**Indian Academy of Sciences, Bengaluru**
**Indian National Science Academy, New Delhi**
**The National Academy of Sciences India, Prayagraj**

# SUMMER RESEARCH FELLOWSHIPS — 2023

## Format for the final Report*,^

| | | |
|---|---|---|
| Name of the candidate | : | Manjeet Singh |
| Application Registration no. | : | MATS862 |
| Date of joining | : | 9th JUNE 2023 |
| Date of completion | : | 6th August 2023 |
| Total no. of days worked | : | 59 Days |
| Name of the guide | : | Prof. Nitin Saxena |
| Guide's institution | : | Indian Institute of Technology Kanpur |
| Project title | : | Integer Programming with fixed no. of variables |

**Address with pin code to which the certificate could be sent:**

P-10, Old Navy Nagar, Colaba
Mumbai, 400005
E-mail ID: manjeetazin@gmail.com
Phone No: 7386807731

TA Form attached with final report : [✓ YES] NO
If, NO, Please specify reason

[illegible]
Signature of the candidate
Date: 8th August 2023

Signature of the guide
Date: [illegible] 8th August 2023

**IMPORTANT NOTES:**
*** This format should be the first page of the report and should be stapled with the main report. The final report could be anywhere between 20 and 25 pages including tables, figures etc.**
**^ The final report must reach the Academy office within 10 days of completion. If delayed fellowship amount will not be disbursed.**

(For office use only; do not fill/tear)

| | |
|---|---|
| Candidate's name: | Fellowship amount: |
| Student: Teacher: | Deduction: |
| Guide's name: | TA fare: |
| KVPY Fellow: INSPIRE Fellow: | Amount to be paid: |
| PFMS Unique Code: | A/c holder's name: |
| Others | |

Summer Research Fellowship Programme, 2023

# INTEGER PROGRAMMING WITH A FIXED NUMBER OF VARIABLES

**Manjeet Singh MATS862**
Indian Institute of Information Technology, Ranchi

**Guide: Prof. Nitin Saxena**
Indian Institue of Technology, Kanpur

# Contents

# Chapter 1

# Introduction

Linear and Integer programs belong to the class of optimization problems and find applications in various real-world fields. The main objective of a general linear programming problem is to either maximize or minimize a specific goal while adhering to certain constraints. For instance, one application involves scheduling students, faculty, and classrooms to minimize the number of students unable to enroll in their preferred classes. Alternatively, a manufacturer may utilize linear and integer programming to allocate resources efficiently while maximizing profits. These types of problems are solvable using linear algebra when the constraints are linear, ensuring polynomial time complexity. The key distinction between linear and integer programs lies in the requirement that integer programs' solutions must be integral rather than real-valued. It is much more difficult to find integer-only solutions to the problem, and currently, integer programs are not solvable in polynomial time (they are NP-complete)[1].The general integer program is the problem:

**Integer Programming Problem:** $Let\ a_{ij}\ \in \mathbb{Z}, b_i\ \epsilon\ \mathbb{Z},\ and\ c_{ij}\ \epsilon\ \mathbb{R}$

$$i = 1, ...., n, j = 1, ..., m;$$

$$we\ wish\ to\ find\ a\ solution(\sigma_1, \sigma_2, \sigma_3, ....., \sigma_m)\ in\ \mathbb{N}^m\ of\ the\ system$$

$$\begin{gathered}
a_{11}\sigma_1 + a_{12}\sigma_2 + \cdots + a_{1m}\sigma_m = b_1 \\
a_{21}\sigma_1 + a_{22}\sigma_2 + \cdots + a_{2m}\sigma_m = b_2 \\
. \\
. \\
. \\
a_{n1}\sigma_1 + a_{n2}\sigma_2 + \cdots + a_{nm}\sigma_m = b_n
\end{gathered}$$

$which\ minimizes\ the\ "cost\ function"$

$$c(\sigma_1, \sigma_2, \cdots, \sigma_m) = \sum_{j=1}^{m} c_j \sigma_j$$

**Types of Integer Programming Problems:**

- **Pure Integer Programming Problem**: All variables are required to be integer.
- **Mixed Integer Programming Problem:** Some variables are restricted to be integers; the others can take any value.
- **Binary Integer Programming Problem:**All variables are restricted to be 0 or 1.

Methods to solve integer programming whose time complexity is also order of n include:

- **Branch and Bound**
- **Gomory's Cutting Plane**
- **Lenstra's Algorithm**

We will looking at different ways of solving Pure Integer Programming Questions, their proofs, algorithms and examples.

# Chapter 2

# Branch And Bound

## 2.1 Introduction

For most discrete optimization problems, performing a complete enumeration or explicit listing of all possible solutions to find the optimal one is impractical. Even for relatively small problem sizes, the number of solution points in the feasible region can be incredibly large. For instance, if a single point can be enumerated in $10^{10}$ seconds, enumerating all points in a small problem with only 75 binary variables would take about 120,000 years!

To address this issue, a general-purpose approach called ”branch-and-bound” was introduced by Land and Doig [2] for solving integer programming problems. This method avoids explicitly enumerating every solution point and relies on a few simple principles to efficiently explore the feasible region. By selectively exploring promising branches of the solution space, the branch-and-bound technique significantly reduces the computational burden and allows for practical solution finding in discrete optimization problems..

