_{k,k-1}| \le \frac{1}{2}$, Lovászova podmínka s konstantou $c = \frac{3}{4}$ implikuje splnění Siegelovy podmínky a naopak pokud nějaká báze splňuje Siegelovu podmínku, pak

splňuje také Lovászovu podmínku s konstantou $c = \frac{1}{2}$. V případě použití Siegelovy podmínky místo Lovászovy podmínky bychom mohli volání $\mathrm{RED}(k, k-1)$ provádět spolu s voláním ostatních $\mathrm{RED}(k, l)$.

(5) V algoritmu předpokládáme, že zadané vektory $\mathbf{b_i}$ jsou lineárně nezávislé – pokud nejsou, vypíše algoritmus chybové hlášení. Jistě by nebylo těžké upravit algoritmus tak, aby pro zadané lineárně závislé vektory byla výstupem LLL-redukovaná báze mřížky, kterou tyto vektory generují. Takto upravený algoritmus lze najít v [7, Algorithm 2.6.8].

(6) Pokud bychom uvažovali, že Gramova matice může mít reálné koeficienty (ne jen racionální), museli bychom čísla $\mu_{i,j}$ a B_i počítat přibližně v pohyblivé řádové čárce. Nepřesnosti způsobené takovouto reprezentací čísel pak mohou způsobit chybné výstupy LLL algoritmu (přičemž můžeme získat bázi, která je „téměř“ LLL-redukovaná, ale také bázi, která má k LLL-redukovanosti „velice daleko“). Podrobnosti týkající se této situace tu více rozebírat nebudeme, zvídavého čtenáře však můžeme odkázat na [7, s. 90] a pro podrobnější informace na [43, 45].

(7) Neustále se objevují nové verze LLL algoritmu, či jsou na jeho základě vytvářeny nové algoritmy specializované pro určitou oblast použití. Nebudeme se zde snažit všechny možné varianty ani vyjmenovat, jen namátkou však můžeme dychtivým čtenářům doporučit [7, s. 90-97] a [41, 43, 44, 45, 46].

3. Aplikace

A. K. Lenstra, H. W. Lenstra a L. Lovász publikovali LLL algoritmus jako algoritmus pro faktorizaci polynomů s racionálními koeficienty (viz článek [31]). Od té doby byla nalezena řada oblastí uplatnění tohoto algoritmu v matematice, informatice i kryptologii[1]. Zároveň samotný algoritmus procházel řadou změn, vylepšení i modifikací vhodných pro různé oblasti použití a stal se základem či součástí mnoha komplikovanějších a efektivnějších algoritmů. Pokusíme se zde alespoň stručně popsat hlavní oblasti použití algoritmu – vzhledem k tomu, že rozsah práce nedovoluje věnovat se jednotlivým oblastem podrobněji, uvádíme alespoň odkazy na další literaturu k jednotlivým tématům.

LLL algoritmus se používá především v těchto oblastech:

1. Faktorizace polynomů nad celými nebo racionálními čísly, případně nad dalšími tělesy. Přestože LLL algoritmus je polynomiální, exponenciální algoritmus Berlekamp-Zassenhaus byl pro mnoho polynomů prakticky rychlejší, protože LLL algoritmus vedl na mřížky velké dimenze s velkými koeficienty. V současnosti se používá[2] algoritmus Marka von Hoeije (viz [16]) z roku 2000, ve kterém se podařilo zkombinovat dobré vlastnosti obou algoritmů.
2. Problémy teorie mřížek: Problém nejmenší báze mřížky, problém nejkratšího vektoru mřížky a problém nejuzavřenějšího vektoru mřížky. Problém nejkratšího vektoru mřížky L dimenze n (SVP) je najít vektor $\mathbf{v} \in L$ tak, že $\|\mathbf{v}\| = \lambda_1(L)$ (viz definice 2.1.1). Vzhledem k tomu, že nejkratší vektor mřížky zatím obecně nedokážeme v polynomiálním čase najít, postačí nám často najít vhodnou aproximaci nejkratšího vektoru mřížky, tedy vektor $\mathbf{v} \in L$ tak, že $\|\mathbf{v}\| = f(n)\lambda_1(L)$ pro nějaký aproximační faktor $f(n)$. Z 2.1.5 (iv) vidíme, že LLL algoritmus dokáže aproximovat SVP s faktorem $2^{\frac{n-1}{2}}$.

   Problém nejuzavřenějšího vektoru mřížky (CVP) je následující: pro danou mřížku $L \subseteq \mathbb{Q}^n$ a daný vektor $\mathbf{u} \in \mathbb{R}^n$ najít vektor $\mathbf{v} \in L$ tak, aby $\|\mathbf{u} - \mathbf{v}\|$ byla minimální. Jedná se tedy o nehomogenní verzi SVP.

   Problém nejmenší báze mřížky: pro danou mřížku L najít bázi $M = \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$, která minimalizuje buď $\max_{1 \le i \le m} \{\|\mathbf{b_i}\|\}$ nebo součin $\|\mathbf{b_1}\| \cdot \cdots \cdot \|\mathbf{b_n}\|$. LLL algoritmus umožňuje nejmenší bázi aproximovat.

   Mnoho různých matematických problémů lze převést na některý z problémů teorie mřížek volbou vhodné mřížky.

[1] Kryptologií zde rozumíme nauku o šifrování, která zahrnuje kryptografii – vědu o vývoji šifrovacích systémů i kryptoanalýzu – umění prolomit šifrovací systémy.

[2] také v nových verzích systémů počítačové algebry, např. v Maple a Pari

3. Nalezení minimálního polynomu, jehož kořenem je zadané algebraické číslo (dané aproximací).
4. Pro danou posloupnost reálných čísel $(x_1, \ldots, x_n)$ najít posloupnost celých čísel $(a_1, \ldots, a_n)$, tak aby platilo: $\sum_{i=1}^{n} a_i x_i = 0 \wedge \exists i : a_i \neq 0$. Např. pro $\left(\mathrm{arctg}(1), \mathrm{arctg}(\frac{1}{5}), \mathrm{arctg}(\frac{1}{239})\right)$ zadané desetinnými aproximacemi, získáme výsledek $(1, -4, 1)$, tedy $\mathrm{arctg}(1) - 4\,\mathrm{arctg}(\frac{1}{5}) + \mathrm{arctg}(\frac{1}{239}) = 0$, což je tzv. Machinova formule pro výpočet čísla π[3].
5. Celočíselné lineární programování, které je významným odvětvím optimalizace. Základním problémem celočíselného lineárního programování je rozhodnout, jestli existuje celočíselné řešení m racionálních nerovnic o n neznámých. Formálně: Je dána matice $A \in \mathbb{Q}^{m \times n}$ a vektor $b \in \mathbb{Q}^m$ a máme rozhodnout, jestli množina $\mathbb{Z}^n \cap \{x \in \mathbb{R}^n | Ax \leq b\}$ je neprázdná. Použitím LLL algoritmu získáme řešení pro pevný počet proměnných v polynomiálním čase. Více o celočíselném programování např. [48] a [40], o použití LLL algoritmu v celočíselném programování [38] a [32].
6. Široké využití v kryptologii. Nejdříve byl LLL algoritmus používán v kryptoanalýze jako nástroj útoků na různé systémy – jako první byly prolomeny různé systémy založené na principu batohu (knapsack based cryptographic systems) – princip jednoho z těchto útoků bude v této práci rozebrán ještě podrobněji. Později byl LLL algoritmus použit i k útokům na šifrovací systém RSA (především v případech, kdy je malý veřejný či soukromý šifrovací exponent) a na některé systémy digitálních podpisů. Tuto problematiku shrnují například [15] a [49].

