**IE 511: Integer Programming, Spring 2021** **23 Mar, 2021**

# Lecture 17: Lattice Problems, Structured IPs

*Lecturer: Karthik Chandrasekaran* *Scribe: Karthik*

**Disclaimer**: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

## 17.1 Computational Problems on Lattices

We were considering lattices and computational problems over lattices.

**Recap: Lattices**

For $a_1, \ldots, a_m \in \mathbb{R}^n$, let $\mathcal{L}(a_1, \ldots, a_m) := \{\sum_{i=1}^m \lambda_i a_i : \lambda_1, \ldots, \lambda_m \in \mathbb{Z}\}$

**Observation:** $\mathcal{L}$ is a group under addition.

**Definition 1.** A set $\mathcal{L} \in \mathbb{R}^n$ is a *lattice* if it is generated by linearly independent vectors.

**Definition 2.** For a lattice $\mathcal{L}$, $det(\mathcal{L}) := |\det \begin{bmatrix} b_1 & \ldots & b_n \end{bmatrix}|$ for some basis $b_1, \ldots, b_n$ of $\mathcal{L}$.

### 17.1.1 Membership testing problem

Given: A basis $b_1, \ldots, b_n \in \mathbb{R}^n$ of a lattice $\mathcal{L}$ and a vector $t \in \mathbb{R}^n$.
Goal: Is $t \in \mathcal{L}(b_1, \ldots, b_n)$?
This is the linear equations integer feasibility problem (LEIF).

$$\exists x \in \mathbb{Z}^n : \begin{bmatrix} b_1 & \ldots & b_n \end{bmatrix} x = t? \tag{17.1}$$

We know how to solve (17.1) in polynomial time using the Hermite basis for the lattice.

### 17.1.2 Shortest Vector Problem (SVP)

Given: A basis $b_1, \ldots, b_n \in \mathbb{R}^n$ of a lattice $\mathcal{L}$.
Goal: $\min\{\|y\| : y \in \mathcal{L}(b_1, \ldots, b_n), y \neq 0\}$, i.e., find a shortest non-zero vector in the lattice given by its basis.

Recall that $\|y\| = \sqrt{\sum_{i=1}^n y_i^2}$.

**Example:** $b_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, b_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Then $\mathcal{L}(b_1, b_2) = \mathbb{Z}^2$ (since the basis matrix is unimodular) and a shortest vector in $\mathcal{L}(b_1, b_2)$ is $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

**SVP and IP–a connection.** An equivalent formulation of SVP is

$$\min\left\{\|Bx\|^2 : x \neq 0, x \in \mathbb{Z}^n\right\}, \text{where } B = \begin{bmatrix} b_1 & \ldots & b_n \end{bmatrix}.$$

Note that this formulation is an integer optimization formulation where the objective function is non-linear.

In contrast to the membership testing problem, the shortest vector problem is NP-hard. The hardness of the problem has been used to design crypto systems. Note that it is not even clear how to solve this problem in time $2^{2^n}$, let alone polynomial-time. For example, if $n = 3$, can we solve this problem efficiently? Not so obvious. We will see some results which allows us to solve this problem in $2^{O(n^3)}$ time, which in turn implies an efficient algorithm to solve this problem for $n = 3$, i.e., for fixed constant $n$.

For membership testing problem, we saw that the Hermite basis of the lattice can be used to solve it. For SVP, another basis is useful in order to design fast algorithms. Before we introduce this basis, let us see some useful facts about lattice basis.

**Definition 3.** A basis $B = \begin{bmatrix} b_1 & \dots & b_n \end{bmatrix}$ is an *orthogonal* basis if $b_i^T b_j = 0 \ \forall i, j \in [n] \ i \neq j$.

**Proposition 4.** *If $b_1, \dots, b_n$ is an* orthogonal *basis for $\mathcal{L}$, then*

$$\min\left\{\|Bx\|^2 : x \neq 0, x \in \mathbb{Z}^n\right\} = \min\{\|b_i\|^2 : i \in [n]\},$$

*i.e., a shortest vector in the orthogonal basis is a shortest non-zero vector in the lattice.*

