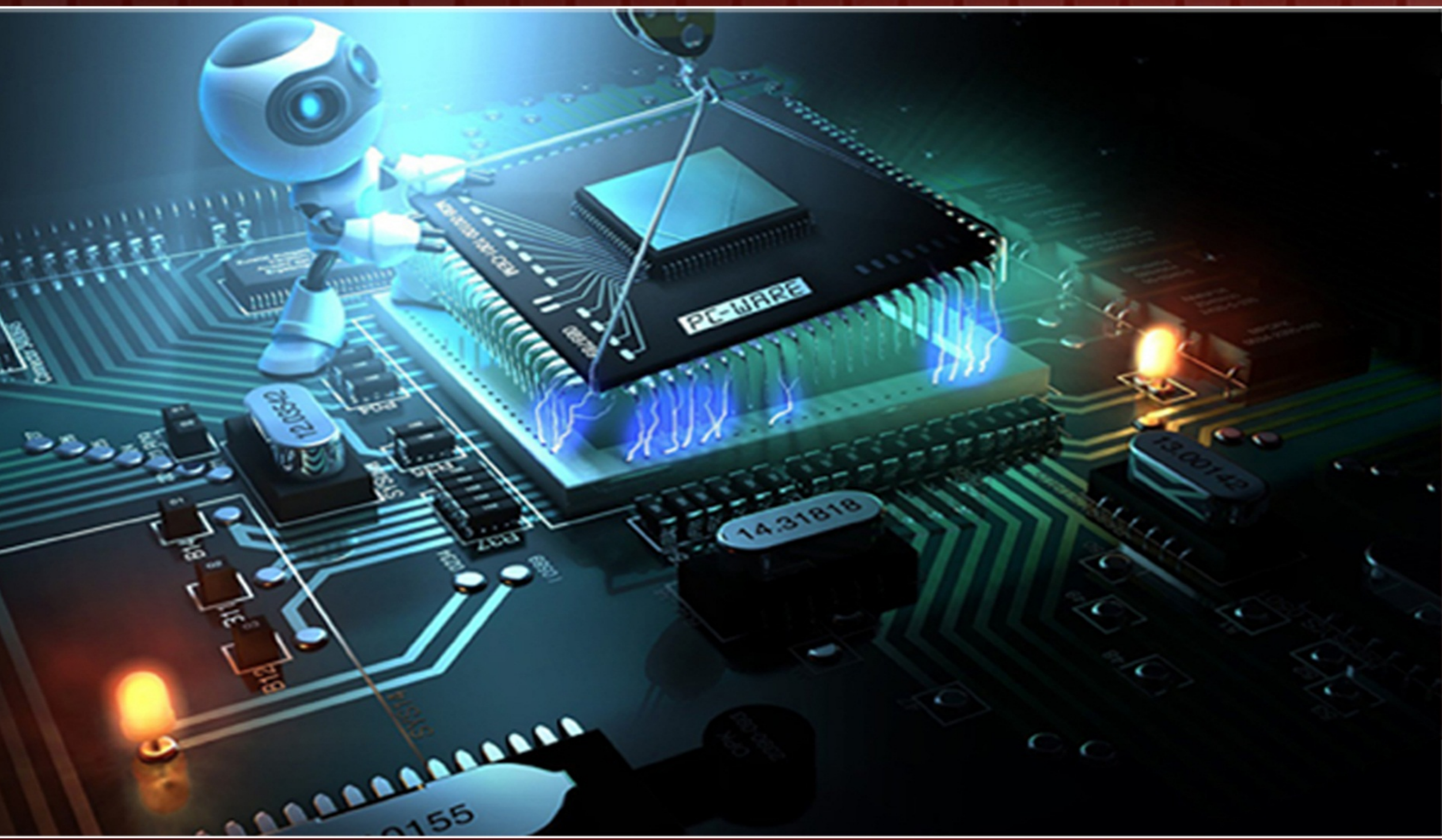
ISSN: 2054-7390

Volume 8 No 2, April 2020

Transactions on Machine Learning and Artificial Intelligence



Society for Science and Education
UNITED KINGDOM

TABLE OF CONTENTS

EDITORIAL ADVISORY BOARD

DISCLAIMER

All the contributions are published in good faith and intentions to promote and encourage research activities around the globe. The contributions are property of their respective authors/owners and the journal is not responsible for any content that hurts someone's views or feelings etc.

SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM

TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Volume 8 No 2
ISSN 2054-7390

Electrical Conductance Analysis of *Solanum lycopersicum* under Biotic Stress

[1,2]Teuma Mbezi M., [4]Ambang Zachée, [1]Ekobena Fouda H. and [5]Kofané Timoleon C.

[1]Laboratory of Biophysics, Department of Physics, Faculty of Sciences, University of Yaoundé I, ,Yaoundé, Cameroon.

[2]National Advanced School of Posts, Telecommunications and Information and Communication Technologies.

[4]Laboratory of Phytopathology, Department of biology and vegetal physiology Faculty of Sciences, University of Yaoundé I, P.O. Box 812, Yaoundé, Cameroon

[5]Laboratory of Mechanics, Department of Physics, Faculty of Sciences.

3michelteuma@gmail.com; zachambang@yahoo.fr;
hekobena@gmail.com; tckofane@yahoo.com

ABSTRACT

Our purpose is to provide different parameters of control from which one can identify a sick plant before the appearance of the first symptoms. We made a stochastic analysis and an analysis according to the theory of information, to deduce those characteristics parameters. It came out from our analysis that the DSP of health plant is above the DSP of the sick plant. Generally, the DSP of health and treated plant is above the DSP of sick and treated plant. However there is an overlapping between the DSP of sick and treated plant, and the health one for the whole value of the normalized reduced frequency. The average conductance of health plant is higher than the average conductance of sick plant. We also observed that, average conductance of health and treated plant is lower than the average conductance of sick and treated plant. The standard deviation of health plant is higher than the standard deviation of sick plant. We also observed that, standard deviation of health and treated plant is lower than the standard deviation of sick and treated plant. The electric conductance signal G(ω,t) of *Solanum lycopersicum* leaf plant is not a statistics process in the broad sense (SSL). Electric conductance G(ω,t) of the plant is a non ergotic signal. The entropy of the sick plant is higher than the entropy of the health one. Those parameters can be used during the development of informatics application, and can be used in I.O.T. (internet of thing)

Key words: Statistics in the broad sense (SSL); ergotic; *Solanum lycopersicum*; spectral density of power (DSP); mildew; electric conductance; entropy.

1 Introduction

Now our days, organic matter is usually study by using electrical circuit [1-4]. Generally, one can identify the sick plants i.e. plants which had undergone a biotic or abiotic stress starting from the appearance of the symptoms on the plants; however the appearance of the symptoms supposes that the plant already underwent a certain number of damages inside their tissues; which could have an influence on the quality and quantity of resulting product produced from these plants [5]. Teuma et al measured the electrical resistance of tomato (*Solanum lycopersicum*) sheets infected by the mildew and untreated with the

DOI: 10.14738/tmlai.82.6799
Publication Date: 18th April, 2020
URL: http://dx.doi.org/10.14738/tmlai.82.6799

ridomil MC, of the infected and treated sheets; treated sheets but not infected and the pilot plants with an aim of using the biophysics methods to diagnose the physiological state of the plants subjected to the disease and the fungicidal treatent [4]. This study aims to firstly make a stochastic analysis [6]; secondly, an analysis according to the information theory, and to deduce from those analysis a characteristics parameters from which one can identify a sick plant to the health one before the apparition of visible symptoms.

2 Materials and Methods

Matlab software was used to analyze data.

Methods of analysis

- **Spectral density of power (DSP)**

The spectral density of power $\Upsilon_G(\nu)$ of G(t) signal is expressed as:

$$\Upsilon_G(\nu) = \left|\widehat{G}(\nu)^2\right| \tag{1}$$

$$\text{Where} \quad \widehat{G}(\nu) = \sum_{n=1}^{N} G(n) e^{-j2\pi n\nu} \quad ; \quad G(n) = \frac{1}{R(n)} \tag{2}$$

$\widehat{G}(\nu)$ is the discrete time Fourier transformation of the electric conductance G(t) signal; N the number of G(t) sample ; ν is the normalized reduced frequency ; $\nu \in [0; 1[$.

The spectral density $\Upsilon_G(\nu)$ was evaluated firstly for the pilot plant (health plant); secondly for the health and treated plant with the ridomil MC; thirdly for the plant infected by the mildew and untreated with the ridomil MC; and fourthly, for the infected and treated plant. The resulting evaluations of the spectral density ϒG (ν) were each time plotted.

2.1 Stochastic process analysis

Since the measure value of G(t) is unpredictable during the 26 hours, we define a probabilized space (Ω,@,P), where $\Omega = \left\{\omega_1, \omega_2, \omega_3, \omega_4\right\}$ is the universe space, @=p(Ω) the entire parts of Ω and P the probability of the plant to be health, or to be sick, or to be health and treated, or to be sick and treated.

We define the random variable ω which takes values:

$\omega = \omega_1 = 1$, when the plant is health,

$\omega = \omega_2 = 2$, when the plant is sick,

$\omega = \omega_3 = 3$, when the plant is health and treated,

$\omega = \omega_4 = 4$, when the plant is sick and treated.

We consider that P is equiprobable on Ω; so we have:

$P = \frac{1}{4} = 0.25$ i.e

$P(\omega=\omega_1=1) = P(\omega=\omega_2=2) = P(\omega=\omega_3=3) = P(\omega=\omega_4=4) = 0.25$

We also define E as the whole possible states of the process, and E_1, E_2, E_3, E_4 as:

$E = \bigcup_{i=1}^{4} E_i$,

where E_1, E_2, E_3 and E_4 are respectively the whole possible state of the process to be: health, sick, health and treated, sick and treated.

T represents the whole discrete time.

