Non-Archimedean Artificial Intelligence

teaching material

Lorenzo Fiaschi

a.y. 2024-2025

Course: Symbolic and Evolutionary Artificial Intelligence
University of Pisa
lorenzo.fiaschi@ing.unipi.it

Alpha Theory

Axiom 1 (Existence)

Exist an ordered field $\mathbb{E} \supset \mathbb{R}$ *and a function* $\lim\limits_{n\uparrow\alpha} \colon \mathbb{R}^{\mathbb{N}} \to \mathbb{E}$, *s.t.:*

- $c_r(n) = r \ \forall n \in \mathbb{N} \Longrightarrow \lim\limits_{n\uparrow\alpha} c_r(n) = r$
- $i(n) = n \ \forall n \in \mathbb{N} \Longrightarrow \lim\limits_{n\uparrow\alpha} i(n) = \alpha \notin \mathbb{N}$
- $\lim\limits_{n\uparrow\alpha} \varphi(n) + \lim\limits_{n\uparrow\alpha} \psi(n) = \lim\limits_{n\uparrow\alpha}(\varphi(n) + \psi(n))$
- $\lim\limits_{n\uparrow\alpha} \varphi(n) \cdot \lim\limits_{n\uparrow\alpha} \psi(n) = \lim\limits_{n\uparrow\alpha}(\varphi(n) \cdot \psi(n))$

- $\xi \in \mathbb{E}$ is called Euclidean number
- $\alpha \notin \mathbb{N}$
- α is infinite
- $\eta := \alpha^{-1} \in \mathbb{E}$

- η is infinitesimal

$$\frac{\alpha^7 - \alpha^\pi + \eta^\alpha}{-3 + 5e^\eta} \in \mathbb{E}$$

What do infinite and infinitesimal mean?

Definition (Infinite number)

$x \in \mathbb{E}$ *is infinite* $\overset{def}{\iff} \nexists\, k \in \mathbb{N}$ *such that* $k > |x|$.

Definition (Finite number)

$x \in \mathbb{E}$ *is finite* $\overset{def}{\iff} \exists\, k \in \mathbb{N}$ *such that* $k > |x| > \frac{1}{k}$.

Definition (Infinitesimal number)

$x \in \mathbb{E}$ *is infinitesimal* $\overset{def}{\iff} \nexists\, k \in \mathbb{N}$ *such that* $|x| > \frac{1}{k}$.

A “more concrete” interpretation of α

Definition (Numerosity α)

Exists a function $\mathfrak{num}$, $\mathfrak{num} : \mathcal{U} \to \mathbb{E}$ *which satisfies*

- $\mathfrak{num}(A) = |A|$, *if* A *is finite*
- $\mathfrak{num}(A) < \mathfrak{num}(B)$ *if* $A \subset B$
- $\mathfrak{num}(A \cup B) = \mathfrak{num}(A) + \mathfrak{num}(B) - \mathfrak{num}(A \cap B)$
- $\mathfrak{num}(A \times B) = \mathfrak{num}(A) \cdot \mathfrak{num}(B)$
- $\alpha := \mathfrak{num}(\mathbb{N})$

Axiom 2 (Internal sets)

If $\psi(n)$ is a sequence of nonempty sets, then

$$\lim_{n\uparrow\alpha}\psi(n)=\left\{\lim_{n\uparrow\alpha}\varphi(n)\,\middle|\,\varphi(n)\in\psi(n)\ \forall n\right\}.$$

Axiom 3 (Extension)

Let $\varphi\colon\mathbb{N}\to\mathbb{R}$ and $f\colon\mathbb{R}\to\mathbb{R}$ be a function for which $f\circ\varphi$ is defined. Then, f can be uniquely extended at $\xi=\lim_{n\uparrow\alpha}\varphi(n)$ as follow

$$^{*}f(\xi)=\lim_{n\uparrow\alpha}(f\circ\varphi)(n),$$

where the symbol $^{*}f$ indicates the extension of f to $\mathbb{E}$.

Definition (Elementary formula)

An elementary formula is a finite string of symbols in the first order language, i.e., a finite-lengths formula using as "alphabet" the variables (x, y, ...), the logic connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$), the quantifiers ($\exists$, $\forall$), the equality ($=$), and the membership ($\in$).