**Orthogonal basis implies SVP is easy**

*Proof of Proposition 4.* We prove the equality by showing inequality in both directions.
$\leq$: By definition.
$\geq$: Let $x := \arg\min\{\|Bx\|^2 : x \neq 0, x \in \mathbb{Z}^n\}$. Then

$$\begin{aligned}
\|Bx\|^2 &= x^T B^T B x \\
&= x^T \begin{bmatrix} \|b_1\|^2 & 0 & \dots & 0 \\ 0 & \|b_2\|^2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & \|b_n\|^2 \end{bmatrix} x \qquad \text{(since columns}(B) \text{ are pairwise orthogonal)} \\
&= \begin{bmatrix} x_1\|b_1\|^2 & x_2\|b_2\|^2 & \dots & x_n\|b_n\|^2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \\
&= \sum_{i=1}^{n} x_i^2 \|b_i\|^2 \\
&\geq \min_{i \in [n]} \|b_i\|^2 \qquad (\text{since } x_i \in \mathbb{Z} \ \forall i \in [n], x \neq 0).
\end{aligned}$$

□

So, if we can find an orthogonal basis for the given lattice, then we are done. But the given lattice may not have an orthogonal basis. Lövász suggested an alternative approach by observing that it is sufficient to solve a bounded norm SVP.

**Lemma 4.1** (Bounded Norm SVP - Lövász)**.** *Let $\mathcal{L} \subseteq \mathbb{R}^n$ be a full dimensional lattice with basis $b_1, \ldots, b_n$ and let*

$$\bar{x} := \arg\min \left\{ \|Bx\|^2 : x \neq 0, x \in \mathbb{Z}^n, |x_j| \leq \frac{\prod_{i=1}^n \|b_i\|}{det(\mathcal{L})} \ \forall j \in [n] \right\}.$$

*Then $v = B\bar{x}$ is a shortest non-zero vector in $\mathcal{L}$.*

*Proof.* **Exercise.** *Hint. Cramer's rule and Hadamard's inequality.* □

Lemma 4.1 reduces the search space for $x$ to solve SVP—namely, it suffices to consider $x$ whose coordinates are bounded. So, if we can find a basis $b_1, \ldots, b_n$ such that $\prod_{i=1}^n \frac{\|b_i\|}{\det(\mathcal{L})}$ is small, then we can do a brute force search over all integral $x$ such that $|x_j| \leq \frac{\prod_{i=1}^n \|b_i\|}{\det(B)} \ \forall j \in [n]$ and return the $x$ with smallest $\|Bx\|^2$. This raises the next computational problem.

### 17.1.3 Minimum Basis Problem

Given: A basis $a_1, \ldots, a_n \in \mathbb{R}^n$ of a lattice $\mathcal{L}$.
Goal: Find a basis $b_1, \ldots, b_n$ for $\mathcal{L}(a_1, \ldots, a_n)$ such that $\prod_{i=1}^n \|b_i\|$ is minimized.

Lövász showed that this problem is NP-hard. Hermite (1850) showed that every lattice $\mathcal{L} \subseteq \mathbb{R}^n$ has a basis $b_1, \ldots, b_n$ such that $\prod_{i=1}^n \|b_i\| \leq c_n \det(\mathcal{L})$ where $c_n$ is constant depending only on $n$. Lövász (1982) showed that there exists a polynomial-time algorithm to find a basis $b_1, \ldots, b_n$ such that $\prod_{i=1}^n \|b_i\| \leq 2^{\frac{1}{2}\binom{n}{2}} \det(\mathcal{L})$. His algorithm used a variant of Gram-Schmidt orthogonalization of the given vectors $a_1, \ldots, a_n$. As a corollary of Lövász's result and Lemma 4.1 showing that it is sufficient to solve bounded norm SVP, we can solve SVP in time $2^{O(n^3)}$—in particular, if $n$ is a fixed constant, then we can solve SVP efficiently.

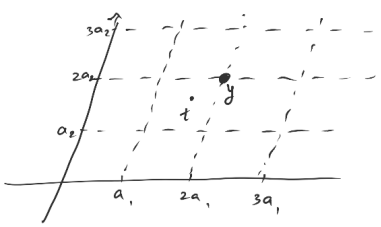
**Corollary 4.1.** *For fixed $n$, SVP can be solved efficiently.*

There has been a long and fruitful line of research addressing the approximability of SVP using various choice of basis. Improving the approximation factors and the runtime of the approximation algorithms remain important research directions.

### 17.1.4 Closest Vector Problem (CVP)

This is an extension of the membership testing problem.

Given: A basis $b_1, \ldots, b_n \in \mathbb{R}^n$ of a lattice $\mathcal{L}$ and a vector $t \in \mathbb{R}^n$
Goal: $\min\{\|t - y\| : y \in \mathcal{L}(b_1, \ldots, b_n)\}$, i.e., find the distance of $t$ to the closest lattice vector.



Van Emde Boas (1981) showed that this problem is NP-hard. CVP can be solved efficiently if $\mathcal{L}$ has an orthogonal basis and such a basis can be found efficiently. There are general families of lattice where CVP and SVP are known to be solvable efficiently. Broadening these families and verifying whether a given lattice belongs to these families are avenues for research.