   V posledním desetiletí se problémy teorie mřížek staly inspirací i pro kryptografy a vznikly i kryptografické systémy s veřejným klíčem postavené na této teorii: nejznámější z nich je NTRU. Více informací o tomto systému lze nalézt v [17] a na domovské stránce NTRU[4]. První verze systému NTRU však byla prolomena paradoxně použitím LLL algoritmu – více o tomto útoku se lze dočíst v [8]. Další kryptografický systém s veřejným klíčem založený na teorii redukce mřížek představuje např. [14]. Použití

[3] Pomocí této formule lze zřejmě počítat π pomocí vztahu $\frac{\pi}{4} = 4\,\mathrm{arctg}(\frac{1}{5}) - \mathrm{arctg}(\frac{1}{239})$. Machinova formule je jedním z prvních zajímavých vztahů, které byly objeveny pro počítání desetiného rozvoje čísla π. Anglický matematik John Machin pomocí této formule roku 1706 spočítal prvních 100 číslic desetinného rozvoje čísla π, aniž by tuto formuli dokázal. Důkaz podal ještě téhož roku Hermann. Gauss, Stirling a další pak našli další formule pro počítání $\frac{\pi}{4}$. Počítání desetinného rozvoje čísla π je pro matematiky atraktivním problémem dodnes, i když nemá žádné praktické použití: v roce 2002 spočítal tým profesora Y. Kanady na jednom z nejmodernějších superpočítačů 1,2411 bilionu číslic desetinného rozvoje čísla π. Podrobnější informace o Machinově formuli (včetně jejího důkazu) lze najít např. na `http://milan.milanovic.org/math/english/pi/machin.html`

[4] `http://www.ntru.com`

LLL algoritmu v kryptoanalýze i kryptografii shrnuje [35].

Nyní podrobněji rozebereme jednu z aplikací LLL algoritmu, a to útoky na kryptografické systémy s veřejným klíčem založené na principu batohu.

3.1 Problém batohu

Existuje mnoho různých verzí problému batohu. Nejjednodušší z nich, tzv. *aditivní problém batohu 0-1*, vypadá takto: Mějme dána čísla $a_1, \ldots, a_n$, $a_i \in \mathbb{N}$ pro $i = 1, \ldots, n$ a $s \in \mathbb{N}$. Hledáme čísla $x_1, \ldots, x_n : x_i \in \{0, 1\}$ tak, aby platilo: $s = \sum_{i=1}^{n} x_i a_i$. Čísla $a_1, \ldots, a_n$ nazveme váhy položek a s nosnost. Problém můžeme také zformulovat ve vektorové notaci: Je dán vektor vah položek $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$ a nosnost $s \in \mathbb{N}$ a máme najít vektor $\mathbf{x} = (x_1, \ldots, x_n) \in \{0, 1\}^n$ tak, aby $s = \mathbf{x} \cdot \mathbf{a}$. Takovýto problém batohu je speciálním případem tzv. sčítání podmnožin (subset sum)[5], který je NP-úplný[6]. Právě uvedený problém batohu je NP-těžký, jak je dokázáno v [34].

Některé speciální případy aditivního batohu 0-1 lze vyřešit velice snadno, například problém batohu se superrostoucí posloupností[7] vah položek. Zřejmě

[5]Problém sčítání podmnožin je následující: máme dánu konečnou množinu obsahující celá čísla. V homogenní verzi se ptáme, zda součet čísel v některé podmnožině dané množiny je nula. V nehomogenní verzi je zadané číslo $s \in N$ a ptáme se, zda součet čísel v některé podmnožině zadané množiny je roven s.

[6]Uvedeme zde pouze neformální představu tříd problémů P, NP, NP-těžkých a NP-úplných. Pro formální definice a pro pochopení významu těchto tříd odkazujeme čtenáře na texty [6] a [37, Kapitola 5]. Označme $t_A(n)$ maximální dobu spotřebovanou algoritmem A k výpočtu přes všechny vstupy velikosti n. Řekneme, že algoritmus A má polynomiální složitost, jestliže existuje polynom $p(x)$ tak, že $t_A(n) \leq p(n)$ pro $\forall n \in \mathbb{N}$. Řekneme, že problém leží ve třídě P (lze ho provést v polynomiálním čase), pokud existuje nějaký algoritmus, který ho řeší a má polynomiální složitost. NP (zkratka pro nedeterministicky polynomiální) je množina problémů, pro které existuje algoritmus, který je vyřeší v polynomiálním čase na nedeterministickém Turingově stroji (umožňuje v každém kroku rozvětvit výpočet na k větví, v nichž se pak řešení hledá současně). U NP-úplných problémů lze v polynomiálním čase ověřit správnost výsledku algoritmu. Zřejmě $P \subseteq NP$. Řekneme, že vlastnost π_1 je polynomiálně redukovatelná na vlastnost π_2, pokud existuje funkce f z P tak, že x má vlastnost $\pi_1 \iff f(x)$ má vlastnost π_2. NP-úplné nazveme takové problémy třídy NP, na které lze polynomiálně redukovat všechny ostatní problémy z NP. Řekneme, že problém π je NP-těžký, pokud existuje nějaká NP-úplná vlastnost π_0 tak, že pokud existuje polynomiální algoritmus pro π, existuje i polynomiální algoritmus pro π_0. Jednou z největších nevyřešených otázek je, zda $P \neq NP$ (za rozhodnutí této otázky dokonce vypsal Clay Mathematics Institute of Cambridge v r. 2000 odměnu \$ 1 000 000.)

[7]Posloupnost $\{a_i\}_{i=1}^{n}, a_i \in \mathbb{Z}$ nazveme *superrostoucí*, pokud platí: $a_j > \sum_{i=1}^{j-1} a_i$ pro $\forall j = 2, \ldots, n$, tedy každý prvek posloupnosti je ostře větší než součet všech prvků, které mu předchází.

můžeme v tomto případě počítat x_i od x_n po x_1 pomocí vztahu:

$$x_i = 1 \iff s - \sum_{j=i+1}^{n} x_j a_j \geq a_i.$$

Jednoduchý algoritmus na počítání x_i můžeme pak sestrojit velice podobně jako algoritmus na převod čísla z desítkové do dvojkové soustavy.

Aditivní problém batohu 0-1 můžeme zobecnit tak, že složky vektoru $\mathbf{x}$ budou z množiny $\{0, 1, 2, \ldots, 2^b - 1\}$, kde $b \in \mathbb{N}_0$, místo z množiny $\{0, 1\}$. Tento problém batohu pak nazveme *zobecněný problém batohu*. Dalším problémem batohu může být tzv. *multiplikativní problém batohu*, kdy pro daný vektor vah položek $\mathbf{a}$ a danou nosnost $s \in \mathbb{N}$ hledáme vektor $\mathbf{x}$, tak aby: $\prod_{i=1}^{n} a_i^{x_i} = s$. Rozhodnout, zda má zadaný multiplikativní problém batohu řešení, je také NP-úplný problém. Vyřešit ho je jednoduché, pokud jsou jednotlivé váhy položek navzájem nesoudělná čísla. Multiplikativní problém batohu lze převést na aditivní zlogaritmováním.

Problém batohu se stal předmětem zájmu zkoumání mnoha kryptografů a matematiků poté, co Diffie a Hellman zveřejnili v roce 1976 svůj nápad šifrovacího systému s veřejným klíčem [11][8]. Zdálo se, že NP-úplnost problému rozhodnutí, zda má problém batohu řešení, by mohla zajistit dostatečnou bezpečnost a přispět k nalezení tzv. jednosměrné funkce (funkce invertovatelné jen za pomoci nějaké „soukromé“ informace)[9], která byla nezbytná pro uvedení systémů s veřejným klíčem do praxe. Proto byly také postupně formulovány různé varianty problému batohu – výše zmíněné, ale i další, o kterých se může zvídavý čtenář více dozvědět např. v [26].

3.2 Šifrovací systémy založené na problému batohu – prohraná sázka kryptografie?

3.2.1 Šifrovací systémy Merkle-Hellman založené na problému batohu

V roce 1978 navrhli Merkle a Hellman jeden z prvních šifrovacích systémů s veřejným klíčem [34][10]. Merkle a Hellman navrhli 2 metody, jak vytvořit šifrovací systém založený na problému batohu. První, známější metoda, využívala převodu aditivního 0-1 batohu se superrostoucí posloupností vah položek

[8] Jejich práce byla ovlivněna myšlenkami R. C. Merkla.

[9] Formálnější definice jednosměrné funkce viz např. [6, s. 138-140], [37, str. 78-80].

[10] Svůj návrh zveřejnili několik měsíců po tom, co Rivest, Shamir a Adelman publikovali první systém s veřejným klíčem: RSA.

na, jak doufali, těžký aditivní 0-1 problém batohu, kde inverzní převod by byl nemožný bez znalosti soukromého klíče. Druhá metoda je založena na multiplikativním problému batohu: jednoduchý multiplikativní problém batohu, kde váhy položek $\mathbf{a} = (a_1, \ldots, a_n)$ jsou navzájem nesoudělná čísla, je pomocí umocňování čísla b modulo m, kde $b < m$, m je prvočíslo a $m > \prod_{i=1}^{n} a_i$, převeden na těžký multiplikativní problém batohu.