Theorem (Transfer Principle)

Let $\sigma(x_1, \ldots, x_k)$ be an elementary formula and let $\varphi_1, \ldots, \varphi_k$ be arbitrary sequences. Then

$$\sigma(\varphi_1(n), \ldots, \varphi_k(n)) \text{ a.e.} \iff \sigma(\lim_{n\uparrow\alpha} \varphi_1(n), \ldots, \lim_{n\uparrow\alpha} \varphi_k(n))$$

Transfer principle

- The Transfer Principle guarantees that some properties are **preserved** by the α-limit
- Examples: continuity, differentiability, etc.

$$\alpha \cdot (\alpha + 2) = \alpha^2 + 2\alpha$$

$$0 < \frac{1}{\alpha} = \alpha^{-1} < \alpha^0 = 1 < \alpha^1 = \alpha < (\alpha + 1)$$

$$\frac{-10.0\alpha^2 + 16.0 + 42.0\eta^2}{5.0\alpha^2 + 7.0} = -2.0 + 6.0\eta^2$$

- Case in which the Transfer Principle does not work?

Definition (Completeness)

X is complete $\overset{def}{\iff}$ every nonempty subset of X having an upper bound must have a least upper bound (or supremum) in X.

- Transfer Principle does not transfer $\mathbb{R}$ **completeness**. An example is the monad of zero, i.e., $\mu(0) \coloneqq \{x \in \mathbb{E} \,|\, x \approx 0\}$.
- Completeness in first order logic:

$$\forall X \in \mathcal{P}(\mathbb{R})(X \text{ “bounded above”} \implies (\exists x \in \mathbb{R}) \text{ “}x = \sup X\text{”})$$

- Non-standard version of the formula:

$$\forall X \in {}^*\mathcal{P}(\mathbb{R}) \ldots$$

Some questions and the Qualified Set Axiom

- Is $\mathbb{E}$ unique? No, yet $\mathbb{E} = {}^*\mathbb{R}$
- Is α even? Not necessarily.
- More generally, $\frac{\alpha}{n} \in {}^*\mathbb{N}\ \forall n \in \mathbb{N}$? Not necessarily.
- Similarly: $\sqrt{\alpha} \in {}^*\mathbb{N}$? $\sqrt[n]{\alpha} \in {}^*\mathbb{N}\ \forall n \in \mathbb{N}$? Not necessarily.

Axiom (Qualified set)

Let $Q := \{m!^{m!} \mid m \in \mathbb{N}\}$. *Then,* $\alpha \in {}^*Q$

- In general, any property of α must be coherent with the others
- $\frac{\alpha}{2} \in {}^*\mathbb{N} \ \wedge \ \alpha$ prime is false
- $\alpha \leq 0$ is false
- $\sin(\alpha) = 0$ is false

Numerical Alpha Theory

Algorithmic fields

- $\mathbb{R}$ **reference** set for theory about data science, machine learning, market analysis, etc.
- In practice, $\mathbb{R}$ is "**too large**" to fit in a computer, e.g., $\frac{1}{3}$, which is a repeating decimal
- Numbers representation is **not** always ***exact***, field properties do **not hold** anymore as well
- Common computers work with **floating point** numbers (IEEE 754 standard), which are a **finite-dimension** encoding
- ***Algorithmic Field***: set of all the numbers which can be **represented exactly** within a machine
- It may contain **symbols not present** in the original field it approximates: `NaN`, $\pm$`Inf`.

An example: lack of associativity

With finite precision it happens that $(-2^{127} + 2^{127}) + 1 = 1$ and $-2^{127} + (2^{127} + 1) = 0$, as reported in the following Matlab code:

```
>> m1 = single(-2^127);
>> m2 = single(2^127);
>> m3 = single(1);
```

```
>> (m1 + m2) + m3
>> ans = 1
>> m1 + (m2 + m3)
>> ans = 0
```

The importance of fixed length

- Fixed-length vs. Symbolic computations
- **Symbolic**:
  - Numbers variable-size typically slows a lot the computations
  - "Infinite precision"
  - **Iterative schemes** makes numbers inner representation grow at each iteration
  - Programs tend to **get slower** and slower during time
- **Fixed-length**:
  - **Standardized** operations, e.g., CPU instructions
  - **Faster** code
  - **Deterministic** time consumption
  - Possibility to build **hardware accelerators**, e.g., FPUs
  - **Noisy** operations which may notably **affect** the results

Algorithmic numbers (ANs)

Definition (Algorithmic Number)

$\xi \in \mathbb{E}$ is an Algorithmic Number $\overset{def}{\iff} \xi = \sum_{k=0}^{\ell} r_k \alpha^{s_k}$ *where* $r_k \in \mathbb{R}$, $s_k \in \mathbb{Q}$, $s_k > s_{k+1}$.