We now define the electric conductance process G(ω,t) as:

$$\mathrm{G}: \ \Omega \times \mathrm{T} \rightarrow \mathrm{E}$$

$$(\omega, t) \rightarrow \mathrm{G}(\omega, \mathrm{t})$$

Where $\mathrm{G}(\omega, \mathrm{t}) = G_t(\omega) = \begin{cases} \mathrm{E1} \ \ if \ \omega = \omega 1, \forall t \in T \\ E2 \ \ if \ \ \omega = \omega 2, \ \ \ \forall t \in T \end{cases}$

$$\mathrm{G}(\omega, \mathrm{t}) = G_t(\omega) = \begin{cases} \mathrm{E3} \ \ if \ \omega = \omega 3, \forall t \in T \\ E4 \ \ if \ \ \omega = \omega 4, \ \ \ \forall t \in T \end{cases}$$

$T = \{1,2,3,4,5,6,7,8,9,10 \ldots,16\}$; $g_{i,j}$ is the j^{em} element of Ei ;

$i \in \{1,2,3,4\}$; $j \in \{1,2,3,4,5,6,7,8,9,10 \ldots,16\}$;

Let us suppose: $\hat{g}_{m,n}$ the m×n matrix of extra- cellular space conductance, and $g_{i,j}$ one of its element; with m=4, n=26. We then have:

$$\hat{g}_{m,m} = (g_{i,j}) 1 \leq i \leq m \leq j \leq n \quad \text{with} \ (g_{i,j}) = G(\omega = \omega_i, t = j) \qquad (3)$$

Statistical properties:

$$\textbf{Average :} \quad \boldsymbol{M_R}\ (\boldsymbol{t}) = \sum_{i=1}^{4} G_t(\omega_i) \times P(\omega = \omega_i) \ \ \text{i.e} \qquad (4)$$

$$\boldsymbol{M_R}\ (\boldsymbol{t} = \boldsymbol{j}) = \sum_{i=1}^{4} g_{i,j} \times P(\omega = \omega_i) \ \ \text{i.e}$$

$$\boldsymbol{M_R}\ (\boldsymbol{t} = \boldsymbol{j}) = \frac{1}{4}(g_{1,j} + g_{2,j} + g_{3,j} + g_{4,j}) \qquad (5)$$

- **Autocorrelation function:**

$$\tau_G(t = i, t = i + k) = \sum_{i=1}^{4} \sum_{l=1}^{4} g_{i,j} \times g_{l+k,j} \times P(\omega = \omega_i, \omega = \omega_l) \qquad (6)$$

Where:

$P(\omega = \omega_1, \omega = \omega_2) = P(\omega = \omega_1, \omega = \omega_4) = P(\omega = \omega_2, \omega = \omega_3) = 0$,
$P(\omega = \omega_2, \omega = \omega_1) = P(\omega = \omega_4, \omega = \omega_1) = P(\omega = \omega_3, \omega = \omega_2) = 0$
$P(\omega = \omega_4, \omega = \omega_3) = P(\omega = \omega_3, \omega = \omega_4) = 0$
because the plant can't be in health and sick at the same time, and
$P(\omega = \omega_1, \omega = \omega_3) = P(\omega = \omega_2, \omega = \omega_4) = \frac{2}{16}$,
$P(\omega = \omega_3, \omega = \omega_1) = P(\omega = \omega_4, \omega = \omega_2) = \frac{2}{16}$
The conjoint probability of plant to be respectively healthy and treated, sick and treated
$P(\omega = \omega_1, \omega = \omega_1) = P(\omega = \omega_2, \omega = \omega_2) = \frac{2}{16}$,
$P(\omega = \omega_3, \omega = \omega_3) = P(\omega = \omega_2, \omega = \omega_3) = \frac{2}{16}$;

k$\in N$;i $+$ k ≤ 4, and l $+$ k ≤ 4.

Considering the above assumption, we then have:

$$\tau_G(t=i,t=i+k)=\frac{1}{8}\left(g_{1,j}\times g_{1+k,j}+g_{2,j}\times g_{2+k,j}+g_{3,j}\times g_{3+k,j}+g_{4,j}\times g_{4+k,j}+\ g_{1,j}\times g_{3+k,j}+g_{3,j}\times g_{1+k,j}+g_{2,j}\times g_{4+k,j}+g_{4,j}\times g_{2+k,j}\right) \tag{7}$$

- **Temporal properties:**

- **Temporal average:**

$$\mu_G(\omega=\omega_i)=\frac{\mathbf{1}}{\boldsymbol{N}}\left(\sum_{\boldsymbol{j=1}}^{\mathbf{16}} g_{i,j}\right) \tag{8}$$

We deduced the average vector **M (**$\mu_G(\omega=1),\mu_G(\omega=2),\mu_G(\omega=3),\mu_G(\omega=4)$**)** and the standard deviation vector $\boldsymbol{\sigma}(\sigma(\omega=1),\sigma(\omega=2),\sigma(\omega=3),\sigma(\omega=4))$

- **Temporal autocorrelation function:**

$$\mu_{GG}(\omega=\omega_i)=\frac{\mathbf{1}}{\boldsymbol{N}}\left(\sum_{\boldsymbol{j=1}}^{\mathbf{16}} g_{i,j}\times g_{i,j+k}\right) \tag{9}$$

k$\in N$; j $+$ k ≤ 26

2.2 Analysis according to information theory

evaluation of the entropy

The average information quantity is evaluated by the entropy, expressed as:

$$H(X)=-\sum_{i=1}^{N} P_i log\,(P_i) \tag{10}$$

Where P_i is the probability of obtaining alphabet x_i.

The entropy was evaluated for two states of tomato plant.

3 Results and Discussion

We observe an oscillation of the electric conductance G(t) whether the plant is health, sick, health and treated, or sick and treated (Figure1). The picks of conductance of sick and treated plant are higher than the conductance picks of sick and untreated plant. The conductance picks of sick and untreated plant are generally up to the electric conductance of the health plant.

- **Spectral density of power of the electric conductance.**

By using equations 1, 2, 3, we have:

$$\Upsilon_G(\nu)=\sqrt{\ \left(A^2(\nu)+B^2(\nu)\right)^2+4\left(A^2(\nu)B^2(\nu)\right)} \tag{11}$$

Concerning the health plant, we have:

$$A(\nu) = \sum_{j=1}^{16} g_{1,j} \cos 2\pi\nu j$$

$$B(\nu) = \sum_{j=1}^{16} g_{1,j} \sin 2\pi\nu j$$

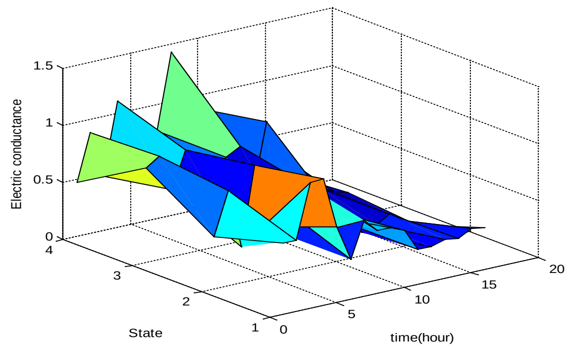


Figure 1: Behavior of electric conductance of plan under biotic stress. We observe an oscillation of the electric conductance G(t) whether the plant is health, sick, health and treated, or sick and treated. The picks of conductance of sick and treated plant are higher than the conductance picks of sick and untreated plant. The conductance picks of sick and untreated plant are generally up to the electric conductance of the health, and sick plant. The conductance maxima of sick plant are also generally up the health one.

Concerning the sick plant, we have:

$$A(\nu) = \sum_{j=1}^{16} g_{2,j} \cos 2\pi\nu j$$

$$B(\nu) = \sum_{j=1}^{16} g_{2,j} \sin 2\pi\nu j$$

Concerning the health and treated plant, we have:

$$A(\nu) = \sum_{j=1}^{16} g_{3,j} \cos 2\pi\nu j$$

$$B(\nu) = \sum_{j=1}^{16} g_{3,j} \sin 2\pi\nu j$$

Concerning sick and treated plant, we have:

$$A(\nu) = \sum_{j=1}^{16} g_{4,j} \cos 2\pi\nu j$$

$$B(\nu) = \sum_{j=1}^{16} g_{4,j} \sin 2\pi\nu j$$

We can observe on Figure2 that the DSP of health plant (green curve) is above the DSP of the sick plant (red curve). We can also observe that generally, the DSP of health and treated plant (yellow curve) is above the DSP of sick and treated plant (blue curve). However there is an overlapping between the DSP of sick and treated plant, and the health one for the whole value of the normalized reduced frequency.

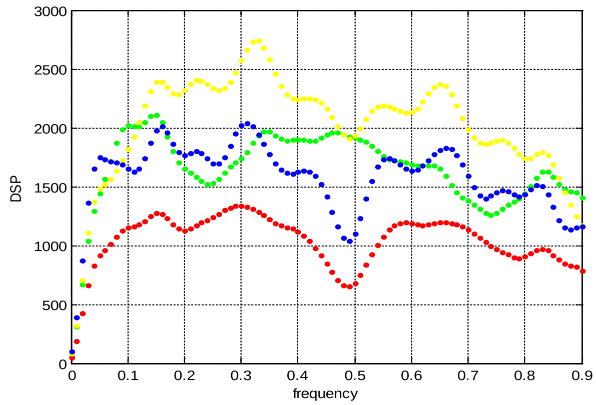


Figure 2. Spectral density of power of the electric conductance process G(ω,t) of *Solanum lycopersicum* leaf plant. We can observe that the DSP of health plant (green curve) is above the DSP of the sick plant (red curve). We can also observe that generally, the DSP of health and treated plant (yellow curve) is above the DSP of sick and treated plant (blue curve). However there is an overlapping between the DSP of sick and treated plant, and the health one for the whole value of the normalized reduced frequency.

- **Statistical properties:**

- **Statistical average**

In Figure3, the curve reveals that the electric conductance process G(ω,t) is non-statistics in the broad sense (non SSL) ; due to the fact that the statistical average of the G(ω,t) process Is not constant during the time of evolution. The statistical average has an oscillatory behavior which decreases according to time.

