GRAVER BASES

Francisco Javier Blázquez Martínez

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Mathematics section,

Chair of discrete optimization

Project director: Dr. Friedrich Eisenbrand

Project supervisor: Jana Cslovjecsek

January 2021

Acknowledgements

First and foremost I would like to express my sincere thanks to Jana Cslovjecsek for helping me throughout the whole project as well as to Professor Friedrich Eisenbrand for giving me the oportunity of doing this. It has been a very didactic experience that I really appreciate.

Moreover, this project is one of my finals steps for obtaining the Double degree in Mathematics and Computer Engineering at the Complutense University of Madrid that, thanks to the help of many people, I have been able to carry out in this fantastic university that is the École Politechnique Fédéreale de Lausanne. I want to expressly thank Professor Katzalin Olcoz and Professor Daniel Chaver for all the facilities and help. I can't help but be grateful to the Complutense University and all the teachers I've had for everything I have learnt.

Finally, I would like to thank Luis Felipe Ramirez for advising me so accurately on the important decisions of these last years as well as my family for their unconditional support.

To all, thank you very much from the heart.

Contents

Chapter 1

Introduction

Hereafter, the underlying problem is the classical *Integer Program* (IP), that we formulate in the following way:

$$(IP) \equiv max\{c^t x : Ax = b, l \leq x \leq u, x \in \mathbb{Z}^n\}$$

$A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$, l and u lower and upper bounds for x

Despite the simplicity of its formulation, allowing only linear constraints and a linear objective function, the importance of IP is well known . A large number of problems in diverse fields of the mathematics and algorithms with an infinity of applications admit an IP formulation. Unfortunately, it's also well known that integer programming is NP-Complete, what means that no efficient algorithm is likely to exist for solving an IP in the general case. This explains the great interest in restricted formulations of the problem and in certain resolution techniques even when they can't be applied to the general IP. In the following sections we present the latest techniques based on the **Graver bases** and its bounds as well as their application to the **N-Fold IP**, a restricted formulation of the IP which has won relevance in the last decades given its theoretical properties and its wide applications.

For this purpose, in chapter 2 we introduce the Graver basis of a given matrix, explore its properties, bounds, and how these can be applied for solving the general IP. We then study in chapter 3 the N-Fold case and show, with the help of Graver bases, that the N-Fold IP can be solved in polynomial time, delving into the best known algorithm for this case.

Chapter 2

Graver bases

In this chapter we first define formally the Graver basis of a matrix, presenting and proving its main properties. After this, we present two algorithms for the general IP based on these properties, studying their complexity and main limitations.

Before introducing the concept of Graver basis of a matrix, we define the partial order $\sqsubseteq$ in $\mathbb{R}^n$ by $u \sqsubseteq v$ if $u_i \cdot v_i \geq 0$ and $|u_i| \leq |v_i|$ for all i. Note that the condition $u_i \cdot v_i \geq 0$ means that $\sqsubseteq$ can only compare *sign compatible* vectors, i.e., vectors with the same sign componentwise. We also introduce the notation $\mathcal{L}(A)$ for the integral kernel of a matrix excluding zero, i.e., $\mathcal{L}(A) := \{z \in \mathbb{Z}^n : Az = 0, z \neq 0\}$.

Definition 2.1 (Graver basis). The Graver basis $\mathcal{G}(A)$ of a given matrix $A \in \mathbb{Z}^{mxn}$ is defined as the set of $\sqsubseteq$-minimal elements in $\mathcal{L}(A)$.

Graver bases were initially defined as *universal integral test set* in [1] by Jack. E. Graver in 1975. They often appear defined in an equivalent way as the nonzero indecomposable elements in $ker(A)$. *Indecomposable* in the sense that they can not be expressed as the sum of two sign compatible vectors. We now study their properties.