There is a rich mathematical theory behind latices and the computational problems associated with them. The mathematical theory of lattices was initiated by Minkowski and it is known as the geometry of numbers. These are the foundations to the theory of IP. The computational aspects of lattices are much more recent owing to the recent advent of computers and the recent development of connections between lattices and cryptography. Consequently, there are numerous unexplored research questions concerning the computational aspects of lattices.

## 17.2 Structured IPs: Solving Techniques

In the next couple of lectures, we will see algorithmic techniques to solve *structured IPs*—structured IPs are those IPs which have some nice underlying *structure* that can be exploited to design efficient algorithms. The algorithms that we will see are iterative in nature and will broadly fall under one of two techniques:

1. **Primal Technique:** In this technique, we always maintain a primal feasible solution and we improve it towards optimum. This is similar to the primal simplex method but for IPs.
2. **Primal Dual Technique:** In this technique, we always maintain a dual feasible solution and try to find a primal feasible solution that satisfies complementary slackness with the current dual. This is similar to the dual simplex method but for IPs.

These techniques are broadly applicable for any optimization problem, let alone discrete optimization problems/IPs. We will illustrate these techniques for discrete optimization problems through specific applications.

### 17.2.1 Primal Technique

We will see max cardinality matching in bipartite graphs as a concrete application where the primal technique is useful to design efficient algorithms.

**Recall: Max Cardinality Matching in Bipartite Graphs**

**Definition 5.** Let $G = (V, E)$ be a graph. A *matching* $M \subseteq E$ is a set of edges with each vertex in $V$ adjacent to at most one edge in $M$.

**Definition 6.** Let $G = (V, E)$ be a graph. A *vertex cover* is a set $R \subseteq V$ such that every edge $e \in E$ is incident to at least one of the vertices of $R$.

**Weak duality:** $|M| \leq |R|$ for all matching $M$ and vertex cover $R$ in $G$.

**Max Cardinality Matching:**

Given: A graph $G$

Goal: $\max\{|M| : M \text{ is a matching in } G\}$

IP: $\max\{\sum_{e \in E} x_e : \sum_{e \in \delta(v)} x_e \leq 1 \ \forall v \in V, x_e \in \{0, 1\} \ \forall e \in E\}$

Recall that if $G$ is bipartite, then the LP-relaxation to the above IP has an integral optimum solution. But how do we solve this LP-relaxation? We will see an algorithm to solve the IP in bipartite graphs without relying on the LP-relaxation.

**Observation.** Feasible solutions to the integer program corresponds to a matching in the graph.

Primal algorithm outline: We will maintain a feasible solution to the IP and aim to improve it in each iteration.

Let us examine how to construct a matching of larger cardinality, or rather, how do we verify if a given matching is indeed a matching of maximum cardinality?

**Example:** See Figure 17.2.1. This example motivates the definition of $M$-alternating and $M$-augmenting paths.

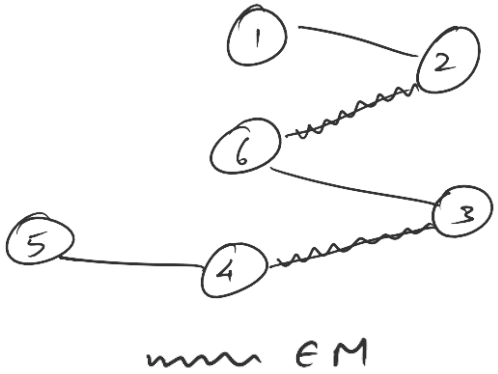



Figure 17.1: A larger matching than $M$ can be obtained by considering the path $1-2-6-3-4-5$ and exchanging the non-matching edges with the matching edges in the path.

**Definition 7.** Let $M$ be a matching in $G = (V, E)$. An *$M$-alternating path* is a path $P = v_0\ e_1\ v_1\ e_2 \dots e_t\ v_t$ such that $e_i = v_{i-1}v_i$ for all $i \in [t]$ and

1. $e_2, e_4, \dots, e_{\text{even}} \in M$ (even edges are in the matching)
2. $e_1, e_3, \dots, e_{\text{odd}} \in E \setminus M$ (odd edges are not in the matching)
3. $v_0$ is not incident to any edge in $M$. We call vertices that are not incident to edges of $M$ to be *$M$-exposed* vertices.

In the above example, the path 1, 12, 2, 26, 6, 63, 3, 34 is an $M$-alternating path. The vertex 1 is an $M$-exposed vertex.