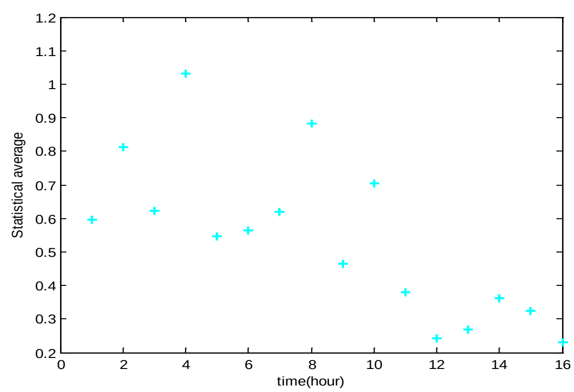


Figure 3. Statistical average of electric conductance signal G(ω,t) according to time of *Solanum lycopersicum* leaf plant. The curve reveals that the electric conductance process G(ω,t) is non-statistics in the broad sense (non SSL) ; due to the fact that the statistical average of the G(ω,t) process Is not constant during the time evolution. The statistical average has an oscillatory behavior which decreases according to time.

- **Autocorrelation function**

The autocorrelation decreases when we pass from the state 1 to the state 4. The signal G(ω,t) of the health plant (state 1) is more correlate to the signal G(ω,t) of the sick plant (state 2) than the signal G(ω,t) of the sick and treated (state 3), and the health and treated (state 4) plant. The signal G(ω,t) doesn't depend only to displacement parameter k when passing from the health plant (state 1) to the health and treated plant (state 4), but also on time; it is not stationary as it is shown in the Figure 4. This implies that the electric conductance signal G(ω,t) of *Solanum lycopersicum* leaf plant is not a statistics process in the broad sense (SSL).

- **Temporal properties:**

- **Temporal average and standard deviation**

The obtained average vector of electric conductance signal G(ω,t) is:

$\mathbf{M}\ (0.423\mu S, 0.342\mu S, 0.453\mu S, 0.492\mu S)$

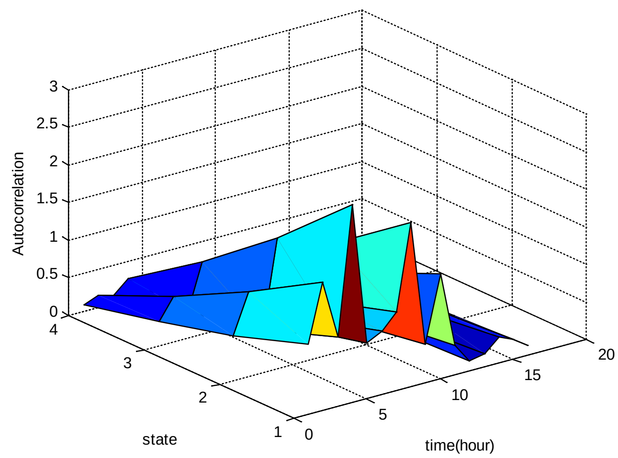


Figure 4. Statistical autocorrelation of electric conductance signal G(ω,t) according to time and the state of *Solanum lycopersicum* leaf plant. The autocorrelation decreases when we pass from the state 1 to the state 4. The signal G(ω,t) of the health plant (state 1) is more correlate to the signal G(ω,t) of the sick plant (state 2) than the signal G(ω,t) of the sick and treated (state 3), and the health and treated (state 4) plant. The signal G(ω,t) doesn't depend only to displacement parameter k when passing from the health plant (state 1) to the health and treated plant (state 4), but also on time; it is not stationary as it is shown in the figure. This implies that the electric conductance signal G(ω,t) of *Solanum lycopersicum* leaf plant is not a statistics process in the broad sense (SSL).

The average conductance of health plant is higher than the average conductance of sick plant. We also observed that, average conductance of health and treated plant is lower than the average conductance of sick and treated plant.

The standard deviation vector of the process is:

$\boldsymbol{\sigma}(0.255, 0.239, 0.276, 0.337)$

The standard deviation of health plant is higher than the standard deviation of sick plant. We also observed that, standard deviation of health and treated plant is lower than the standard deviation of sick and treated plant.

- **Temporal autocorrelation function**

Autocorrelation has an oscillatory behavior according to time. Autocorrelation is higher for the health plant (state 1) than the other states. The lower autocorrelation is observed for the sick plant (state 2). Autocorrelation depends at the same time of the parameter of time evolution and the state of the plant; this implies that electric conductance G(ω,t) of the plant is a non ergotic signal (Figure5).

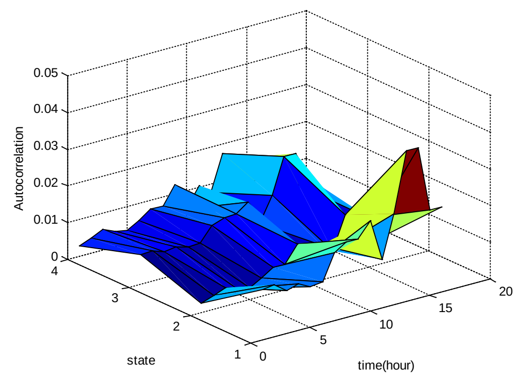


Figure 5. Temporal autocorrelation function according to time and the state of *Solanum lycopersicum* leaf plant. Autocorrelation has an oscillatory behavior according to time. Autocorrelation is higher for the health plant (state 1) than the other states. The lower autocorrelation is observed for the sick plant (state 2). Autocorrelation depends at the same time of the parameter of time evolution and the state of the plant; this implies that electric conductance G(ω,t) of the plant is a non ergotic signal.

- **evaluation of the entropy**
- **Entropy of the health plant**

Alphabet xi	0.625	1.111	0.666	0.357	0.555	0.5	0.312	0.303	0.344	0.322	0.384	0.37
P(X=xi)	2/16	2/16	2/16	1/16	2/16	1/16	1/16	1/16	1/16	1/16	1/16	1/16

$$H(X) = -\sum_{i=1}^{N=12} P_i log\,(P_i) \quad = \quad 1.520sh \tag{12}$$

- **Entropy of the sick plant**

Alphabet xi	0.454	0.833	0.312	1.0	0.434	0.588	0.277	0.714	0.303	0.204	0.243	0.227	0.161
P(X=xi)	1/16	1/16	2/16	1/16	3/16	1/16	1/16	1/16	1/16	1/16	1/16	1/16	1/16

$$H(X) = -\sum_{i=1}^{N=13} P_i log\,(P_i) \quad = \quad 3.577sh \tag{13}$$

Knowing that the passive electric characteristics reflect the degree of viability of life cells [7], one can say that electric conductance reveals the level of viability and vitality of a life cells. The fact that the means conductance of health plant ($0.423\mu S$) is higher than the means conductance of the sick one($0.342\mu S$)

revealed that the infection has decreased the vitality of the plant. One can also say that the infection of the plant by phytophthora (pathogenic agent of the mildew) perturbs the electric fluctuation of plant; what results in the lower standard deviation of the sick plant $(0.239\,)$than the health one (0.255). However the treatment of the sick plant with the ridomil MC fungicide increases the vitality and viability of plant which is revealed through the higher values of conductance means $(0.492\mu S)$ and standard deviation (0.337). The ridomil MC fungicide also increases the vitality and viability of the health plant due to the higher values of conductance means $(0.453\mu S)$ and standard deviation (0.276) of the health and treated plant than conductance means and standard deviation of the health and no treated plant with ridomil.

When the plant was infected, a self-defense mechanism is started by the plant, which results in the higher conductance picks of the sick plant than the health plant (Figure 1). However the highest conductance picks observed in sick and treated plant can be due to the combine action of the self-defense mechanism and the addition of ions in the plant which came from ridomil.

The fact that the DSP of health plant is above the DSP of the sick plant reveals that, the infection of the plant decreases its electrical energy. The treatment of health and sick plant which ridomil increases their DSP i.e. their electrical energy. However, ridomil increases the energy of sick plant to the level of health plant. That is why we can observe there an overlapping between the DSP of sick and treated plant and the health one (Figure 2).

The fact that the statistical average of the G(ω,t) process is not constant during the time evolution may be explained by life nature of plant, i.e. the plant is not an inert matter. The physiological process of our four groups of plants is not uniform during the time evolution (Figure3).

The statistical autocorrelation, when there exists, is accentuated between the health plant and the sick one (Figure4). It reveals the level of reciprocal dependence between the conductance of health plant and the conductance of the sick one. The temporal autocorrelation is higher for the health plant than the sick one (Figure5). The weak correlation observed in the electric conductance may be due to the perturbation of phytophthora (pathogenic agent of the mildew).

Knowing that, the entropy reveals the information quantity, one can say that the entropy of the health plant (1.520) reveals the information quantity directly linked to the physiological activity of the health plant. However, when the plant is infected, the plant will activate a new physiological activity for it self-defense; this will result to the additional information quantity (3.577).

4 Conclusion

The main concerned of our study was to provide different control parameters from which one can identify a sick plant before the appearance of the first symptoms. The tomato plants were set out in 4 groups. The first group was made up of plants into good health, the second group of the sick plants, the third group of plants into good health but treated by the ridomil, and the forth group was made up of sick plants which were treated by the ridomil MC fungicide. It came out from our analysis that the conductance picks of sick plant are generally up to the electric conductance of the health one. The picks of conductance of sick and treated plant are higher than the conductance picks of sick and untreated plant. The DSP of health plant is above the DSP of the sick plant. Generally, the DSP of health and treated plant is above the DSP of sick and treated plant. However there is an overlapping between the DSP of sick and treated plant, and the health one for the whole value of the normalized reduced frequency. The average conductance of health plant is higher than the average conductance of sick plant. We also observed that, average conductance of health and treated plant is lower than the average conductance of sick and treated plant. The standard deviation of health plant is higher than the standard deviation of sick plant. We also observed that, standard deviation of health and treated plant is lower than the standard deviation of sick and treated plant. The electric conductance signal G(ω,t) of *Solanum lycopersicum* leaf plant is not a statistics process in the broad sense (SSL). Electric conductance G(ω,t) of the plant is a non ergotic signal. The entropy of the sick plant is higher than the entropy of the health one.

