FAST ALGORITHM FOR FINDING LATTICE SUBSPACES IN $\mathbb{R}^n$ AND ITS IMPLEMENTATION

ANDREW M. POWNUK

THE UNIVERSITY OF TEXAS AT EL PASO

Goal and Objective

In the literature there are known algorithms with exponential complexity that determine if a given subspace is lattice-ordered.

In this presentation a polynomial time algorithm (serial and parallel) as well as its computer implementation will be presented.

The method can be applied in economics as well as in the theory of vector lattices.

Minimum-cost Portfolio Insurance

In economics it is possible to prove that the minimum-cost insured portfolio exists if and only if the linear space generated by the corresponding financial instruments is lattice-ordered.

Theorem The minimum-cost insured portfolio exists and is price independent for every portfolio and at every floor if and only if the asset span is a lattice subspace of $\mathbb{R}^S$. In this case, the minimum-cost insured portfolio θ^k satisfies $X\left(\theta^k\right) = X\left(\theta^k\right) \vee_M k$.

Source: C.D. Aliprantis, D.J. Brown, and J. Werner, Minimum-cost portfolio insurance,

Journal of Economic Dynamics & Control, 2000, Vol. 24, pp. 1703-1719.

The payoff of security *n* in *S* states is a vector $x_n \in \mathbb{R}_+^S$.

The payoffs $x_1, \ldots, x_N$ are assumed linearly independent.

For a portfolio $\theta = \left(\theta_1, \ldots, \theta_N\right) \in \mathbb{R}^N$, its payoff is

$$X\left(\theta\right) = \sum_{n=1}^{N} \theta_n x_n .$$

The set of payoff of all portfolios is the linear span of payoffs $x_1, x_2, \ldots, x_N$ in the space $\mathbb{R}^S$ of all state contingent claims and is the *asset span* $\mathcal{M}$.

A contingent claim is a marketed payoff if it lies in the asset span $\mathcal{M} = Span\left(\left\{x_1, x_2, \ldots, x_N\right\}\right)$.

It is assumed that the risk-free payoff is marketed, so that $\mathbb{1} \in \mathcal{M}$.

Let $p = \left(p_1, \ldots, p_N\right) \in \mathbb{R}^N$ be a vector of security prices.

A non-zero portfolio θ with positive payoff $X\left(\theta\right) \geq 0$ and zero or negative value $p \cdot \theta \leq 0$ is an *arbitrage portfolio*.

A security price vector $p \in \mathbb{R}^N$ is arbitrage-free if there is no *arbitrage portfolio*, that is, if $p \cdot \theta > 0$ for all non-zero portfolios θ with $X\left(\theta\right) \geq 0$.

Theorem

If $p \cdot \theta \geq 0$ for every arbitrage-free price vector p, then $X\left(\theta\right) \geq 0$.

Source: C.D. Aliprantis, D.J. Brown, and J. Werner, Minimum-cost portfolio insurance,

Journal of Economic Dynamics & Control, 2000, Vol. 24, pp. 1703-1719.

The insured payoff on a portfolio θ at a "floor" $\mathbb{k}$ is the contingent claim $X\left(\theta\right) \vee \mathbb{k}$. This contingent claim may or may not be marketed (element of $\mathcal{M}$).

The minimum cost insurance provides a payoff that dominates the insured payoff at the minimum cost.

Formally, the minimum-cost portfolio insurance is defined by the following minimization problem:

$$\begin{cases} \min\limits_{\eta \in \mathbb{R}^N} p \cdot \eta \\ s.t. X\left(\eta\right) \geq X\left(\theta\right) \vee \mathbb{k} \end{cases}$$

where $X\left(\theta\right) \vee \mathbb{k}$ is the insured payoff and $\mathbb{k} = k\mathbb{1}$ (k is the strike price).

This linear programming problem has a unique solution as long as p is arbitrage-free. We denote the solution by θ^k and refer to it as the *minimum-cost insured portfolio*.

Theorem The minimum-cost insured portfolio exists and is price independent for every portfolio and at every floor if and only if the asset span is a lattice subspace of $\mathbb{R}^S$. In this case, the minimum-cost insured portfolio θ^k satisfies $X\left(\theta^k\right) = X\left(\theta\right) \vee_M k$.