Branch-and-bound is a divide and conquer strategy, which decomposes the problem to sub-problems over a tree structure, which is referred to as branch-and-bound tree. The decomposition works based on a simple idea: If S is decomposed into $S^1$ and $S^2$ such that $S = S^1 S^2$, and we define sub-problems $z^k = max\{f(x) : x \in S^k\} for\ k = 1, 2$, then $z = max_k z^k$. Each subproblem represents a node on the tree. Fig. 1 shows a schematic of a branch-and-bound tree. The main problem with feasible region $S$ (for simplicity we call it problem $S$) is at the root node, and is then divided into two sub-problems (with feasible regions) $S^1$ and $S^2$, where we have $S^1 S^2 = S$. The process of dividing a node sub-problem into smaller sub-problems is called branching and sub-problems $S^1$ and $S^2$ are called branches created at node $S$. In Fig. 1 sub-problem $S^1$ is further branched into smaller sub-problems and so on. The branching does not necessarily have to be two-way, and multi-way branching is also possible. Observe that branching indefinitely will only result in explicit enumeration of the feasible region of $S$. Therefore to avoid explicit enumeration, in branch-and-bound whenever possible a branch is pruned (or fathomed), meaning that its sub-problem is not divided anymore. In other words, the feasible region of the node sub-problem is implicitly enumerated without branching any deeper.But when can we prune a branch? The main idea is to use bounds on the objective value of sub-problems intelligently to prune branches.

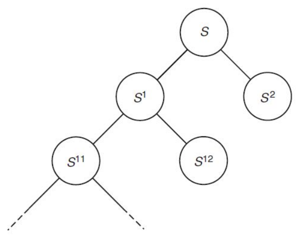


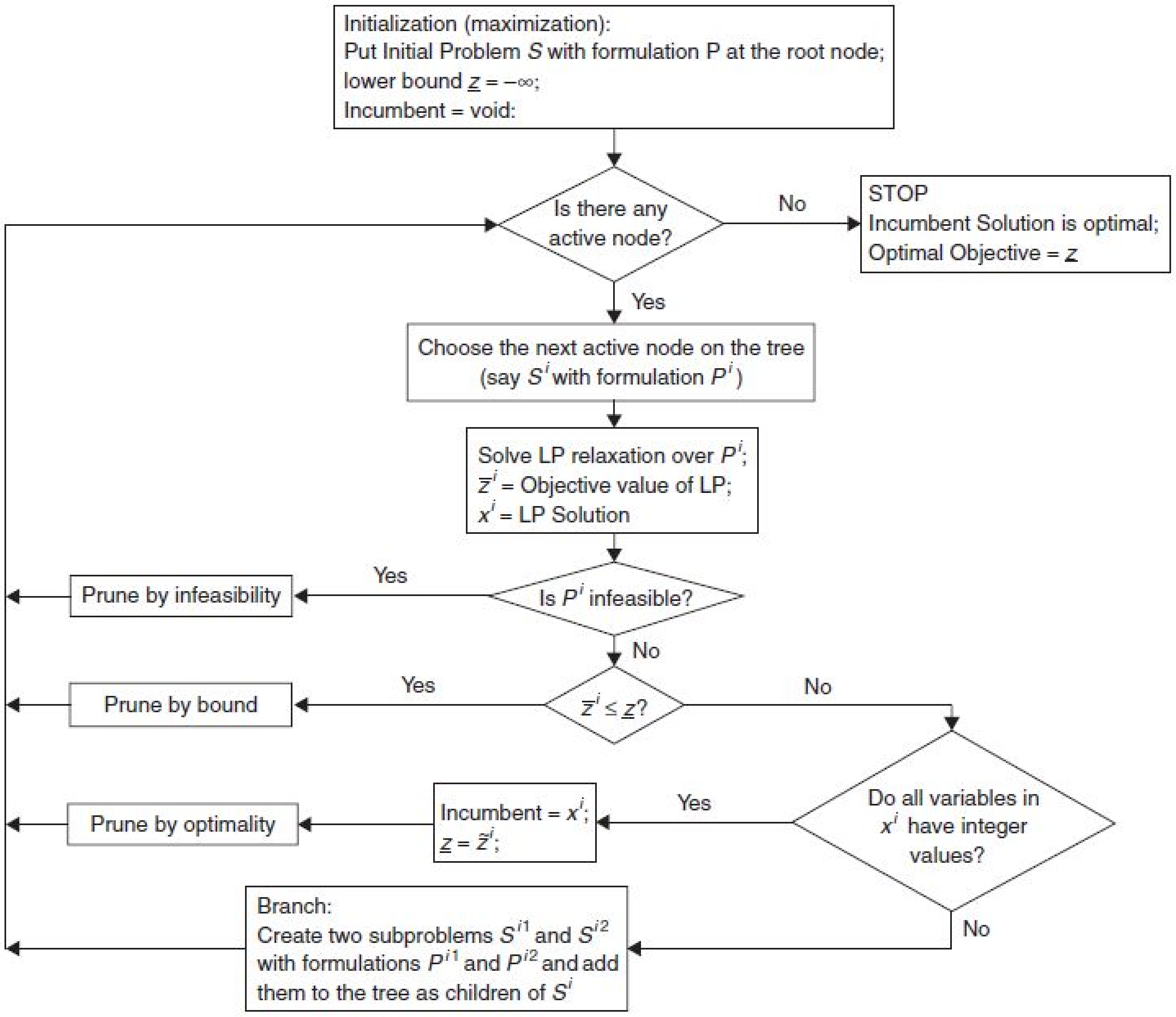
**Fig. 1**: Sub-problems in a branch-and-bound Tree