ACKNOWLEDGEMENTS

We would like to express a special thanks to Pr Ben Boli Germain, the head of Atomic Molecular and Nuclear research group of U.Y.1; the head of Nuclear Technology Section of Cameroon Pr Saidou and the german teacher Nkolo Laure Martine.

REFERENCES

[1] Teuma Mbezi M, Ekobena Fouda H.P., Tabi Conrad B. and T.C. Kofane. Estimated Photosynthetic Activity from Its Electrical Impedance Spectroscopy. *A.S.R.J.E.T.S*. 2015, 13(1), pp.178-193.

[2] Teuma Mbezi M, Ekobena Fouda H.P., Tabi Conrad B. and T.C. Kofane. Estimated Photosynthesic Activity From its Passive Electrical Properties. *International Journal of current*. 2015, (7)10, pp. 21180-21185.

[3] Teuma Mbezi M, Ekobena Fouda H.P., Tabi Conrad B. and T.C. Kofane. Behavior of electrical properties of synthetic chorophyll pigment solution by using the T.E-model. *Indian Journal of Science and Technology*. 2017, 10(38).

[4] Teuma Mbezi M, Ekobena Fouda H.P., Ambang Zachée, Tabi Conrad B. and T.C. Kofane. study of the passive electrical properties of tomato tissues after infection and treatment by fongicide . *Indian Journal of Science and Technology*. 2017, 10(26).

[5] Philippe Lepoivre. Phytopathologie ; Bases moléculaires et biologiques des pathosystèmes et fondements des stratégies de lutte.2003 ; Editions De Boeck Université.

[6] Teuma Mbezi M, Ambang Zachée, Ekobena Fouda H.P., Tabi Conrad B. and T.C. Kofane. Stochastic Electrical Behavior of Splina Liquid Chlorophyll Drink Indian Journal of Science and Technology, Vol 11(12), DOI: 10.17485/ijst/2018/v11i12/120086, March2018

[7] Gounar II, Panishkmine LA, Tihonov FP. Activité bioélectrique du tournesol et de la tomate sur quelques types de pathogénéité. Jounal de l'académie agricole de Moscou. 1974; 5:3–8.

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON
MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 8 No 2
ISSN 2054-7390

Primality Test and Primes Enumeration using Odd Numbers Indexation

WOLF Marc, WOLF François
Independent researchers;
marc.wolf3@wanadoo.fr; francois.wolf@dbmail.com

ABSTRACT

Odd numbers can be indexed by the map $k(n) = (n-3)/2\,, n \in 2\mathbb{N}+3$. We first propose a basic primality test using this index function that was first introduced in [8]. Input size of operations is reduced which improves computational time by a constant. We then apply similar techniques to Atkin's prime-numbers sieve which uses modulus operations and finally to Pritchard's wheel sieve, in both case yielding similar results.

Keywords: odd number index, primality test, primes enumeration, Atkin sieve, composite odd numbers, wheel sieve.

1 Introduction

1.1 Primality test and prime enumeration

An odd number n is prime when it is not divisible by any prime p lower than or equal to $\sqrt{n}$. This basic primality test requires too much computational time for large integers. Faster and more efficient deterministic and probabilistic primality tests have been designed for large numbers [1]. A deterministic polynomial primality test was proposed by M. Agrawal, N. Kayal and N. Saxena in 2002 [2].

Enumeration of primes up to a given limit can be done by using a primality test but prime number sieves are preferred from a performance point of view. A sieve is a type of fast algorithm to find all primes up to a given number. There exists many such algorithms, from the simple Erastosthenes' sieve (invented more than 2000 years ago), to the wheel sieves of Paul Pritchard ([3], [4], [5]) and the sieve of Atkin [6]. In [7], Gabriel Paillard, Felipe Franca and Christian Lavault present another version of the wheel sieve and give an overview of all the existing prime-numbers sieves.

In theory, indices are a way to represent odd numbers. By adapting results from [8], we show how odd number indices may be used in applied mathematics. In the last part, we apply [8] to Pritchard's wheel sieve, which leads to a $dynamical$ wheel sieve. Using the linear diophantine equation resolution method first introduced in [9], we introduce an original way of "turning the wheel".

1.2 Notation

We will use the following notations:

DOI: 10.14738/tmlai.82.8054
Publication Date: 30th April, 2020
URL: http://dx.doi.org/10.14738/tmlai.82.8054

1. I designates the set of odd integers greater than 1, i.e.:

$$I = \{N_k = 2k + 3 | k \in \mathbb{N}\};$$

2. P the set of prime numbers, P_n the set of prime numbers not greater than n;

3. C the set of composite odd integers, i.e.:

$$C = I \backslash P = \{N_k \in I | \exists (a, b) \in I, N_k = ab\}$$

The function $f: k \in \mathbb{N} \longmapsto N_k \in I$ is bijective. The inverse function is $f^{-1}: N_k \in I \longmapsto k = \frac{N_k - 3}{2}$. $k = f^{-1}(N_k)$ is the index of N_k. The preimage of C is denoted by W:

$$W = f^{-1}(C) = \{k \in \mathbb{N} | N_k \in C\}$$

4. For x and y two integers, we denote by $x \bmod y$ the remainder of the Euclidean division of x by y, which belongs to $[\![0, y - 1]\!]$.

5. N_1 and N_2 are the subsets of I given by:

$$N_1 = \{N_k \in I | N_k \bmod 4 = 1\}$$

$$N_2 = \{N_k \in I | N_k \bmod 4 = 3\}$$

Similarly:

$$C_1 = N_1 \cap C$$

$$C_2 = N_2 \cap C$$

Finally, S_1 and S_2 designate the set of indices corresponding to elements of C_1 and C_2 respectively, i.e. $S_1 = f^{-1}(C_1)$ and $S_2 = f^{-1}(C_2)$.

2 Basic primality test and primes enumeration

2.1 Two families of infinite sequences with arithmetic difference

[8] shows that W is the union of two families of finite sequences with arithmetic difference. Actually proposition 2-5 says that any composite odd number $N_k \in C$ can be written as a difference of two squares, and more precisely that there exists $j \in \mathbb{N}$ and $x \in [\![0, j]\!]$ such that:

$$\begin{cases} (\mathbf{1})\ N_k \in C_1 \Rightarrow N_k = (2j + 3)^2 - (2x)^2, \\ (\mathbf{2})\ N_k \in C_2 \Rightarrow N_k = (2j + 4)^2 - (2x + 1)^2 \end{cases}$$

<u>Corollary 2-1</u>: Let $k_j(n) = (2j + 3)n + j$. One has:

$$W = S_1 \cup S_2$$

and:

$$\begin{aligned} S_1 &= \{k_i(x) = k_i(i + 1) + 2(2i + 3)x;\ i \in \mathbb{N}, x \in \mathbb{N}\} \\ S_2 &= \{k_i(x) = k_i(i + 2) + 2(2i + 3)x; i \in \mathbb{N}, x \in \mathbb{N}\} \end{aligned}$$

Thus W is the union of two families of infinite arithmetic sequences. The indices $k_i(i+1)$ of first type reference points (or remarkable points, see[8]) are the initial terms of sequences ranging in S_1. Similarly, the indices $k_i(i+2)$ of second type reference points are the initial terms of sequences ranging in S_2.

Proof: We substitute j by $i+x$ in relations **(1)** and **(2)**:

$$\begin{aligned}(2j+3)^2-(2x)^2 &= (2i+2x+3)^2-(2x)^2 = (2i+3)(2i+4x+3)\\ &= 2[k_i(i+1)+2(2i+3)x]+3\end{aligned}$$

and similarly:

$$\begin{aligned}(2j+4)^2-(2x+1)^2 &= (2i+2x+4)^2-(2x+1)^2 = (2i+3)(2i+4x+5)\\ &= 2(2i+3)(i+2x+2)+2i+3 = 2[k_i(i+2)+2(2i+3)x]+3\end{aligned}$$

Proposition 2-1: For any $N_k \in C$ there exists $X \in P$, $X \le \sqrt{N_k}$ and $x \in \mathbb{N}$ such that:

$$N_k \in C_1 \Rightarrow N_k = X(X+4x)$$

$$N_k \in C_2 \Rightarrow N_k = X(X+4x+2)$$

Thus, writing $X = 2i+3$, we get:

$$W = S_1' \cup S_2'$$

where:

$$\begin{aligned}S_1' &= \{k_i(x) = k_i(i+1)+2(2i+3)x;\ i \in \mathbb{N}\setminus W, x \in \mathbb{N}\}\\ S_2' &= \{k_i(x) = k_i(i+2)+2(2i+3)x; i \in \mathbb{N}\setminus W, x \in \mathbb{N}\}\end{aligned}$$

Proof: Take X the smallest prime dividing $N_k \in C$. Thus $X \in P_{\sqrt{N_k}}$ and if $Y = \frac{N_k}{X}$ then $Y \ge X$ and $Y - X$ is even, and we can write it either $4x$ or $4x+2$. These two cases clearly correspond respectively to $N_k \in C_1$ and $N_k \in C_2$. Thus the index k can be decomposed as in corollary 2-1, but with i the index of a prime number, hence in $\mathbb{N}\setminus W$.

2.2 Basic primality test

In this section, we describe a basic primality test using the previous infinite sequences.

Definition 2-2: For any $p = 2i+3 \in P$ and $N \in I$ we let:

1- $A(N,p) = N - p^2$ and $f_A(p) = p^2$.

2- $B(N,p) = N - p(p+2)$ and $f_B(p) = p(p+2)$.

Proposition 2-2: $N \in N_1$ is a prime number when:

$$\forall p = 2i+3 \in P_{\sqrt{N}}, \frac{A(N,p)}{4} \bmod p \ne 0$$

$N \in N_2$ is a prime number when:

$$\forall p = 2i+3 \in P_{\sqrt{N}}, \frac{B(N,p)}{4} \bmod p \ne 0$$

Proof: This follows from the fact that $A(N,p) \bmod p = N \bmod p$ and likewise for $B(N,p)$.