Merkle s Hellmanem také navrhovali různá vylepšení obou metod, která by ztížila kryptoanalýzu tohotu systému: permutovat váhy položek či přenásobit vektor vah položek libovolnou konstantou. Navrhli také iterativní metodu šifrování a dešifrování: v každé iteraci jsou zvoleny nové M a W, pomocí kterých se původní jednoduchý problém batohu převede několikrát až na výsledný těžký problém batohu. Tato verze je známá jako multi-iterativní, základní verze s jedním převodem jako jedno-iterativní.

Dále se budeme pro jednoduchost zabývat podrobněji pouze aditivní jedno-iterativní variantou šifrovacího systému Merkle-Hellman. Pseudokód tohoto šifrovacího systému je uveden v příloze. Nechť $\mathbf{s} = (s_1, \ldots, s_n)$ je superrostoucí posloupnost vah položek, $M, W \in \mathbb{Z}$ tak, že: $M > \sum_{i=1}^{n} s_i$, $\gcd(M, W) = 1$. Merkle a Hellman převedli váhy položek batohu $\mathbf{s}$ na jinou posloupnost vah položek $\mathbf{a}$ transformací:

$$a_i = W s_i \mod M \quad 1 \leq i \leq n.$$

Veřejným klíčem je nová posloupnost vah položek $\mathbf{a}$. Soukromým klíčem je $(M, W, \mathbf{s})$. Zprávu $\mathbf{x} \in \{0,1\}^n$ zašifrujeme: $c = (\mathbf{x}, \mathbf{a})$. K rozšifrování, tedy k získání $\mathbf{x}$ z c, bychom při znalosti veřejného klíče $\mathbf{a}$ potřebovali vyřešit problém batohu s vahami položek $\mathbf{a}$ a nosností c, což je obecně NP-těžký problém, pro který nemáme zatím k dispozici žádný deterministický polynomiální algoritmus. Exponenciální algoritmus pro řešení tohoto problému můžeme snadno zkonstruovat tak, že budeme rekurzivně procházet všechny možnosti součtů vah položek, přičemž zkoumání každé možnosti ukončíme v případě, že součet vah položek je větší nebo roven c. Vzhledem k exponenciální časové náročnosti takového algoritmu však pro dostatečný počet položek batohu však řešení budeme hledat opravdu dlouho. Jak však brzy ukážeme, při kryptoanalýze nám může výrazně pomoct například to, jak byl problém batohu určený veřejným klíčem získán z problému batohu se superrostoucí posloupností vah položek.

Pokud známe soukromý klíč, získáme zprávu $\mathbf{x}$ tak, že vyřešíme problém batohu s vahami položek $\mathbf{s}$ a nosností $c' = cW^{-1} \mod M$, což je jednoduchý problém, protože $\mathbf{s}$ je superrostoucí posloupnost. Označme $U = W^{-1} \mod M$, pak:

$$c' = cU \equiv (\mathbf{x} \cdot \mathbf{a})\, U \equiv \mathbf{x} \cdot (\mathbf{a}U) \equiv \mathbf{x} \cdot \mathbf{s} \pmod{M}.$$

Protože $\sum_{i=1}^{n} x_i s_i < M$, platí: $c' = \mathbf{x} \cdot \mathbf{s}$.

Merkle si byl jistý, že tato verze je dostatečně bezpečná a vypsal odměnu \$ 100 za její prolomení.

Příklad 3.2.1. Alice a Bob si chtějí posílat důvěrné zprávy. Protože vědí, že linku, po které si zprávy posílají, odposlouchává pan X, rozhodli se své zprávy šifrovat – po lince tedy budou putovat pouze kryptogramy (zašifrované zprávy), o kterých doufají, že je pan X nedokáže rozluštit. Domluvili se, že pro šifrování použijí Merkle-Hellman jedno-iterativní aditivní systém. Nechť $n = 5$. Alice zvolí $W = 37$, $M = 43$ a superrostoucí posloupnost $\mathbf{s} = (1, 3, 5, 11, 21)$ a spočítá svůj veřejný klíč $\mathbf{a} : a_i \equiv W \cdot s_i \pmod{M}$, tedy $\mathbf{a} = (37, 25, 13, 20, 3)$. Alice má tedy veřejný klíč $(37, 25, 13, 20, 3)$ a soukromý klíč $(43, 37, (1, 3, 5, 11, 21))$.

Zašifrování zprávy: Bob, který chce Alici poslat zprávu $\mathbf{x} = (1, 0, 1, 1, 0)$, spočítá $c = 37 + 13 + 20 = 70$ a pošle toto číslo Alici.

Dešifrování zprávy: Aby Alice dešifrovala zprávu, kterou obdržela od Boba, spočítá $d \equiv W^{-1}c \pmod{M} = 7 \cdot 70 \equiv 17 \pmod{M}$ a vyřeší problém batohu se superrostoucí posloupností $\mathbf{a}$ a nosností d: $17 - 0 \cdot 21(0) = 17 - 1 \cdot 11(1) = 6 - 1 \cdot 5(1) = 1 - 0 \cdot 3(0) = 1 - 1 \cdot 1(1) = 0$. Tak získá zprávu $\mathbf{x} = (1, 0, 1, 1, 0)$.

Útok na systém: Pan X zachytí kryptogram, který poslal Bob Alici a zná pouze Alicin veřejný klíč. Pokud chce rozluštit zprávu, mohl by vyřešit problém batohu s vektorem vah položek $(37, 25, 13, 20, 3)$ a nosností 70. V tomto případě pan X snadno uhodne, že řešením je $37 + 13 + 20 = 70$, tedy původní zpráva je $\mathbf{x} = (1, 0, 1, 1, 0)$. Pro větší n a větší čísla a_i by měl pan X s hledáním řešení obecného batohu mnohem více práce. Jak však brzy ukážeme (v částech 3.2.3 a 3.2.4), pokud pan X dokáže dobře použít LLL algoritmus, s velice vysokou pravděpodobností se mu podaří kryptogram rozšifrovat v polynomiálním čase.

Alice a Bob dnes tedy tento systém použijí pouze, pokud jim záleží spíše na rychlosti a jednoduchosti šifrování a dešifrování zpráv než na bezpečnosti celého systému.

3.2.2 Kryptoanalýza Merkle-Hellman šifrovacího systému založeného na aditivním problému batohu

V roce 1982 Shamir předvedl útok na tento systém [42] a získal slíbenou odměnu \$ 100. Jeho útok byl založen na vyřešení problému lineárního programování. Po úspěšném útoku Shamira byl Merkle přesvědčen, že multi-iterativní verze šifrovací systému Merkle-Hellman je absolutně bezpečná a vypsal za její prolomení odměnu \$ 1 000. K jeho velké smůle však v roce 1982 také A.

K. Lenstra, H. W. Lenstra a L. Lovász publikovali LLL algoritmus, který se záhy začal používat v kryptoanalýze – už v tomtéž roce předvedl Adelman útok na šifrovací systém Graham-Shamir [1], byl výrazně zefektivněn útok na jedno-iterativní šifrovací systém Merkle-Hellman, a konečně roku 1984 Brickell předvedl úspěšný útok na multi-iterativní verzi šifrovacího systému Merkle-Hellman založený na redukci mřížek [4]. Nezbývá než doufat, že to byla poslední Merkleova sázka na tenkém ledě kryptografie[11].