Source: C.D. Aliprantis, D.J. Brown, and J. Werner, Minimum-cost portfolio insurance,

Journal of Economic Dynamics & Control, 2000, Vol. 24, pp. 1703-1719.

Theorem

(Abramovich-Aliprantis-Polyrakis, 1994). The asset span $\mathcal{M}$ is a lattice-subspace of $\mathbb{R}^S$ if and only if there is a fundamental set of states.

Example

$$x_1 = (1,1,1), x_2 = (0,1,2)$$

$$\mathcal{M} = Span(\{x_1, x_2\})$$

$$\mathcal{M} = \{\theta_1(1,1,1) + \theta_2(0,1,2) : \theta_1 \in \mathbb{R}, \theta_2 \in \mathbb{R}\} \subseteq \mathbb{R}^3$$

$$\dim \mathcal{M} = 2$$

$\mathbb{1} = (1,1,1) \in \mathcal{M}$ then $\mathcal{M}$ is a lattice-subspace.

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$ then $y_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, y_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, y_3 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$ then $y_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, y_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, y_3 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$ or $y_2 = \alpha_{2,1} y_1 + \alpha_{2,3} y_3$

Where $\alpha_{2,1} = \frac{1}{2} \geq 0, \alpha_{2,3} = \frac{1}{2} \geq 0$

and $\dim\left(Span\left(\left\{y_1, y_2\right\}\right)\right) = 2$ then y_1, y_3 are fundamental.

Minimum portfolio insurance is a solution of the following optimization problem

$$\begin{cases} \min\limits_{(\eta_1,\eta_2)\in\mathbb{R}^2} p_1\eta_1 + p_2\eta_2 \\ (1,1,1)\eta_1 + (0,1,2)\eta_2 \geq (1,1,2) \end{cases} \Rightarrow \left(1,\frac{1}{2}\right)$$

where insured payoff is $X(\theta) \vee \mathbb{k} = x_2 \vee \mathbb{1} = (1,1,2)$.

Then $\left(1,\frac{1}{2}\right)$ is minimum-cost insured portfolio at every arbitrage-free price p.

Theorem Suppose that there exists a fundamental set of states F for the asset span $\mathcal{M}$. Then for every arbitrage-free price system p and for every portfolio θ and floor *k*, the minimum-cost insured portfolio θ^k is the unique portfolio that replicates the insured payoff $X\left(\theta\right) \vee \mathbb{k}$ in the fundamental states. That is,

$$X\left(\theta^k\right) =_F X\left(\theta\right) \vee \mathbb{k}$$

The portfolio θ^k is the solution to the equation $X_F \theta^k =_F X\left(\theta\right) \vee \mathbb{k}$, that is, $\theta^k =_F \left(X_F\right)^{-1} \left[X\left(\theta\right) \vee \mathbb{k}\right]_F$

In the example of two securities with payoffs $x_1 = \mathbb{1}$ and $x_2 = \left(0,1,2\right)$, the insured payoff on security 2 at "floor" $k = 1$ is the contingent claim $X\left(\theta\right) \vee \mathbb{k} = x_2 \vee \mathbb{1} = \left(1,1,2\right)$ and is not in the asset span. Since states y_1 and y_3 are fundamental, the minimum-cost insurance on security x_2 replicates the claim $\left(1,1,2\right)$ in states 1 and 3. The portfolio $\left(1,\frac{1}{2}\right)$ has payoff $x_1 1 + x_2 \frac{1}{2} = \left(1,\frac{3}{2},2\right)$ and provides the minimum-cost insurance at arbitrary arbitrage-free prices.

Partially ordered set is a pair $\left(P, \leq\right)$ where P is a set and $\leq$ is a relation such that:

1) $a \leq a$ (reflexivity),

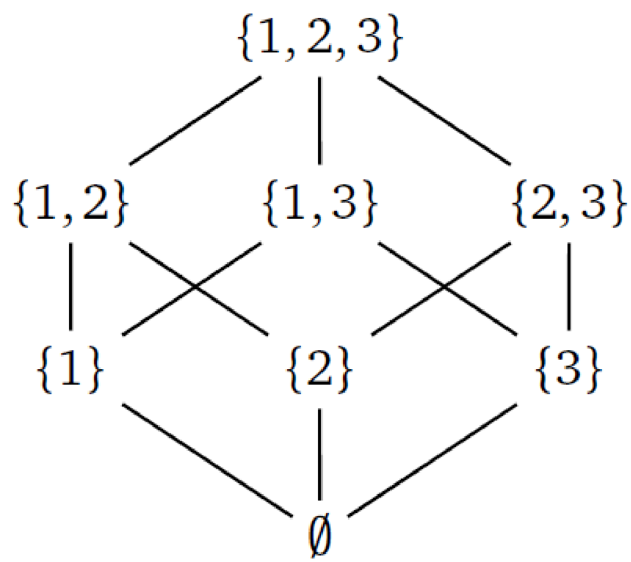
2) if $a \leq b$ and $b \leq a$ then $a = b$ (antisymmetry),

3) if $a \leq b$ and $b \leq c$ then $a \leq c$ (transitivity).

Example, A pair $\left(\mathbb{R}^2, \leq_{\mathbb{R}^2}\right)$ is an example of the partial set and for example $\left(1,2\right) \leq_{\mathbb{R}^2} \left(3,5\right) \Leftrightarrow \left(1 \leq 3\right) \text{and} \left(2 \leq 5\right)$

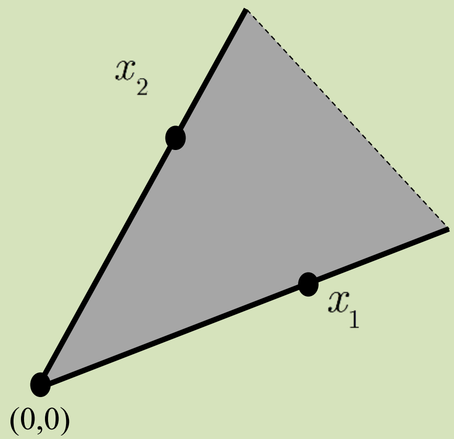
A **lattice** is a partially ordered set in which every two elements have a least upper bound and also called a greatest lower bound.

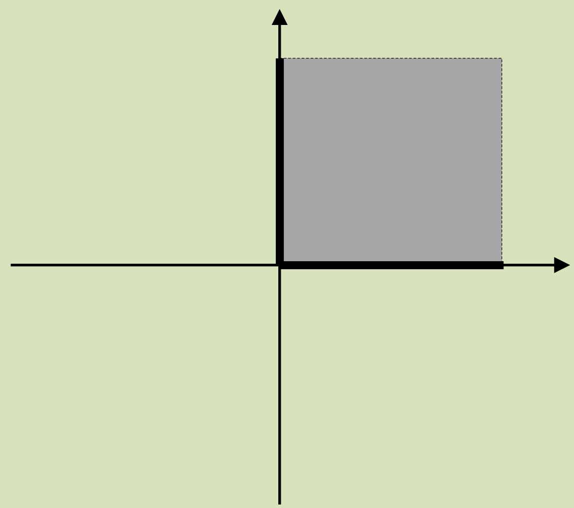
Example $\left(2^{\{1,2,3\}}, \subseteq\right)$



Generalized inequality

$$x \ge y \Rightarrow x - y \in K$$





$$\left(1,2\right) \le_{\mathbb{R}^2} \left(3,5\right) \Leftrightarrow \left(3-1 \ge 0\right) \text{and} \left(5-2 \ge 0\right)$$

V.N. Katsikis, Computational methods in portfolio insurance, Applied Mathematics and Computation, 189, 1, pp.9-22, 2007.

The algorithm requires $\binom{m}{n}$ steps, which grows exponentially with m.

$$2^{\frac{n}{2}} \le \binom{n}{\frac{n}{2}} \le (2e)^{\frac{n}{2}}$$

Definition

A set of n indices $\{m_1,\ldots,m_n\}$ is called a *negative fundamental set* of indices for the vectors $x_1,\ldots,x_n \in \mathbb{R}^m$ whenever the n vectors $y_{m_1},\ldots,y_{m_n}$ are linearly independent; and for at least one $j \in \{m_1,\ldots,m_n\}$, all the coefficients in the expansion $y_j = \sum_{r=1}^{n} \alpha_{j,r} y_{m_r}$ are non-positive.