Remark 2-2: In order to reduce computation of $A(N,p)$ and $B(N,p)$ for two consecutive prime numbers, we only decrement the value.

More precisely, if $p < p'$ are two primes, we let $\alpha(p,p') = p' - p$ and we compute:

$$\begin{cases} \Delta A(N,p,p') = A(N,p) - A(N,p') = \alpha(\alpha + 2p) \\ \Delta B(N,p,p') = B(N,p) - B(N,p') = \Delta A(N,p,p') + 2\alpha \end{cases}$$

These two expressions are independent of N.

2.3 Primality test with indices

We adapt here the results of the previous section with indices.

Definition 2-3: For any i index of a prime number $p \in P$ and $k \in \mathbb{N}$, we let:

1- $A'(k,i) = (k-3)/2 - i(i+3)$, $f'_A(i) = i(i+3)$, $g'_A(k) = (k-3)/2$

2- $B'(k,i) = (k-6)/2 - i(i+4)$ and $f'_B(i) = i(i+4)$, $g'_B(k) = (k-6)/2$

Proposition 2-3: $k \in S_1$ is a prime number index when:

$$\forall p = 2i+3 \in P_{\sqrt{2k+3}}, A'(k,i) \bmod p \neq 0$$

$k \in S_2$ is a prime number index when:

$$\forall p = 2i+3 \in P_{\sqrt{2k+3}}, B'(k,i) \bmod p \neq 0$$

Proof: This follows from proposition 2-2 and definition 2-2 because if we let $N = 2k+3$ then $A'(k,i) = \frac{A(N,p)}{4}$ and $B'(k,i) = \frac{B(N,p)}{4}$.

Remark 2-3: In order to reduce computation of $A'(k,i)$ and $B'(k,i)$ for two consecutive prime number indices, we only decrement their values.

More precisely, if $i < i'$ are two prime indices we let $\alpha'(i,i') = i' - i$ and we compute:

$$\Delta A'(k,i,i') = A'(k,i) - A'(k,i') = \alpha'(\alpha' + 2i + 3)$$

$$\Delta B'(k,i,i') = B'(k,i) - B'(k,i') = \Delta A'(k,i,i') + \alpha'$$

These two expressions are independent of k.

2.4 First algorithms of prime enumeration

In this section, we present prime enumeration algorithms based on propostion 2-2 and 2-3. The first one manipulates numbers and the second one indices.

2.4.1 Primality test using numbers

This first algorithm named ***PrimeEnumeration*** consists in two functions:

- The main function which determines primes in up to N_{Max} and returns them in a list, along with its size.
- An auxiliary function which returns whether a number N is prime, based on precomputed list of primes and values of ΔA and ΔB. It is called ***LocalTest.*** It is also in charge of updating the

lists ΔA and ΔB if needed.

Three zero-based lists are used and built recursively in this algorithm: the list of primes itself L_p, and the lists of values for ΔA and ΔB respective to L_p (remember it is independent from N). Only numbers which are not multiples of 2 and 3 are tested. Thus we restrict to $N = 6m + 1$ and $N = 6m + 5$. The congruence of N modulo 4 depends on the parity of m, i.e. when m is even, $N \bmod 4 = 1$ and when m is odd, $N \bmod 4 = 3$.

Algorithm 2-4-1a Function *PrimeEnumeration(*N_{Max}*)*: N_{Max} is an odd integer such that $N_{Max} \geq 7$. This function returns the list of primes up to N_{Max} and its size.

First step : intialisation of variables

$L_p \leftarrow \{5\}$ → List of primes from 5, initialized with one element

$i_l \leftarrow 1$ → Size of the list L_p

→ **About the next two lists, see the remark 2-2**

$\Delta A \leftarrow \{16\}$ → $\Delta A(N,3,5) = 2 \times (2 + 6) = 16$

$\Delta B \leftarrow \{20\}$ → $\Delta B(N,3,5) = \Delta A(N,3,5) + 2 \times 2 = 20$

$i_{r1} \rightarrow 0$

$Cap1 \leftarrow 25$

$i_{r2} \rightarrow 0$

$Cap2 \leftarrow 35$

Second step : iteration

$(m, N) \leftarrow (1,7)$

$ModEqOne \leftarrow$ **False** → $m = 1$ so $(6m + 1) \bmod 4 = 3$

While $N \leq N_{Max}$ **Do** → Loop to get odd primes in range $[\![5, N_{Max}]\!]$

If *LocalTest*$(N, L_p, \Delta A, \Delta B, i_{r1}, Cap1, i_{r2}, Cap2, ModEqOne)$ **Do**

$L_p(i_l) \leftarrow N$

$i_l \leftarrow i_l + 1$

End If

$N \leftarrow 6m + 5$

If $N \leq N_{Max}$ **And** *LocalTest*$(N, L_p, \Delta A, \Delta B, i_{r1}, Cap1, i_{r2}, Cap2, ModEqOne)$ **Do**

$L_p(i_l) \leftarrow N$

$i_l \leftarrow i_l + 1$

End If

$m \leftarrow m + 1$

$N \leftarrow 6m + 1$

$ModEqOne \leftarrow !ModEqOne$ → Switch the boolean value

End While

Return $(\{2,3\} + L_p, i_l + 2)$ → Return the list of primes and the number of primes.

Algorithm 2-4-1b Function *LocalTest* ($N, L_p, \Delta A, \Delta B, i_{r1}, Cap1, i_{r2}, Cap2, ModEqOne$ **):** N is an odd integer. i_{root} stands for i_{r1} or i_{r2} depending on $ModEqOne$. This function decides whether for all $p \in L_p[0 \dots i_{root}]$, $A(N,p)/4$ or $B(N,p)/4$ is not divisible by p. It will also potentially update ΔA, ΔB, i_{r1}, i_{r2}, $Cap1$ and $Cap2$ which must be passed by reference.

First step : intialisation of variables

$A \leftarrow 9$ → stands for $f_A(3) = 3^2$

$B \leftarrow 15$ → stands for $f_B(3) = 3 \times 5$

If $ModEqOne$ **Do** → initiate references that might be updated

$i_{root} \leftarrow i_{r1}$

$Cap \leftarrow Cap1$

$\Delta \leftarrow \Delta A$

Else

$i_{root} \leftarrow i_{r2}$

$Cap \leftarrow Cap2$

$\Delta = \Delta B$

End If

If $N = Cap$ **Do**

Return False → The cap is a composite number

End If

If $N > Cap$ **Do** → update references because we always want $N \leq Cap$

$i_{root} \leftarrow i_{root} + 1$

$\alpha \leftarrow \left(L_p(i_{root}) - L_p(i_{root} - 1)\right)$

If $ModEqOne$ **Do**

$\Delta(i_{root}) \leftarrow \alpha(\alpha + 2L_p(i_{root} - 1))$ → ΔA

Else

$\Delta(i_{root}) \leftarrow \Delta A(i_{root}) + 2\alpha$ → ΔB, using ΔA which must already be updated

End If

$Cap \leftarrow Cap + \Delta(i_{root})$

End If

Second step : iteration

If $ModEqOne$ **Do**

$N \leftarrow N - A$

Else

$N \leftarrow N - B$

End If

$i \leftarrow 0$

While $i \leq i_{root}$ **Do** → Iteration at most up to $i = i_{root}$

$N \leftarrow N - \Delta(i)$

If $(N/4) \bmod L_p(i) = 0$ **Do** → N is a multiple of 4, division by 4 can be done bitwise

Return False → Test is negative

End If

$i \leftarrow i + 1$

End While

Return True → Test is positive

2.4.2 Primality test using infinite sequences and indices

This second algorithm ***IndexPrimeEnumeration*** also consists in two functions, mirroring the previous algorithm:

- The main function which determines primes up to N_{Max} and returns them in a list along with its size.
- An auxiliary function which returns whether a number N is prime based on precomputed list of primes and values of $\Delta A'$ and $\Delta B'$. It is called ***LocalTest.***

Four zero-based lists are used and built recursively: the list of primes L_p, the corresponding indices IL_p (indices of primes), and the lists $\Delta A'$ and $\Delta B'$ respective to L_p.

Only numbers which are not multiple of 2 and 3 are tested, i.e. indices of the form $k = 3m - 1$ and $k = 3m + 1$.

Remark 2-4-2: To avoid any division in the computation of A' and B' we will write $m = 2t + 1$ or $2t + 2$.

Algorithm 2-4-2a Function *IndexPrimeEnumeration*(N_{Max}): N_{Max} is an odd integer such that $N_{Max} \geq 7$. This function returns the list of primes up to N_{Max} and its size.