3.2.3 Řešení batohů s nízkou hustotou

Jde o skupinu problémů batohu, které lze řešit pomocí redukce mřížky. Nechť $\mathbf{a} = (a_1, \ldots, a_n)$ je vektor vah položek s maximální váhou A, tedy $A = \max_{1 \leq i \leq n} a_i$. *Hustotu* vah položek $a_1, \ldots, a_n$ označíme d a definujeme: $d = \frac{n}{log_2 A}$. Brickell v [3] a Lagarias a Odlyzko v [24] nezávisle na sobě dokázali, že většinu batohů s nízkou hustotou pro velké n lze vyřešit, pokud máme tip na nejkratší vektor příslušné mřížky. Útok Lagariase a Odlyzka funguje pro většinu batohů, ve kterých je hustota vah položek $d < 0{,}6463\ldots$: spočívá v nalezení nejkratšího vektoru mřížky L_0 generované řádky matice

$$B_0 = \begin{pmatrix} 1 & 0 & \ldots & 0 & Na_1 \\ 0 & 1 & \ddots & \vdots & Na_2 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ \vdots & & \ddots & 1 & Na_n \\ 0 & \ldots & \ldots & 0 & Ns \end{pmatrix},$$

kde $N > \frac{\sqrt{n}}{2}$ je libovolné celé číslo. Nechť $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ je řešení batohu s vahami položek $\mathbf{a}$ a nosností s. Vektor souřadnic $\mathbf{c_0} = (x_1, \ldots, x_n, -1) \in \mathbb{Z}^{n+1}$ generuje vektor mřížky

$$\mathbf{b_0} = \mathbf{c_0} B_0 = \Big(x_1, \ldots, x_n, N\left(x_1 a_1 + \cdots + x_n a_n - s\right)\Big) = (x_1, \ldots, x_n, 0)\,.$$

Nalezením nejkratšího vektoru $\mathbf{b_0}$ v mřížce L_0 tedy získáme $\mathbf{x} = (x_1, \ldots, x_n)$. Za předpokladu, že nejvýš $\frac{1}{2}$ hodnot x_i je různá od 0, je tento vektor velice malý ($\|\mathbf{b_0}\|^2 \leq \frac{1}{2}n$). Pro velká n autoři ukázali, že pravděpodobnost P, že $\mathbf{b_0}$ je opravdu nejmenším vektorem v mřížce L_0 je dána vzorcem:

$$P \leq n\Big(2n\sqrt{\frac{1}{2}n} + 1\Big)\frac{2^{c_0 n}}{A},$$

[11] Ralph C. Merkle je profesorem na College of Computing v Georgii a kromě kryptografie se zabývá také nanotechnologiemi. Za spoluobjevení kryptografie s veřejným klíčem získal ceny společností ACM a IEEE.

kde $c_0 = 1{,}54724\ldots$ Tento odhad pravděpodobnosti byl získán počítáním bodů mřížky ve vysokých dimenzích. Při volbě $A = 2^{cn}$ konverguje tato pravděpodobnost k nule a n se zvětšuje, kdykoliv $c > c_0$. Tedy většinu problémů batohu s velkým n a hustotou $d = \frac{n}{log_2 A} = \frac{1}{c} < 0{,}6463\ldots$ můžeme vyřešit, pokud máme tip na nejkratší vektor mřížky. Kompletní důkaz těchto tvrzení lze najít v [24].

Tato hranice hustoty byla nezávisle zvýšena Costerem, LaMacchiou a Schnorrem a také Jouxem a Sternem. Jejich výsledky jsou shrnuty v [9]. Coster, LaMacchia, Odlyzko a Schnorr uvažují mřížku L_1 generovanou řádky matice

$$B_1 = \begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 & Na_1 \\ 0 & 1 & \ddots & & \vdots & Na_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & & \ddots & 1 & 0 & Na_{n-1} \\ 0 & 0 & \ldots & 0 & 1 & Na_n \\ \frac{1}{2} & \frac{1}{2} & \ldots & \frac{1}{2} & \frac{1}{2} & Ns \end{pmatrix}.$$

V tomto případě vektor souřadnic $\mathbf{c_1} = (x_1, \ldots, x_n, -1) \in \mathbb{Z}^{n+1}$ generuje nejkratší vektor mřížky L_1: $\mathbf{b_1} = \mathbf{c_1} B_1 = \left(x_1 - \frac{1}{2}, \ldots, x_n - \frac{1}{2}, 0\right)$. Zřejmě každá složka vektoru $\mathbf{b_1}$ je rovna $-\frac{1}{2}$, 0 nebo $\frac{1}{2}$, protože $x_i \in \{0, 1\}$ pro každé $i = 1, \ldots, n$. Bez ohledu na skutečné hodnoty x_i vektor $\mathbf{b_1}$ musí splňovat: $\|\mathbf{b_1}\|^2 = \frac{1}{4}n$. Tedy vektor $\mathbf{b_1}$ je malým vektorem v mřížce L_1. Autoři ukazují, že pravděpodobnost P, že $\mathbf{b_1}$ není opravdu tím nejmenším vektorem v mřížce, je ohraničena:

$$P \leq n\left(4n\sqrt{n} + 1\right)\frac{2^{c_1 n}}{A},$$

kde $c_1 = 1{,}0628\ldots$. Pokud $A = 2^{cn}$, pravděpodobnost konverguje k 0 a n se zvětšuje kdykoli $c > c_1$. Tedy téměř všechny problémy batohu s velkým n a hustotou $d < \frac{1}{c} = 0{,}9408\ldots$ lze vyřešit, pokud dokážeme najít nejkratší vektor mřížky.

Zatím nedokážeme najít nejkratší vektor mřížek dimenze $n > 4$, ale v praxi nám většinou stačí vektor, který najde LLL algoritmus.

Příklad 3.2.2. Předpokládejme, že pan X zachytil kryptogram $c = 1605$, který poslal Bob Alici, ví, že zpráva je zašifrována Merkle-Hellman šifrovacím systémem a našel si Alicin veřejný klíč $\mathbf{a} = (319, 196, 250, 477, 200, 559)$. Pouze se znalostí těchto údajů chce pan X zjistit původní zprávu $\mathbf{x}$. Zkusí tedy spočítat hustotu problému batohu zadaného Alicinými veřejným klíčem: $d = \frac{6}{log_2 559} \doteq 0{,}66 < 0{,}9408$. Po tomto zjištění už má pan X poměrně vysokou pravděpodobnost, že se mu podaří zprávu $\mathbf{x}$ zjistit v polynomiálním čase takto:

Pan X zvolí $N > \frac{\sqrt{n}}{2}$, např. $N = \left\lceil \frac{\sqrt{n}}{2} \right\rceil = 2$ a vytvoří matici A, na jejíž řádkové vektory pak uplatní LLL algoritmus:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 2 \cdot 319 \\ 0 & 1 & 0 & 0 & 0 & 0 & 2 \cdot 196 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 \cdot 250 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 \cdot 477 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 \cdot 200 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 \cdot 559 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 2 \cdot 1605 \end{pmatrix}.$$

Výsledkem aplikace LLL algoritmu na řádkové vektory matice A, jsou řádkové vektory matice A' :

$$A' = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & -2 & 2 & 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{3}{2} & -\frac{3}{2} & \frac{1}{2} & 2 \\ -3 & -1 & 0 & 2 & 1 & 0 & 2 \\ \frac{3}{2} & -\frac{5}{2} & -\frac{3}{2} & -\frac{1}{2} & \frac{3}{2} & -\frac{1}{2} & 0 \\ \frac{1}{2} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & -\frac{1}{2} & -\frac{3}{2} & 2 \end{pmatrix}.$$

Pana X z těchto vektorů zajímá pouze první vektor $\mathbf{b_1} = (y_1, \ldots, y_n)$, $y_i \in \left\{-\frac{1}{2}, 0, \frac{1}{2}\right\} \forall i = 1, 2, \ldots, n+1$. Zkusí spočítat $x_i = -y_i + \frac{1}{2}\ \forall i = 1, 2, \ldots, n$, čímž získá tip na vektor $\mathbf{x} = (1, 0, 1, 1, 0, 1)$ a ověří, jestli opravdu našel řešení: $319 + 250 + 477 + 559 = 1605$. Tedy zpráva, kterou poslal Bob Alici, byla 101101.

Na závěr poznamenejme, že problém batohu uvedený v příkladu 3.2.1, by se panu X touto metodou vyřešit nepodařilo, protože hustota problému batohu určeného veřejným klíčem je větší než 0,9408.

3.2.4 Využití diofantických aproximací

Definice 3.2.3. Nechť $\alpha \in \mathbb{R}, p \in \mathbb{Z}, q \in \mathbb{N}$ a $\gcd(p, q) = 1$. Racionální číslo $\frac{p}{q}$ se nazývá *dobrá diofantická aproximace čísla* α, jestliže pro všechna $q' \in \mathbb{N}, q' < q$ platí $|q'\alpha - p| > |q\alpha - p|$.

Nyní ukážeme útok na jedno-iterativní Merkle-Hellman šifrovací systém, založený na redukci báze mřížky a zároveň na diofantických aproximacích.

