Fast(ish) Algorithms for Integer Programming: The Lost Lecture of 6.854

Advanced Algorithms Final Project

Logan Weber* loganweb@mit.edu
Josh Pollock* jopo@mit.edu

Abstract

In 2017, Eisenbrand and Weismantel improved a longstanding runtime bound for pseudo-polynomial integer programming algorithms with a fixed number of constraints, m, most notably dropping an exponent of $\mathcal{O}(m^2)$ to an exponent of only $\mathcal{O}(m)$ [2].

Their secret weapon? The Steinitz Lemma, an unassuming statement on the existence of vector walks that don't stray too far from the origin [15]. In this reading paper we contrast Eisenbrand and Weismantel's approach with Papadimitriou's previously-state-of-the-art result from 1981 [14]. We then provide intuition for Jansen and Rohwedder's 2019 improvement to the coefficients of this new algorithm [8]. Along the way, we explore connections between combinatorial optimization and linear algebra.

1 Introduction

Integer linear programs, often referred to as integer programs (IPs) (Section 3), are a natural way to reason about, solve, and approximate many NP-hard problems. Fundamental advancements in our understanding of IPs can thus provide insight into many other interesting domains including scheduling, voting, and graph coloring problems [7, 11, 4].

In just the past five years, researchers have significantly improved runtimes for now-classical state-of-the-art result for IPs [2, 10]. In this reading paper, we contextualize and provide intuition for one such advancement: Eisenbrand and Weismantel's improved pseudo-polynomial algorithm for IPs with a fixed number of constraints. Building on Papadimitriou's generalization of the dynamic programming pseudo-polynomial algorithm from 1981 [14] (Section 4), Eisenbrand and Weismantel dramatically reduce Papadimitriou's search space (Section 5) by reasoning directly about intermediate *outputs* rather than intermediate *inputs* (Section 6). Their secret weapon? The Steinitz Lemma (Section 7), an unassuming statement on the existence of vector walks that don't stray too far from the origin [15, 5]. To use this bound, Eisenbrand and Weismantel must modify Papadimitriou's DP, replacing it with Bellman-Ford (Section 8).

Though this Steinitz-based algorithm is a big improvement over Papadimitriou, we discuss a recent improvement by Jansen and Rohwedder [8] that takes advantage of the shape of Eisenbrand and Weismantel's search space to construct a fuzzy divide-and-conquer strategy, improving the coefficients of the algorithm (Section 9). Finally we ask: can we do better (Section 10)? Unfortunately, this is quite unlikely. Knop et al. recently proved Eisenbrand and Weismantel's approach is probably optimal for general IPs, assuming 3-SAT runs in exponential time [12]. We conclude by looking at the bigger picture and suggesting there is something more fundamental to the structure of fast(ish) algorithms for IPs (Section 11).

*equal contribution

2 Notation

2.1 Norms

The IP algorithms we discuss in this paper search over integer coordinates in Euclidean space. Norms bound these spaces, so we briefly review their important properties.

Definition 1: Norm

A *norm*, $\|\cdot\| : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$, is a function that satisfies the following:

1. **triangle inequality:** $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$
2. **absolute homogeneity:** $\|a\boldsymbol{x}\| = |a| \, \|\boldsymbol{x}\|$ for all $a \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^m$
3. **positive definite:** $\|\boldsymbol{x}\| = 0$ iff $\boldsymbol{x} = \boldsymbol{0}$ for all $\boldsymbol{x} \in \mathbb{R}^m$

Though we often care about the 2-norm, $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i}$, in this paper we are concerned with norms that connect more closely to the computational size of a vector:

- **1-norm $\|\cdot\|_1$:** $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$
- **∞-norm $\|\cdot\|_\infty$:** $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$
- **max-norm for matrices $\|\cdot\|_{\max}$:** $\|A\|_{\max} = \max_{i,j} |a_{i,j}|$

We can combine these norms to bound matrix-vector products, which we leave as an exercise for the reader. Let $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$. Then

$$\|A\boldsymbol{x}\|_\infty \leq \|A\|_{\max} \|\boldsymbol{x}\|_1 \leq n \|A\|_{\max} \|\boldsymbol{x}\|_\infty . \tag{1}$$

2.2 Matrices

Throughout our paper, we find it useful to refer to the column vectors of the constraint matrix A. To keep our notation disjoint from the standard notation A_i for indexing *row* vectors, we adopt the convention of referring to the i^{th} *column* vector of A as $\boldsymbol{a}_i$.

3 IP Generalizes Knapsack

The IPs we will consider in this paper are those in *standard form*:

$$\max \left\{ \boldsymbol{c}^T \boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m \times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n \right\} .$$

One can think of this IP as a generalized version of knapsack. Column i of A represents the list of properties of a particular item (which may be arbitrary integers), say its weight, volume, and price. The coordinate x_i of $\boldsymbol{x}$ represents the number of copies of item i in the knapsack, and we can thus say $\boldsymbol{x}$ represents a *multiset* of items. The vector $\boldsymbol{b}$ represents the requirements on the sums of the properties in the matrix A for all the items in the knapsack. The coordinate c_i of $\boldsymbol{c}$ represents the profit of item i. The goal is thus to maximize the profit of the knapsack subject to a set of constraints and allowing multiple copies of each item.

4 Dynamic Programming: From Knapsack to IP

Knapsack has a well-known pseudo-polynomial algorithm that runs in time $O(nb)$ where n is the number of items and b is the total size of the knapsack [6]. In 1981, equipped with the insight that IP is a generalized form of knapsack, Papadimitriou extended this algorithm to a pseudo-polynomial algorithm for general IPs (with a fixed number of constraints) [14]. Eisenbrand and Weismantel's algorithm builds on the ideas of Papadimitriou, so we first build an intuition for his work.

4.1 Knapsack

Consider the knapsack problem,

$$\max\left\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{a}^T\boldsymbol{x} = b, \boldsymbol{x} \in \{0,1\}^n, \boldsymbol{a} \in \mathbb{N}^n, b \in \mathbb{N}, \boldsymbol{c} \in \mathbb{Z}^n\right\},$$

where c_i and a_i are the profit and size of the i^{th} item, respectively, and b is the **exact** size requirement.[1]

It has the familiar dynamic program (DP) recurrence:

$$T(0, v) = \begin{cases} 0, & v = 0 \\ -\infty, & \text{otherwise} \end{cases}$$
$$T(i+1, v) = \max(T(i, v), c_{i+1} + T(i, v - a_{i+1}))$$

We compute $T(n, b)$ to solve the problem, building a table of size $\mathcal{O}(nb)$. Each entry takes $\mathcal{O}(1)$ time to compute, so we have an $\mathcal{O}(nb)$ runtime.

We now incrementally relax the constraints on knapsack and incrementally generalize the recurrence to obtain an algorithm for a generic IP.

4.2 Multiple Items and Multiple Constraints

We first allow both multiple copies of each item in the knapsack and subject the knapsack items to arbitrarily many constraints:

$$\max\left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{N}^{m\times n}, \boldsymbol{b} \in \mathbb{N}^m, \boldsymbol{c} \in \mathbb{Z}^n\right\}.$$

The DP is nearly unchanged. The biggest difference is that we must search over all possible multiplicities for item i that do not exhaust the "budget" granted by the constraint vector $\boldsymbol{b}$. The value of k chosen at step i gives us x_i^*.

$$T(0, \boldsymbol{v}) = \begin{cases} 0, & \boldsymbol{v} = \boldsymbol{0} \\ -\infty, & \text{otherwise} \end{cases}$$
$$T(i+1, \boldsymbol{v}) = \max_{k \geq 0}(k \cdot c_{i+1} + T(i, \boldsymbol{v} - k \cdot \boldsymbol{a}_{i+1}))$$

We compute $T(n, \boldsymbol{b})$ to solve the problem, building a table of size $\mathcal{O}(n \left\|\boldsymbol{b}\right\|_\infty^m)$. Each entry takes $\mathcal{O}(\left\|\boldsymbol{b}\right\|_\infty)$ time to compute, so we have an $\mathcal{O}(n \left\|\boldsymbol{b}\right\|_\infty^{m+1})$ runtime.