Theorem (AN normal form)

Any AN ξ can be represented as $\xi = \alpha^p P\left(\eta^{\frac{1}{m}}\right)$, *where* $p \in \mathbb{Z}$, $m \in \mathbb{N}$ *and* $P(x)$ *is a polynomial with real coefficients such that* $p_0 \neq 0$ *if* $x \neq 0$.

- Subset of $\mathbb{E}$ which can be **better manipulated** by computers
- Two main issues:
  - Not close with respect to inversion, e.g., $(\alpha + 1)^{-1}$
  - Variable length (simplifications between p and m)

Truncation and Bounded ANs (BANs)

Definition (Truncation)

Let $P(x) = p_0 x^{z_0} + \ldots + p_m x^{z_m}$, $z_{i-1} < z_i$, $i = 1, \ldots, m$. *Then*

$$\mathfrak{tr}_n [P(x)] := \begin{cases} P(x) & n \geq m \\ p_0 x^{z_0} + \ldots + p_n x^{z_n} & n < m \end{cases}$$

Definition (BAN)

$\xi \in \mathbb{E}$ *is a BAN* $\overset{def}{\Longleftrightarrow}$ ξ *is an AN and* $m = 1$.

$$\xi = \alpha^p P(\eta)$$

BANs library



```
abstract type AbstractAlgNum <: Number end

const SIZE = 3;

# Ban declaration
mutable struct Ban <: AbstractAlgNum
    # Members
    p::Int
    num::Array{T,1} where T<:Real

    # Constructor
    Ban(p::Int,num::Array{T,1}, check::Bool) where T <: Real = new(p,copy(num))
    Ban(p::Int,num::Array{T,1}) where T <: Real =
                                (_constraints_satisfaction(p,num) && new(p,copy(num)))
    Ban(a::Ban) = new(a.p,copy(a.num))
    Ban(x::Bool) = one(Ban)
    Ban(x::T) where T<:Real = ifelse(isinf(x), Ban(0, ones(SIZE).*x), one(Ban)*x)
end

# α constant
const α = Ban(1, [one(Int64); zeros(Int64, SIZE-1)], false);
# η constant
const η = Ban(-1, [one(Int64); zeros(Int64, SIZE-1)], false);
```

Non-Archimedean Artificial Intelligence

teaching material

Lorenzo Fiaschi

a.y. 2024-2025

Course: Symbolic and Evolutionary Artificial Intelligence
University of Pisa
lorenzo.fiaschi@ing.unipi.it

Introduction to LMOP

Lexicographic multi-objective optimization problems (LMOPs)

- Problem to **optimize** a scalar **function** (objective) over a set

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{D} \end{aligned}$$

- **Multi-objective**: series of objectives $f_1, f_2, \ldots, f_n$
- **Lexicographic**: objectives are ordered by priority

$$f_1 \gg f_2 \gg \ldots \gg f_n$$

- Analytical **model**:

$$\begin{aligned} \text{lex} \min_{x} \quad & f_1, f_2, \ldots, f_n \\ \text{s.t.} \quad & x \in \mathcal{D} \end{aligned}$$

Algorithm 1 Solving a LMOPs

$\Omega = \mathcal{D}$
for $i = 1, \ldots, n$ **do**
  if $\Omega == \emptyset$ **or** $|\Omega| == 1$ **then**
    break
  end if
  $\Omega_i := \left\{ \arg\min_{x \in \Omega} f_i(x) \right\}$
  $\Omega = \Omega_i$
end for
return Ω

LMOPs: preemptive approach

- Problem **reformulation**:

$$\begin{aligned} \min_{x} \quad & f_i(x) \\ \text{s.t.} \quad & f_j(x) = \min_{x \in \mathcal{D}} f_j(x) \quad \forall j = 1, \ldots, i-1, \\ & x \in \mathcal{D} \end{aligned}$$