Začneme s kongruencí $a_i \equiv s_i W \pmod{M}$. Definujeme $k_i \in \mathbb{Z}$ pro $i = 1, \ldots, n$ tak, že:

$$a_i U - k_i M = s_i \tag{3.1}$$

a budeme se snažit uhodnout čísla k_i. Prvním důležitým faktem je, že daný problém batohu s vahami položek $\mathbf{a}$ a nosností c lze převést na nekonečně mnoho problémů batohu se superrostoucí posloupností vah $\mathbf{a}'$ a nosností c'. To bylo nezávisle dokázáno Eierem a Laggerem [13] a Desmedtem, Vanderwallem a Govaertsem [10]. Jejich výsledky (jejichž důkaz zde neuvedeme), lze shrnout do následujícího lemmatu:

Lemma 3.2.4. *Nechť $M, U, \mathbf{a}, k_i$ pro $i = 1, \ldots, n$ jsou definována stejně jako výše. Pak existuje $\varepsilon > 0$ tak, že:*

$$\frac{U'}{M'} \in \mathbb{Q} \wedge \left| \frac{U'}{M'} - \frac{U}{M} \right| \leq \varepsilon \Longrightarrow \mathbf{s}' = (s'_1, \ldots, s'_n)\,,\ \textit{kde}\, s'_i = a_i U' - k_i M'$$

$$\textit{pro}\, i = 1, \ldots, n,\ \textit{je superrostoucí.}$$

Stačí nám tedy najít takovou vhodnou dvojici čísel (U', M'), která nám převede původní problém batohu na problém batohu se superrostoucí posloupností vah položek. Ten už umíme jednoznačně vyřešit, takže znalost této dvojice je ekvivalentní znalosti tajných U a M. Z rovnosti 3.1 zřejmě pro každé $i = 1, \ldots n$ platí: $\frac{U}{M} - \frac{k_i}{a_i} = \frac{s_i}{a_i M}$, tedy každé $\frac{k_i}{a_i}$ je dobrou aproximací $\frac{U}{M}$. Tedy, pokud budeme znát k_i, pomocí $\frac{k_i}{a_i}$ můžeme dopočítat U' a M', které splňují podmínky lemmatu 3.2.4. Pro hodnoty k_i odvodil Shamir v [42] následující vztah:

$$\left| \frac{a_i}{a_1} - \frac{k_i}{k_1} \right| \leq \frac{M}{2^{n-i}\,|a_1 k_1|} \text{ pro } \quad \text{i} = 1,\ \ldots,\ \text{n}, \tag{3.2}$$

tedy každé $\frac{k_i}{k_1}$ je dobrou aproximací $\frac{a_i}{a_1}$. To vede k problému nalezení simultánní diofantické aproximace – tedy aproximace vektoru $(\frac{a_2}{a_1}, \frac{a_3}{a_1}, \ldots, \frac{a_n}{a_1}) \in \mathbb{Q}^{n-1}$ vektorem $(\frac{k_2}{k_1}, \frac{k_3}{k_1}, \ldots, \frac{p_n}{k_1}) \in \mathbb{Q}^{n-1}$, $k_1 < a_1$.

Definice 3.2.5. Nechť $\delta \in \mathbb{R}$ pevné. Řekneme, že vektor $(\frac{p_1}{p}, \frac{p_2}{p}, \ldots, \frac{p_n}{p}) \in \mathbb{Q}^n$ je *simultánní diofantickou aproximací δ-kvality vektoru* $(\frac{q_1}{q}, \frac{q_2}{q}, \ldots, \frac{q_n}{q}) \in \mathbb{Q}^n$, když $p < q$ a zároveň:

$$\left| p\frac{q_i}{q} - p_i \right| \leq q^{-\delta} \text{ pro } i = 1, 2, \ldots, n.$$

(Čím větší je δ, tím lepší je aproximace.) Dále řekneme, že se jedná o *neobyčejně dobrou simultánní diofantickou aproximaci* (UGSDA), pokud $\delta > \frac{1}{n}$.

Takovou aproximaci nazýváme neobvykle dobrou, neboť pravděpodobnost toho, že taková aproximace existuje, je poměrně malá vzhledem ke všem

možným volbám q_i. Nalezení simultánní diofantické aproximace δ-kvality i nalezení UGSDA lze řešit LLL algoritmem např. následujícím algoritmem popsaným v [25, 33].

Algoritmus (Nalezení diofantické simultánní aproximace δ-kvality). K zadanému vektoru $(\frac{q_1}{q}, \frac{q_2}{q}, \ldots, \frac{q_n}{q}) \in \mathbb{Q}^n$ najdeme simultánní diofantickou aproximaci $(\frac{p_1}{p}, \frac{p_2}{p}, \ldots, \frac{p_n}{p})$ δ-kvality, $\delta \in \mathbb{R}^+$ takto:
[Krok 1] $\lambda \leftarrow \lfloor q^\delta \rfloor$
[Krok 2] Najdeme LLL redukovanou bázi $B = \{\mathbf{b_1}, \ldots \mathbf{b_n}\}$ mřížky dimenze $(n+1)$, která je generována řádkovými vektory matice A :

$$A = \begin{pmatrix} \lambda q & 0 & 0 & \ldots & 0 & 0 \\ 0 & \lambda q & 0 & \ldots & 0 & 0 \\ 0 & 0 & \lambda q & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\lambda q_1 & -\lambda q_2 & -\lambda q_3 & \ldots & -\lambda q_n & 1 \end{pmatrix}.$$

[Krok 3] Pro každý vektor $\mathbf{v} = (v_1, v_2, \ldots, v_n, v_{n+1}) \in B$, pro který $v_{n+1} \neq q$, polož: $p \leftarrow v_{n+1}$, pro $i = 1, \ldots, n$ polož $p_i \leftarrow \frac{1}{q}\left(\frac{v_i}{\lambda} + pq_i\right)$. Jestliže $\left|p\frac{q_i}{q} - p_i\right| \leq q^{-\delta}$ pro každé $i, 1 \leq i \leq n$, vrať $(\frac{p_1}{p}, \frac{p_2}{p}, \ldots, \frac{p_n}{p})$.
[Krok 4] Vrať chybové hlášení – Buď simultánní diofantická aproximace δ-kvality neexistuje, nebo se ji nepodařilo tímto algoritmem najít.

Volbou $\delta > \frac{1}{n}$ můžeme právě uvedeným algoritmem najít UGSDA.

Algoritmus (Kryptoanalýza jedno-iterativního šifrovacího systému Merkle-Hellman založeného na aditivním problému batohu). Pro vektor vah položek $\mathbf{a}$ a kryptogram c se pokusí najít původní zprávu $\mathbf{x}$:
[Krok 1] Nalezení neobvykle dobré simultánní diofantické aproximace vektoru $(\frac{a_2}{a_1}, \frac{a_3}{a_1}, \ldots, \frac{a_n}{a_1})$. Pokud se nepodaří UGSDA najít, algoritmus skončí neúspěchem.
[Krok 2] Nalezení čísla $\frac{U'}{M'}$, které bude dobrou aproximací $\frac{k_1}{a_1}$ tak, aby platilo $\gcd(U', M') = 1$.
[Krok 3] Problém batohu s vahami položek $\mathbf{a}$ a nosností c převedeme na jednoduchý problém batohu se superrostoucí posloupností vah položek $\mathbf{a}'$ a nosností c', kde $a'_i = a_i U' \mod M'$ pro $i = 1, \ldots, n$ a $c' = cU' \mod M'$.

Poznámka 3.2.6. Uvedený postup funguje tím lépe, čím větší je n a jemu odpovídající $M, W, \mathbf{s}$ a $\mathbf{a}$. Neobvykle dobré simultánní diofantické aproximace jsou čím dál vzácnější pro vzrůstající n. Malé hodnoty n se v systému Merkle-Hellman z bezpečnostních důvodů nepoužívají, protože pak by

bylo možné řešit obecný prolém batohu zadaný veřejným klíčem a kryptogramem exponenciálním algoritmem. Merkle a Hellman v [34] doporučují používat $100 \leq n \leq 200$ a pro $n = 100$: $2^{201} + 1 \leq M2^{202} - 1$, $(2^{i-1} - 1) \cdot 2^{100} \leq s_i \leq 2^{i-1}2^100$, $2 \leq U \leq M - 2$.

Ukážeme teď jeden „malý“ příklad na tento způsob kryptoanalýzy s tím, že zdaleka ne pro všechny takto „malé“ příklady lze najít řešení LLL algoritmem a ani tento příklad nelze řešit úplně bez potíží.