First step : intialisation of variables

$L_p \leftarrow \{5\}$ → List of primes from 5, initialized with one element

$IL_p \leftarrow \{1\}$ → List of index of primes

$i_l \leftarrow 1$ → Size of the two lists L_p and IL_p

→ **About the next two lists, see the remark 2-3**

$\Delta A' \leftarrow \{4\}$ → $\Delta A'(k, 0,1) = 1 \times (1+3) = 4$

$\Delta B' \leftarrow \{5\}$ → $\Delta B'(k, 0,1) = \Delta A'(k, 0,1) + 1 = 5$

$k_{Max} \leftarrow (N_{Max} - 3)/2$

$i_{r1} \rightarrow 0$

$Cap1 \leftarrow 11$

$i_{r2} \rightarrow 0$

$Cap2 \leftarrow 16$

Second step : iteration

$(t, k, g') \leftarrow (0, 2, -2)$ → k starts at $3(2t+1) - 1$, g' stands for $g'_A(k)$ or $g'_B(k)$

While $k \leq k_{Max}$ **Do** → Loop to get odd prime indices in range $[\![1, k_{Max}]\!]$

If *LocalTest*($g', k, L_p, IL_p, \Delta A', \Delta B', i_{r2}, Cap2$,**False**) **Do**

$IL_p(i_l) \leftarrow k$

$L_p(i_l) \leftarrow 2k + 3$

$i_l \leftarrow i_l + 1$

End If

$k \leftarrow k + 2$ → $k = 3(2t+1) + 1$

$g' \leftarrow g' + 1$

If $k \leq k_{Max}$ **And** *LocalTest*($g', k, L_p, IL_p, \Delta A', \Delta B', i_{r2}, Cap2$,**False**) **Do**

$IL_p(i_l) \leftarrow k$

$L_p(i_l) \leftarrow 2k + 3$

$i_l \leftarrow i_l + 1$

End If

$k \leftarrow k + 1$ → $k = 3(2t+2) - 1$

$g' \leftarrow g' + 2$

If $k \leq k_{Max}$ **And** *LocalTest*$(g', k, L_p, IL_p, \Delta A', \Delta B', i_{r1}, Cap1,$**True**$)$ **Do**

$IL_p(i_l) \leftarrow k$

$L_p(i_l) \leftarrow 2k + 3$

$i_l \leftarrow i_l + 1$

End If

$k \leftarrow k + 2$ → $k = 3(2t + 2) + 1$

$g' \leftarrow g' + 1$

If $k \leq k_{Max}$ **And** *LocalTest*$(g', k, L_p, IL_p, \Delta A', \Delta B', i_{r1}, Cap1,$**True**$)$ **Do**

$IL_p(i_l) \leftarrow k$

$L_p(i_l) \leftarrow 2k + 3$

$i_l \leftarrow i_l + 1$

End If

$t \leftarrow t + 1$ → We do not use t but keep it for the sake of readability

$k \leftarrow k + 1$ → $k = 3(2t + 1) - 1$

$g' \leftarrow g' - 1$

End While

Return $(\{2,3\} + L_p, i_l + 2)$ → Return the list of primes and the number of primes.

Algorithm 2-4-2b Function *LocalTest(*$g', k, L_p, IL_p, \Delta A', \Delta B', i_{root}, Cap, ModEqOne$*)*: g' stands for $g'_A(k)$ or $g'_B(k)$ depending on $ModEqOne$. This function decides whether for all $p \in L_p[0 \dots i_{root}]$, $A'(k,i)$ or $B'(k,i)$ is coprime with p.

First step : intialisation of variables

If $ModEqOne$ **Do** → initiate references that might be updated

$\Delta \leftarrow \Delta A'$

Else

$\Delta = \Delta B'$

End If

If $k = Cap$ **Do**

Return False → The cap is the index of a composite number

End If

If $k > Cap$ **Do** → update references because we always want $k \leq Cap$

$i_{root} \leftarrow i_{root} + 1$

$\alpha \leftarrow \left(IL_p(i_{root}) - IL_p(i_{root} - 1)\right)$

If $ModEqOne$ **Do**

$\Delta(i_{root}) \leftarrow \alpha(\alpha + L_p(i_{root} - 1))$ → $\Delta A'$

Else

$\Delta(i_{root}) \leftarrow \Delta A'(i_{root}) + \alpha$ → $\Delta B'$, using $\Delta A'$ which must already be updated

End If

$Cap \leftarrow Cap + \Delta(i_{root})$

End If

Second step : iteration

$R \leftarrow g'$

$i \leftarrow 0$

While $i \leq i_{root}$ **Do** → Iteration at most up to $i = i_{root}$

$R \leftarrow R - \Delta(i)$

If $R \bmod L_p(i) = 0$ **Do**

Return False → Test is negative

End If

$i \leftarrow i + 1$

End While

Return True → Test is positive

2.5 Performance of the algorithms

In this section, we present the performance of the previous two algorithms of prime enumeration. We first give a theoretical complexity, followed by empirical results.

Proposition 2-5: Time complexity (in terms of number of arithmetic operations) and space complexity are the same for both ***PrimeEnumeration*** and ***IndexPrimeEnumeration*** algorithms.

Time complexity is:

$$O\left(\frac{(N_{Max})^{\frac{3}{2}}}{\ln(N_{Max})}\right).$$

Space complexity is:

$$O\left(\frac{\sqrt{N_{Max}}}{\ln(N_{Max})}\right)$$

Proof: Any number n's primality is tested with primes in $[\![5,\sqrt{n}]\!]$, in $O(1)$ operations. There are $\pi(\sqrt{n}) - 2 \sim \frac{\sqrt{n}}{\ln(\sqrt{n})} = O\left(\frac{\sqrt{n}}{\ln(n)}\right)$ such primes. We loop over range $[\![7, N_{Max}]\!]$, time complexity is thus $\sum_{t=7}^{N_{Max}} O\left(\frac{\sqrt{t}}{\ln(t)}\right) = O\left(\frac{(N_{Max})^{\frac{3}{2}}}{\ln(N_{Max})}\right)$ (actually we skip two thirds of the terms in this sum by not testing multiples of 2 and 3, but complexity remains $O\left(\frac{(N_{Max})^{\frac{3}{2}}}{\ln(N_{Max})}\right)$ albeit with smaller constant.

The space complexity is related to the lists we keep in memory, which are at most of size $\pi(N_{Max})$. This space complexity is $O\left(\frac{\sqrt{N_{Max}}}{\ln(N_{Max})}\right)$.

Both algorithms have been implemented in Visual Studio C++ 2012. We measured execution time for various values of N_{Max} and produced a regression using Maple 2017.3. Details of the Maple options used to get the regression are given in appendix 8.1.

On the graph 2-5 below, we represent the computation time in seconds for both algorithms. Curve T_1 corresponds to the algorithm ***PrimeEnumeration*** and curve T_2 to ***IndexPrimeEnumeration***. The correlation coefficient R of each curve is given on the graph. We observe that computation time of both algorithms is consistent with theoretical complexity, although exponent is a bit smaller than 1.5.

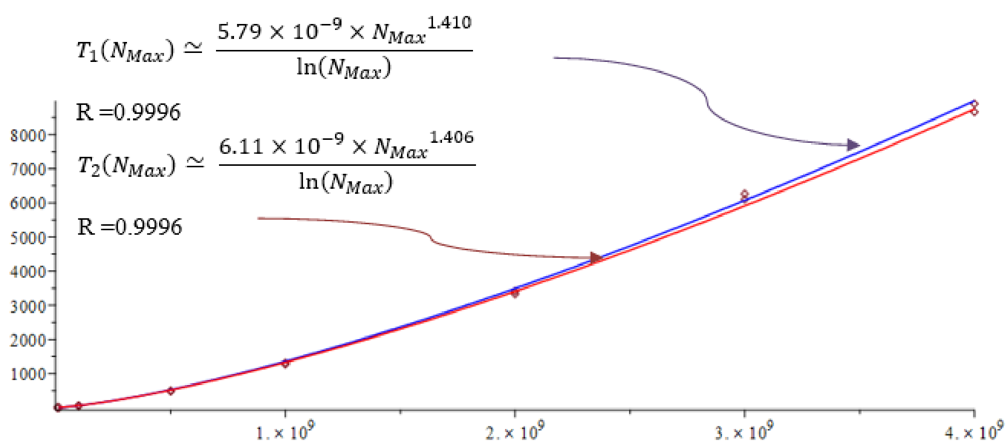


Figure 1: computation time $T\ (N_{Max})$ in seconds for both algorithms (Prime enumeration)

Both algorithms ***PrimeEnumeration*** and ***IndexPrimeEnumeration*** have the same number of modulo operations. But the computation of the input of modulus operations is done with larger inputs for the former than for the latter, which allows to marginally save time for large values of N_{Max}.

3 The sieve of Atkin

The sieve of Atkin [6] is a modern and efficient algorithm for primes enumeration. We present two algorithms based on it, one using numbers and the other indices. Both are based on the version which has a complexity $O(N_{Max})$ in time and space. Modified versions achieve up to $O\left(\frac{N_{Max}}{\ln\ln(N_{Max})}\right)$ in time and $O\left(N_{Max}{}^{\frac{1}{2}+o(1)}\right)$ in space.

3.1 Atkin algorithm

This algorithm is based on the following three results from [6].

Proposition 3-1 Let $n > 3$ be a square-free integer. Then n is prime if and only if one of the three following conditions is true:

a. $n \in 1 + 4\mathbb{N}$ and there is an odd number of solutions to $n = 4x^2 + y^2, (x, y) \in \mathbb{N}^2$,
b. $n \in 7 + 12\mathbb{N}$ and there is an odd number of solutions to $n = 3x^2 + y^2, (x, y) \in \mathbb{N}^2$,
c. $n \in 11 + 12\mathbb{N}$ and there is an odd number of solutions to $n = 3x^2 - y^2$, $x > y$, $(x, y) \in \mathbb{N}^2$.

We observe that the first congruence condition on n can also be replaced by $n \in 1 + 12\mathbb{N}$ or $n \in 5 + 12\mathbb{N}$. We also observe the following for an odd integer n:

- If $n = 4x^2 + y^2$, y must be odd.
- If $n = 3x^2 + y^2$ or $n = 3x^2 - y^2$, x and y must have opposite parity.

Furthermore if n is square-free, x and y must be in $\mathbb{N}^*$, with $x < \sqrt{n/2}$ and $y < \sqrt{n}$.

Remark 3-1 We can compute the remainder modulo 12 of $ax^2 + by^2$ depending on remainders modulo 12 of x and y. This gives us the different cases to check in Atkin sieve. We present them in table 3-1, noting that there is no case for $y \bmod 12 = 0$ and $y \bmod 12 = 6$.

Table 1: Atkin sieve cases depending on remainders modulo 12 of x and y.

$x\backslash y$	1	2	3	4	5	7	8	9	10	11
0	$4x^2+y^2$ $3x^2-y^2$				$4x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$ $3x^2-y^2$				$4x^2+y^2$ $3x^2-y^2$
1	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$
2	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$
3	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$		$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$		$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$
4	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$
5	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$
6	$4x^2+y^2$ $3x^2-y^2$				$4x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$ $3x^2-y^2$				$4x^2+y^2$ $3x^2-y^2$
7	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$
8	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$
9	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$		$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$		$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$
10	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$ $3x^2-y^2$		$4x^2+y^2$		$4x^2+y^2$ $3x^2-y^2$
11	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$	$3x^2+y^2$ $3x^2-y^2$	$4x^2+y^2$

We could run the sieve looping through 12x12 blocks of (x, y) according to this table, but for readability we do not implement this optimization in the algorithms below. We note however that this would save all the modulo operations.