4.3 From Natural to Integral Constraints

We now allow the constraints to be negative:

$$\max\left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n\right\}.$$

Unlike the previous generalization, this one poses a significant problem: if we use the DP in Section 4.2, it does not even appear to terminate! Previously, we relied on the fact that $\boldsymbol{a}_{i+1} \in \mathbb{N}^m \setminus \{\boldsymbol{0}\}$ to give us an implicit upper bound $k \leq \left\|\boldsymbol{v}\right\|_\infty$. But with negative entries in A, k might be arbitrarily large in an optimal solution. To guarantee termination, we need an explicit upper bound K for k. With such a bound, we could use the following DP:

$$T(0, \boldsymbol{v}) = \begin{cases} 0, & \boldsymbol{v} = \boldsymbol{0} \\ -\infty, & \text{otherwise} \end{cases}$$
$$T(i+1, \boldsymbol{v}) = \max_{0 \leq k \leq K}(k \cdot c_{i+1} + T(i, \boldsymbol{v} - k \cdot \boldsymbol{a}_{i+1}))$$

Papadimitriou's key contribution was to provide a simple, yet relatively small, value for K.

[1] Note we can roughly use the same DP to solve the more common knapsack problem $\boldsymbol{a}^T\boldsymbol{x} \leq b$, where we are allowed unused space. We need only change the base case $T(0, v)$ to be 0 when $0 \leq v \leq b$, rather than when $v = 0$.

5 Papadimitriou's DP: As Easy As $\Delta_{A\mathbf{bc}}$ (to the $\mathcal{O}(m^2)$)

Recall that the k chosen at step i gives us x_i^*, so the crux of our problem is bounding $\|\boldsymbol{x}^*\|_\infty$. After obtaining this bound, we will have a DP that terminates, but we would, of course, like to know the runtime. To that end, we additionally bound $\|\boldsymbol{v}\|_\infty$ (the size of intermediate $\boldsymbol{v}$ components) before composing both bounds to arrive at our final runtime.

In this section, we focus on building intuition for the *feasibility* problem, but we give a sketch of how to extend the algorithm to solve the *optimization* problem (details in [14]).

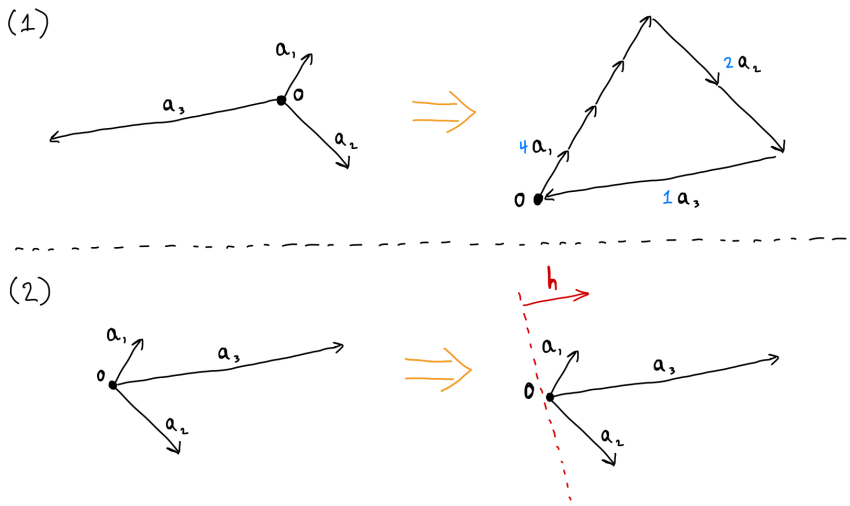
5.1 Bounding $\boldsymbol{x}$

Let $\Delta_{A\boldsymbol{b}} = \max(\|A\|_{\max}, \|\boldsymbol{b}\|_\infty)$, i.e., the largest (in magnitude) numeric value in the input. We present the following variant of Farkas' lemma without proof, though roughly it follows by using Cramer's rule to show that any solution to a matrix-vector equation with an integral nonsingular matrix must have a bounded, rational solution.

Lemma 1: Integral Farkas' Lemma

Let $A \in \mathbb{Z}^{m\times k}$ be an integral matrix such that $\|A\|_{\max} \le \Delta_{A\boldsymbol{b}}$. Then exactly one of the following is true:

1. There exists an $\boldsymbol{x} \in \mathbb{N}^k \setminus \{\mathbf{0}\}$ such that $\|\boldsymbol{x}\|_\infty \le (m\Delta_{A\boldsymbol{b}})^m$ and $A\boldsymbol{x} = \mathbf{0}$.
2. There exists a splitting hyperplane, $\boldsymbol{h} \in \mathbb{R}^m$, such that $\|\boldsymbol{h}\|_\infty \le (m\Delta_{A\boldsymbol{b}})^m$ and $\boldsymbol{h}^T A \ge \mathbf{1}$.



Intuitively, if we interpret the columns of A as force vectors, these vectors can be scaled so their net force is $\mathbf{0}$. Otherwise, if such a scaling does not exist, then there is a hyperplane, $\boldsymbol{h}$, such that all of the force vectors lie on one side of it. We now use this lemma to find a "small" solution to an arbitrary IP.

Consider a standard form IP with constraints $A\boldsymbol{x} = \boldsymbol{b}$, and suppose we have a feasible solution, $\boldsymbol{x}$. If $\|\boldsymbol{x}\|_\infty \le (m\Delta_{A\boldsymbol{b}})^m$ as in Lemma 1, we are done. Otherwise, WLOG reorder the columns of A and the corresponding entries in $\boldsymbol{x}$, such that the first k entries in $\boldsymbol{x}$ are strictly larger than $(m\Delta_{A\boldsymbol{b}})^m$ and the rest are at most $(m\Delta_{A\boldsymbol{b}})^m$. We can thus split the constraints into two pieces where A_u consists of the first k columns:

$$\begin{bmatrix} A_u & A_l \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_u \\ \boldsymbol{x}_l \end{bmatrix} = A_u\boldsymbol{x}_u + A_l\boldsymbol{x}_l = \boldsymbol{b} \tag{2}$$

Using the integral Farkas' lemma on A_u, we get the following two cases:

Case 1: There exists a $\boldsymbol{y}_u \in \mathbb{N}^k \setminus \{\mathbf{0}\}$ such that $\|\boldsymbol{y}_u\|_\infty \le (m\Delta_{A\boldsymbol{b}})^m$ and $A_u\boldsymbol{y}_u = \mathbf{0}$.

Notice $\begin{bmatrix} \boldsymbol{x}_u - \boldsymbol{y}_u \\ \boldsymbol{x}_l \end{bmatrix}$ is a strictly smaller feasible solution since $\|\boldsymbol{x}_u\|_\infty > (m\Delta_{A\boldsymbol{b}})^m$. We can thus replace our feasible $\boldsymbol{x}$, re-sort the constraint columns and entries, and apply Farkas' lemma again.

We continue until we hit the other case of Farkas' lemma.

Case 2: There exists a splitting hyperplane, $\boldsymbol{h} \in \mathbb{R}^m$, such that $\|\boldsymbol{h}\|_\infty \leq (m\Delta_{A\boldsymbol{b}})^m$ and $\boldsymbol{h}^T A_u \geq \mathbf{1}$.

We use this hyperplane to bound $\boldsymbol{x}_u$. If we left-multiply Equation 2 by $\boldsymbol{h}^T$, we find

$$\boldsymbol{h}^T A_u \boldsymbol{x}_u + \boldsymbol{h}^T A_l \boldsymbol{x}_l = \boldsymbol{h}^T \boldsymbol{b}.$$

Using the properties of $\boldsymbol{h}^T$ and a little bit of algebra, we establish

$$\|\boldsymbol{x}_u\|_1 = \mathbf{1}^T \boldsymbol{x}_u \leq \boldsymbol{h}^T A_u \boldsymbol{x}_u = \boldsymbol{h}^T \left(\boldsymbol{b} - A_l \boldsymbol{x}_l\right).$$

We have a bound on each of the terms on the right-hand side involving m and $\Delta_{A\boldsymbol{b}}$. Combining them using Equation 1 twice incurs a factor of n and yields the following theorem:

Theorem 1: [14]

If a standard form IP with constraints $A\boldsymbol{x} = \boldsymbol{b}$ is feasible and bounded then it has a solution $\boldsymbol{x}^* \geq \mathbf{0}$ such that $\|\boldsymbol{x}^*\|_\infty \leq n\mathcal{O}(m\Delta_{A\boldsymbol{b}})^{2m+1}$.