Příklad 3.2.7. Tentokrát předpokládejme, že pan X zachytil kryptogram $c = 6665$ a chce ho dešifrovat metodou diofantických aproximací. Najde Alicin veřejný klíč

$$\mathbf{a} = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570, 726).$$

Pan X spočítá $\delta \leftarrow \frac{1}{n-1}$, $\lambda \leftarrow \lfloor \sqrt[n-1]{575} \rfloor = 2$ a použije LLL algoritmus na řádkové vektory matice B:

$$B = \begin{pmatrix} 2 \cdot 575 & 0 & 0 & \dots & 0 & 0 \\ 0 & 2 \cdot 575 & 0 & \dots & 0 & 0 \\ 0 & 0 & 2 \cdot 575 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ -2 \cdot 436 & -2 \cdot 1586 & -2 \cdot 1030 & \dots & -2 \cdot 726 & 1 \end{pmatrix}.$$

Výsledkem LLL algoritmu budou řádkové vektory $\{\mathbf{b'_1}, \dots, \mathbf{b'_n}\}$. Pokud bude pan X postupovat podle algoritmu pro nalezení simultánní diofantické aproximace δ-kvality, získá pro $\mathbf{b'_6}$ koeficienty

$$\mathbf{k} = (k_1, \dots, k_n) = (91, 69, 251, 163, 304, 90, 114, 187, 249, 115).$$

(Přestože nalezené hodnoty k_i přesně odpovídají hodnotám k_i z rovnosti 3.1, k_9 nesplňuje podmínku na UGSDA.) Vzhledem k tomu, že $\gcd(k_1, a_1) = \gcd(91, 575) = 1$, položí pan X: $U' \leftarrow 91$ a $M' \leftarrow 575$. Pak spočítá nový vektor vah položek

$$\mathbf{a}' = (a'_1, \dots, a'_n) = (0, 1, 1, 5, 11, 29, 61, 128, 270, 516),$$

$$\text{kde } a'_i \equiv a_i \cdot 91 \mod 575 \quad \text{a nové } c' \equiv 6665 \cdot 91 \equiv 465 \mod 575.$$

Přestože nový vektor vah položek $\mathbf{a}'$ neobsahuje úplně superrostoucí posloupnost, pan X už snadno uhodne řešení (váhat bude pouze u prvních 3 jeho složek): $\mathbf{x} = (1, 0, 1, 1, 0, 0, 1, 1, 1, 0)$. A skutečně $575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665$, tedy původní zpráva byla 1011001110.

Další možnost hledání UGSDA

Pro velká n můžeme použít jiný postup pro hledání UGSDA uvedený v [15], kde stačí použít vhodnou mřížku dimenze $t \leq n$. Uvažujme t-dimenzionální mřížku L, jejíž bází jsou řádky matice B:

$$B = \begin{pmatrix} a_2 & a_3 & \dots & a_t & \lfloor a_1^{\frac{1}{t}} \rfloor \\ -a_1 & 0 & \dots & \dots & 0 \\ 0 & -a_1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -a_1 & 0 \end{pmatrix}.$$

Každý vektor $\mathbf{u} \in L$ je tvaru: $\mathbf{u} = \left(u_1 a_2 - u_2 a_1, \dots, u_1 a_t - u_t a_1, u_1 \lfloor a_1^{\frac{1}{t}} \rfloor\right)$, kde $u_1, \dots, u_t \in \mathbb{Z}$. Pokud $\mathbf{v}$ je nejmenší vektor mřížky L nalezený LLL algoritmem, pak podle 2.1.5 (iii) platí:

$$\|\mathbf{v}\| \leq 2^{\frac{t-1}{4}} d(L)^{\frac{1}{t}}.$$

Nyní potřebujeme spočítat determinant mřížky L – podle 1.2.7 (2) (Pohybujeme se v $\mathbb{R}^n$ a uvažujeme standartní skalární součin – ortonormální bází je kanonická báze $E = \{\mathbf{e_1}, \dots, \mathbf{e_n}\}$, kde $e_{ii} = 1, e_{ij} = 0$ pro $i \neq j$.) je bází mřížky L absolutní hodnota z determinantu matice přechodu od K k bázi mřížky, tedy $\left|\det(B^T)\right| = |\det(B)|$. Laplaceovým rozvojem tohoto determinantu podle prvního řádku získáváme ($\tilde{a}_{ij}$ označuje algebraický doplněk prvku a_{ij}): $d(L) = \sum_{j=1}^{t} a_{1j}\tilde{a}_{1j}$, kde zřejmě všechny $\tilde{a}_{1j}$ pro $0 < i \leq t-1$ jsou rovny nule, $\tilde{a}_{1t} = (-1)^{2t} a_1^{t-1}$. Tedy $d(L) = \lfloor a_1^{\frac{1}{t}} \rfloor a_1^{t-1}$ a platí:

$$\|\mathbf{v}\| \leq 2^{\frac{t-1}{4}} d(L)^{\frac{1}{t}} \leq 2^{\frac{t-1}{4}} a_1^{\frac{t-1}{t}} \lfloor a_1^{\frac{1}{t}} \rfloor^{\frac{1}{t}} \leq 2^{\frac{t-1}{4}} a_1^{\frac{t^2-t-1}{t^2}}.$$

Pak pro $i = 1, \dots, t-1$ platí:

$$\left|\frac{a_i}{a_1} v_1 - v_i\right| \leq \|\mathbf{v}\| \leq 2^{\frac{t-1}{4}} a_1^{-\frac{t+1}{t^2}} = a_1^{\frac{t-1}{4\log_2 a_1} - \frac{t+1}{t^2}} \leq a_1^{-\frac{1}{t-1}},$$

kde poslední nerovnost platí, pokud $t \geq 2\sqrt{\log_2 a_1}$. Tedy, pokud použijeme dostatečně velké t, pomocí LLL algoritmu najdeme vektor, který odhalí UGSDA $\left(\frac{a_2}{a_1}, \dots, \frac{a_t}{a_1}\right)$. Protože UGSDA jsou poměrně vzácné, můžeme předpokládat, že $v_i = k_i$ pro $i = 1, \dots, n$.

3.2.5 Bezpečnost a budoucnost těchto šifrovacích systémů

V roce 2000 přijala IEEE standard P1363 (a v roce 2004 ho aktualizovala jako P1363a): Standard Specifications For Public Key cryptography [20], ve kterém specifikuje 3 hlavní oblasti kryptografických funkcí: faktorizaci celých čísel, diskrétní logaritmus v konečném tělese a diskrétní logaritmus v grupě bodů eliptické křivky nad konečným tělesem. V roce 2005 přijala další standard P1363.1 týkající se kryptografických systémů založených na teorii mřížek. Tento fakt vyjadřuje názor kryptografické komunity na systémy založené na problému batohu: Přestože ani o jednom ze 3 problémů v P1363 není dokázáno, že by byl NP-těžký nebo NP-úplný, je jim v aplikacích v kryptografii důvěřováno mnohem víc než problému batohu, který je prokazatelně NP-těžký.

Počet šifrovacích systémů založených na problému batohu, které už byly prolomeny[12] je dostatečně odstrašující na to, aby pro tyto systémy byly vytvářeny nějaké standardy, přestože některé šifrovací systémy na principu batohu dosud prolomeny nebyly. Problém je v tom, že problém batohu je NP-těžký jako celek, ale konkrétní zadání nemusí být NP-těžké – i kdyby pouhé 1 % konkrétních zadání problému batohu bylo polynomiálně řešitelných, oslabuje to bezpečnost šifrovacího systému. Mnoho dosavadních šifrovacích systémů bylo prolomeno díky své konstrukci – snažili se jen zakrýt původní snadno řešitelný batoh, ale už tato informace poskytla dostatek informací k jejich prolomení. Dalším problémem je, že batohy s hustotou menší než 0,9408 dokážeme s poměrně velkou pravděpodobností řešit a batohy s hustotou větší než 1 už pro šifrovací systémy nelze použít, protože by se mohlo stát, že z jednoho kryptogramu bychom mohli obdržet více platných zpráv.

Na druhou stranu systémy založené na principu batohu umožňují poměrně rychlé šifrování a dešifrování. Jak je uvedeno v [36]: „Jedno-iterativní šifrovací systém Merkle-Hellman se 100 položkami může být až 100× rychlejší než RSA s modulem zhruba 500 bitů.“ Pokud by se tedy podařilo dobře využít NP-těžkosti obecného problému batohu, mohl by vzniknout velice zajímavý šifrovací systém.

V současné době lze jen těžko odhadnout, jestli budou vznikat nějaké další šifrovací systémy založené na principu batohu, pokud už jsou známé tak populární systémy jako RSA či tak nadějné systémy jako NTRU. Jisté je jen to, že systémů založených na principu batohu vzniklo od 80. let minulého století mnoho. Přestože většina z nich už byla prolomena, jejich vývoj se zatím úplně nezastavil.