Proposition 2.2. *For every matrix A, $\mathcal{G}(A)$ is a finite set.*

Proof. Dickson's lemma states that every subset of $\mathbb{N}^n$ has a finite number of minimal elements with the order $\leq$ componentwise. It's easy to see that this implies that $\mathcal{L}(A)$ has a finite number of $\sqsubseteq$-minimal elements in every orthant. As the elements in different orthants are not comparable we have that $\mathcal{G}(A)$ is the union of 2^n finite sets, concluding the proof. □

Unfortunately, the cardinality of $\mathcal{G}(A)$ may be exponential in n, the number of columns of A. This limits the explicit computation and usage of the Graver basis to only certain cases but doesn't limit its theoretical properties. The most important of these properties is expressed in the following proposition:

Proposition 2.3. *Every integral element in $ker(A)$ can be expressed as positive integral linear combination of sign compatible elements in $\mathcal{G}(A)$.*

Proof. The proof is by induction in the ℓ_1 norm. For the base case we see that given $u \in \mathbb{Z}^n \cap ker(A)$ such that $||u||_1 = 1$ then u belongs to $\mathcal{G}(A)$ and the result holds.

For the induction case lets suppose the result is given up to k and take $u \in \mathbb{Z}^n \cap ker(A)$ such that $||u||_1 = k$. Again, if u is minimal in $\mathbb{Z}^n \cap ker(A) \setminus \{0\}$ the result is clear so let's suppose this is not the case. Therefore it exists u_1 s.t. $u_1 \sqsubseteq u, u_1 \neq u$. We take $u_2 = u - u_1$. Note that thanks to the definition of $\sqsubseteq$ u, u_1 and u_2 are sign compatible and thanks to $u \neq u_1$ necessarily $||u_1||_1, ||u_2||_1 \leq k$. With this appreciations, the proof concludes after applying the induction hypothesis:

$$u = u_1 + u_2 = \sum \alpha_{1i} g_{1i} + \sum \alpha_{2i} g_{2i} = \sum \alpha_j g_j$$

□

This proposition is the reason why Graver bases were introduced as *universal integral test set.* It ensures that, given any feasible point, the whole feasible region can be expressed in terms of elements in $\mathcal{G}(A)$. Note that thanks to requiring positive coefficients and sign compatible elements we avoid cancellations in every component.

In the next proposition we see how, thanks to this property, we can do an optimality test for any feasible point using only elements in the Graver basis.

Proposition 2.4. *Given a feasible point z of the IP, z is not optimum if and only if there exists $g \in \mathcal{G}(A)$ s.t. $c^t g > 0$ and $l \leq z + g \leq u$.*

Proof. If there exists $g \in \mathcal{G}(A)$ such that $c^t g > 0$ is clear that $z + g$ is a feasible point which strictly improves the objective function, so z is not an optimum.

For the other implication, if z is not an optimum we can take a feasible point y improving z. Thanks to the previous proposition there exist $g_i \in \mathcal{G}(A)$, $\alpha_i \geq 0$ s. t. $(y - z) = \sum \alpha_i g_i$. Then $0 < c^t(y - z) = \sum \alpha_i c^t g_i$ so at least one $g_i \in \mathcal{G}(A)$ verifying $c^t g_i > 0$. Finally, $z + g_i$ is feasible because thanks to $\alpha_i \geq 0$ and g_i being sign compatible with $y - z$ for all i we have: $l \leq z \leq z + g_i \leq z + \sum \alpha_i g_i = y \leq u$. □

2.1 Graver basis greedy augmentation algorithm

We now consider how to solve the general IP with the help of Graver bases. Note that proposition 2.4 doesn't only give us an optimality test but also provides us an improvement direction if the feasible point is not optimal. We can follow that improvement direction to get a better feasible point and then repeat this process. That is the idea of the following procedure (introduced in [1]):

General IP algorithm using Graver basis

1. From a feasible solution z_i
2. Find g^* optimum for the sub-problem:
   $max\{c^t g : g \in \mathcal{G}(A), l \leq z_i + g \leq u\}$
   - $c^t g^* \leq 0 \implies z_i$ optimal solution.
   - $c^t g^* > 0 \implies g^*$ improvement direction, loop back to 1 with:
     $z_{i+1} = z_i + \lambda \cdot g^*$ with the biggest λ respecting the bounds.

Note that the algorithm finishes in a finite number of steps thanks to the lower and upper bounds l and u. We can assume the bounds are finite and, this way, the objective function is also bounded. Since every iteration we are strictly increasing the objective function, no infinite loop is possible. As is well known, it is always possible to add suitable polynomial upper and lower bounds without excluding some optimal solution if any, so assuming l and u to be finite is no loss of generality.