- It leverages the **exact** definition of LMOP to solve it
  - It solves ***n*** different optimization **problems**
  - After each run, a **further constraint** is **added**
- **Issues**:
  - The single tasks to be solved get more and **more complex**
  - The **nature** of the problem may **change**
  - Bad use of **hardware**, e.g., cache
  - Bad use of **software** information, e.g., warm start

LMOPs: scalarization

- **Weighted** scalarization is applied
- Weights are **chosen** "in accordance to" the **priority**
- $\{w_i\}_{i=1}^n$ such that $\frac{w_{i+1}}{w_i} \ll 1$, $i = 1, \ldots, n-1$
- Problem **reformulation**:

$$\min_x \quad \sum_{i=1}^{n} w_i f_i(x) \quad \Longleftrightarrow \quad \min_x \quad w_1 f_1(x) + \ldots + w_n f_n(x)$$
$$\text{s.t.} \quad x \in \mathcal{D} \qquad\qquad \text{s.t.} \quad x \in \mathcal{D}$$

- Issues:
  - **No** guarantee of problems **equivalence** (new and original one)
  - The **choice** of the weights is **arbitrary** and error-prone
  - The choice of the weights is a **trial-and-error** approach
  - Very different weights may induce **numerical instabilities**

LMOPs: non-standard scalarization

- **Hybrid approach** between the previous two
- The weights choice is: $w_i = \alpha^{1-i}$, $i = 1, \ldots, n$
- Problem **reformulation**:

$$\min_x \quad \sum_{i=1}^{n} f_i(x)\alpha^{1-i} \quad \Longleftrightarrow \quad \min_x \quad f_1(x)\alpha^0 + \ldots + f_n(x)\alpha^{1-n}$$
$$\text{s.t.} \quad x \in \mathcal{D} \qquad\qquad \text{s.t.} \quad x \in \mathcal{D}$$

- It keeps the **positive aspects** of the two
  - Guarantee to **converge** to the optimal solution (if any)
  - Use of (non-standard) **single-objective** algorithms
  - The problems are **equivalent**
- Non-standard **versions exist** because of Transfer Principle
- The choice of weights is **not unique**

Theorem

Consider a LMOP whose objective functions $f_1, \ldots, f_n$ are real functions and the priority is induced by the natural order. Then, there exists an equivalent scalar program over the same domain, whose objective function is non-standard and has the following form:

$$F(x) = \beta_1 f_1(x) + \ldots + \beta_n f_n(x),$$

where $\beta_1, \ldots, \beta_n$ are non-standard values s.t. $\frac{\beta_{i+1}}{\beta_i} \approx 0$, $i = 1, \ldots, n$-1.

Proof.

Let Ω be the domain of the two programs, and $\omega \in \Omega$ be a global maximum of the lexicographic optimization problem, i.e., $\nexists \omega' \in \Omega$ such that $f_1(\omega') > f_1(\omega)$, or $\nexists i = 2, \ldots, n$ such that $f_j(\omega') = f_j(\omega)$, $j = 1, \ldots, i$-1, and $f_i(\omega') > f_i(\omega)$. However, this is true if and only if $\nexists \omega' \in \Omega$ such that $F(\omega') > F(\omega)$, by the definition of F itself. The latter fact precisely means that ω is a global maximum for the scalar non-standard program.

LMOLP

Lexicographic multi-objective linear programming (LMOLP)

- **Model**:

$$\begin{array}{ll} \text{lex}\min\limits_{x} & c_1^T x, \ldots, c_n^T x \\ \text{s.t.} & Ax = b, \\ & x \geq 0 \end{array} \iff \begin{array}{ll} \min\limits_{x} & \sum\limits_{i=1}^{n} c_i^T x \alpha^{1-i} \\ \text{s.t.} & Ax = b, \\ & x \geq 0 \end{array}$$

- Solving algorithm: **NA-c-Simplex**
- Simplex coping with **non-standard cost functions**
- It **keeps** "all" the **properties** of its standard version
- The **transfer principle** holds for the set of **linear functions**

A toy example: Kite[1]