Algorithm 3-1 ***SieveOfAtkin(**N_{Max}**)**: $N_{Max} > 3$ is an integer. This function returns the list of all prime numbers less than N_{Max}.

First step : intialisation of variables

$L_p \leftarrow \{2, 3\}$ → Dynamic list of odd primes

$i_l \leftarrow 2$ → Number of primes in the list

Sieve[N_{Max}] ← {**False**, …,**False**} → Array of N_{Max} entries all initialized to **False**

$\boldsymbol{x_{max}} \leftarrow \left\lceil\sqrt{N_{Max}/2}\right\rceil - 1$ → Bound for x

$\boldsymbol{y_{max}} \leftarrow \left\lceil\sqrt{N_{Max}}\right\rceil - 1$ → Bound for y

Second step : iteration for first case

For $x = 1$ **To** $\boldsymbol{x_{max}}$

 For y $= 1$ **To** $\boldsymbol{y_{max}}$ **Step** 2 → y must be odd

  $n \leftarrow 4x^2 + y^2$

  If $n < N_{Max}$ **And** $(n \bmod 12 = 1$ **Or** $n \bmod 12 = 5)$ **Do**

   Sieve[n] ← !Sieve[n] → Switch the boolean value Sieve[n]

  End If

 End For

End For

Third step : iteration for second and third cases

For $x = 1$ **To** $\boldsymbol{x_{max}}$ **Step** 2

 For $y = 2$ **To** $\boldsymbol{y_{max}}$ **Step** 2 → case where x is odd and y even

  $n \leftarrow 3x^2 + y^2$

  If $n < N_{Max}$ **And** $(n \bmod 12 = 7)$ **Do**

   Sieve[n] ← !Sieve[n]

  End If

  If $x > y$ **Do**

$n \leftarrow 3x^2 - y^2$

If $n < N_{Max}$ **And** $(n \bmod 12 = 11)$ **Do**

Sieve$[n] \leftarrow$!Sieve$[n]$

End If

End If

End For

End For

For $x = 2$ **To** $\boldsymbol{x_{max}}$ **Step** 2

For $\mathrm{y} = 1$ **To** $\boldsymbol{y_{max}}$ **Step** 2 → case where x is even and y is odd

$n \leftarrow 3x^2 + y^2$

If $n < N_{Max}$ **And** $(n \bmod 12 = 7)$ **Do**

Sieve$[n] \leftarrow$!Sieve$[n]$

End If

If $x > y$ **Do**

$n \leftarrow 3x^2 - y^2$

If $n < N_{Max}$ **And** $(n \bmod 12 = 11)$ **Do**

Sieve$[n] \leftarrow$!Sieve$[n]$

End If

End If

End For

End For

Fourth step : remove multiples of prime squares

For $n = 5$ **To** $\boldsymbol{y_{max}}$ **Step** 2 → multiples of 2 and 3 are ignored by the previous iterations

If Sieve$[n]$ **Do**

For $i = n^2$ **To** $N_{Max} - 1$ **Step** $2n^2$

Sieve$[i] \leftarrow$ **False**

End For

End If

End For

Last step : return list of primes from the sieve

For $n = 5$ **To** $N_{Max} - 1$ **Step** 2

If Sieve$[n]$ **Do**

$L_p(i_l) \leftarrow n$

$i_l \leftarrow i_l + 1$

End If

End For

Return (L_p, i_l)

3.2 Atkin algorithm with indices

We can rewrite proposition 3-1 as:

Corollary 3-2: k is the index of a prime number if and only if $2k + 3$ is square-free and one of the three following conditions is true:

a. $k \in (1 + 6\mathbb{N}) \cup (5 + 6\mathbb{N})$ and there is an odd number of solutions to $k = 2x^2 + \frac{y^2-3}{2}$,
b. $k \in 2 + 6\mathbb{N}$ and there is an odd number of solutions to $k = \frac{3x^2+y^2-3}{2}$,
c. $k \in 4 + 6\mathbb{N}$ and there is an odd number of solutions to $k = \frac{3x^2-y^2-3}{2}$ with $y < x$.

The relationships presented in the following remark are used in the next algorithm.

Remark 3-2: For the fourth step (square multiples elimination), we note that if $n = 2k + 3$, the index of n^2 is $2k^2 + 6k + 3$ and that the step of $2n^2$ translates into a step of $n^2 = (2k + 3)^2$ for indices.

Algorithm 3-2 *IndexSieveOfAtkin(*N_{Max}*)*: $N_{Max} > 3$ is an odd integer. This function returns the list of all prime numbers less than N_{Max}.

First step : intialisation of variables

$L_p \leftarrow \{2, 3\}$ → Dynamic list of primes

$i_l \leftarrow 2$ → Number of primes in the list

$k_{Max} \leftarrow (N_{Max} - 3)/2$ → Index of N_{Max}

Sieve$[k_{Max}] \leftarrow \{$**False**, …,**False**$\}$ → Array of k_{Max} entries all initialized to **False**

$\boldsymbol{x_{max}} \leftarrow \lceil\sqrt{N_{Max}/2}\rceil - 1$ → Bound for x

$\boldsymbol{y_{max}} \leftarrow \lceil\sqrt{N_{Max}}\rceil - 1$ → Bound for y

Second step : iteration for first case

For $x = 1$ **To** $\boldsymbol{x_{max}}$

For $\mathrm{y} = 1$ **To** $\boldsymbol{y_{max}}$ **Step** 2 → y must be odd

$k \leftarrow 2x^2 + \frac{y^2-3}{2}$

If $k < k_{Max}$ **And** ($k \bmod 6 = 1$ **Or** $k \bmod 6 = 5$) **Do**

Sieve[n] ← !Sieve[n] → Switch the boolean value Sieve[n]

End If

End For

End For

Third step : iteration for second and third cases

For $x = 1$ **To** $\boldsymbol{x_{max}}$ **Step** 2

For $y = 2$ **To** $\boldsymbol{y_{max}}$ **Step** 2 → case where x is odd and y even

$k \leftarrow \frac{3x^2+y^2-3}{2}$

If $k < k_{Max}$ **And** ($k \bmod 6 = 2$) **Do**

Sieve[n] ← !Sieve[n]

End If

If $x > y$ **Do**

$k \leftarrow \frac{3x^2-y^2-3}{2}$

If $k < N_{Max}$ **And** ($k \bmod 6 = 4$) **Do**

Sieve[n] ← !Sieve[n]

End If

End If

End For

End For

For $x = 2$ **To** $\boldsymbol{x_{max}}$ **Step** 2

For $\mathrm{y} = 1$ **To** $\boldsymbol{y_{max}}$ **Step** 2 → case where x is even and y is odd

$k \leftarrow \frac{3x^2+y^2-3}{2}$

If $k < k_{Max}$ **And** ($k \bmod 6 = 2$) **Do**

Sieve[n] ← !Sieve[n]

End If

If $x > y$ **Do**

$k \leftarrow \frac{3x^2-y^2-3}{2}$

If $k < k_{Max}$ **And** $(k \bmod 6 = 4)$ **Do**

Sieve$[n]$ ← !Sieve$[n]$

End If

End If

End For

End For

Fourth step : remove multiples of prime squares

For $k = 1$ **To** $\frac{y_{max}-3}{2}$ → multiples of 3 are ignored by the previous iterations

If Sieve$[k]$ **Do**

For $i = 2k^2 + 6k + 3$ **To** $k_{Max} - 1$ **Step** $(2k+3)^2$

Sieve$[i]$ ← **False**

End For

End If

End For

Last step : return list of primes from the sieve

For $k = 1$ **To** $k_{Max} - 1$

If Sieve$[k]$ **Do**

$L_p(i_l) \leftarrow 2k + 3$

$i_l \leftarrow i_l + 1$

End If

End For

Return (L_p, i_l)

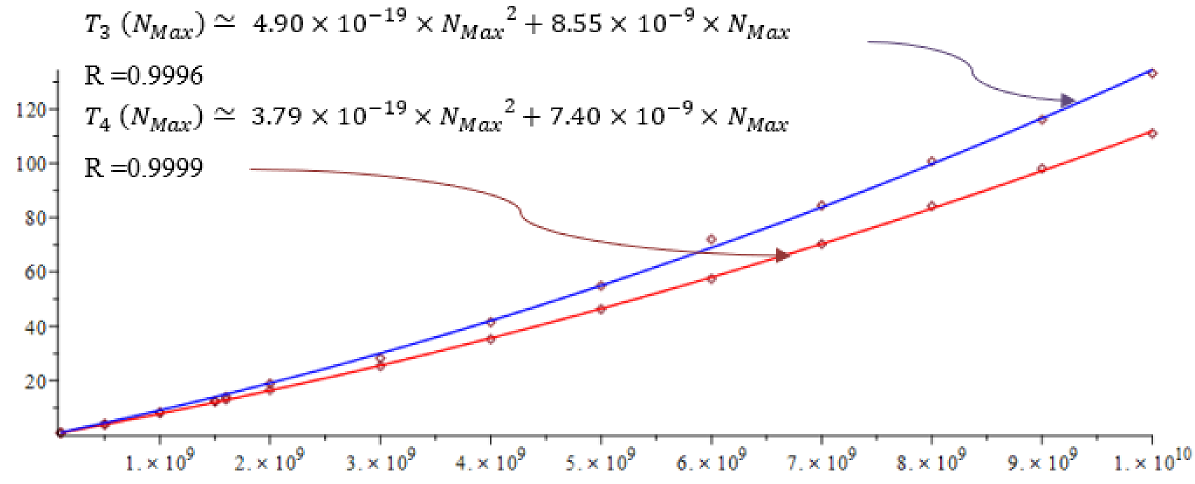
3.3 Performance of algorithms

In this section, we discuss theoretical complexity and present our results with the two algorithms implementing the sieve of Atkin.