The question that arises now is the complexity of this algorithm. It was analyzed in [2, Theorem 3.3] showing that it's polynomial. This of course doesn't mean we have a polynomial algorithm for the general IP, it means that, given an IP along with its Graver basis, we have a polynomial algorithm in this input size. The complexity of the problem remains in computing the Graver basis which, as we announced, may be exponential. This makes the algorithm non-viable but for small matrices.

Another way to estimate the complexity of the algorithm is using [3, Theorem 2.b], which precises that the number of augmentation steps is polynomial. Assuming the Graver basis is given as part of the input, we could solve any augmentation step by iterating over the elements in $\mathcal{G}(A)$, checking the constraints and the objective function, which has complexity $n \cdot |\mathcal{G}(A)|$. Therefore, if the Graver basis is given the algorithm is polynomial in $n \cdot |\mathcal{G}(A)|$.

2.2 Graver basis bound augmentation algorithm

Up to this point we have seen how Graver bases allow a straightforward algorithm for the general IP. However, we have also seen that its main drawback is that it requires the explicit computation of the Graver basis. In this section we show how we can avoid computing the Graver basis thanks to bounds on the ℓ_1-norm of its elements.

Proposition 2.5 (Graver basis bounds). *Given $A \in \mathbb{Z}^{mxn}$ and Δ an upper bound for the absolute value of each component of A, for every $g \in \mathcal{G}(A)$:*

- $||g||_1 \leq m^{m/2}\Delta^m \cdot (n-m)$ *[Onn 2010]*
- $||g||_1 \leq (2m\Delta+1)^m$ *[Eisenbrand,Hunkenschröder,Klein 2018]*

We refer to [4] and [5] for the proofs. Note that both bounds are exponential in the number of rows of A but the second one has the advantage of being independent in the number of columns. This will be a key fact in the next chapter.

Why should bounds to the Graver bases help? Because thanks to the proposition 2.4, the search of an improvement direction can be restricted to the elements in the Graver basis and, thanks to the bounds, we can restrict our search space without excluding any element of the Graver basis. This is the idea of the following algorithm [6]:

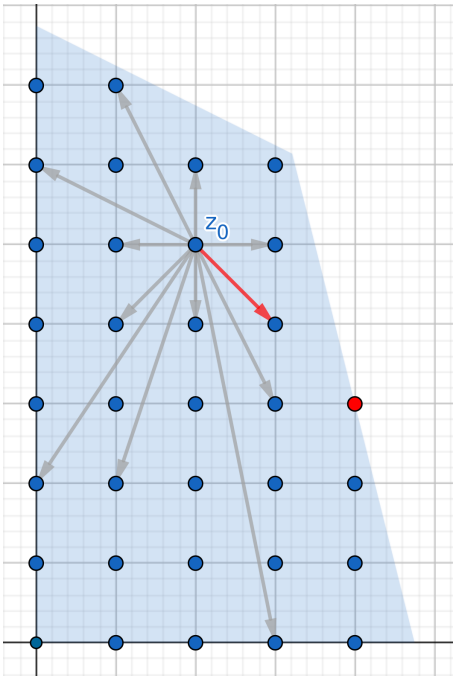


FIGURE 2.1: Feasible region

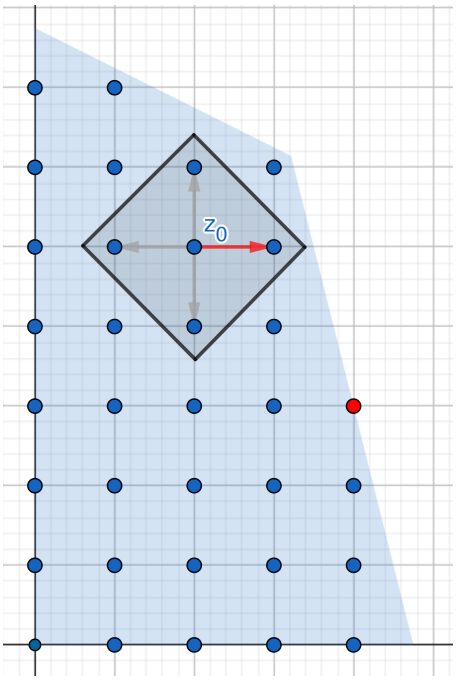


FIGURE 2.2: Bounded region

General IP algorithm using Graver basis norm bound

1. From a feasible solution z_i
2. Find g^* optimum for the sub-problem:
   $max\{c^t g : Ag = 0, l - z_i \leq g \leq u - z_i, g \in \mathbb{Z}^n, ||g||_1 \leq ||\mathcal{G}(A)||\}$
   - $g^* = 0 \implies z_i$ optimal solution.
   - $g^* \neq 0 \implies g^*$ improvement direction, loop back to 1 with:
     $z_{i+1} = z_i + \lambda \cdot g^*$ with the biggest λ respecting the bounds.

As we advanced, the main advantage of this algorithm is that it doesn't require the explicit computation of the Graver basis. However, the complexity is totally dependent on the added restriction $||g||_1 \leq ||\mathcal{G}(A)||$ and, as we saw in the proposition 2.2, the only bounds we have for the general case are exponential. This means that for the general IP the lower and upper bounds are much more restrictive than the Graver basis bound and therefore we have to explore the whole feasible region.

In certain cases we can get a much tighter bound for the Graver basis elements and this can help us to get a faster algorithm. The N-Fold IP is an iconic example.

Chapter 3

N-Fold IP

A generalized N-Fold IP has constriction matrix A of the form ($A_i \in \mathbb{Z}^{rxt}, B_i \in \mathbb{Z}^{sxt}$):

$$N^{(n)} = \begin{pmatrix} A_1 & A_2 & \cdots & A_n \\ B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_n \end{pmatrix}$$

It was presented in [2] in 2006 in a simplified version in which $\forall i, j\ A_i = A_j, B_i = B_j$. This simplified N-Fold matrix is totally determined given $A \in \mathbb{Z}^{r\times t}$, $B \in \mathbb{Z}^{s\times t}$ and the order n and we denote it by $N^{(n)}_{A,B}$. Hereafter we'll refer to the generalized formulation as simply the N-Fold IP.

The N-Fold IP has a withe range of applications. In [2] it's applied to the multiway transportation and cutting-stock problems and in [6] to privacy and disclosure control in statistical databases just to name a few examples beyond the typical in operations research. In fact, the N-Fold IP is universal. Every IP can be expressed as an N-Fold IP. This is because, as shown in [7], every IP can be modeled as a slim 3-way transportation program, which can be expressed as an N-Fold IP. However, this is more a theoretical achievement which shows the expressiveness power of the N-Fold rather than a useful practical approach for solving any IP.

In any case, the N-Fold IP is interesting by itself since it has good theoretical properties. In the following sections we will study it with the help of Graver bases to finally obtain a roughly linear algorithm for its resolution. We start with the following proposition:

Proposition 3.1. *For any $A \in \mathbb{Z}^{r\times t}$ and $B \in \mathbb{Z}^{s\times t}$ and any n there is a polynomial time algorithm in n that computes the Graver basis of the N-Fold matrix $N^{(n)}_{A,B}$.*

Again, we refer to Apendix A for more details about Graver bases computation. We won't go into the details of the proof of this proposition, they can be seen in [2] (Theorem 4.2). However, we consider important to remark that this proposition implies that the cardinality of $\mathcal{G}(A)$ is bounded by a polynomial function of n.

This has an important consequence, the greedy Graver basis augmentation algorithm presented before has, in this case, polynomial complexity for every augmentation step and therefore polynomial complexity. However it's still remaining the problem of obtaining an initial feasible solution, that is precisely what solves the next proposition.