5.2 Bounding $\boldsymbol{v}$

With a bound on $\boldsymbol{x}$, we know the DP in Section 4.3 terminates, but if we want to know its *runtime*, we need to know how many table entries we need to fill. The number of table entries is determined by the number of columns of A **and** by the space of intermediate $\boldsymbol{v}$'s, the latter of which we need a bound for.

We can think of an intermediate $\boldsymbol{v}$ as the value of the constraints of an intermediate solution, $\boldsymbol{x}$. Thus $A\boldsymbol{x} = \boldsymbol{v}$ for some $\boldsymbol{x} \in \mathbb{N}^n$ such that $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x^*}\|_\infty$, where $\boldsymbol{x^*}$ is some feasible solution. We already have a bound on the entries of A, $\|A\|_{\max}$, which we can compose with the bound on $\|\boldsymbol{x}^*\|_\infty$ from Theorem 1 (using Equation 1) to get a bound on $A\boldsymbol{x}$. Succinctly, we have

$$\|\boldsymbol{v}\|_\infty = \|A\boldsymbol{x}\|_\infty \leq n \|A\|_{\max} \|\boldsymbol{x}\|_\infty \leq n \|A\|_{\max} \left(n\mathcal{O}(m\Delta_{A\boldsymbol{b}})^{2m+1}\right) = n^2\mathcal{O}(m\Delta_{A\boldsymbol{b}})^{2m+2}$$

5.3 Papadimitriou's DP

These bounds for $\|\boldsymbol{x}\|_\infty$ and $\|\boldsymbol{v}\|_\infty$ are certainly not pretty, but what's particularly troubling is the factor of n, since we often consider m to be fixed but allow n to vary. The following lemma shows that n is in fact dominated by a function of $\|A\|_{\max}$ and m, which will allow us to simplify our bounds.

Lemma 2: [2]

In a standard form IP, we can safely assume $n = \mathcal{O}(\|A\|_{\max})^m$.

Proof. Consider a single column vector. Each of its entries is in the range $[-\|A\|_{\max}, \|A\|_{\max}]$ and thus can be one of only $2\|A\|_{\max} + 1 = \mathcal{O}(\|A\|_{\max})$ possible integers. Therefore A has at most $\mathcal{O}(\|A\|_{\max})^m$ unique column vectors. A matrix with duplicate columns can be deduplicated by removing repeats but keeping the copy with the highest corresponding objective value. □

Using Lemma 2, we rewrite our bound on $\|\boldsymbol{x}\|_\infty$ to $\mathcal{O}(\|A\|_{\max})^m\mathcal{O}(m\Delta_{A\boldsymbol{b}})^{2m+1} = \mathcal{O}(m\Delta_{A\boldsymbol{b}})^{\mathcal{O}(m)}$ and our bound on $\|\boldsymbol{v}\|_\infty$ to $(\mathcal{O}(\|A\|_{\max})^m)^2\mathcal{O}(m\Delta_{A\boldsymbol{b}})^{2m+2} = \mathcal{O}(m\Delta_{A\boldsymbol{b}})^{\mathcal{O}(m)}$. With a bound on the largest component of $\boldsymbol{v}$ (i.e., $\|\boldsymbol{v}\|_\infty$), there can be at most $(2\|\boldsymbol{v}\|_\infty + 1)^m$ unique values $\boldsymbol{v}$ can take on at each of the $n+1$ steps in the DP. Thus, there are $\mathcal{O}(n(2\|\boldsymbol{v}\|_\infty + 1)^m) = (m\Delta_{A\boldsymbol{b}})^{\mathcal{O}(m^2)}$ table entries, each of which takes $\mathcal{O}(\|\boldsymbol{x}^*\|_\infty) = O(m\Delta_{A\boldsymbol{b}})^{\mathcal{O}(m)}$ time to compute. We now have our first runtime for solving IPs that is pseudo-polynomial with fixed m: $\mathcal{O}\left((m\Delta_{A\boldsymbol{b}})^{\mathcal{O}(m^2)}\right)$.

Optimization. We have given bounds to use for K in our DP, such that we will find a feasible solution if any exists. To extend this algorithm to an optimization setting, Papadimitriou uses an LP relaxation to bound how large the optimal integral **cost** can be. With this bound, he then adds the

constraint $\boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{c}'$ to the IP, where $\boldsymbol{c}'$ is the current guess for the cost. He then binary searches over values of $\boldsymbol{c}'$ within the cost range, using the feasibility DP as a subroutine. The asymptotic bounds remain the same, except we now depend upon $\boldsymbol{c}$, which we express with $\Delta_{A\boldsymbol{b}\boldsymbol{c}} = \max\{\Delta_{A\boldsymbol{b}}, \|\boldsymbol{c}\|_\infty\}$.

Runtime 1: Papadimitriou [14]

The IP $\max\left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n\right\}$ can be solved in time $\mathcal{O}\left((m\Delta_{A\boldsymbol{b}\boldsymbol{c}})^{\mathcal{O}(m^2)}\right)$.

This runtime is better than nothing, but the exponent scales quite poorly with the number of constraints, m. In the next section, we will provide intuition for how Eisenbrand and Weismantel improved this running time, not only dropping the $\mathcal{O}(m^2)$ exponent to $\mathcal{O}(m)$, but also lowering $\Delta_{A\boldsymbol{b}\boldsymbol{c}}$ to $\|A\|_{\max}$, moving the dependence of $\|\boldsymbol{b}\|_\infty$ outside the exponent, and removing the dependence upon $\boldsymbol{c}$ entirely. In the following sections, we let $\Delta = \|A\|_{\max}$.

6 What's Wrong With Papadimitriou?

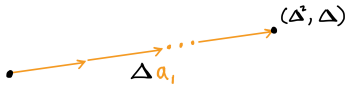
What accounts for the $\mathcal{O}(m^2)$ exponent? One of Papadimitriou's key assumptions was that a good bound on $\boldsymbol{x}$ would translate to a good bound on intermediate $\boldsymbol{v}$'s. However, as we have seen, a bound on $\|\boldsymbol{v}\|_\infty$ still leaves us with a space of size $\mathcal{O}(\|\boldsymbol{v}\|_\infty)^m$ to search. Can we shrink this space?

Let's return to the generalized knapsack interpretation from Section 3. A solution, $\boldsymbol{x}$, corresponds to a *multiset* of vectors. Papadimitriou's DP builds this multiset incrementally, one column at a time. Bounding intermediate $\boldsymbol{v}$'s amounts to bounding the intermediate constraint values of the partially completed knapsack. As we'll see, the order in which one builds the knapsack matters *significantly*. Consider the following IP:

$$\begin{bmatrix}\Delta & -\Delta \\ 1 & 0\end{bmatrix}\mathbf{x} = \begin{bmatrix}0 \\ \Delta\end{bmatrix}$$

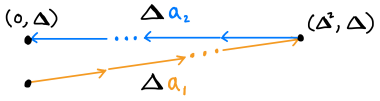
$\mathbf{x} = \begin{bmatrix}\Delta \\ \Delta\end{bmatrix}$ is the smallest (and indeed only) solution to this system. However, the partial sums are quite large! After including Δ copies of $\boldsymbol{a}_1$, we arrive at the intermediate result

$$\begin{bmatrix}\Delta & -\Delta \\ 1 & 0\end{bmatrix}\begin{bmatrix}\Delta \\ 0\end{bmatrix} = \begin{bmatrix}\Delta^2 \\ \Delta\end{bmatrix}$$

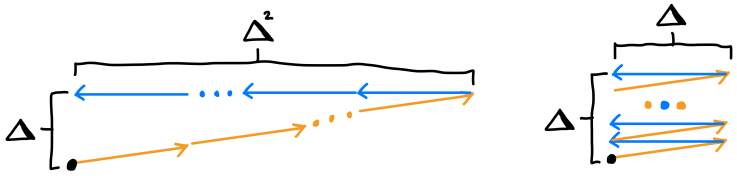


Then we accumulate Δ copies of $\boldsymbol{a}_2$, meeting the constraints:

$$\begin{bmatrix}\Delta \\ \Delta^2\end{bmatrix} + \begin{bmatrix}\Delta & -\Delta \\ 1 & 0\end{bmatrix}\begin{bmatrix}0 \\ \Delta\end{bmatrix} = \begin{bmatrix}0 \\ \Delta\end{bmatrix}$$



Papadimitriou's DP forces us to first choose how many copies of $\boldsymbol{a}_1$ to collect, *then* choose how many copies of $\boldsymbol{a}_2$ to collect, after which, the algorithm is finished. On the other hand, if we are allowed to "revisit" earlier columns, we can just alternate between them, keeping the partial sums small.