## 2.2 Branch and Bound To Solve Integer Programming Problems

Consider an Integer Programming Based Problem as

$$z = max\{cx : x \in S\}$$

where $S = \{x : x \in Z^n \cap P\}$ and $P$ is a polyhedron. As before we refer to this problem as problem $S$. For simplicity we talk about pure integer programming problems but what follows can be easily extended to mixed integer programming problems too. Figure 2 shows the flow chart of the branch-and-bound algorithm for solving the IP problem S. This flow chart is a special case of the general flow customized in order to solve the IP problem.


Initialization (maximization):
Put Initial Problem S with formulation P at the root node;
lower bound z = −∞;
Incumbent = void:
Is there any active node?
No
STOP
Incumbent Solution is optimal;
Optimal Objective = z
Yes
Choose the next active node on the tree (say S^i with formulation P^i)
Solve LP relaxation over P^i;
z̄^i = Objective value of LP;
x^i = LP Solution
Is P^i infeasible?
Yes
Prune by infeasibility
No
z̄^i ≤ z?
Yes
Prune by bound
No
Do all variables in x^i have integer values?
Yes
Incumbent = x^i;
z = z̄^i;
Prune by optimality
No
Branch:
Create two subproblems S^i1 and S^i2 with formulations P^i1 and P^i2 and add them to the tree as children of S^i

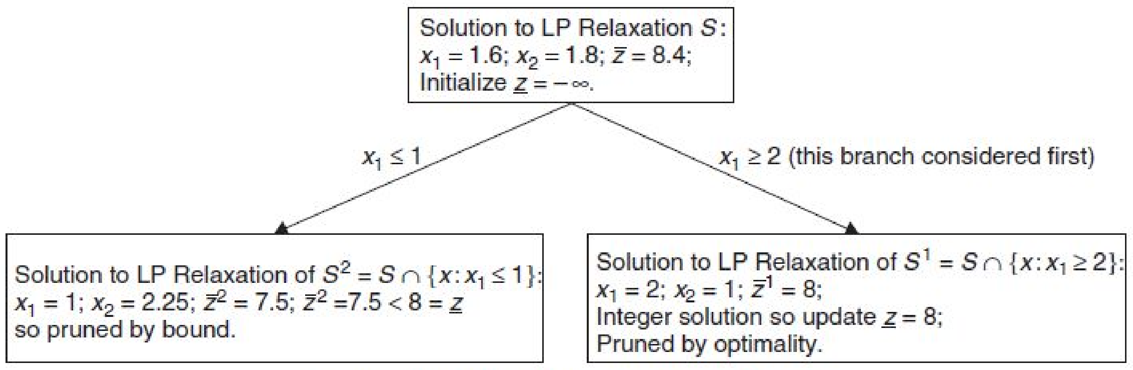
**Example:** Consider an IP problem

$$max(z) = 3x_1 + 2x_2$$

$$3x_1 + 4x_2 \leq 12$$

$$2x_1 + x_2 \leq 5$$

$$x_1, x_2 \geq 0 \; and \; x_i \epsilon \mathbb{Z}$$



Solution to LP Relaxation $S$:
$x_1 = 1.6$; $x_2 = 1.8$; $\bar{z} = 8.4$;
Initialize $\underline{z} = -\infty$.

$x_1 \leq 1$

$x_1 \geq 2$ (this branch considered first)

Solution to LP Relaxation of $S^2 = S \cap \{x : x_1 \leq 1\}$:
$x_1 = 1$; $x_2 = 2.25$; $\bar{z}^2 = 7.5$; $\bar{z}^2 = 7.5 < 8 = \underline{z}$
so pruned by bound.

Solution to LP Relaxation of $S^1 = S \cap \{x : x_1 \geq 2\}$:
$x_1 = 2$; $x_2 = 1$; $\bar{z}^1 = 8$;
Integer solution so update $\underline{z} = 8$;
Pruned by optimality.

So the optimal IP solution is $x_1 = 2$; $x_2 = 1$; $z = 8$.

# Chapter 3

# Cutting Plane Method

## 3.1 Introduction

The concept of adding linear inequality constraints, also known as cutting planes, to the linear Integer Programming (LP) problem was first introduced by Dantzig, Fulkerson, and Johnson in 1954 while working on the traveling salesman problem. These additional constraints effectively eliminate parts of the convex feasible region that do not contain any feasible integer points.

In 1958, Gomory [3] presented the first cutting plane algorithm, which systematically generated cuts applicable to integer programming (IP) problems. Subsequently, in 1960, he developed a second cutting plane algorithm for IP problems that maintains all-integer tableaux (Gomory [4]). Later, Gomory [5] extended his first cutting plane algorithm to handle mixed-integer programming (MIP) problems. All these algorithms fall under the category of dual cutting plane algorithms because they preserve dual feasibility when applied to the optimal solution of the LP-relaxation of the IP problems.