Definition

A solution ξ to the equation $b = \sum_{i=1}^{n} \xi_i x_i$ is called *basic nonnegative solution* if for the set $L = \left\{ i : \xi_i > 0 \right\}$, the set of vectors $\left\{ y_i : i \in L \right\}$ is linearly independent.

Example

$$X = \begin{bmatrix} 1 & 1 & 0 & 4 & 1 \\ 2 & 3 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Fundamental set of indices.

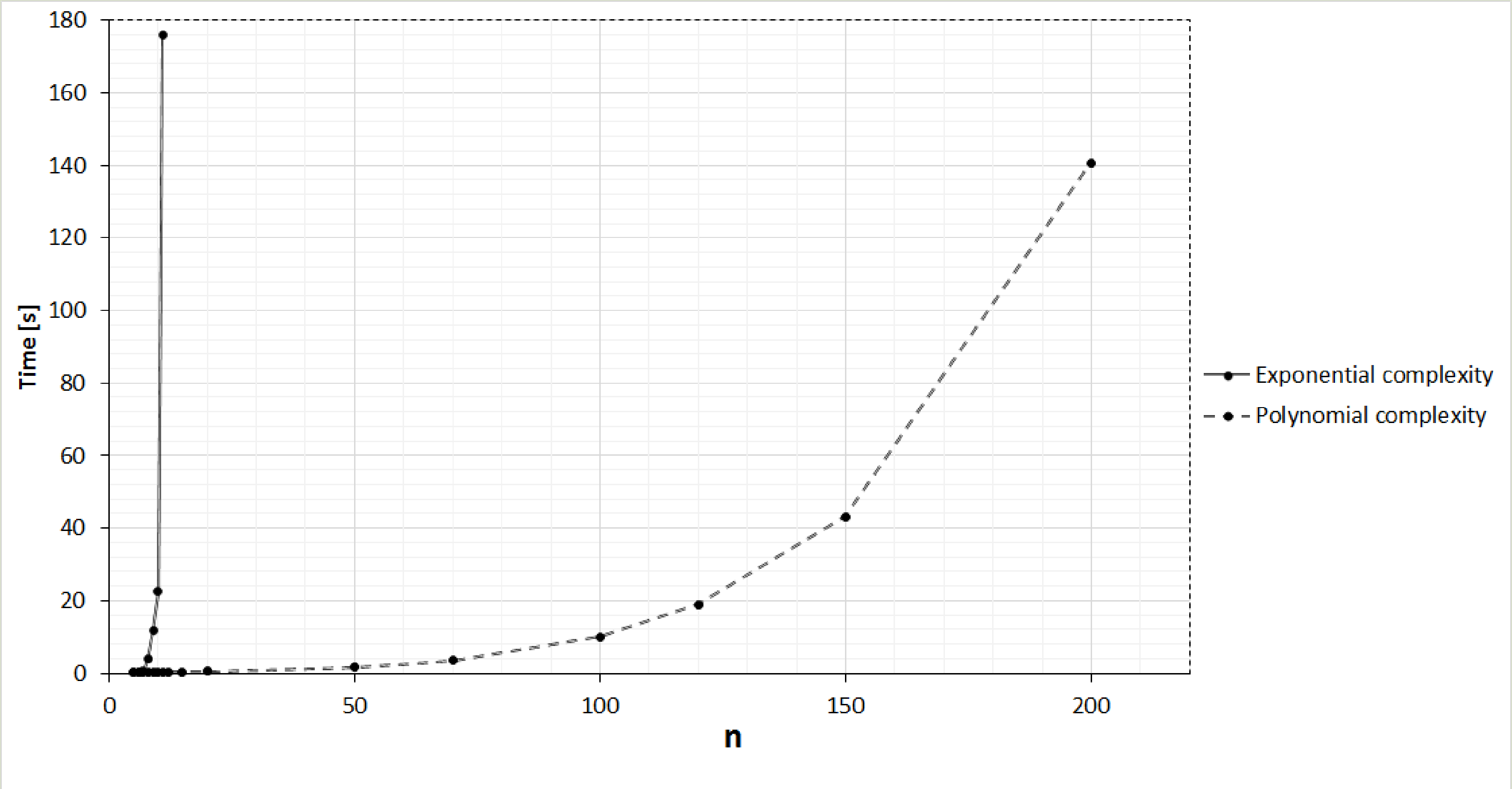
$$I = \left[0, 2, 3, 4, 0\right], Y_I = \begin{bmatrix} 1 & 0 & 4 \\ 3 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Lattice “YES”.

