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-Computer Engineering at 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 Prof. Katzalin Olcoz and Prof. 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 my family for their unconditional support and also Luis Felipe Ramirez, for advising me so well in the important decisions these last years.

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:

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

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

Despite the simplicity of its formulation, allowing only linear constraints and a linear objective function, it's well known the importance of IP. 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 IP is NP-Complete, what means that no efficient algorithm is likely to exist for solving the 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 last 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, we first introduce the Graver basis of a given matrix, explore its properties, bounds, and how can these be applied for solving the general IP. We then study the N-Fold case and show, with the help of Graver bases, that the N-Fold IP can be solved in polynomial time. In the last sections we go further improving this polynomial complexity, obtaining two different efficient algorithms. One based on augmenting a feasible solution and another based on a proximity bound.

Chapter 2

Graver bases

Before introducing the concept of Graver basis of a matrix, we define a 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., with the same sign componentwise. The Graver basis of a matrix is the set of minimal elements (for this order $\sqsubseteq$) in its integral kernel excluding zero. Formally:

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 $\{z \in \mathbb{Z}^n : Az = 0, z \neq 0\}$.

Graver bases were initially defined as *universal integral test set* in [1] by Jack. E. Graver, in 1975. They often appear also 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. It's easy to see the equivalence of both definitions.

Now that Graver bases are formally defined, we present their main properties in the form of propositions which will be the theoretical basis for the algorithms presented in the next sections.

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 the integral kernel of A (excluding zero) 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 of course limits the explicit computation and usage of the Graver basis to only certain cases but of course 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 $g \in \mathcal{G}(A)$ (therefore $\in ker(A)$) $s.t. c^t g > 0$ 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, then $y - z$ verifies the hypothesis of the previous proposition so there exist $g_i \in G(A)$, $\alpha_i \geq 0$ s. t. $0 < c^t(y - z) = \sum \alpha_i c^t g_i$ and it's then clear that exists at least one $g_i \in \mathcal{G}(A)$ verifying $c^t g_i > 0$. Finally, thanks of $\alpha_i \geq 0$ and g_i being sign compatible with $y - z$ we have that for all i, $l \leq z \leq z + g_i \leq z + \sum \alpha_i g_i = y \leq u$. □

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 provide 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 (idea 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.

We can affirm that it's an algorithm, i.e., it finishes in a finite number of steps, thanks to the lower and upper bounds l and u. We can assume they 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. In the Appendix A we go further to analyze how to compute the Graver basis of a given matrix and we introduce the tool 4ti2.

Another way to estimate the complexity of the algorithm is using [3] (Theorem 2.b), which states that the number of augmentation steps is polynomial. Since once obtained the Graver basis the cost of each augmentation step is in the order $n \cdot |\mathcal{G}(A)|$ (search over $\mathcal{G}(A)$), it's clear that the algorithm is polynomial in $n \cdot |\mathcal{G}(A)|$.

2.2 Graver Basis norm bounds

Up to this point we have seen how Graver bases allow a straightforward algorithm for the general IP. However, we have 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 it thanks to bounds on the ℓ_1-norm of the Graver basis 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 proof. 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.

Why should bounds to the Graver bases help? Because thanks to 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 (see [6]).

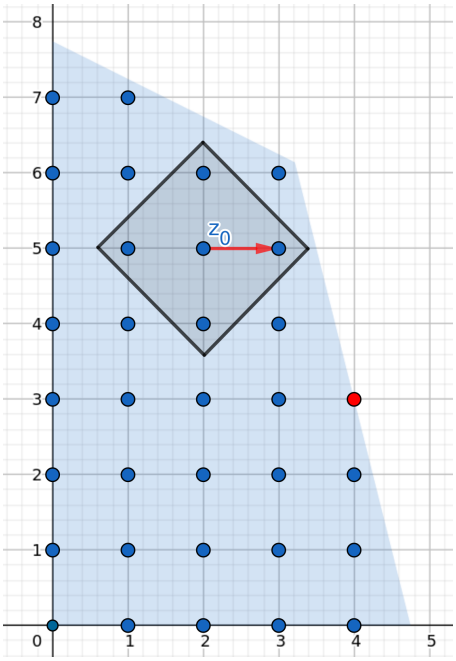


FIGURE 2.1: Complete this

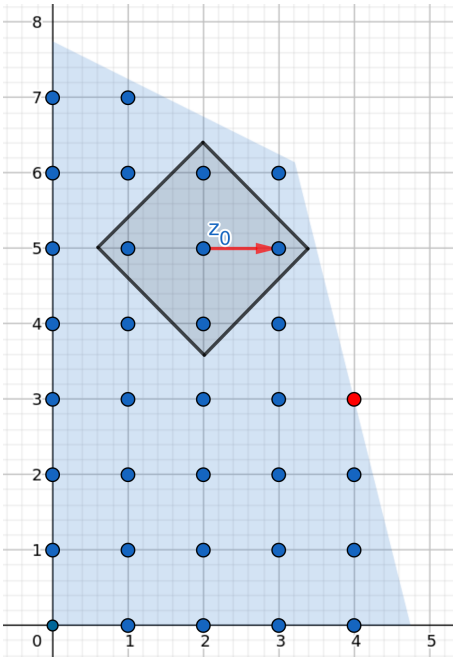


FIGURE 2.2: Also this

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 the only bounds we have for the general case are exponential. In this case, the additional restriction to the problem doesn't improves the lower and upper bounds and the complexity is exponential.