## 3.2 Gomory's Cutting Plane Method

### 3.2.1 Fractional Algorithm

Indeed, the first cutting plane algorithm introduced by Gomory [3] to solve pure Integer Programming (PIP) problems is commonly referred to as "Gomory's fractional cuts." The reason behind this name is that all the nonzero coefficients of the generated cuts are strictly less than one. Additionally, this algorithm is sometimes known as the "method of integer form."

What makes Gomory's fractional cuts particularly significant is that they were the first cutting plane algorithm to be mathematically proven to be finitely convergent. This means that when applying the algorithm, it is guaranteed to reach an optimal solution within a finite number of iterations. This property is crucial in practical implementations, as it ensures that the algorithm will eventually terminate and find an optimal solution for the pure Integer Programming problems it is applied to.
Consider PIP Problem:

$$\begin{aligned} maximize \quad & \sum\nolimits_{j=1}^{n} c_j x_j \\ subject\ to \quad & \sum\nolimits_{j=1}^{n} a_{ij} x_j \le b_i \quad i = -1 \cdots, m \\ & x_j \ge 0\ and\ integer \quad j = -1 \cdots, n \end{aligned} \tag{3.1}$$

where the data are required to be integral (constraints are cleared of fractions) to ensure that the value of objective function and the slack variables are integral for any integer solution. Gomory's fractional cuts can be derived as follows. Suppose that we have an optimal solution of the LP-relaxation which is noninteger. Consider the pth row of the noninteger solution. This can be expressed as

$$x_p = \bar{a}_{p0} - \sum\nolimits_{jeJ} \bar{a}_{pj} x_i \tag{3.2}$$

where J is the set of nonbasic variables, which can be written as

$$x_p = [\bar{a}_{p0}] + f_{p0} - \sum\nolimits_{juJ} [\bar{a}_{pj}] x_j - \sum\nolimits_{jeJ} f_{pj} x_j \tag{3.3}$$

Where $\bar{a} - pj = [\bar{a}_{pj}] + f_{pj}\ 0 \le f_{pj} < 1$ and [y] denotes the largest less then or equal to y. This can be rearrange as

$$x_p = [\bar{a} - p0] + \sum\nolimits_{juJ} [\bar{a} - pj] x_j = f_{p0} - \sum - juJ f_{pj} x_j \tag{3.4}$$

Given that $f - pj \ge 0$ and $x_j \ge 0, j \epsilon J$

$$\sum f_{pJ} x_j \ge 0 \tag{3.5}$$

Thus

$$f_{p0} - \sum f_{pJ} x_j \le f_{p0} \tag{3.6}$$

Since all the variables are required to be integer, it follows that the left hand side of the equation (4.4) will be integer, hence the right hand side must also be integer. Thus the left hand side of (4.6) is required to be integer and therefore

$$f_{p0} - \sum\nolimits_{jeJ} f_{pJ} x_j \leq 0 \tag{3.7}$$

and can be written in the form

$$s_p = -f_{p0} + \sum - p0 f_{pJ} x_j \tag{3.8}$$

where $s_p$ is a non-negative integer slack variable. This is Gomory's cut for solving PIP problems. Adding the new constraint (4.8) to the optimal tableau of the related LP problem results in a primal infeasible problem which can be solved using the dual simplex method.

The basic algorithm for Gomory's fractional cuts is as follows:

**Step 1: Initialisation**
Solve the LP-relaxation. If it is infeasible, so is the IP problem-terminate. Otherwise go to step 2.

**Step 2: Optimality test**
If the optimal solution to LP-relaxation is integer feasible then it is optimal to IP terminate. Otherwise go to step 3.
**3: Cutting and pivoting**
Choose a row $r$ with and add to the bottom of the simplex tableau, the $f_{r0} > 0$ fractional cut or constraint (4.8).
**4: Reoptimization**
the new LP using the dual simplex method. If the solution to the new LP problem is infeasible, the IP has no solution - terminate. If the new optimum is integer feasible, the IP is solved - terminate. Otherwise go to step 2.

It is a standard practice to drop each added inequality immediately after its slack variable re-enters the set of basic variables. That is, when a new inequality is introduced it is used as the pivot row so its slack becomes nonbasic. When this slack variable returns to the basic set, it and its defining row are omitted. The tableau contains exactly n nonbasic variables always. Hence if this rule is followed, there will never more than n cuts at any stage of the algorithm.

### 3.2.2 Gomory's all-integer algorithm

That's correct. Gomory's dual fractional algorithm is known to suffer from round-off errors and may not be efficient for solving large problems. To address this issue, Gomory developed a second cutting plane algorithm as an improvement. This algorithm, called the "dual all-integer algorithm," is a direct extension of the classical dual simplex method. The key difference between the two algorithms is that in the all-integer algorithm, the pivot row is generated at each iteration, ensuring a pivot value of -1. This pivot value guarantees that the resulting tableau will maintain all-integer coefficients.

Unlike the fractional algorithm, the all-integer algorithm is applied directly to the initial tableau without any optimization, constraint generation, or reoptimization steps. In this method, inequalities are generated at every iteration, starting from the very first one. Each of these constraints is designed to have integral coefficients and ensure that the pivot value is precisely -1. By maintaining all-integer tableaux throughout the process, the dual all-integer algorithm overcomes the round-off errors associated with fractional algorithms and performs better in practice, especially for larger problems.