Above we saw that if we don't enforce column-order, we might be able to find a sequence of partial sums that doesn't grow too large. But we are left with two unresolved questions:

1. We showed the existence of this ordering for a hand-crafted example, but how do we know such an ordering **always** exists (Section 7)?
2. How do we construct a DP that can search these more general orderings efficiently (Section 8)?

7 Home Is Where the Steinitz Strip Is

Enter the Steinitz Lemma, a statement at the intersection of combinatorial optimization and linear algebra. This lemma guarantees that, provided our given vectors sum to $\mathbf{0}$, we can always order them such that their partial sums never get too big.

Theorem 2: Steinitz Lemma [15, 5]

Let $\|\cdot\|$ be any norm, and let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^m$ such that $\|\boldsymbol{v}_i\| \leq B$ for all i.

Assume

$$\sum_{i=1}^{n} \boldsymbol{v}_i = \mathbf{0},$$

then there exists a permutation $\pi \in S_n$ such that

$$\left\| \sum_{j=1}^{k} \boldsymbol{v}_{\pi(j)} \right\| \leq mB, \quad k = 1, \ldots, n.$$

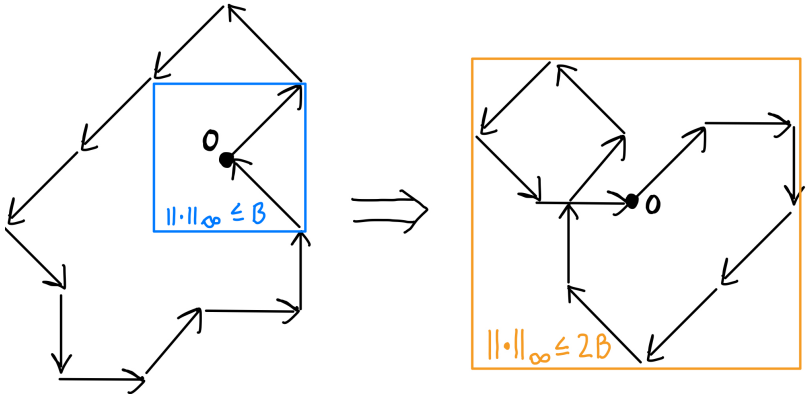


Figure 1: Depiction of the Steinitz lemma, as presented in [2]. The vectors on the left all have bounded norms and sum to 0, so they can be re-ordered to fit within a ball of radius mB (here $m = 2$).

Proof. Our proof follows Sevast'janov's original proof [5] and Lau et al.'s modern presentation [13].

Motivation

Rather than directly considering a permutation of vectors, we instead reason about a collection of nested sets of vectors. A permutation of vectors, π, is in correspondence with a collection of nested sets, $A_1 \subset A_2 \subset \cdots \subset A_n$, where A_k contains the first k vectors of the permutation.

Now let A_k be *any* subset of our given vectors such that $|A_k| = k$. By the triangle inequality, we have the naive bound

$$\left\| \sum_{\boldsymbol{v} \in A_k} \boldsymbol{v} \right\| \leq \sum_{\boldsymbol{v} \in A_k} \|\boldsymbol{v}\| \leq kB,$$

which could be much larger than mB. Luckily, if $k \leq m$, the desired inequality holds regardless of the contents of the set. On the other hand, if $k > m$ the vectors must be linearly dependent, a property we can exploit to lower the bound on these sums:

Suppose we have a non-trivial convex combination of the vectors in A_k. That is, there exist some coefficients $\lambda_{\boldsymbol{v}} \in [0, 1]$, not all 0, such that $\sum_{\boldsymbol{v} \in A_k} \lambda_{\boldsymbol{v}} \boldsymbol{v} = \mathbf{0}$. Then we could improve our bound to

$$\left\| \sum_{\boldsymbol{v} \in A_k} \boldsymbol{v} \right\| = \left\| \sum_{\boldsymbol{v} \in A_k} (1 - \lambda_{\boldsymbol{v}}) \boldsymbol{v} \right\| \leq \sum_{\boldsymbol{v} \in A_k} (1 - \lambda_{\boldsymbol{v}}) \|\boldsymbol{v}\| \leq \left(k - \sum_{\boldsymbol{v} \in A_k} \lambda_{\boldsymbol{v}} \right) B.$$

If we can find $\lambda_{\boldsymbol{v}}$ that also satisfy $\sum_{\boldsymbol{v}\in A_k} \lambda_{\boldsymbol{v}} = k - m$, we will have the desired inequality.

This suggests the following LP to find $\lambda_{\boldsymbol{v}}$ for a given set A_k:

$$\sum_{\boldsymbol{v}\in A_k} \lambda_{\boldsymbol{v}} \boldsymbol{v} = \mathbf{0} \tag{3}$$

$$\sum_{\boldsymbol{v}\in A_k} \lambda_{\boldsymbol{v}} = k - m \tag{4}$$

$$0 \leq \lambda_{\boldsymbol{v}} \leq 1 \quad \forall \boldsymbol{v} \in A_k \tag{5}$$

Our goal is thus to produce a nested collection of sets where each set with more than m elements has a corresponding feasible LP. This will allow us to bound every partial sum by mB. We proceed by induction starting with $k = n$, i.e., the entire set of vectors, and ending with $k = m + 1$, since the property is straightforward for the first m vectors, and they can be ordered arbitrarily.

Iterative LP Algorithm

Base Case $k = n$: We start the induction with our assumption on the sum of all the vectors, namely $\sum_{\boldsymbol{v}\in A_n} \boldsymbol{v} = \mathbf{0}$. Let $\lambda_{\boldsymbol{v}} = \dfrac{n-m}{n}$. Constraint families 4 and 5 are clearly satisfied. Constraint family 3 holds by our assumption.

Inductive Step $k+1 \implies k$:

Assume there exists a set of vectors, A_{k+1} and a corresponding set of coefficients, $\lambda_{\boldsymbol{v}}^{(k+1)}$, satisfying the following LP:

$$\sum_{\boldsymbol{v}\in A_{k+1}} \lambda_{\boldsymbol{v}}^{(k+1)} \boldsymbol{v} = \mathbf{0}$$

$$\sum_{\boldsymbol{v}\in A_{k+1}} \lambda_{\boldsymbol{v}}^{(k+1)} = k + 1 - m$$

$$0 \leq \lambda_{\boldsymbol{v}}^{(k+1)} \leq 1 \quad \forall \boldsymbol{v} \in A_{k+1}$$

We wish to construct a set $A_k \subset A_{k+1}$ and corresponding coefficients $\lambda_{\boldsymbol{v}}^{(k)}$ that satisfy the LP when $k+1$ is replaced by k. We do so in two phases.

Scale $\lambda_{\boldsymbol{v}}^{(k+1)}$. First, in the LP above, shrink $k + 1 - m$ to $k - m$. By scaling $\lambda_{\boldsymbol{v}}^{(k+1)}$ to $\frac{k-m}{k+1-m}\lambda_{\boldsymbol{v}}^{(k+1)}$, we obtain a feasible solution.

Basic Feasible Solution. Now comes the key insight of this inductive scheme. By reasoning about tight constraints, we will argue at least one of the $\lambda_{\boldsymbol{v}}^{(k+1)}$ is 0 at a basic feasible solution. We can thus throw out this variable and its corresponding vector to produce the set A_k.

Consider a basic feasible solution. There are m equalities in family 3, one for each coordinate, and one equality in family 4. There are $k + 1$ variables. Thus at least $(k + 1) - (m + 1) = k - m$ of the inequalities must be tight. But at most $k - m$ of these inequalities can be tight at 1, since they must sum to $k - m$. If exactly $k - m$ are tight at 1, the rest of the coefficients must be 0, so WLOG we pick one of the corresponding vectors to remove. If fewer than $k - m$ are tight at 1, then at least one is tight at 0, so we again select a vector to remove WLOG. This yields the desired set A_k.

We have thus inductively constructed a collection of nested sets satisfying the LP and thus a permutation satisfying the desired bound.

□

We now use the Steinitz Lemma to assert there is always a small sequence of partial sums for any feasible point of an IP. Our proofs are adapted from Jansen and Rohwedder's re-presentation of Eisenbrand and Weismantel's results [8].

We first consider the special case when $\boldsymbol{b} = \mathbf{0}$, ignoring the trivial solution $\boldsymbol{x} = \mathbf{0}$ since we will ultimately generalize this idea.