[12] Podrobnosti o různých systémech a útocích na ně viz [26]

Seznam použité literatury

[1] Adelman, L. M. (1983). On Breaking Generalized Knapsack Public Key Cryptosystems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (s. 402-412). ACM, New York. Dostupný na `http://delivery.acm.org/10.1145/810000/808771/p402-adleman.pdf?key1=808771&key2=8238442411&coll=GUIDE&dl=GUIDE&CFID=71438707&CFTOKEN=58602468`

[2] Bican, L. *Lineární algebra a geometrie.* Praha: Academia, 2000. ISBN 80-200-0843-8.

[3] Brickell, E. F. Solving low density knapsacks. In D. Chaum, editor, *Advances in Cryptography - Proceedings of CRYPTO '96*, svazek 1666 *Lecture NOtes in Computer Science*, s. 129-142. Springer-Verlag, 1996.

[4] Brickell, E. F. (1985). Breaking Iterated Knapsacks. In G. R. Blakley, David C. Chaum (Ed.) *Advances in Cryptology – CRYPTO '84*, Lecture Notes in Computer Science: vol. 196 (s. 342-358). Springer-Verlag, Berlin. Dostupný na `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C84/342.PDF`

[5] Cassels, J. W. S. *An Introduction to Diophantine Approximation.* University Press, Cambridge: 1965.

[6] Černá, I. *Úvod do teórie zložitosti.* Učební text Fakulty informatiky Masarykovy univerzity, Brno 1998. Dostupný na `http://www.fi.muni.cz/usr/cerna/Slozitost/ia012.html.windows-1250`

[7] Cohen, H. *A course in computational algebraic number theory.* 4th print. New York: Springer-Verlag, 2000. ISBN 0-387-55640-0.

[8] Coppersmith, D. a Shamir, A. Lattice Attacks on NTRU. In *Advances in Cryptology – Eurocrypt'97.* Springer-Verlag, 1997. Dostupný na `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E97/52.PDF`

[9] Coster, M. J., Joux, A., LaMacchia, B. A., Odlyzko, A. M., Schnorr, C. P. a Stern, J. Improved low-density subset sum algrithms. *Computational Complexity*, 1992.

[10] Desmedt, Y. G., Vandewalle, J. P. a Govaerts, R. J. M. A critical analysis of the security of knapsack public-key algorithms. *IEEE Transactions on Information Theory*, IT-30: s. 601-611, 1984.

[11] Diffie, M. a Hellman, M. E. New Directions in Cryptography *IEEE Transactions on Information Theory*, IT-22(6): s. 644-654, 1976. Dostupný na `http://www.cs.rutgers.edu/~tdnguyen/classes/cs671/presentations/Arvind-NEWDIRS.pdf`

[12] Dummit, D. S. a Foote, R. M. *Abstract Algebra*. 3rd edition. New York: John Wiley & Sons, 2004. ISBN 0-471-43334-9.

[13] Eier, R. a Lagger, H. (1983). Trapdoors in knapsack cryptosystems. In T. Beth (Ed.) *Advances in Cryptology – Proceedings of CRYPTO '82*, Vol. 149 *Lecture Notes in Computer Science* (s. 316-322). Springer-Verlag. Dostupný na `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E82/316.PDF`

[14] Goldreich, O., Goldwasser, S. a Halevi, S. Public-Key Cryptosystems from Lattice Reduction Problems. *Lecture Notes in Computer Science*, 1294: s. 112-131, 1997. Dostupný na `http://theory.lcs.mit.edu/~oded/cryptography.html`

[15] Hinek, M. J. *Lattice Attacks in Cryptography: A Partial Overview*. 2004. Dostupný na `http://www.cacr.math.uwaterloo.ca/techreports/2004/cacr2004-08.pdf`

[16] Hoeij, M. van. Factoring polynomials and the knapsack problem. *Journal of Number Theory*, 95: s. 167-189, 2002. Dostupný na: `http://www.math.fsu.edu/~hoeij/knapsack/paper/knapsack.pdf`

[17] Hoffstein, J., Pipher, J. a Silverman, J. H. NTRU: A Ring-Based Public Key Cryptosystem *Lecture Notes in Computer Science*, 1423: s. 267-288, 1998. Dostupný na `http://ntru.com/cryptolab/pdf/ANTS97.pdf`

[18] Horák, P. a Janyška, J. *Analytická geometrie*. Brno: Masarykova univerzita, 2002. ISBN 80-210-1623-X.

[19] Horák, P. *Úvod do lineární algebry*. Učební text UJEP Brno, 1978.

[20] IEEE. P1363: Standard Specifications For Public Key Cryptography. Dostupný na `http://grouper.ieee.org/groups/1363/`

[21] Janyška, J. a Sekaninová, A. *Analytická teorie kuželoseček a kvadrik.* Brno: Masarykova univerzita, 1996. ISBN 80-210-1435-0.

[22] Klazar, M. *Kaleidoskop teorie čísel, kapitola 2: diofantické aproximace.* Učební text MMF UK Praha, 2000. Dostupný na `http://kam.mff.cuni.cz/~klazar/lectnotes.html`

[23] Knuth, D. E. *The art of computer programming. Vol. 2, Seminumerical algorithms.* Addison-Wesley Publishing Company, 1981. ISBN 02-010-3822-6.

[24] Lagarias, J. C. a Odlyzko, A. M. Solving low-density subset sums problems. *J. Assoc. Comput. Mach.*, 31, s. 229-246, 1985.

[25] Lagarias, J. C. Knapsack public key cryptosystems and Diophantine approximation. In D. Chaum (Ed.), *Advances in Cryptology: Proceedings of Crypto 83* (s.3-23). New York and London: Plenum Press, 1984. Rozšířený abstrakt dostupný na `http://www.math.lsa.umich.edu/~lagarias/doc/1218knap.pdf`

[26] Lai, M. K. *Knapsack Cryptosystems: The Past and the Future*, 2001. Dostupný na `http://www.cecs.uci.edu/~mingl/knapsack.html`

[27] LaMacchia, B. *Basis Reduction Algorithms and Subset Sum Problems.* Thesis, MIT Artificial Intelligence Lab, 1991. Dostupný na `http://www.farcaster.com/papers/sm-thesis/index.htm`

[28] Mac Lane, S. a Birkhoff, G. *Algebra*, 2. vydání, přeložili Legéň, A. a Smítal, J., Bratislava: Vydavatelstvo technickej a ekonomickej literatúry, 1974.

[29] Lang, S. *Algebra.* 3rd edition. New York: Springer Verlag, 2002. ISBN 0-387-95385-X.

[30] Lang, S. *Introduction to Diophantine approximations.* Springer-Verlag, New York, 1995. ISBN 0-387-94456-7.

[31] Lenstra, A. K., Lenstra, H. W. a Lovász, L. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4): s. 515-534, 1982. ISSN: 0025-5831.

[32] Lenstra, H. W. Jr. Integer Programming with a fixed number of variables *Math. Op. Res.*, 8: s. 538-548, 1983.

[33] Menezes, A. J., Oorschot, P. C. a Vanstone, S. C. *Handbook of Applied Cryptography*, 5th printing. CRC Press, 1996. ISBN 0-8493-8523-7. Dostupný na `http://www.cacr.math.uwaterloo.ca/hac/`

[34] Merkle, R. C. a Hellman, M. E. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, IT-24: s. 525-530, 1978. Dostupný na `http://www-ee.stanford.edu/~hellman/publications/30.pdf`

[35] Ngyen, P. O. a Stern, J. The Two Faces of Lattices in Cryptology. *Lecture Notes in Computer Science*, 2146: s. 146-180, 2001. Dostupný na `http://www.di.ens.fr/~pnguyen/pub.html#NgSt01`

[36] Odlyzko, A. M. The Rise and Fall of Knapsack Cryptosystems. In C. Pomerance (Ed.), *Cryptology and Computational Number Theory, Proceedings of Symposia in Applied Mathematics*, vol. 42 (s. 75-88), American Mathematics Society, 1990. Dostupný na `http://www.dtc.umn.edu/~odlyzko/doc/arch/knapsack.survey.pdf`

[37] Paseka, J. *Kryptografie*. Učební text Přírodovědecké fakulty Masarykovy univerzity, 1998. Dostupný na `ftp://www.math.muni.cz/pub/math/people/Paseka/lectures/CRYPTO/`

[38] Regev, O. *Integer Programming (Lattices in Computer Science: Lecture 5)*. Učební text Tel Aviv University, 2004. Dostupný na `http://www.cs.tau.ac.il/~odedr/teaching/lattices_fall_2004/ln/integer_programming.pdf`

[39] Regev, O. *LLL Algorithm (Lattices in Computer Science: Lecture 5)*. Učební text Tel Aviv University, 2004. Dostupný na `http://www.cs.tau.ac.il/~odedr/teaching/lattices_fall_2004/ln/lll.pdf`

[40] Ryjáček, Z. *Teorie grafů a diskrétní optimalizace 2*. Učební text Západočeské univerzity, 2001. Dostupný na `http://cam.zcu.cz/~ryjacek/students/ps/TGD2.pdf`

[41] Seysen, M. Simultaneous reduction of a lattice basis and its reciprocal basis. *Combinatorica*, 13(3): s. 363-376, 1993.