J.J. Del Valle, V. Kreinovich, and P.J. Wojciechowski, Feasible algorithms for lattice and directed subspaces, Mathematical Proceedings of the Royal Irish Academy, Vol. 112A, No. 2, pp. 199-204, 2014.

```
GET_FUNDAMENTAL_INDEX( m,Y )
{
    INDEX:= {1,...,m} ;
    Z := Y ;
    for( i := 1; i <= m; i++)
        if NONNEGCOMB( y[i], Z )
        { Z := Z \ {y_i} ;
        INDEX := INDEX \ {i} ; }
    return INDEX, PREFUND( Y ,INDEX);
}
```

Time of calculations



Calculations was done on Dell Precision 690 with two quad-core processors Intel Xeon X5365 and 16 GB memory and MATLAB Version 8.0.0.783 R2012b.

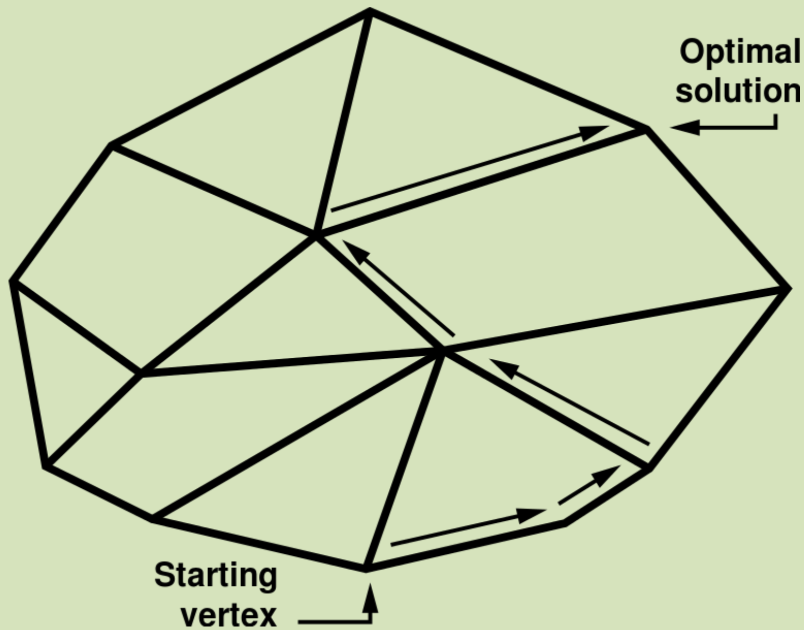
Computational Complexity of the Linear Programming Problem

$$\min c^T x$$

$$s.t. \begin{cases} Ax \le b \\ x \ge 0 \end{cases}$$

L.V. Kantorovich, A new method of solving some classes of extremal problems, Doklady Akad Sci USSR, 28, 1940, 211-214.

G.B Dantzig, Maximization of a linear function of variables subject to linear inequalities, 1947. Published pp. 339–347 in T.C. Koopmans (ed.): Activity Analysis of Production and Allocation, New York-London 1951 (Wiley & Chapman-Hall).



Interior Point Method

N. Karmarkar, A new polynomial time algorithm for linear programming, Combinatorica, 4, 1984.

$O\left(n^4 L\right)$ - computational complexity

KKT conditions

$$L = c^T x + (Ax - b)^T y - x^T y$$

$$\begin{cases} \nabla_x f(x) + \nabla_x h^T(x) y - z = 0 \\ h(x) = 0 \\ XZe = 0 \end{cases} \Rightarrow \begin{cases} c + A^T y - z = 0 \\ Ax - b = 0 \\ XZe = 0 \end{cases} \Rightarrow \begin{cases} A^T y - z = -c \\ Ax = b \\ XZe = 0 \end{cases}$$

$$\nabla_x f(x) = \nabla_x c^T x = c$$

Perturbed KKT conditions

$$\begin{cases} \nabla_x f(x) + \nabla_x h^T(x) y - z = 0 \\ h(x) = 0 \\ XZe = \boldsymbol{\mu}_k e \end{cases}$$