Lemma 3

Let $\max\left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{0}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{c} \in \mathbb{Z}^n\right\}$ be bounded and feasible. For any feasible solution, $\boldsymbol{x}$, there exists a sequence of partial sums, $A\boldsymbol{x}_1, \ldots, A\boldsymbol{x}_{\|\boldsymbol{x}\|_1 - 1}, A\boldsymbol{x}$, such that every sum, $A\boldsymbol{x}_i$ lies in the set $\{\boldsymbol{y} \colon \|\boldsymbol{y}\|_\infty \leq m\Delta\}$. This region contains at most $(2m\Delta + 1)^m = O(m\Delta)^m$ integral points.

Proof. Let $\boldsymbol{x}$ be any feasible point of the IP. Since $A\boldsymbol{x} = \boldsymbol{0}$ we can use the Steinitz Lemma to produce a sequence of column vectors, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\|\boldsymbol{x}\|_1}$, where column i appears x_i times.

This sequence has small partial sums, in particular since $\|\boldsymbol{v}_i\|_\infty \leq \Delta$ by the definition of Δ,

$$\left\|\sum_{j=1}^{k} \boldsymbol{v}_j\right\|_\infty \leq m\Delta, \quad k = 1, \ldots, \|\boldsymbol{x}\|_1 .$$

Every partial sum is thus contained inside a ball with $(2m\Delta + 1)^m$ integral points, since there are $2m\Delta + 1$ integral points in the interval $[-m\Delta, m\Delta]$. □

We now discuss how to generalize this result to arbitrary $\boldsymbol{b}$.

7.1 Arbitrary b

With the normal Steinitz Lemma, we can think of the partial sums as staying close to the origin. We will now think of them as marching towards $\boldsymbol{b}$. If there are n vectors, we can think of each one of them as getting us to roughly $\dfrac{\boldsymbol{b}}{n}$. We generalize the Steinitz Lemma to formalize this intuition.

Theorem 3: Generalized Steinitz Lemma [2, 5]

Let $\|\cdot\|$ be any norm, and let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^m$ such that $\|\boldsymbol{v}_i\| \leq B$ for all i.

Assume

$$\sum_{i=1}^{n} \boldsymbol{v}_i = \boldsymbol{b},$$

then there exists a permutation $\pi \in S_n$ such that

$$\left\|\sum_{j=1}^{k} \boldsymbol{v}_{\pi(j)} - \left(\frac{k}{n}\right)\boldsymbol{b}\right\| \leq m\left(B + \frac{\|\boldsymbol{b}\|}{n}\right), \quad k = 1, \ldots, n.$$

Proof. Translate each of the n vectors $\boldsymbol{v}_i$ to $\boldsymbol{v}_i - \dfrac{\boldsymbol{b}}{n}$. This translates the sum to $\sum_i \left(\boldsymbol{v}_i - \dfrac{\boldsymbol{b}}{n}\right) = \boldsymbol{0}$. By the triangle inequality, $\left\|\boldsymbol{v}_i - \dfrac{\boldsymbol{b}}{n}\right\| \leq \left(B + \dfrac{\|\boldsymbol{b}\|}{n}\right)$. Thus by the Steinitz Lemma,

$$\left\|\sum_{j=1}^{k} \left(\boldsymbol{v}_{\pi(j)} - \frac{\boldsymbol{b}}{n}\right)\right\| \leq m\left(B + \frac{\|\boldsymbol{b}\|}{n}\right), \quad k = 1, \ldots, n.$$

The result follows. □

We take this opportunity to name the path produced by the Generalized Steinitz Lemma:

Definition 2: Steinitz-Ordered Path

Let $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. A *Steinitz-ordered path* of a solution $\boldsymbol{x} \in \mathbb{N}^n$ to the equation $A\boldsymbol{x} = \boldsymbol{b}$ is a path in $\mathbb{R}^m$ connecting the partial sums given by the Generalized Steinitz Lemma.

We are now ready to generalize Lemma 3 to arbitrary $\boldsymbol{b}$.

Theorem 4

Let $\max\left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n\right\}$ be bounded and feasible. For any feasible solution, $\boldsymbol{x}$, there exists a Steinitz-ordered path contained in the set $\{\boldsymbol{y} : \exists\, \gamma \in [0,1], \|\boldsymbol{y} - \gamma\boldsymbol{b}\|_\infty \leq 2m\Delta\}$. This region contains at most $(4m\Delta + 2)^m(\|\boldsymbol{b}\|_\infty + 1) = O(m\Delta)^m(\|\boldsymbol{b}\|_\infty + 1)$ integral points.

Proof. Let $\boldsymbol{a}_i$ be a column of A. By properties of norms,

$$\begin{aligned}\left\|\boldsymbol{a}_i - \tfrac{\boldsymbol{b}}{\|\boldsymbol{x}\|_1}\right\|_\infty &\leq \|\boldsymbol{a}_i\|_\infty + \tfrac{\|\boldsymbol{b}\|_\infty}{\|\boldsymbol{x}\|_1} \\ &\leq \Delta + \|A\|_{\max} && \text{(Equation 1)} \\ &= 2\Delta.\end{aligned}$$

Thus, by the Generalized Steinitz Lemma, for any feasible $\boldsymbol{x}$ we have a Steinitz-ordered path, $\{\boldsymbol{v}_{\pi(i)}\}$, such that

$$\left\|\sum_{i=1}^{k} \boldsymbol{v}_{\pi(i)} - (k/\|\boldsymbol{x}\|_1)\,\boldsymbol{b}\right\|_\infty \leq 2m\Delta, \quad k \in \{1, \ldots, \|\boldsymbol{x}\|_1\} \tag{6}$$

The vertices of the path are thus within $2m\Delta$ of the set $\left\{\left(\frac{1}{\|\boldsymbol{x}\|_1}\right)\boldsymbol{b}, \left(\frac{2}{\|\boldsymbol{x}\|_1}\right)\boldsymbol{b}, \ldots, \boldsymbol{b}\right\}$. If we use this region directly to bound our search space, that bound will depend on $\boldsymbol{x}$. We instead increase the region slightly to the set of vectors $\boldsymbol{v}$ such that, for some $\gamma \in [0,1]$,

$$\|\boldsymbol{v} - \gamma\boldsymbol{b}\|_\infty \leq 2m\Delta.$$

This removes $\boldsymbol{x}$ from the description. Now consider the discrete points $\{\boldsymbol{b} \cdot {}^{i}/_{\|\boldsymbol{b}\|_\infty} : i = 0, \ldots \|\boldsymbol{b}\|_\infty\}$. Notice, for any j,

$$\left\|\boldsymbol{b} \cdot \frac{j+1}{\|\boldsymbol{b}\|_\infty} - \boldsymbol{b} \cdot \frac{j}{\|\boldsymbol{b}\|_\infty}\right\|_\infty = \left\|\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|_\infty}\right\|_\infty = 1,$$

so the points are close. In particular, this implies that for any $\gamma\boldsymbol{b}$ with $\gamma \in [0,1]$, there is a discrete point at most $1/2$ a unit away. Thus, for some j,

$$\left\|\boldsymbol{v} - \boldsymbol{b} \cdot \frac{j}{\|\boldsymbol{b}\|_\infty}\right\|_\infty \leq \|\boldsymbol{v} - \gamma\boldsymbol{b}\|_\infty + \left\|\gamma\boldsymbol{b} - \boldsymbol{b} \cdot \frac{j}{\|\boldsymbol{b}\|_\infty}\right\|_\infty \leq 2m\Delta + \tfrac{1}{2}.$$

This means we can search $\|\boldsymbol{b}\|_\infty + 1$ balls of size $4m\Delta + 2$ in the infinity norm. The size of this set is $(4m\Delta + 2)^m(\|\boldsymbol{b}\|_\infty + 1)$.