[42] Shamir, A. (1982) A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *Proceedings of the IEEE*

Symposium on Foundations of Computer Science (s. 145-152). New York: IEEE. Dostupný na http://www.bits-pilani.ac.in:12356/faculty/sundarb/courses/netsec/readings/crypto/complexity/breaking-mh.pdf

[43] Schnorr, C. P. A more efficient algorithm for lattice basis reduction. *J. Algorithms*, 9: s. 47-62, 1988.

[44] Schnorr, C. P. Fast LLL-Type Lattice Reduction. *Inf. Comput.*, 204(1): s. 1-25, 2006. Dostupný na http://www.mi.informatik.uni-frankfurt.de/research/papers.html

[45] Schnorr, C. P. a Euchner, M. (1991). Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. In L. Budach (Ed.), *Proc. Fundamentals of Computational Theory '91*, LNCS 529 (s. 68-85), Springer-Verlag. http://www.mi.informatik.uni-frankfurt.de/research/papers.html#1994

[46] Schnorr, C. P. a Hörner, H. H. Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction. *Lecture Notes in Computer Science*, 921: s. 1-12, 1995. Dostupný na http://www.mi.informatik.uni-frankfurt.de/research/papers.html

[47] Škarvada, L. a Pitner, T. *Návrh algoritmů I.* Učební text FI MU. Dostupný na http://www.fi.muni.cz/~libor/vyuka/IB002/

[48] Schrivjer, A. *Theory of Linear and Integer Programming.* Chichester: John Wiley & Sons, 1986. ISBN 0-471-90854.

[49] Stern, J. a Toffin, P. (1990). Cryptoanalysis of a Public-key Cryptosystem Based on Approximations by Rational Numbers *Advances in Cryptology – Eurocrypt'90.* Springer-Verlag.

[50] Zlatoš, P. *Lineárna algebra a geometria.* Elektronický učební text FMFI UK, Bratislava 2003. http://thales.doa.fmph.uniba.sk/katc/pages/member.php?jazyk=sk&clen=zlatos

A pseudokód LLL algoritmu

Mějme danou mřížku (L, q) a její libovolnou bázi $M = \{\mathbf{b_i}, \ldots, \mathbf{b_n}\}$ (buď souřadnicemi vzhledem ke kanonické bázi $\mathbb{R}^p$ pro $p \geq n$ nebo její Gramovou maticí). Výstupem algoritmu budou nové vektory $\mathbf{b_1}, \ldots, \mathbf{b_n}$, které budou tvořit LLL-redukovanou bázi L a matice H, která bude udávat souřadnice této LLL-redukované báze vzhledem k původně zadané bázi, sloupcové vektory matice H označíme $\mathbf{h_i}$.

Poznamenejme, že v následujícím algoritmu $c \leftrightarrow d$ označuje „prohození“ hodnot v proměnných c a d. Předpokládáme, že proměnné i, j, m, r jsou lokální (tedy unikátní v každém podprogramu algoritmu), zatímco ostatní proměnné jsou globální (tedy všechny funkce i hlavní program k nim mohou přistupovat a měnit jejich hodnotu).

Algorithm 1: LLL algoritmus

```
k := 2; k_max := 1; H := E_n                    /* inicializace */
b*_1 := b_1
B_1 := (b_1, b_1)                               /* konec inicializace */
repeat
    if k > k_max then                           /* Aktualizace G-S procesu */
        k_max := k; b*_k := b_k
        for j = 1, ..., k-1 do
            μ_{k,j} := (b_k, b*_j) / B_j; b*_k := b*_k - μ_{k,j} b*_j; B_k := (b*_k, b*_k)
        end
        if B_k = 0 then error: Vektory b_i netvoří bázi; exit
    end
                                                /* Konec aktualizace G-S procesu */
    RED (k, k-1)                                /* Ověřování LLL podmínek */
    while B_k < (3/4 - μ^2_{k,k-1}) B_{k-1} do
        SWAP (k); k := max{2, k-1}; RED (k, k-1)
    end
    for m = k-2, k-3, ..., 1 do RED (k, m)
    k := k+1
until k = n+1
Vypiš LLL-redukovanou bázi b_i a matici H.
```

Function `RED`(k,m)

if $\|\mu_{k,m}\| > \frac{1}{2}$ **then**
  $r := \lfloor \mu_{k,m} \rceil = \lfloor \frac{1}{2} + \mu_{k,m} \rfloor$
  $\mathbf{b_k} := \mathbf{b_k} - r\mathbf{b_m}; \mathbf{h_k} := \mathbf{h_k} - r\mathbf{h_m}$
  for $j = 1, \ldots m-1$ **do**
    $\mu_{k,j} := \mu_{k,j} - r\mu_{m,j}$
  end
  $\mu_{k,m} := \mu_{k,m} - r$
end

Function `SWAP`(k)

$\mathbf{b_{k-1}} \leftrightarrow \mathbf{b_k}; \mathbf{h_{k-1}} \leftrightarrow \mathbf{h_k}$
if $k > 2$ **then**
  for $j = 1, 2, \ldots, k-2$ **do**
    $\mu_{k-1,j} \leftrightarrow \mu_{k,j}$
  end
  $\mu := \mu_{k,k-1}; B := B_k + \mu^2 B_{k-1}; \mu_{k,k-1} := \frac{\mu B_{k-1}}{B}$
  $\mathbf{b} := \mathbf{b^*_{k-1}}; \mathbf{b^*_{k-1}} := \mathbf{b^*_k} + \mu\mathbf{b}; \mathbf{b^*_k} := -\mu_{k,k-1}\mathbf{b^*_k} + \frac{B_k}{B}\mathbf{b}$
  $B_k := \frac{B_{k-1}B_k}{B}; B_{k-1} := B$
  for $i = k+1, k+2, \ldots, k_{max}$ **do**
    $t := \mu_{i,k}; \mu_{i,k} := \mu_{i,k-1} - \mu t$
    $\mu_{i,k-1} := t + \mu_{k,k-1}\mu_{i,k}$
  end
end

B Šifrovací systém Merkle-Hellman

Aditivní jedno-iterativní šifrovací systém Merkle-Hellman

Function `Merkle-Hellman: Generování-klíče(`n`)`

Input: parametr n
Output: soukromý klíč $(M, W, (b_1, \ldots, b_n))$ a veřejný klíč $(a_1, \ldots, a_n)$
$(b_1, \ldots, b_n) :=$ libovolná superrostoucí posloupnost
$M :=$ libovolné $M \in \mathbb{N}$ tak, že $M > \sum_{i=1}^{n} b_i$
$W :=$ nahodné $W \in \mathbb{N}$ tak, že $W \leq M - 1 \wedge \gcd(M, W) = 1$
for $i = 1, 2, \ldots, n$ **do**
  $a_i := W \cdot b_i \mod M$
end
Vypiš soukromý klíč $(M, W, (b_1, \ldots, b_n))$ a veřejný klíč $(a_1, \ldots, a_n)$.

Function `Merkle-Hellman: Šifrování`

Input: veřejný klíč $(a_1, \ldots, a_n)$ a otevřený text $m \in \{0, 1\}^n$,
  $m = m_1 \ldots m_n$
Output: šifrový text $c \in \{0, 1\}^n$
$c := \sum_{i=1}^{n} m_i a_i$
Vypiš c.

Function `Merkle-Hellman: Dešifrování`

Input: Soukromý klíč $(M, W, (b_1, \ldots, b_n))$ a šifrový text c
Output: zpráva m odpovídající c
$U := W^{-1} c \mod M$
$(m_1, \ldots, m_n) := m_i \in 0, 1$ tak, že $U = \sum_{i=1}^{n} m_i b_i$
Vypiš m.