Proposition 3.2. *For any $A \in \mathbb{Z}^{r\times t}$ and $B \in \mathbb{Z}^{s\times t}$ and any n there is a polynomial time algorithm in n that, given a demand vector $b \in \mathbb{Z}^{s+nr}$, either finds a feasible point $x \in \mathbb{N}^{nq}$ to the N-Fold IP of order n, or asserts that no feasible solution exists.*

Proof. Adding $2n(s+r)$ artificial variables (restricted to be positive, that's why we add $I and -I$) we can construct an N-Fold IP for which an initial feasible solution is trivial for any right side b. Applying the two phase method over this gives us the result.

$$N = \begin{pmatrix} A & I_s & -I_s & 0 & 0 & A & I_s & -I_s & 0 & 0 & \cdots & A & I_s & -I_s & 0 & 0 \\ B & 0 & 0 & -I_r & I_r & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & B & 0 & 0 & -I_r & I_r & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & B & 0 & 0 & -I_r & I_r \end{pmatrix}$$

□

Theorem 3.3 (N-Fold IP is polynomially solvable). *Fix any pair of integer matrices A, B of compatible sizes. Then there is a polynomial time algorithm that solves the generalized n-fold integer programming problem on any input n, b, c.*

Proof. As we advanced before, thanks to the bound on the cardinality of the Graver basis of $N^{(n)}_{A,B}$, once we get a feasible point we can apply the algorithm described in Section 2.1 to compute a solution in polynomial time. The previous proposition ensures that this initial feasible point can also be computed in polynomial time, therefore, the global algorithm is polynomial. □

3.1 N-Fold augmentation algorithm

A generalized N-Fold IP has constriction matrix A of the form ($A_i \in \mathbb{Z}^{rxt}, B_i \in \mathbb{Z}^{sxt}$):

$$N^{(n)} = \begin{pmatrix} A_1 & A_2 & \cdots & A_n \\ B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_n \end{pmatrix}$$

In this section we compute a bound for the ℓ_1-norm of the elements in the Graver basis of the N-Fold matrix. For this purpose we present the *Steinitz lemma*. We won't prove it but it can be seen in [8].

Lemma 3.4 (Steinitz Lemma). *Let $v_1, ..., v_n$ be vectors with $||v_i|| \leq \Delta$ for $i = 1, ..., n$. If $\sum_{i=1}^{n} v_i = 0$, then there is a reordering $\pi \in S_n$ such that for each $k \in \{1, ..., n\}$ the partial sum $p_k := \sum_{i=1}^{k} v_{\pi(i)}$ satisfies $||p_k|| \leq n\Delta$.*

It's possible (using Steinitz Lemma) to obtain a much tighter bound for the norm of the elements in the Graver basis than the ones mentioned before. This implies a restriction in the space of search for the improvement direction in the augmentation algorithm making it much faster.

Lemma 3.5 (N-Fold Graver basis bound). *For all $g \in Gr(N)$ $||g||_1 \leq L_B(2r\Delta L_B + 1)^r =: L_A$ where $L_B = (2s\Delta + 1)^s$*

Proof. Let y be a Graver basis element of $\mathcal{N}$, in coherence with the block decomposition of $\mathcal{N}$ we split it in blocks $y^t = ((y^{(1)})^t, ..., (y^{(n)})^t)$. Since every $y^{(i)} \in ker(B^{(i)}$ we decompose it as the sum of Graver basis elements $y^{(i)} = y_1^{(i)} + ... + y_{N_i}^{(i)}$. We have:

$$0 = A_1y^{(1)}+...+A_ny^{(n)} = A_1y_1^{(1)}+...+A_1y_{N_1}^{(1)}+...+A_ny_1^{(n)}+...+A_ny_{N_n}^{(n)} =: v_1+...+v_N$$

For $N = \sum_{i=1}^{n} N_i$ and $||v_i||_\infty \leq \Delta L_B$ for all $i \in \{1, ..., N\}$. Note that $N \leq (2r\Delta L_B + 1)^r$. At this point we can apply the Steinitz Lemma to reorder the v_i to bound each partial sum by $r\Delta L_B$ in the l_∞ norm.

$$||y||_1 \leq L_B(2r\Delta L_B + 1)^r = (2s\Delta + 1)^s(2r\Delta(2s\Delta + 1)^s + 1)^r = L_A$$

□

Thank to this bound we can prove this. We won't prove it since we'll see in the next section another version of this algorithm in a more efficient schema. [5] (Lemmas 4 and 5).

Lemma 3.6. *The augmentation step of the bounded algorithm (Section 2.2) can be solved in time* $nt(rs\Delta)^{\mathcal{O}(r^2s+rs^2)}$.