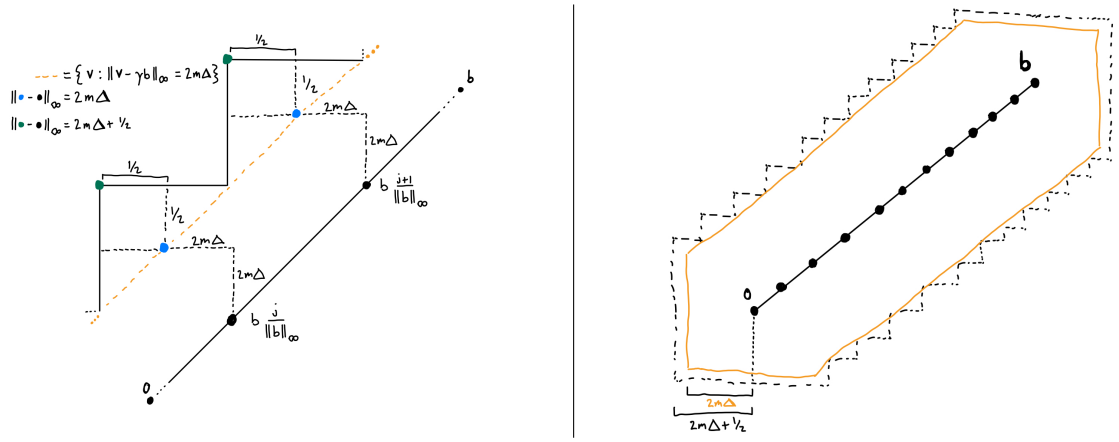


Figure 2: **(left)** Summary of proof above **(right)** The yellow bounding region which is the goal of the theorem and the dotted-black overapproximation used to prove it.

□

We will study the region introduced in Theorem 4 in the following sections, so we give it a name:

Definition 3: Steinitz Strip

The *Steinitz strip* for the constraint $A\boldsymbol{x} = \boldsymbol{b}$ is the set $\{\boldsymbol{y} : \exists \gamma \in [0,1], \|\boldsymbol{y} - \gamma \boldsymbol{b}\|_\infty \leq 2m\Delta\}$ (i.e., the region in Figure 2). It contains a Steinitz-ordered path for every feasible $\boldsymbol{x}$.

Our first application of the Steinitz strip is to bound the size of an optimal solution to an IP.

Corollary 1: [8]

Let $\max\left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n\right\}$ be bounded and feasible. There exists an optimal solution $\boldsymbol{x}^*$ such that $\|\boldsymbol{x}^*\|_1 \leq \mathcal{O}(m\Delta)^m(\|\boldsymbol{b}\|_\infty + 1)$.

Proof. Consider an optimal solution $\boldsymbol{x}^*$. By Theorem 4, it has a Steinitz-ordered path inside the Steinitz strip, which has at most $\mathcal{O}(m\Delta)^m(\|\boldsymbol{b}\|_\infty + 1)$ distinct points. The length of this path is $\|\boldsymbol{x}^*\|_1$, so if we can ensure the path visits each point in the strip at most once, we will have proved the corollary. In general we cannot guarantee this, but we can *reduce* any solution to one that visits unique points.

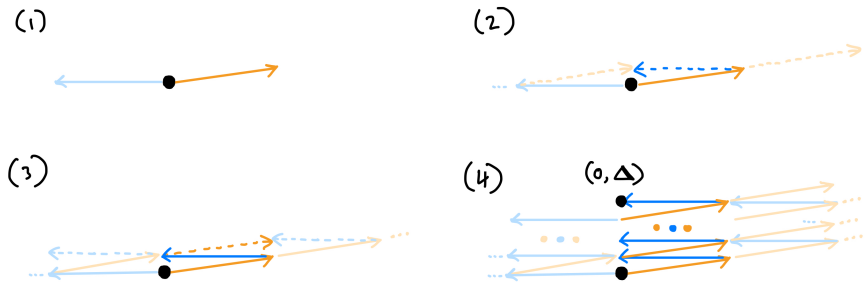
If $\boldsymbol{x}^*$ visits a point more than once there is a cycle in the path, i.e., there is some subpath of the Steinitz-ordered path $\{\boldsymbol{v}_{\pi(j)}\}$ such that $\sum_{j=k_1}^{k_2} \boldsymbol{v}_{\pi(j)} = \boldsymbol{0}$. We claim we can remove this cycle while maintaining optimality. If the cycle had positive cost, then we could traverse the cycle again, violating the optimality of $\boldsymbol{x}^*$. Similarly, if the cycle had negative cost, then we could remove the cycle, again violating optimality. Thus the cycle must have 0 cost, and so removing it produces another optimal solution. After removing all 0-cost cycles, we obtain a new optimal solution $\|\boldsymbol{y}^*\|$ that traverses every point at most once. Thus $\|\boldsymbol{y}^*\| \leq \mathcal{O}(m\Delta)^m(\|\boldsymbol{b}\|_\infty + 1)$. □

8 Eisenbrand's BF

In the previous section we showed there is always a Steinitz-ordered path from $\boldsymbol{0}$ to $\boldsymbol{b}$ within the Steinitz strip. However, recall Papadimitriou's DP forced us to pick clumps of vectors at a time, so despite its large search space, it might not even discover this path!

Recall a Steinitz-ordered path is a path in $\mathbb{R}^m$ starting at $\boldsymbol{0}$. In fact, since our constraints are integral, it's a path in $\mathbb{Z}^m$ starting at $\boldsymbol{0}$. What is the structure containing *all* such paths? It has vertices, which are points reachable from $\boldsymbol{0}$ via integer-linear combinations of column vectors, $\boldsymbol{a}_i$, and that lie within the Steinitz strip. These vertices are connected by edges, where for each vertex $\boldsymbol{u}$, we have an edge to another vertex $\boldsymbol{v}$ iff $\boldsymbol{u} + \boldsymbol{a}_i = \boldsymbol{v}$ for some i (that is, iff $\boldsymbol{v}$ is reachable from $\boldsymbol{u}$ using only a single column vector). These properties describe a graph structure embedded in $\mathbb{R}^m$!

Consider the example in Section 6. If we run breadth-first search (BFS) over its implicit graph (dotted edges denote the frontier), we see how the space of feasible solutions begins to unfold, and how we find the Steinitz-ordered path (denoted by the bolded vectors).



A BFS will give us a *feasible* solution, if any exists, but in order to find an *optimal* solution we need to consider the weights on these edges. Each edge corresponds to adding a column vector $\boldsymbol{a}_i$ to our solution. With an objective, $\boldsymbol{c}^T\boldsymbol{x}$, adding this column vector adds c_i to the objective value. Thus the weight of an edge between two vertices $\boldsymbol{u}$ and $\boldsymbol{v}$ is c_i when $\boldsymbol{v} + \boldsymbol{a}_i = \boldsymbol{u}$.

We can thus run the Bellman-Ford algorithm to compute the longest path through this graph in order to find an optimal solution!

What is the size of our graph? As stated earlier, the number of integral points, $|V|$, in the Steinitz strip is at most $\mathcal{O}(m\Delta)^m(\|\boldsymbol{b}\|_\infty + 1)$, and the number of edges, $|E|$, is at most $n|V|$, because each vertex has at most n directions (column vectors) to choose from. Substituting into the Bellman-Ford runtime gives $\mathcal{O}(|V| \cdot |E|) = \mathcal{O}(n|V|^2) = \mathcal{O}(n((m\Delta)^m(\|\boldsymbol{b}\|_\infty + 1))^2) = \mathcal{O}(n(m\Delta)^{2m}\|\boldsymbol{b}\|_\infty^2)$.

Recall in Lemma 2 we showed $n = \mathcal{O}(\Delta)^m$, allowing us to rewrite to our final bound.

Runtime 2: Eisenbrand and Weismantel [2]

The IP $\max\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n\}$ can be solved in time $\mathcal{O}((m\Delta)^{\mathcal{O}(m)}\|\boldsymbol{b}\|_\infty^2)$.

Compared to Papadimitriou's DP runtime of $\mathcal{O}\left((m\Delta_{A\boldsymbol{b}})^{\mathcal{O}(m^2)}\right)$, we have dropped the exponent from quadratic to linear **and** we have factored our dependence on $\|\boldsymbol{b}\|_\infty$ out of the exponential term.

By reasoning locally about the graph induced by column vectors $\boldsymbol{a}_i$, we were able to tap into an existing graph algorithm, but if we take a more global view on the search space (i.e., the Steinitz strip), a little guesswork allows us to formulate a divide-and-conquer procedure, which we walk through in the next section.

9 Fuzzy Divide and Conquer for Faster Search

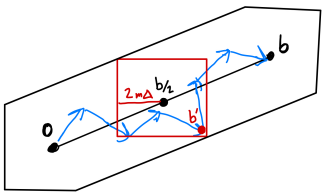
Though Eisenbrand and Weismantel's algorithm runs much faster than Papadimitriou's, we can still do better. They considered a much more general search space than Papadimitriou, but they were able to bound the search space more effectively, because its structure "unlocked" the Steinitz lemma. However, the Steinitz strip has a nice structure that Eisenbrand and Weismantel don't take full advantage of. It's a narrow strip around the line segment from $\mathbf{0}$ to $\boldsymbol{b}$, so we expect that half the path should land somewhere near the midpoint $\boldsymbol{b}/2$.