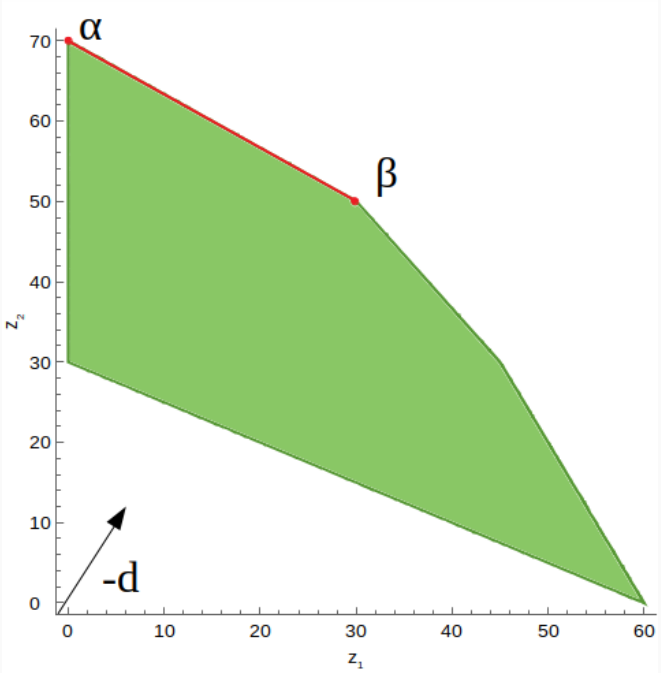


Figure 1: Standard Kite

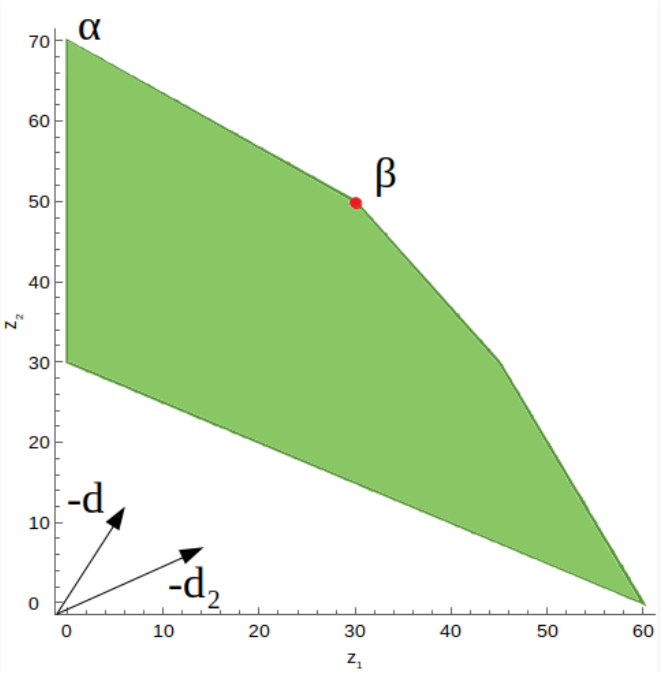


Figure 2: NA Kite

$$\begin{aligned} \underset{x}{\text{lexmax}} \quad & 8x_1 + 12x_2,\ 14x_1 + 10x_2 \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

$$\begin{aligned} \max_{x} \quad & (8 + 14\eta)x_1 + (12 + 10\eta)x_2 \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