The only thing remaining then for determining the complexity is then bound the number of augmentation steps needed for the algorithm to finish. This is provided by the following lemma.

Lemma 3.7. *Consider the bounded algorithm for an N-Fold IP. Let* $\Gamma := max_i(u_i - l_i)$. *Given an initial feasible solution we can find an optimum solution applying the bounded algorithm, which executes the augmentation step at most* $\mathcal{O}(ntlog(\Gamma)log(nt\Gamma))$.

This complexity depends on the lower and upper bounds since, at the end of the day, they are restrictions to the feasible region and therefore to the problem's complexity. In [5] they solve this issue by first solving the linear relaxation problem and then creating artificial l and u constraints which, thanks to a proximity bound they prove, keeps at least one optimum.

Lemma 3.8 (N-Fold augmentation algorithm complexity). *The N-Fold IP can be solved in time* $(nt)^2log^2(nt) \cdot \varphi(rs\Delta)^{O(r^2s+rs^2)} + LP$

3.2 N-Fold via LP rounding

The main drawback of the augmentation algorithm is that after solving the linear relaxation, we can be arbitrarily far from the optimal solution of the IP, what means that we need many augmentation steps. In this section we follow another approach introduced in [9] based on a more restricted linear relaxation which optimum is closer to the optimum of the IP and we prove this. We then show how to take advantage of this proximity bound to obtain the current fastest algorithm for the N-Fold case, running in roughly linear time.

3.2.1 Restricted linear relaxation

Following the previous block decomposition of the variables in the N-Fold IP we can express the N-Fold IP as:

$$(N-FoldIP) \equiv max\{\sum (c^{(i)})^t y^{(i)} : \sum A^{(i)} y^{(i)} = b_0, l_i \leq y^{(i)} \leq u_i, y^{(i)} \in P_i\}$$

$$A \in \mathbb{Z}^{mxn},\ b \in \mathbb{Z}^m,\ c \in \mathbb{Z}^n,\ l \text{ and } u \text{ lower and upper bounds for x}$$

The main idea of the restricted linear relaxation is that we restrict each region P_i to $Q_i = conv(P_i \cap \mathbb{Z}^t)$.

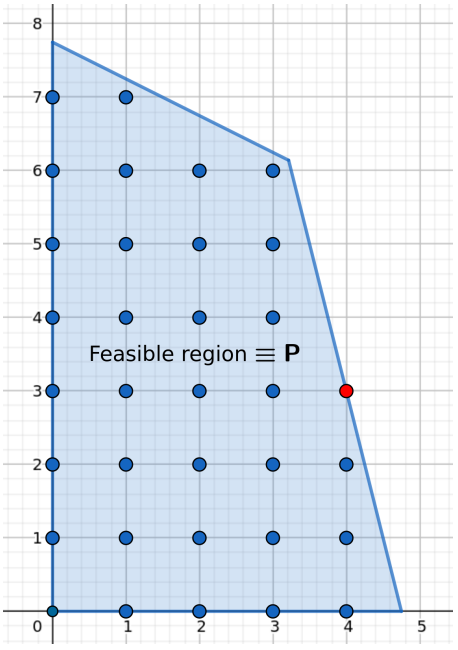


FIGURE 3.1: Complete this

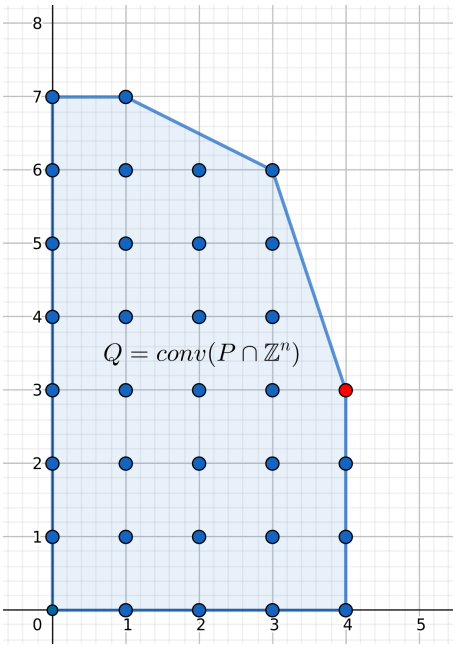


FIGURE 3.2: Also this

Solving the restricted linear relaxation problem is not an easy task in practice at all but it was proved in [9] that it can be solved in linear time (ignoring logarithmic factors).