In an ideal world, we'd know the length of the path, $\|\boldsymbol{x}\|_1$, and its halfway point, $\boldsymbol{b'}$. Using these, we could then search for paths of length $\frac{\|\boldsymbol{x}\|_1}{2}$ from $\mathbf{0}$ to $\boldsymbol{b'}$ and from $\boldsymbol{b'}$ to $\boldsymbol{b}$. (Note if we translate the second path to the origin, it becomes a path from $\mathbf{0}$ to $\boldsymbol{b} - \boldsymbol{b'}$.) This would give us two half-sized problems! Alas, our world is far from ideal, but it is close enough that we still get a speedup.

Determining $\|\boldsymbol{x}\|_1$. By Corollary 1, there is an optimum $\boldsymbol{x}^*$ such that $\|\boldsymbol{x}^*\|_1 \le O(m\Delta)^m(\|\boldsymbol{b}\|_\infty + 1)$. By adding a $\mathbf{0}$ column to the constraint matrix and a corresponding 0 entry to $\boldsymbol{c}$, we can pad this solution so that the inequality is tight. We thus know the exact size of the solution we're looking for!

Guessing the Halfway Point. Now that we know the exact size of the solution we're searching for, we can use Equation 6 (a consequence of the Generalized Steinitz Lemma) with $k = \|\boldsymbol{x}\|_1/2$ to obtain a region containing the halfway point of the path:

$$\left\| \sum_{j=1}^{\|\boldsymbol{x}\|_1/2} \boldsymbol{v}^{(j)} - \frac{\boldsymbol{b}}{2} \right\|_\infty \le 2m\Delta.$$



Thus if we guess every value in the ball with radius $2m\Delta$ around $\frac{\boldsymbol{b}}{2}$, one of our guesses will be the halfway point.

Guessing the Quarterway Point. Suppose we have a guess, $\boldsymbol{b'}$, for the halfway point. How do we guess a quarterway point? By the same logic as the halfway point, we have

$$\left\| \sum_{j=1}^{\|\boldsymbol{x}\|_1/4} \boldsymbol{v}^{(j)} - \frac{\boldsymbol{b'}}{2} \right\|_\infty \leq 2m\Delta.$$

Unfortunately, since this bound is in terms of $\boldsymbol{b'}$ and not $\boldsymbol{b}$, it's difficult to determine how many entries we'll need for the entire DP. We thus relate the quarterway point back to $\frac{\boldsymbol{b}}{4}$ via $\frac{\boldsymbol{b'}}{2}$:

$$\begin{aligned}
\left\| \boldsymbol{b''} - \frac{\boldsymbol{b}}{4} \right\|_\infty &\leq \left\| \boldsymbol{b''} - \frac{\boldsymbol{b'}}{2} \right\|_\infty + \left\| \frac{\boldsymbol{b'}}{2} - \frac{\boldsymbol{b}}{4} \right\|_\infty && \text{(triangle inequality)} \\
&\leq 2m\Delta + \frac{1}{2} \left\| \boldsymbol{b'} - \frac{\boldsymbol{b}}{2} \right\|_\infty \\
&\leq 2m\Delta + \frac{1}{2}(2m\Delta) && \text{(halfway point bound)} \\
&= 3m\Delta.
\end{aligned}$$

Guessing the 2^{-i}-way Point. In general, we have the recurrence

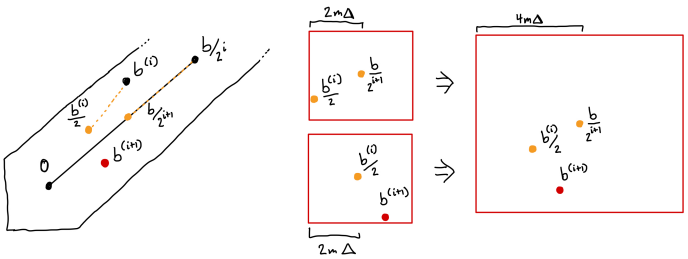
$$\left\| \boldsymbol{b}^{(i+1)} - \frac{\boldsymbol{b}}{2^{i+1}} \right\|_\infty \leq \left\| \boldsymbol{b}^{(i+1)} - \frac{\boldsymbol{b}^{(i)}}{2} \right\|_\infty + \left\| \frac{\boldsymbol{b}^{(i)}}{2} - \frac{\boldsymbol{b}}{2^{i+1}} \right\|_\infty \leq 2m\Delta + \frac{1}{2} \left\| \boldsymbol{b}^{(i)} - \frac{\boldsymbol{b}}{2^i} \right\|_\infty$$

and

$$\left\| \boldsymbol{b'} - \frac{\boldsymbol{b}}{2} \right\|_\infty \leq 2m\Delta.$$

We approximate these bounds from above by using the fixpoint: $B = 2m\Delta + \frac{1}{2}B$. Thus $B = 4m\Delta$ and we have

$$\left\| \boldsymbol{b}^{(i)} - \frac{\boldsymbol{b}}{2^i} \right\|_\infty \leq 4m\Delta.$$



This is only a factor of two worse than for $\boldsymbol{b'}$!

Defining the Recurrence.

Let $\|x\|_1 = 2^K$ where $K = \mathcal{O}(m \log(m\Delta) + \log(\|\boldsymbol{b}\|_\infty))$. We are now ready to formally define the recurrence:

$$T(0, \boldsymbol{v}) = \begin{cases} c_j, & \text{The } j^{\text{th}} \text{ column of } A \text{ is } \boldsymbol{v}. \\ -\infty, & \text{otherwise} \end{cases}$$

$$T(i, \boldsymbol{v}) = \max \left\{ T(i-1, \boldsymbol{v'}) + T(i-1, \boldsymbol{v} - \boldsymbol{v'}) \colon \left\| \boldsymbol{v'} - \frac{\boldsymbol{b}}{2^{K-i+1}} \right\|_\infty \leq 4m\Delta \right\}$$

We compute $T(K, \boldsymbol{b})$ to solve the original IP. There are $K+1$ levels of tables, each indexed by i. Each level will store the entries for every $\boldsymbol{v'}$ in its corresponding ball. There are $(8m\Delta + 1)^m = \mathcal{O}(m\Delta)^m$ entries. Similarly, the max must be computed over $\mathcal{O}(m\Delta)^m$ terms, so computing each entry in the table costs $\mathcal{O}(m\Delta)^m$. This yields an overall runtime of $\mathcal{O}(m\Delta)^{2m}(K+1) = \mathcal{O}(m\Delta)^{2m}(m \log(m\Delta) + \log(\|\boldsymbol{b}\|_\infty)) = \mathcal{O}(m\Delta)^{2m}(\log(\Delta) + \log(\|\boldsymbol{b}\|_\infty)$ since the m terms are absorbed by the first part, which is exponential in m.

Runtime 3: Jansen and Rohwedder (Naïve) [8]

The IP $\max\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n\}$ can be solved in time $\mathcal{O}(m\Delta)^{2m}(\log\Delta + \log\|\boldsymbol{b}\|_\infty)$.