Consider the same PIP problem(4.1)
A general dual all-integer algorithm is as follows:

**Step 1 :**
Start with an all-integer simplex tableau which contains a dual feasible solution.Go to step 2.

**Step 2:**
Select a primal infeasible row p (i.e $\bar{a}_{p0} < 0, p \neq 0$) . If none exists, the tableau exhibits the optimal integer solution - terminate. Otherwise go to step 3.

**Step 3:**
Designate the pivot column q to be the lexicographically smallest among those having $\bar{a}_{pj} < 0$ . If none exists (i.e $\bar{a}_{pj} \geq 0 \; for \; j = 1, \cdots, n$) there is no integer feasible solution terminate. Go to step 4.

**Step 4:**
Derive an all-integer inequality for row $p$ which is not satisfied at the current primal solution. (Its slack will be negative). It must also have a -1 coefficient in column q. Append it to the bottom of the tableau and label it the pivot row. Perform a dual simplex pivot operation and return to step 2.

Note that a vector $A$ is $lexicographically\ positive$ $(denoted\ as\ A > 0)$ if its first nonzero 0¿A element is positive. A vector $A$ is said to be $lexicographically\ greater$ than a vector $B$ if $A - B > 0$. If $-A > 0$, $A$ is called $lexicographically\ negative$ $(denoted\ as\ A- < 0)$ and if $A - B < 0, A$ is $lexicographically$ smaller than $B$.

To begin the calculations, it is necessary to have all-integer dual feasible tableau with $a_{0j} > j$ for all $j$ where $J$ is the set of nonbasic variables. If the initial solution is not dual feasible, that is, at least one $a_{0j} < 0$ , then a redundant constraint of the form

$$s = p - \sum_{j \epsilon J} x_j \tag{3.9}$$

where $p$ is a suitably large integer, is added.

Let the all-integer cut generated on the pth row be as follows:

$$s_p = [\bar{a}_{p0}|\lambda] - (\sum\nolimits_{jeJ})[(\bar{a}_{pj}|\lambda)x_j] \tag{3.10}$$

where $s_p$ is a nonnegative integer slack variable and $\Lambda$ is a positive number found by the following rules:

**Step 1 :**

select a column With $p$ as the generating row, let $q$ be the lexicographically smallest column among those having $\bar{a}_{pj} < 0$ for all $j\epsilon J$.

**Step 2:**

Let $h_q = 1$, and for every $j \geq 1 (j \neq q)$ with $\bar{a}_{pj} < 0$, let $h_j$ be the largest integer satisfying

$$(1|h_j)p_j > p_q \qquad j\epsilon J \tag{3.11}$$

where $p_j$ denotes the column vector of elements $\bar{a}_{ij}, i = 0, 1, 2, \cdots, m$. The symbol '>' denotes lexicographically greater than.

**Step 3 :**
For each $\bar{a}_{pj} < 0 (j \geq 1)$, set

$$\lambda = max\ (\bar{a} - pj|h_j)_{jej} \tag{3.12}$$

Note that $\lambda_j$ is not necessarily an integer

# Chapter 4

# Lenstra's Algorithm

This Algorithm originally introduced by H.W. Lenstra was later simplified by A Paze [6]. It depicts an step by step method for solving Integer Programming Problem and Linear Diophantine Equations as given by Lenstra [7].

This work relies on Lenstra's paper [7], all the definations rermain the same as well as the notations.

## 4.1 Definitions

Let $K$ be a convex set of the form:

$$k = \{x \in R^n, (1, x)B \geq 0\} \tag{4.1}$$

where $B$ is an $(n+1) \times m$ matrix of integers and $(1, x) = (1, x_n, x_{n-1}, \cdots, x_1)$. The number of variables involved $(n)$ is the dimension of the set. $K$ can also be given as the convex hull of its vertices and those vertices can be found by solving at most $\binom{m}{n} \leq m^n$ system of n equations from $B$.
Let $v_1, v_2, c \cdots, v_i$ be the vertices of $K, l > 1$. Let $d$ be the dimension o linear subspace generated by the vector $v_j - v_0 \leq j \leq l$. Then $d$ will be called the rank of $K$.
If $d = n$ then $K$ is of full rank.

Consider the problem:

$$\sum_{i=n}^{1} a_i x_i = M \tag{4.2}$$

where the $a_i s$ and $M$ are assumed to be non negative integers and a solution is sought over the nonnegative integer.

An n-dimensional hyperplane is a set of points given in the form

$$P = \{x \in R^n : x\eta = M\} \tag{4.3}$$

where $\eta$ is a column vector of integers and $M$ is an integer.

A translate of a hyperplane is another hyperplane with same vector $\eta$ but a different $M$.

## 4.2 Intersection of convex set and hyperplane

Two convex set $k_1$ of dimensions $n_1$ and $k_2$ of dimensions $n_2$ will be called I-equivalent if there is a 1-1 mapping $\tau$ from $k_1$ to $k_2$ such that

$$x \in k_1 \cap Z^{n_1} \iff \tau x \in k_2 \cap Z^{n_2}$$

**Procedure 'Cut'**

Convex set of n-dimensions of the form:

$$K^{(0)} = \{x \in R^n : x\eta = M\}$$

we need to find a new convex set $K^{(1)}$ of dimensions n-1 and I-equivalent to $P \cap K^{(0)}$ of the form

$$k^{(1)} = \{y \in R^{n-1} : (1, y)B^{(1)} \geq 0\}$$

where $B^{(1)}$ is $n \times m$ with all integer entries.