Proposition 3.9 (N-Fold RLR complexity). *The N-Fold IP restricted linear relaxation problem can be solved in time*

$$O(nt \cdot log^2(nt) \cdot \varphi p(r)(s\Delta)^{O(s^2)})$$

Proposition 3.10 (N-Fold proximity to RLR). *Let x^* be an optimal vertex solution of a N-Fold RLR, then there exists an optimal solution z^* for the N-Fold IP verifying:*

$$||z^* - x^*||_1 \leq (rs\Delta)^{O(rs)}$$

[Cslovjecsek, Eisenbrand, Weismantel 2020]

3.2.2 Dynamic program

We present in this section a dynamic program which takes us from an optimal solution to the restricted linear relaxation to the optimal solution of the N-Fold IP.

Proposition 3.11 (N-Fold RLR to optimum complexity). *Given an optimal vertex of an N-Fold RLR, the N-Fold IP can be solved in time*

$$O(nt \cdot (rs\Delta)^{O(r^2s+s^2)})$$

Proof. We now define the set S_ℓ as the set of $y \in \mathbb{Z}^r$ such that:

$$\sum_{i=1}^{\ell} A_i x_i^* - \gamma \leq y \leq \sum_{i=1}^{\ell} A_i x_i^* + \gamma$$

We can construct a weighted directed acyclic graph G(V,E) with vertices:

$$V = \{(\ell, y)/y \in S_\ell\} \cup \{(0,0), (n, b_0)\}$$

Weighted edges between $(\ell - 1, y)$ and (ℓ, y') if the problem is feasible:

$$max\{c_\ell^t x : A_\ell x = (y' - y), B_\ell x = b_\ell, l_\ell \leq x \leq u_\ell\}$$

A longest path from $(0,0)$ to (n, b_0) in G(V,E) corresponds to an optimal solution of the N-Fold integer program.

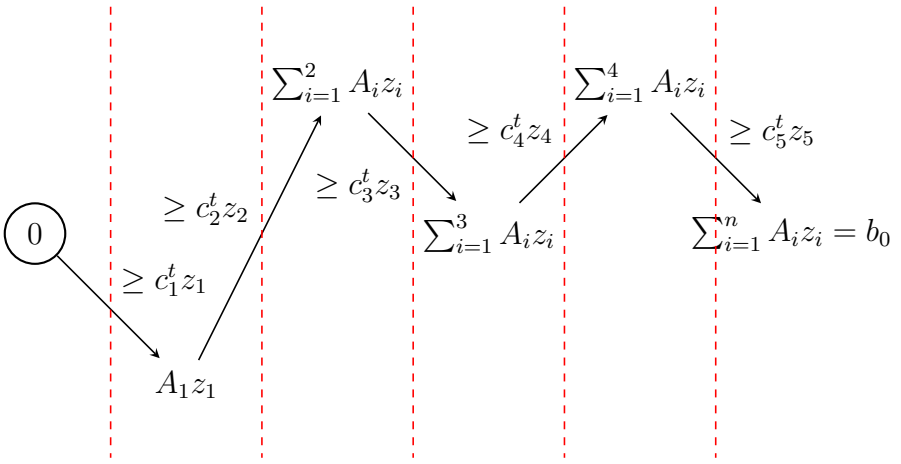
Let $\gamma := (rs\Delta)^{O(rs)}$. For every $1 \leq \ell \leq n$:

$$\begin{aligned} ||x^* - z||_1 \leq \gamma &\implies \sum_{i=1}^{\ell} ||x_i^* - z_i||_1 \leq \gamma \\ &\implies \sum_{i=1}^{\ell} ||A_i(x_i^* - z_i)||_\infty \leq \Delta\gamma = \gamma \end{aligned}$$

Therefore, for every feasible point z verifying the proximity bound we have:

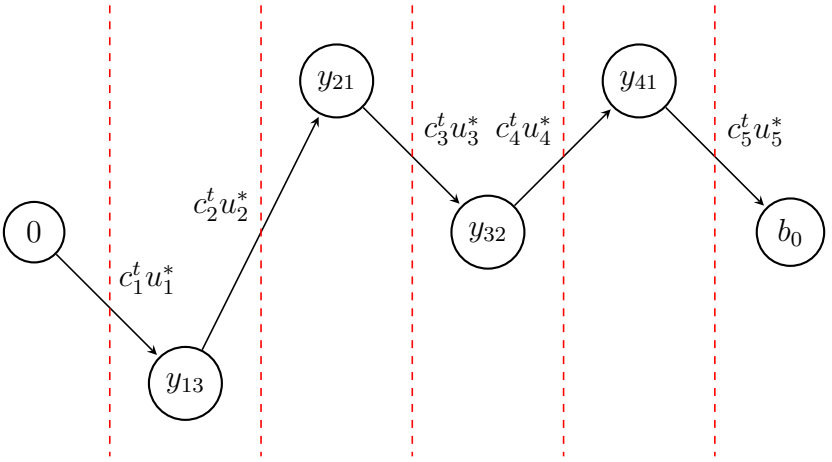
$$\sum_{i=1}^{\ell} A_i x_i^* - \gamma \leq \sum_{i=1}^{\ell} A_i z_i \leq \sum_{i=1}^{\ell} A_i x_i^* + \gamma$$

Graph plot for the proof of algorithm complexity: ————————————————————



What this means is that we have vertices in S_ℓ (mention explicitly!) and edges (thanks that the problem has a feasible point given by the next element of the partial sum). Note also that these feasible point is of course lower or equal than the optimal solution of this edge problem and therefore the path has cost greater or equal than the objective function in that point.

Analogously if we have a path from 0 to b_0 we have vertices and edges. The edges are associated with optimal solutions for the edge IP and thanks to start in 0 and finishing in b_0 we can construct a feasible point since this optimal solutions for the edges problem. Also the path cost is the value of the objective function in this point so every path is associated with a feasible point and it's cost is bounded by the optimal solution of the IP.



□

Facts for N-Fold complexity

- $|S_l| \leq (rs\Delta)^{O(r^2 s)}$
- $|V| + |E| \leq O(n(rs\Delta)^{O(r^2 s)})$
- The edge IP can be computed in time $t((r+s)\Delta)^{O(r+s)^2}$
- Longest path problem in a acyclic digraph can be solved in linear time.

N-Fold complexity

- **N-Fold complexity**
  The N-Fold IP can be solved in time $nt(rs\Delta)^{O(r^2 s+s^2)} + RLR$

[Cslovjecsek, Eisenbrand, Weismantel 2020]

Bibliography

[1] Jack E. Graver. On the foundations of linear and integer linear programming i. 1975.

[2] Jesús A. De Loera, Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. N-fold integer programming. 2006.

[3] Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. A polynomial oracle-time algorithm for convex integer minimization. 2009.

[4] Shmuel Onn. Nonlinear discrete optimization. 2010.

[5] Friedrich Eisenbrand, Christoph Hunkenschröder, and Kim-Manuel Klein. Faster algorithms for integer programs with block structure. 2018.

[6] Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. N-fold integer programming in cubic time. 2011.

[7] Jesús A. De Loera and Shmuel Onn. All linear and integer programs are slim 3-way transportation programs. 2006.

[8] Ernst Steinitz. Bedingt konvergente reihen und konvexe systeme. 1913.

[9] Jana Cslovjecsek, Friedrich Eisenbrand, and Robert Weismantel. N-fold integer programming via lp rounding. 2020.

[10] Bernd Sturmfels. Algebraic recipes for integer programming. 2003.

[11] Elisabeth Finhold and Raymond Hemmecke. Lower bounds on the graver complexity of m-fold matrices. 2013.

[12] Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. 2019.

[13] Raymond Hemmecke. Exploiting symmetries in the computation of graver bases. 2004.

[14] Shmuel Onn. Convex discrete optimization. 2007.

[15] 4ti2 team. 4ti2—a software package for algebraic, geometric and combinatorial problems on linear spaces. URL `https://4ti2.github.io`.