**Definition 8.** An *$M$-augmenting path* is an $M$-alternating path such that

4. the vertex $v_t$ is not incident to any edge in $M$.

*Note:* An $M$-augmenting path always has $t$ to be odd.

An $M$-augmenting path tells us how to augment a matching i.e., improve the current matching $M$ to achieve a larger matching. See Figure 17.2.1.

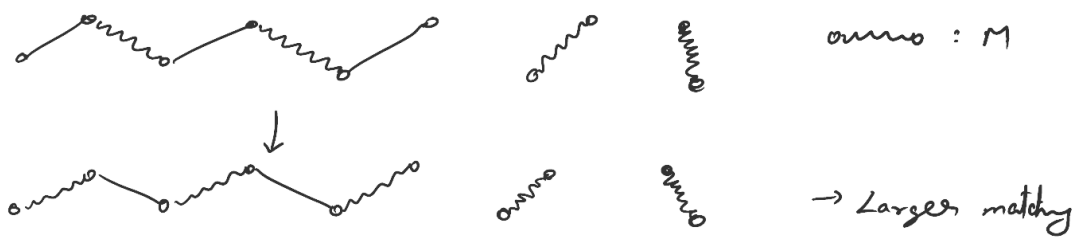



Figure 17.2: An $M$-augmenting path tells us how to augment a matching

**Lemma 8.1.** *Let $M$ be a matching and $P$ be an $M$-augmenting path. Then $M' = M \triangle E(P)$ is a matching with $|M'| > |M|$ where $M \triangle E(P) := (M \cup E(P)) \setminus (M \cap E(P))$ (here, $M \triangle E(P)$ is known as the symmetric difference between $M$ and $E(P)$).*

*Proof.* The vertices $v_0$ and $v_t$ are not incident to $M$, therefore, $M'$ is a matching.

Since $t$ is odd, $|P \cap (E \setminus M)| = |P \cap M| + 1$. It means that the number of non-matching edges in the path is one more that the number of matching edges in the path. Therefore, $|M'| = |P \cap (E \setminus M)| + |M \setminus P| = |P \cap M| + 1 + |M \setminus P| = |M| + 1 > |M|$. □

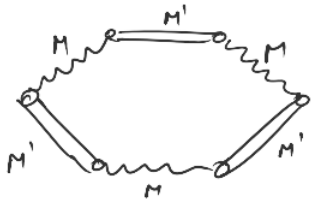
So, the existence of $M$-augmenting path implies that $M$ is not optimal. We will now show that this is the only possible reason for $M$ to be sub-optimal.

**Lemma 8.2.** *A matching $M$ is a maximum cardinality matching in $G$ iff there is no $M$-augmenting path in $G$.*

*Proof.* $\Longrightarrow$: Lemma 8.1

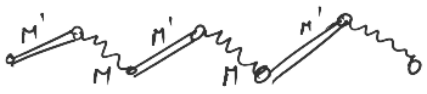
$\Longleftarrow$: We will show the contrapositive. Suppose $M$ is not a maximum matching in $G$. We will show that there exists an $M$-augmenting path. Since $M$ is not maximum, there exists a matching $M'$ in $G$ with $|M'| > |M|$. Let $H = (V, F)$ where $V = V(G)$ and $F = M \triangle M'$. The degree of each vertex in $H$ is 0, 1, or 2 which implies that the connected components in $H$ are vertices, paths, or cycles.

Let $C$ be a cycle in $H$.




Edges in $C$ alternate between $M$ and $M'$, therefore, each cycle in $H$ contains the same number of edges from $M$ and $M'$.

Let $P$ be a path in $H$.



The path $P$ can be even or odd length (can all paths in $P$ be of even length? No; if so, then $|M| = |M'|$ which is a contradiction).

We have $|M| > |M'|$. Therefore, at least one of the paths $P$ must contain more edges from $M'$ than $M$. This path $P$ is an $M$-augmenting path.

□

So, finding a maximum cardinality matching boils down to finding an $M$-augmenting path if one exists or confirming that no $M$-augmenting paths exist, i.e., it suffices to solve the following matching augmentation problem:

**Matching Augmentation Problem**

Given: A graph $G = (V, E)$ and a matching $M$ in $G$.
Goal: Find an $M$-augmenting path or prove that no $M$-augmenting path exists.

We will design an algorithm for the matching augmentation problem in bipartite graphs.

**Remark.** This primal approach for max cardinality matching (by solving the matching augmentation problem) is very similar to finding max $s, t$ flow. Recall that in order to solve max $s, t$ flow, we repeatedly find a flow-augmenting path and use it to augment the flow. Such techniques are called *primal augmentation techniques*.