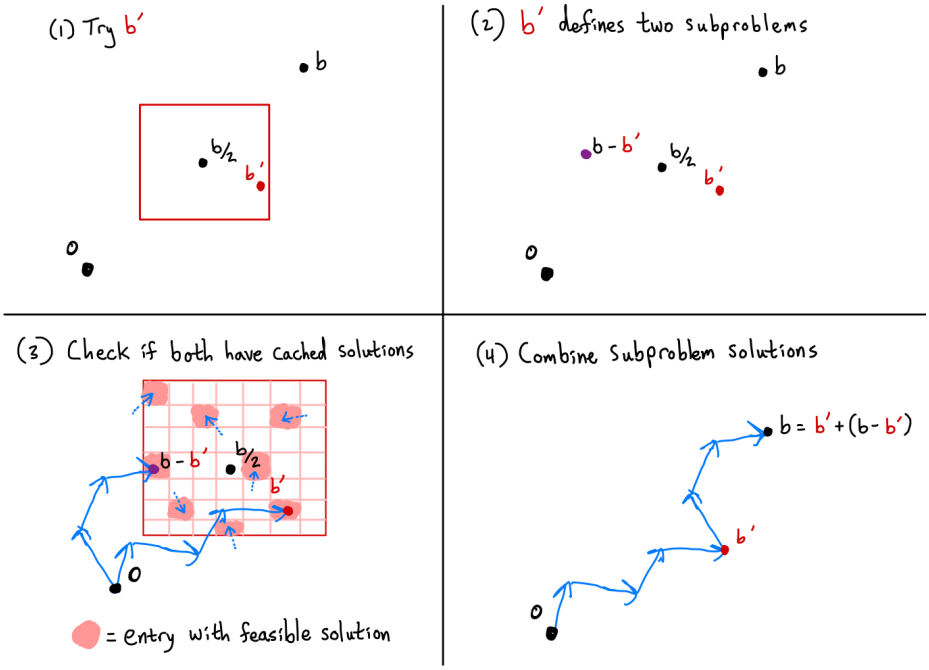


Figure 3: **Visual summary of the final level of the DP**. With all steps $i \le K$ computed, the problem $A\boldsymbol{x} = \boldsymbol{b}$ remains, and we first guess a $\boldsymbol{b}'$ near $\boldsymbol{b}/2$. This $\boldsymbol{b}'$ defines the subproblems $A\boldsymbol{x} = \boldsymbol{b}'$ and $A\boldsymbol{x} = \boldsymbol{b} - \boldsymbol{b}'$. If they have solutions $\boldsymbol{x}_1^*, \boldsymbol{x}_2^*$, we are guaranteed they have $\|\boldsymbol{x}_1^*\|_1, \|\boldsymbol{x}_2^*\|_1 = 2^{K-1}$, meaning they will be cached in the DP table at level $K-1$. We can then string the subproblem solutions together to arrive at a solution to $A\boldsymbol{x} = \boldsymbol{b}$. Beyond feasibility, to optimize for cost, we simply take the max-cost solution over **all** choices of $\boldsymbol{b}'$ near $\boldsymbol{b}/2$.

Compared to Eisenbrand's runtime of $\mathcal{O}((m\Delta)^m\|\boldsymbol{b}\|_\infty^2)$, we have drastically improved our dependence upon $\|\boldsymbol{b}\|_\infty$, going from quadratic to **logarithmic**, while incurring a new $\log(\Delta)$ multiplicative overhead. By improving the speed of the max computation and recognizing that a slightly weaker statement than Steinitz is necessary (we only need to reason about the halfway point, not the entire path), Jansen and Rohwedder have reduced this runtime even further.

Runtime 4: Jansen and Rohwedder [9]

The IP $\max\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{N}^n, A \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n\}$ can be solved in time $\mathcal{O}(\sqrt{m}\Delta)^{2m}\log\|\boldsymbol{b}\|_\infty$.

Exhausted by the breadth of techniques we've covered, we are nevertheless compelled by our eternal quest for optimality. Ready to collapse, the words still find their way past our lips: Can we do better?

10 Can We Do Better?

10.1 Probably Not.

In parallel with algorithmic advancements for IPs, Fomin et al. [3] and Knop et al. [12] proved conditional lower bounds on IP algorithms with similar structures to the ones we've discussed. In particular, they showed that fast IP algorithms that running in pseudo-polynomial time for a fixed

number of constraints would disprove the Exponential Time Hypothesis (ETH), which conjectures that 3-SAT runs in exponential time [1]. More formally, ETH hypothesizes that a 3-SAT instance with n variables and m clauses does not run in time $2^{o(n+m)}$. To provide some intuition for these results, we sketch their proofs.

Reducing 3-SAT to IP.
At the heart of Fomin et al.'s result is an encoding of a 3-SAT formula as an IP. Suppose we have a 3-SAT instance with n variables and m clauses.

Variables. For each 3-SAT variable, x_i, we define two IP variables, x_i and $\overline{x_i}$. We wish to have exactly one of these variables be 1 and the other 0, denoting true and false, respectively, so we add the constraint $x_i + \overline{x_i} = 1$. Since the IP is in standard form, the variables are already integral and non-negative, so the constraint suffices. This requires $2n$ variables and n constraints.

Clauses. For each clause, $C_i = x \vee y \vee z$, we want to constrain the sum of the component variables such that $1 \leq x + y + z \leq 3$. This ensures at least one of the variables is true. To do so, we introduce two slack variables per clause, Y_i and Z_i, with corresponding constraints, $x + y + z + Y_i = 3$ and $Y_i + Z_i = 2$. The second equation ensures $Y_i \leq 2$, so together these constraints encode $1 \leq x+y+z \leq 3$. This requires $2m$ additional variables and $2m$ additional constraints.

Our IP instance totals $2n + 2m$ variables and $n + 2m$ constraints. Notice also that $\|A\|_{\max} = 1$ and $\|\boldsymbol{b}\|_\infty = 3$, which are constant in the size of the 3-SAT instance. Thus if we can solve an IP with k constraints and small values in $2^{o(k)}|I|^{\mathcal{O}(1)}$ time, where $|I|$ is the total bitsize of the input, we can solve 3-SAT in $2^{o(n+2m)} = 2^{o(n+m)}$ time, contradicting ETH. This bound is good, but not great. Both Eisenbrand and Weismantel's and Jansen and Rohwedder's algorithms run in $\mathcal{O}(k^k)|I|^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)}|I|^{\mathcal{O}(1)}$ time on these encodings.

Shrinking the IP.
Knop et al. strengthened Fomin et al.'s claim. They showed that an IP algorithm running in $2^{o(k \log k)}$ time when A is a matrix of 0's and 1's, $\boldsymbol{b}$'s entries are non-negative, and the number of columns and the size of $\boldsymbol{b}$ are both $O(k \log k)$ is sufficient to disprove ETH. To improve the bound, Knop et al. take Fomin et al.'s original 3-SAT encoding and shrink it to one with just $\mathcal{O}((n+m)/\log(n+m))$ constraints. Thanks to this reduction, an IP algorithm with a runtime of $2^{o(k \log k)}|I|^{\mathcal{O}(1)}$ would suffice. This means any algorithm that runs significantly faster Eisenbrand and Weismantel's would violate ETH, which would be a very surprising result.

11 Conclusion

Having seen three different algorithms for solving IPs, which follow quite logically from their predecessors, we take a step back to ponder *why* these new algorithms are so much better. By definition, IPs sit at the intersection of combinatorial optimization and linear algebra, since they require discrete solutions to otherwise-linear problems. At the core of Papadimitriou's algorithm lies a straightforward integral variant of Farkas' lemma. This statement fits largely on the linear algebra side, mostly blind to the combinatorial structure of the problem. On the other hand, Eisenbrand and Weismantel's and Jansen and Rohwedder's algorithms revolve around the Steinitz Lemma, whose proof is a discrete optimization algorithm on nested linear programs. By leveraging the essence of IPs, these latter algorithms achieve runtimes that are probably optimal. Though the Steinitz lemma is the engine behind the algorithms in this paper, the larger takeaway is that the synthesis of ideas from linear algebra and combinatorics may be the key to efficient IP algorithms.

References

[1] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[2] Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. *ACM Trans. Algorithms*, 16(1), November 2019.

[3] Fedor V. Fomin, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. On the optimality of pseudo-polynomial algorithms for integer programming, 2018.

[4] Tomáš Gavenčiak, Martin Kouteckỳ, and Dušan Knop. Integer programming in parameterized complexity: Five miniatures. *Discrete Optimization*, page 100596, 2020.

[5] V. S. Grinberg and S. V. Sevast'yanov. Value of the steinitz constant. *Functional Analysis and Its Applications*, 14(2):125–126, 1980.

[6] Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, October 1975.

[7] Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. Empowering the configuration-ip − new ptas results for scheduling with setups times, 2018.

[8] Klaus Jansen and Lars Rohwedder. On Integer Programming and Convolution. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[9] Klaus Jansen and Lars Rohwedder. On integer programming, discrepancy, and convolution, 2019.

[10] Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Mathematical Programming*, 184(1-2):1–34, November 2019.

[11] Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. *ACM Trans. Econ. Comput.*, 8(3), June 2020.

[12] Dušan Knop, Michał Pilipczuk, and Marcin Wrochna. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Trans. Comput. Theory*, 12(3), June 2020.

[13] Lap Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2011.

[14] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, October 1981.

[15] E. Steinitz. Bedingt konvergente reihen und konvexe systeme. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1913:128 – 176.