Vector form of the equations

$$F_{\mu_k}(X) = \begin{bmatrix} \nabla_x f(x) + \nabla_x h^T(x) y - z \\ h(x) \\ XZe - \boldsymbol{\mu}_k e \end{bmatrix}$$

$$X = (x, y, z)$$

Newton method for $F_{\mu_k}(X) = 0$

1) For initial points $x_0, z_0 \in \mathbb{R}^n$, $y_0 \in R^m$ and $\mu_0 \in R$

2) For k=0,1,2,... until convergence

3) Newton step $F_{\mu}^{'}(X_k)\Delta X_k = -F_{\mu}\left(X_k\right)$

4) Update $X_{k+1} = X_k + \Delta X_k$

Newton's steps

$$F_{\mu}^{'}(X_k)\Delta X_k = -F_{\mu}\left(X_k\right)$$

$$\begin{bmatrix} \left(\nabla_x^2 f(x) + \nabla_x^2 h(x) y\right)_{n\times n} & \nabla_x h(x)_{n\times m} & -I_{n\times n} \\ \left(\nabla_x h(x)\right)^T_{m\times n} & 0_{n\times m} & 0_{m\times n} \\ Z_{n\times n} & 0_{n\times m} & X_{m\times n} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = -\begin{bmatrix} \nabla_x f(x) + \nabla_x h^T(x) y - z \\ h(x) \\ XYe - \mu_k e \end{bmatrix}$$

S. Mehrotra , On the Implementation of a Primal-Dual Interior Point Method, SIAM Journal on Optimization, Vol. 2, pp 575–601, 1992.

Y. Zhang, Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment, Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD, Technical Report TR96-01, July, 1995.

XSEDE

Extreme Science and Engineering Discovery Environment

TACC

Parallel Method

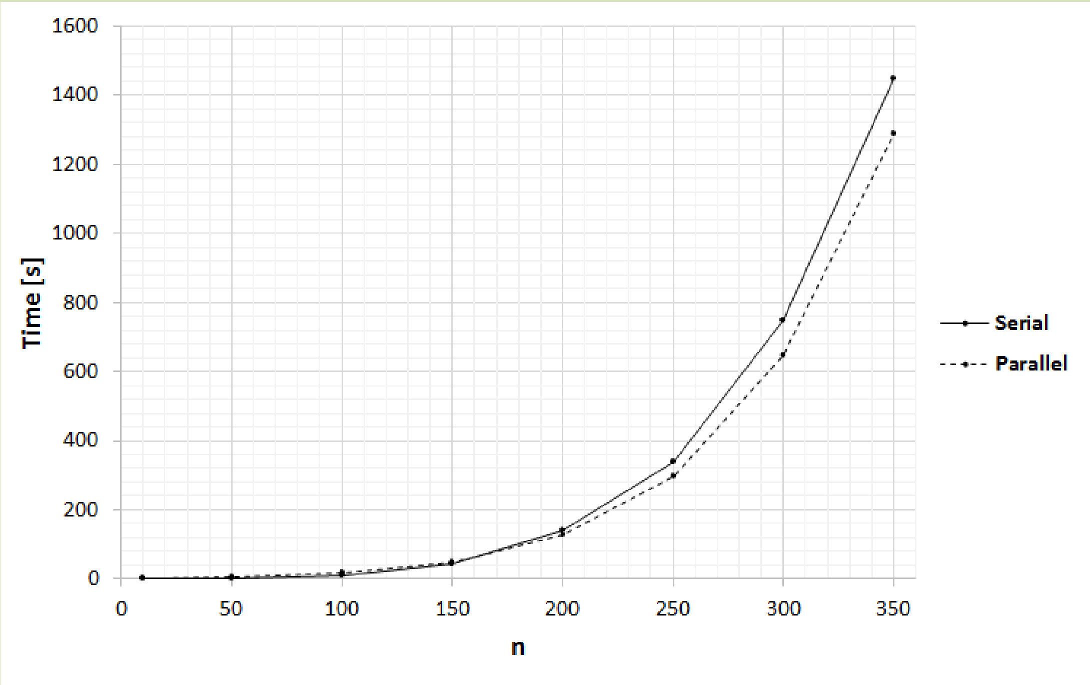
```
GET_FUNDAMENTAL_INDEX_PARALLEL( m,Y )
{
    INDEX:= {1,...,m}_{1×m} ;
    Z := Y ;
    while( NUMBER_OF_VECTORS_TO_BE_REMOVED( Z ) > 0 ){
        FIND_NONEGATIVE_VECTORS _PARALLEL( INDEX,Z );
        REMOVE_NONEGATIVE_VECTOR( INDEX,Z );
    }
return INDEX, PREFUND( Y ,INDEX);
}
```