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

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. □

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

3.2 N-Fold via LP rounding

The main drawback for improving the complexity 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

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

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

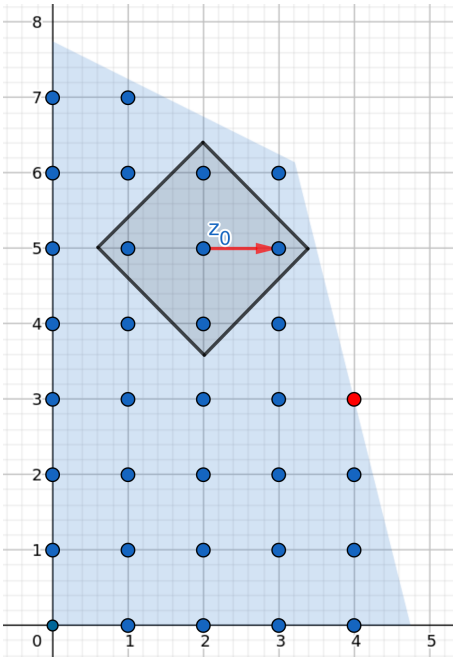


FIGURE 3.1: Complete this

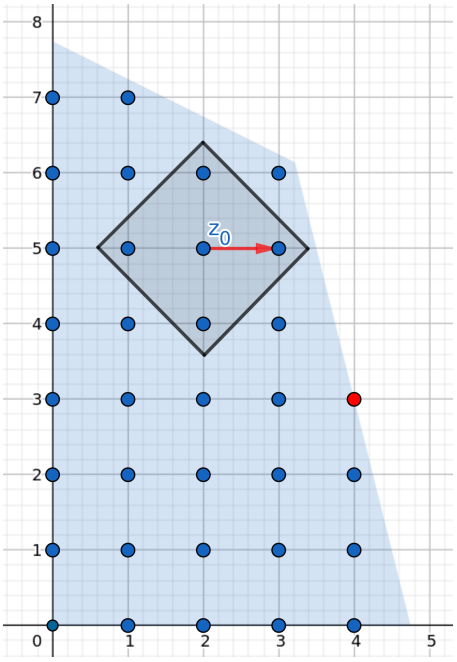


FIGURE 3.2: Also this

3.2.2 Dynamic program

Proposition 3.10 (N-Fold RLR to optimum complexity). *Given an optimal vertex of an N-Fold RLR, the N-Fold IP can be solved in time*

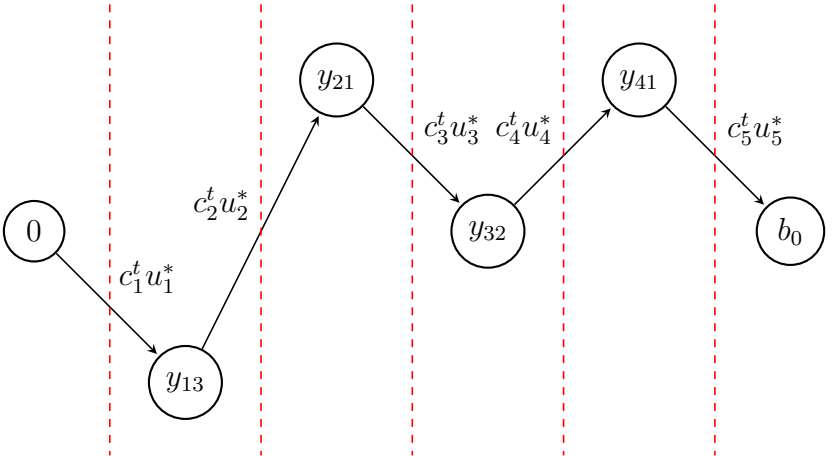
$$O(nt \cdot (rs\Delta)^{O(r^2s+s^2)})$$

Proposition 3.11 (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:*

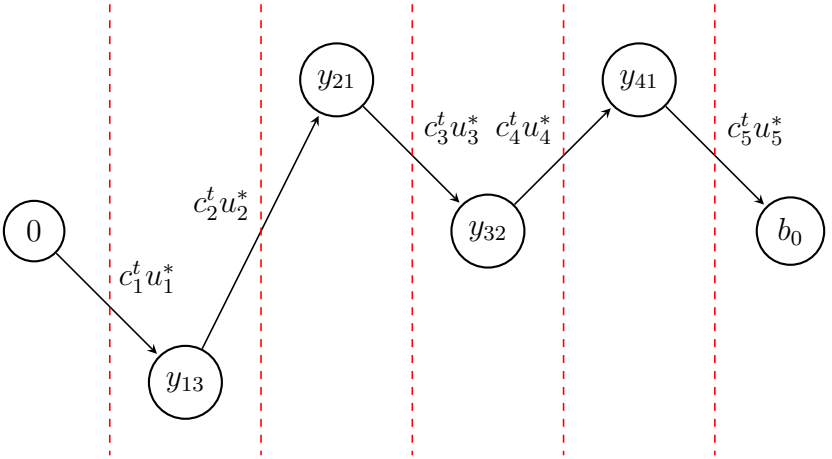
$$||z^* - x^*||_1 \leq (rs\Delta)^{O(rs)}$$

[Cslovjecsek, Eisenbrand, Weismantel 2020]

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



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



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]

Appendix A

Graver basis computation with 4ti2

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.