Let $\eta = (a_n, a_{n-1}, \cdots, a_1)^\tau$ and assume that $(a_n, a_{n-1}, \cdots, a_1)$ are co-prime.
Procedure Steps: **1.** Let $f_n = a_n$ be definition and construct pairs of integers $(t_i, s_i); 1 \leq n-1$ satisfying

$$t_i f_{i+1} - s_i a_i = gcd(t_i, a_i) = f_i$$

Evaluate the pairs with extended Euclidean algorithm. After evaluation we get $a_n = f_n > f_{n-1} \cdots > f_k = f_{k-1} \cdots = f_1 = 1$ as they're co-prime $f_1 = 1$.

**2.** Construct the matrix

$$A = \begin{bmatrix} a_n & s_{n-1} & a_n s_{n-2}/f_{n-1} & a_n s_{n-3}/f_{n-2} & a_n s_k/f_{k+1} & 0 & \cdots & 0 \\ a_{n-1} & t_{n-1} & a_{n-1} s_{n-2}/f_{n-1} & a_{n-1} s_{n-3}/f_{n-2} & a_{n-1} s_k/f_{k+1} & 0 & \cdots & 0 \\ a_{n-2} & 0 & t_{n-2} & a_{n-2} s_{n-3}/f_{n-2} & a_{n-2} s_k/f_{k+1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{k+1} & & & & a_{k+1} s_k/f_{k+1} & 0 & \cdots & 0 \\ a_k & & & & t_k & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 1 & \cdots & 0 \\ a_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

where the lower right corner of $A$ is the unit matrix of dimension $(k-1) \times (k-1)$

Let $A[j]$ be the $j \times j$ upper left corner of $A$ and let $A[i, j+1]$ be the submatrix of A containing the first j rows and column 2 to $j+1$. It is proven by Induction that

$$det(A[j]) - f_{n-j+1} \quad 2 \leq j \leq n$$

$$det(A[1, j+1]) = s_{n-j} \quad 2 \leq j \leq n-1$$

this implies

$$det(A) = det[A(n)] = f_k = 1$$

Thus $A$ is unimodular matrix.
**3.** Construct $A^{-1}$. As the coefficients of $A$ are integer and $det(A) = 1$, $A^{-1}$ must have the same properties. Let $\hat{A}$ be the matrix resulting from $A^{-1}$ when it's first row is multiplied by M. For any $(n-1)$ dimensional vector $y$ we have that

$$(M, y)A^{-1} = (1, y)\hat{A}$$

**4.** Set

$$B^{(1)} = \begin{pmatrix} 1 & \\ 0 & \\ \cdots & \hat{A} \\ 0 & \end{pmatrix} B^{(0)}$$

and

$$k^{(1)} = \{y \in R^{n-1} \ \ (1, y)B^{(1)} \geq 0\}$$

**Lemma 1.** *$K^{(1)}$ is I-equivalent to $P \cap K^{(0)}$. By definition $B^{(1)}$ is $n \times m$.*

For any $x \in P \cap K^{(0)} \cap Z^n$. $y$ to be the $n-1$ last entries of the vector $xA = (M, y)$. Notice that, as $x \in L$ and the first column of $A$ is $y$, by construction, the first entry of $xA$ must be equal to $M$. The coefficients of $x$ and $A$ are integers implying that the coefficients of $y$ are integer too.
Given $y \in K^{(1)} \cap Z^{n-1}$

$$x = (1, y)\hat{A} = (M, y)A^{-1}$$

so that

$$x\eta = (M, y)A^{-1}\eta = (M, y)(1, 0, 0, \cdots, 0)^T = M$$

Again all the coefficients of $x$ are integers, following from the fact that the coefficients of $A^{-1}$ and $y$ are such. we have

$$(1, y)B^{(1)} = (1, y) \begin{pmatrix} 1 & \\ 0 & \\ \cdots & \hat{A} \\ 0 & \end{pmatrix} B^{(0)}$$

$$= (1,1,y)\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \cdots & \hat{A} & & \\ 0 & & & \end{pmatrix} B^{(0)} = (1,(1,y)A)B^{(1)}$$

$$= (1,(M,y)A^{-1})B^{(0)} = (1,x)B^{(0)}$$

Thus $(1,y)B^{(1)} \geq 0$ and $x\eta = M$.
**Remarks:**

1. Number of columns of $B^{(1)}$ are same as that of $B^{(0)}$ i.e the number of inequalities defining consecutive convex set remains same.