Example

$$X_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 6 & 10 & 15 & 21 & 28 & 36 & 45 & 55 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 10 & 18 & 28 & 40 & 54 & 70 & 88 & 108 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 9 & 9 & 21 & 36 & 54 & 75 & 99 & 126 & 156 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 12 & 24 & 16 & 36 & 60 & 88 & 120 & 156 & 196 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 5 & 15 & 30 & 50 & 25 & 55 & 90 & 130 & 175 & 225 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 6 & 18 & 36 & 60 & 90 & 36 & 78 & 126 & 180 & 240 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 7 & 21 & 42 & 70 & 105 & 147 & 49 & 105 & 168 & 238 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 8 & 24 & 48 & 80 & 120 & 168 & 224 & 64 & 136 & 216 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 27 & 54 & 90 & 135 & 189 & 252 & 324 & 81 & 171 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 10 & 30 & 60 & 100 & 150 & 210 & 280 & 360 & 450 & 100 \end{bmatrix}$$

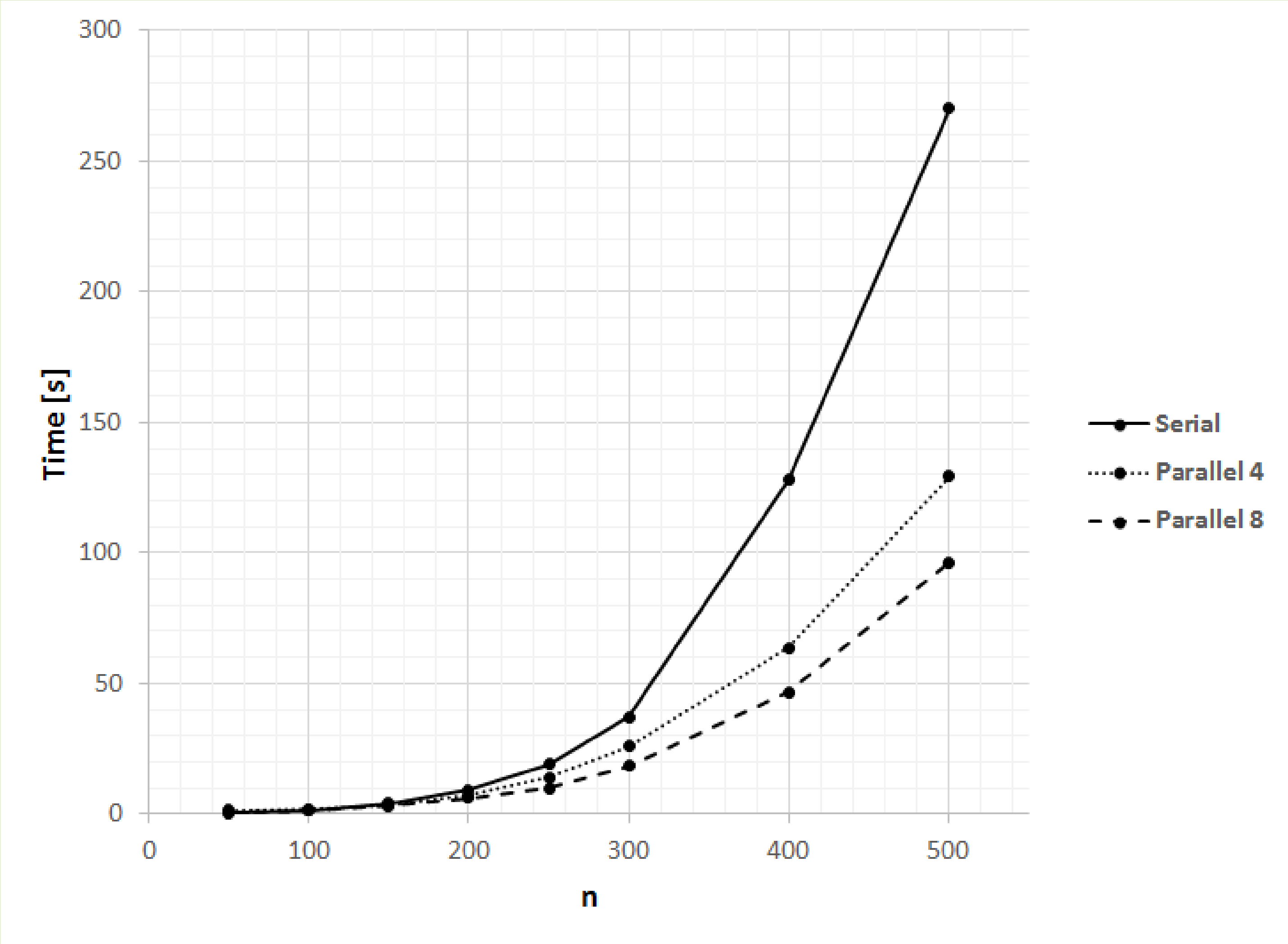
$$X_2 = \begin{bmatrix} 1 & 1 & 0 & 3 & 0 & 6 & 0 & 10 & 0 & 15 & 0 & 21 & 0 & 28 & 0 & 36 & 0 & 45 & 0 & 55 \\ 0 & 2 & 2 & 4 & 0 & 10 & 0 & 18 & 0 & 28 & 0 & 40 & 0 & 54 & 0 & 70 & 0 & 88 & 0 & 108 \\ 0 & 3 & 0 & 9 & 3 & 9 & 0 & 21 & 0 & 36 & 0 & 54 & 0 & 75 & 0 & 99 & 0 & 126 & 0 & 156 \\ 0 & 4 & 0 & 12 & 0 & 24 & 4 & 16 & 0 & 36 & 0 & 60 & 0 & 88 & 0 & 120 & 0 & 156 & 0 & 196 \\ 0 & 5 & 0 & 15 & 0 & 30 & 0 & 50 & 5 & 25 & 0 & 55 & 0 & 90 & 0 & 130 & 0 & 175 & 0 & 225 \\ 0 & 6 & 0 & 18 & 0 & 36 & 0 & 60 & 0 & 90 & 6 & 36 & 0 & 78 & 0 & 126 & 0 & 180 & 0 & 240 \\ 0 & 7 & 0 & 21 & 0 & 42 & 0 & 70 & 0 & 105 & 0 & 147 & 7 & 49 & 0 & 105 & 0 & 168 & 0 & 238 \\ 0 & 8 & 0 & 24 & 0 & 48 & 0 & 80 & 0 & 120 & 0 & 168 & 0 & 224 & 8 & 64 & 0 & 136 & 0 & 216 \\ 0 & 9 & 0 & 27 & 0 & 54 & 0 & 90 & 0 & 135 & 0 & 189 & 0 & 252 & 0 & 324 & 9 & 81 & 0 & 171 \\ 0 & 10 & 0 & 30 & 0 & 60 & 0 & 100 & 0 & 150 & 0 & 210 & 0 & 280 & 0 & 360 & 0 & 450 & 10 & 100 \end{bmatrix}$$

	Time of the calculations [s]	
n	Serial Method	Parallel Method
10	0.75	1.8
50	1.7	5.2
100	10	16.4
150	43.2	48.8
200	140.6	126
250	340.2	298
300	750	647
350	1449	1287



Example

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 9 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 12 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 5 & 15 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 6 & 18 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 7 & 21 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 8 & 24 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 27 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 10 & 30 \end{bmatrix}$$


300
250
200
150
100
50
0
Time [s]
0
100
200
300
400
500
n
Serial
Parallel 4
Parallel 8

Conclusions

Known method for verification if a given subspace is a lattice ordered subspace of $\mathbb{R}^m$ can be applied for very small computational problems (n<20).

Serial method has polynomial complexity and can be effectively applied for large problems (n<500). In order to solve larger problems it is necessary to apply parallel computing.

In presented thesis theoretical background as well as numerical results were presented. Parallel method can be applied to the larger problems depending on available hardware. Current implementation of the parallel method is more effective than the serial method for sufficiently big *n*. More optimized parallel code written some HPC language (e.g. c/c++, FORTRAN) would be more effective.

Thank you