[1] https://doi.org/10.1016/j.amc.2017.05.058

A toy example: Kite

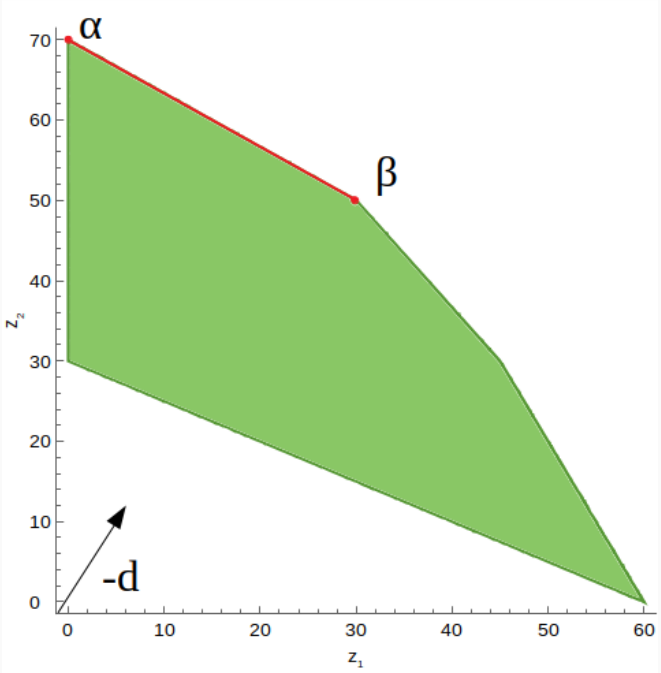


Figure 3: Standard Kite

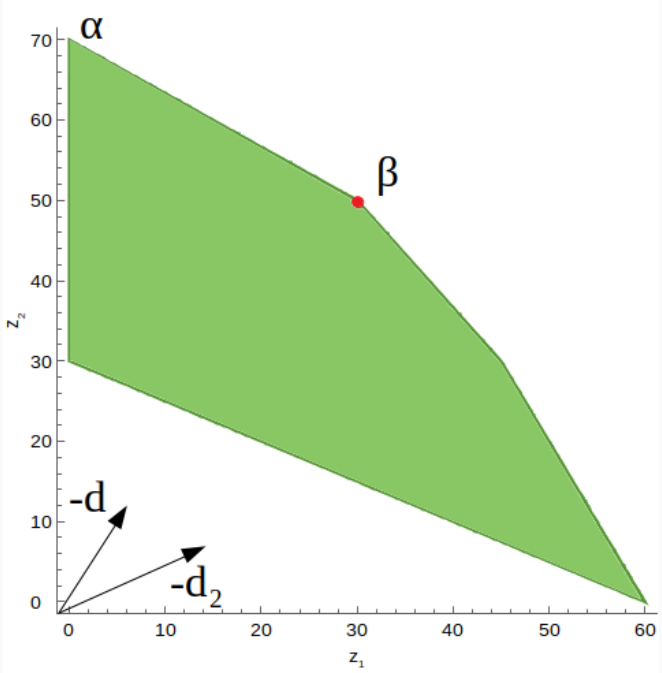


Figure 4: NA Kite

Table 1: NA-c-Simplex iterations solving NA Kite

Iter.	B	$\boldsymbol{x}$	$\boldsymbol{c}^T\boldsymbol{x}$
1	$\{4, 5\}$	$[0,\ 30]^T$	$360 + 300\eta$
2	$\{2, 4\}$	$[0,\ 70]^T$	$840 + 700\eta$
3	$\{2, 3\}$	$[30,\ 50]^T$	$840 + 920\eta$

I-Big-M

Towards another application: penalization terms

- What happens if $\mathtt{B}$ is **not available**?
- Problem **embedding** adding slack variables

$$Ax = b \Longrightarrow Ax + Is = b$$

- **Guarantee** of problem **feasibility**
- Feasible starting basic solution exists: $\mathtt{B}'$

$$(x, s) = (\mathbf{0}, b) \text{ is feasible} \Longrightarrow \mathtt{B}' = \{n+1, \ldots, n+m\}$$

- How to make the two problems **equivalent**?
  - Two Phases method
  - Big-M method

Two Phases method

- First phase: seeking for **feasibility** in the **original** problem

$$\begin{aligned} \min_{x,\,s} \quad & \mathbf{1}^T s \\ \text{s.t.} \quad & Ax + Is = b, \\ & x, s \geq 0 \end{aligned}$$

  - Original problem feasible solution have cost zero
  - Simplex starting basis: $\mathtt{B}'$
  - If $\overline{s} = 0$ move to the second phase
  - Otherwise, the original problem is infeasible
- Second phase: **solve the original problem**
  - $x = \overline{x}$
  - $\mathtt{B} := \mathtt{B}'[1 : n]$
  - Use Simplex algorithm using $\mathtt{B}$ as starting basis

Big-M method

- Tentative to solve the problem in one shot

$$\begin{aligned} \min_{x,\, s} \quad & \mathbf{1}^T sM + d^T x \\ \text{s.t.} \quad & Ax + Is = b, \\ & x, s \geq 0 \end{aligned}$$

- $M \geq 0$ very **big penalization** weight (Big-M)
- Arbitrary choice

Theorem

$\exists\, M \geq 0$ sufficiently large such that $\overline{x}' = [\mathbf{0}, \overline{x}]$ is solution of the larger problem $\Longleftrightarrow$ $\overline{x}$ is solution of the original problem.