2. After the final reduction to dimension $n-1$, a new $(n-1)$ dimensional "cutting" hyperplane is provided allowing for another for another reduction to $n-2$ and so on. Now, find a integral vector in the original convex set corresponding to the integer in the final set (which has dimension 1). Done as follows:

Let the sequence of $\hat{A}$ matrices as constructed in stage 3 of the procedure be denoted by $\hat{A}^{(1)}, \hat{A}^{(2)}, \hat{A}^{(3)}, \cdots, \hat{A}^{(n-1)}$ where $\hat{A}^{(1)}$ is a $n \times n$ matrix and $\hat{A}^{(2)}$ is a $(n-1) \times (n-1)$ matrix, so on $\hat{A}^{(n-1)}$ is $2 \times 2$. Let $x^{(0)}, x^{(1)}, x^{(2)}, \cdots, x^{(n-1)}$ be the corresponding solution vectors i.e $x^{(0)}$ is $n-$dimensional, $x^{(1)}$ is $n-1$ and so on. Then

$$x^{(n-2)} = (1, x^{(n-1)})\hat{A}^{(n-1)}$$
$$x^{(n-3)} = (1, x^{(n-2)})\hat{A}^{(n-2)}$$

to

$$x^{(0)} = (1, x^{(1)})\hat{A}^{(1)}$$

Define $C^{(i)}$ as follows:

$$C^{(i)} = \begin{pmatrix} 1 & \hat{A}^{(i)} \\ 0 & \end{pmatrix}$$

Thus $(1, x^{(0)} = (1, x^{(n-1)})C^{n-1}$
Therefore, at the i-th iteration step find corresponding $C^{(i)}$ matrix which is computed out of $C^{(i-1)}$ and corresponding $A^{(i)}$ matrix.

## 4.3 Convex set whose Rank is not full

The second procedure, describes the reduction of dimension of convex set whose rank is not full, using previous procedure.
Let $K$ be given as in $(1)$ and $rank(K) < n$ **Procedure "Reduce Dimension"** Let $v_1; \cdots, v_l$ be the vertices of $K$, assumed to be of $rankd < n$. The procedure consists of the following steps.

1. Find a subset of the $v's$ $w_0, w_1, \cdots, w_d$ such that the vectors $w_i - w_0$ ,$1 \leq i \leq d$ span the space spanned by the vectors $v_i - v_0$ ,$1 \leq i \leq l$. Such a subset can be found easily, using methods from linear algebra.

2. Construct a determinant $D(v_j)$ as follows. The $i - th$ row of $D(v_j)$ for $1 \leq i \leq d + 1$ equals the vector $w_{i+1}$ extended to an $n + 1$ dimensional vector with last entry equal to 1. The $i - th$ row of $D(v_j)$ for $d + 2 \leq i \leq n$ equals the $(n + 1)$-dimensional vector with all its entries equal to 0 except for the $i - th$ entry which is equal to 1 unit vectors). The $(n + 1)$-st (and last) row of $d(v_j)$ is the vector $v_j$ extended to an $n + 1$-dimensional vector with its last entry equal to 1. As all the $v_j$'s are linear combinations of the $w_i$'s by definition, we have that $D(v_j) = 0, \leq j \leq i$. Let $x$ be the vector $x = (x_n, \cdots, x_1)$ It follows that all the vectors $v_j, 0 \leq j \leq M$ are included in the hyperplane defined by the linear equation $D(x) = 0$ and therefore the whole convex set is included in that hyperplane. We have thus found a hyperplane such that the given convex set can be defined in the form
$$K = P \cap K^{(0)}$$
where $K^{(0)} = conv(v_0, \cdots, v_i)$ and $P = \{(x_n, \cdots, x_1) : D(x) = 0\}$. As all the constant entries in $D(x)$ are rational $D(x) = 0$ can be expressed as hyperplane with integer coefficients.

3. Use procedure "Cut" in order to find $I$-equivalent convex set of dimension $n - 1$.

## 4.4 Basis Reduction

**Procedure "Unimodular Transformation"**

Given a basis $u_1, \cdots, u_n$ for a lattice $L$ such that all the coefficients of the vectors $u_i, 1 \leq i \leq n$ are integers. Let $U$ be a matrix whose rows are vectors $u_i$. Find a unimodular transformation $W$ such that the rows of the unimodular matrix $WU$ are **reduced** basis of $L$ ('reduced' as in [2]).We want $W$ only and not $WU$.

## 4.5 Finding the cutting hyperplane

Consider a convex set of the form

$$K = \{(x_n, \cdots, x_1) \in \mathbb{R}^n : \quad (1, x_n, \cdots, x_1)B \leq 0\}$$

as defined in (1). Assume that the vertices of $K$, denoted by $v_0, v_1, \cdots, v_l$ are given and assume that $K$ is of full rank. We want to find, a hyperplane of the form

$$P_M = \{(x_n \cdots x_1) \in \mathbb{R}^{(n))} : x\eta = M\}$$

as defined in (2) such that the number of translates of $P_M$ intersecting $K$ is proportional to number of integral points in the set $K \cap \mathbb{Z}^n$.

**Procedure "Cutting Planes"**

1. Choose $n+1$ vertices out of the given vertices such that the volume of n-simplex spanned by them is maximal. Le those vertices be $v_0, v_1, \cdots, v_n$. Construct a (non-singular) $n \times m$ matrix whose rows are the vectors $v_i - v_0, 1 \leq i \leq n$. Denote this matrix by $U$. Note that all entries in $U$ are rational.

2. Construct the matrix $U^{-1}$. The entries of $U^{-1}$ are rational. Let $U^{-1}$ be considered as an afine transformation $\mathbb{R}^n \to \mathbb{R}^n$. It follows from the definitions that $conv(v_0, \cdots, v_n)$ is transformed by

$U^{-1}$ into a simplex which is a translate of simplex whose vertices are origin with $n$ unit vectors, while $K$ is transformed into a convex set which is included in a translate of the simplex whose vertices are the origin and the unit vectors doubled.Moreover the natural is transformed by $U^{-1}$ into a lattice such that the rows of $U^{-1}$ can taken as basis for it. to be denoted by $u_1, \cdots, u_n$.

3. Use "Unimodular Transformation" procedure to get unimodular matrix $W$ for $U^{-1}$ above.
   Let $u'_i$ be the rows of the matrix $WU^{-1}$, we get

$$det(U^{-1}) \leq \Pi_n^{i=1}|u_i| \leq 2^{\frac{n(n-1)}{4}}\, det(U^{-1})$$

4. Let $u'_i$ be the rows of the matrix $WU^{-1}$, let $W = [w_{ij}]$ and let $|u'_j| = max\{u'_i| : 1 \leq i \leq n$. Let $W^{(j)}$ be the matrix resulting from $W$ when its $j$-th row is replaced by the row of variables $(x_{n1})$, Return the hyperplanes

$$P_M = \{(x_n \cdots x_1) : det(W^{(j)}) = M$$

For the proofs of whether the following procedure are correct go through [6]

## 4.6 Algorithm

Let our Integer programming algorithm be

$$IPA(n, K, C)$$

where $n$ is n-dimensional convex set $K = \{x \in \mathbb{R}^n : (1, x)B \geq 0\}$
$B$ is an $(n+1) \times m$ matrix of integers, and $K$ is assumed to be bounded so that $m \geq n+1$; $C$ is a $(n+1) \times m_1$ matrix of integer where $n+1 \leq m_1 \leq m+1$

1. if $n = 1$ then for every integer $x \in K$ ($x$ is vector dimension 1) return the $m_1$the dimensional vector $(1, x)C$ and stop.

2. Find the vertices and the dimension $d$ of $K$

3. id $d < n$ then apply procedure ”Reduce dimension” to $K$ with output (of procedure)$K^{(1)}$ and reducing matrix $\hat{A}^{(1)}$ then set

$$K \leftarrow K^{(1)}$$

$$C \leftarrow \begin{bmatrix} 1 & \\ 0 & \\ \cdots & \hat{A}^{(1)} \\ 0 & \end{bmatrix}$$

$$n \leftarrow n - 1$$

   apply $IPA(n, K, C)$.

4. Here $d = n$ apply procedure ”Cutting Planes” to $K$.

5. For every cutting hyperplane $P^M, M_0 \leq M \leq M_1$, returned in 4 and $K$ apply procedure ”Cut”. Let $K_M^{(1)}$ and $A_M^{(1)}$ be the remaining $(n-1)$ dimensional convex sets and corresponding reducing matrices. Set

$$K_M \leftarrow K_M^{(1)}$$

$$C_M \leftarrow \begin{bmatrix} 1 & \\ 0 & \\ \cdots & \hat{A}_M^{(1)} \\ 0 & \end{bmatrix} C$$

   For every $M_0 \leq M \leq M_1$
   Apply $IPA(n-1, K_M, C_M)$

6. For given $n$ and $K$ apply $IPA(n, K, B)$

## 4.7 Complexity:

All the steps of the above algorithm are polynomial (even linear) in the length of the input. The number of times recursion is applied is at most $n$ and whenever recursion is applied the coefficients grow (in their length) by a factor which depends on $n$-only. It follows that the algorithm is polynomial in the length of the input.

# Bibliography

[1] M. R. Garey and D. S. Johnson. Computers and Intractabihty, A Guide to the Theory of NP-Completenes. Freeman & Co. San Francisco, 1979.

[2] A. H. Land and A. G. Doig. "An Automatic Method of Solving Discrete Programming Problems". In: Econometrica 28.3 (1960), pp. 497–520. ISSN: 00129682, 14680262. URL: `http://www.jstor.org/stable/1910129` (visited on 07/18/2023).

[3] R.E. Gomory. "An algorithm for integer solutions to linear programs. Recent Advances in Mathematical Programming ed. Graves, R.L. and Wolfe, P. (Mcgraw Hill)". In: (1963).

[4] R.E. Gomory. "An all-integer Integer programming algorithm. Industrial Scheduling, Muth, J.F. and Thompson, G.L. (Prentice Hall)," in: (1963).

[5] R.E. Gomory. "n algorithm for the mixed integer problem. RAND Report RM2597, July 1960, in M. Simmonard (ed) Linear Programming, (Prentice Hall), Englewood Cliffs, NJ." In: (1966).

[6] A. Paz. A Simplified Version of H. W. Lenstra's Integer Programming Algorithm and Some Applications. Tech. rep. UCB/CSD-83-116. EECS Department, University of California, Berkeley, Aug. 1983. URL: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/1983/6327.html`.

[7] H. W. Lenstra. "Integer Programming with a Fixed Number of Variables". In: Mathematics of Operations Researc 8.4 (1983), pp. 538–548. ISSN: 0364765X, 15265471. URL: `http://www.jstor.org/stable/3689168` (visited on 08/03/2023).