The Infinitely Big-M method (I-Big-M)[2]

- What really is the **role** of M or of the two phases?
- To force s to $\mathbf{0}$
- They **prioritize** the zeroing over the minimization of $d^T z$
- Priority means **lexicographicity**
- Idea: to ***always*** set $M = \alpha$
- This **solves** all the **issues** of Big-M method
  - The problem is **no** more **parametric**
  - "No" **numerical instabilities** using infinite numbers
  - Guarantee of **convergence** because α is always big enough

[2] https://link.springer.com/article/10.1007/s11590-020-01644-6

I-Big-M at work: Standard Kite

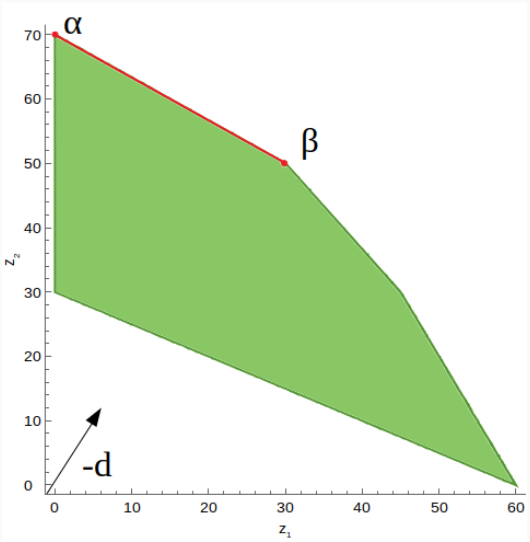


Figure 5: Standard Kite

$$
\begin{aligned}
\max_{x'} \quad & -(e+r)\alpha + 8x_1 + 12x_2 + 7x_3 \\
\text{s.t.} \quad & 2x_1 + x_2 - 3x_3 + s_1 = 90, \\
& 2x_1 + 3x_2 - 2x_3 + s_2 = 190, \\
& 4x_1 + 3x_2 + 3x_3 + s_3 = 300, \\
& x_3 + e = 10, \\
& x_1 + 2x_2 + x_3 + r - p = 70, \\
& x,\ s,\ e,\ r\ p \geq 0
\end{aligned}
$$

Iter.	x'	$c^T x'$	Basis	x
1	$[0, 0, 0, 90, 190, 300, 10, 70, 0]$	-80α	{4,5,6,7,8}	$[0, 0, 0]$
2	$[0, 35, 0, 55, 85, 195, 10, 0, 0]$	$-10\alpha + 420$	{4,5,6,7,2}	$[0, 35, 0]$
3	$[0, 30, 10, 90, 120, 180, 0, 0, 0]$	430	{4,5,6,3,2}	$[0, 30, 10]$
4	$[0, 70, 10, 50, 0, 60, 0, 0, 0, 80]$	910	{4,9,6,3,2}	**[0,70,10]**

I-Big-M effectiveness

- Is the I-Big-M method **truly effective** in real world benchmarks?
- Benchmark neos-1425699, maximization problem
- 63 inequalities; 26 equalities
- I-Big-M finds the optimal solution at the **first attempt**

Table 2: A real problem where setting M in Big-M method is not trivial

Problem: **neos-1425699.mps** (known optimal value of the cost function: $\mathbf{3.14867 \cdot 10^9}$)		
Value of M	**Optimum cost function found by Big-M**	**Number of iterations**
10^2	$1.36403 \cdot 10^7$	26
10^3	$1.29867 \cdot 10^8$	26
10^4	$1.25666 \cdot 10^9$	30
10^5	$2.95207 \cdot 10^9$	88
10^6	$2.95535 \cdot 10^9$	88
10^7	$2.98818 \cdot 10^9$	88
10^8	$\mathbf{3.14867 \cdot 10^9}$	89
10^9	$\mathbf{3.14867 \cdot 10^9}$	89