The reference algorithm ***SieveOfAtkin*** has less operations index-based ***IndexSieveOfAtkin***, which juggles between numbers and indices. But on the other hand ***SieveOfAtkin*** performs Euclidian divisions by 12, whereas ***IndexSieveOfAtkin*** does divisions by 6. This is due to the conversion of number n into its index

$k = (n-3)/2$. Furthermore, the latter only performs the sieve on odd numbers, which means effectively the memory space for the sieve is twice smaller.

On the graph 3-3 below, we plot the computation time in seconds for both algorithms. The curve T_3 corresponds to ***SieveOfAtkin*** and the curve T_4 to ***IndexSieveOfAtkin***. We observe empirically that computation time of both algorithms looks slightly higher than linear, even though theoretically the number of operations appears to be linear in N_{Max}. Details of the Maple options used to get the regression are given in appendix 8.2.



Graph 2: computation time $T\ (N_{Max})$ in seconds for both algorithms (Sieve of Atkin)

The second algorithm is faster for larger values of N_{Max}, roughly for $N_{Max} > 10^9$. For such values the cost of encoding numbers to indices is offset by the gain on modulo operations and halving the size of the sieve. We note also that memory size is halved for the second algorithm.

4 Wheel sieve with indices

We first describe Pritchard's wheel sieve. Then we adapt it to indices and discuss a way to generate the integers of the *turning wheel*.

4.1 Description of Pritchard's wheel sieve

This description is based on [7] and [4]. The wheel sieve operates by generating a set of numbers that are coprime with the first q prime numbers. The second of these is the next prime, multiples of which are then eliminated (*by turning the wheel*).

More precisely, let $p_0 = 2, p_1 = 3$ … the sequence of prime numbers and let:

$$\Pi_q = \prod_{k=0}^{q} p_k$$

$$\mathcal{R}(m) = \{x \in [\![1, m-1]\!] \mid \gcd(x, m) = 1\}$$

$$\mathcal{W}_q = \mathcal{R}(\Pi_q)$$

The following proposition describes a "turn of the wheel".

Proposition 4-1-1: We have the following inductive formula for $\mathcal{W}_q$:

$$\mathcal{W}_0 = \{1\}, \mathcal{W}_1 = \{1,5\}, \mathcal{W}_2 = \{1,7,11,13,17,19,23,29\}$$

$$\forall q \in \mathbb{N}, \mathcal{W}_{q+1} = \left(\bigcup_{x=0}^{p_{q+1}-1} (\mathcal{W}_q + x\Pi_q) \right) \setminus p_{q+1}[\![1, \Pi_q - 1]\!]$$

Proof: The Chinese theorem ensures that $m \in \mathcal{W}_{q+1}$ if and only if $m \bmod \Pi_q \in \mathcal{W}_q$ and $m \notin p_{q+1}\mathbb{N}$. This gives the desired set equality.

Furthermore, induction formula for $\mathcal{W}_q$ can also be used to recursively build the sequence of prime numbers:

Proposition 4-1-2: The second smallest element of $\mathcal{W}_q$ ($q \geq 1$) is the next prime p_{q+1}.

Proof: The first element is 1, which is obviously not prime. For $q \geq 1$, $p_q \geq 3$ and from proposition 4-1-1 we can show (see corollary 4-2-2 later on) that $\mathcal{W}_q$ has at least 2 elements. The second one must then be the smallest integer coprime with $p_0 \dots p_q$, and thus must be p_{q+1}.

The elements of $\mathcal{W}_q$ are called pseudo-primes (at order q). Some of them are primes and others are not. However, we have a boundary condition to identify some of the primes:

Proposition 4-1-3: All integers in $\mathcal{W}_q$ and less than p_q^2 are sure to be primes.

Proof: Any integer less than p_q^2 is either prime or has a divisor among $p_0 \dots p_q$. The latter is impossible by definition of $\mathcal{W}_q$.

To enumerate primes up to N_{Max}, we thus have to keep turning the wheel as long as $p_{q+1}^2 < N_{Max}$.

As Π_q grows exponentially (in particular it can be easily proven from Bertrand's postulate that $\Pi_q > p_q^2$ from $q = 2$), while we are only interested in pseudo-primes up to N_{Max}, we may replace in practice $\mathcal{W}_q$ by $\mathcal{W}_q^{N_{Max}} = \mathcal{W}_q \cap [\![1, N_{Max}]\!]$.

Proposition 4-1-4: The following inductive formula (or wheel turn) is true for all N_{Max}:

$$\forall q \in \mathbb{N}, \mathcal{W}_{q+1}^{N_{Max}} = \left[\left(\bigcup_{x=0}^{\max\left(p_{q+1}-1, \left\lfloor \frac{N_{Max}}{\Pi_q} \right\rfloor\right)} (\mathcal{W}_q + x\Pi_q) \right) \setminus p_{q+1} \left[\!\!\left[1, \left\lfloor \frac{N_{Max}}{p_{q+1}} \right\rfloor \right]\!\!\right] \right] \cap [\![1, N_{Max}]\!].$$

Furthermore, if $N_{Max} > 9$, then as soon as $p_q^2 \geq N_{Max}$, $P_{N_{Max}} = \{p_0 \dots p_q\} \cup \left(W_q^{N_{Max}} \setminus \{1\}\right)$.

Proof: By double inclusion (cf. proof of proposition 4-2-3). The second identity comes from the fact that if $N_{Max} > 9$, $p_q^2 \geq N_{Max}$ implies $q \geq 2$.

Thus, when we turn the wheel, we remove integers that are, for a given $m \in \mathcal{W}_q$, and x, y integers, of the form:

$$m + x\Pi_q = yp_{q+1}$$

One way to do that is to remove all multiples of p_{q+1}. We will show however in section 4.2 that there is a relationship between the value of x, the multiples of Π_q which are added to $\mathcal{W}_q$, and the composite numbers yp_{q+1} which must be removed of the wheel $\mathcal{W}_{q+1}$, so that the index x to remove can be predicted from m or conversely.

4.2 Index wheel sieve

Definition 4-2: We note Π'_q the product of all odd primes up to p_q, i.e. $\Pi_q = 2\Pi'_q$.

We also note:

$$N(m,a,q) = m\Pi_q + a$$

and, with a' the index of a:

$$k(m,a',q) = \frac{N(m,2a'+3,q)-3}{2} = m\Pi'_q + a'$$

the index of $N(m,a,q)$.

We let $\mathcal{W}'_q$ be the set of indices corresponding to $\mathcal{W}_q$, with 1 replaced by $\Pi_q + 1$ (which index is $\Pi'_q - 1$):

$$\mathcal{W}'_q = \left\{\frac{n-3}{2}, n \in \mathcal{W}_q \setminus \{1\}\right\} \cup \{\Pi'_q - 1\}$$

In this section, we describe how we adapt the wheel sieve to work with indices of odd integers. The limit N_{Max} is supposed to be an odd integer of index k_{Max}.

Recurrence relation verified by the index wheel sieve:

The initial index wheels are $\mathcal{W}'_0 = \{0\}$, $\mathcal{W}'_1 = \{1,2\}$, $\mathcal{W}'_2 = \{2,4,5,7,8,10,13,14\}$.

Remark 4-2-1: The first element of $\mathcal{W}'_q$ is the index of the prime number p_{q+1}. $\mathcal{W}'_q$ is included in $\left[\!\left[\frac{p_{q+1}-3}{2}, \Pi'_q - 1\right]\!\right]$.

Proof: Since we remapped 1 to $\Pi_q + 1$ in $\mathcal{W}_q$ to define $\mathcal{W}'_q$, and because the indexing map is increasing, the first element of $\mathcal{W}'_q$ is the index of prime p_{q+1} from proposition 4-1-2 (we note that it works even for $q = 0$), and its last element is $\Pi'_q - 1$.

Proposition 4-2-1: The index wheel sieve is the only sequence of sets verifying:

$$\mathcal{W}'_0 = \{0\}$$

$$\forall q \in \mathbb{N}, \mathcal{W}'_{q+1} = \left(\bigcup_{m=0}^{p_{q+1}-1} (\mathcal{W}'_q + m\Pi'_q)\right) \setminus \left\{\frac{p_{q+1}-3}{2} + y'p_{q+1}, y' \in [\![0, \Pi'_q - 1]\!]\right\}$$

Furthermore, indices in the wheel $\mathcal{W}'_q$ up to k_{Max} correspond to all remaining prime numbers up to N_{Max} (on top of $p_0 \dots p_q$) as soon as:

$$\frac{{p_q}^2 - 3}{2} \geq k_{Max}$$

Proof: This comes from the definition 4-2 of the index wheel sieve, the proposition 4-1-1 and from observing that the index of any odd multiple yp_q of p_q is of the form:

$$\frac{yp_q - 3}{2} = \frac{p_q - 3}{2} + y'p_q, y' = \frac{y-1}{2}$$

If we let $p = 2i + 3$, this corresponds to the definition of $k(y', i)$ in [8]: $k(y', i) = i + (2i + 3)y'$.

Eliminating multiples of the next prime by solving a Diophantine equation:

Proposition 4-2-2: For a given $c \in [\![0, \Pi'_q - 1]\!]$, there exists a unique $(m_c, y_c) \in [\![0, p_{q+1} - 1]\!] \times \mathcal{W}_q$ such that $c + m_c\Pi'_q = y_c p_{q+1}$. Furthermore, m_c only depends of $c \bmod p_{q+1}$, $m_0 = 0$ and for $c_1 = (-\Pi'_q) \bmod p_{q+1}$,

$$m_{c_1} = 1.$$

For all $c \in \mathcal{W}_q$ one has $c \bmod p_{q+1} = m_c c_1 \bmod p_{q+1}$

Remark 4-2-2: Using indices, we must solve (m, y') in the following equations for $a' \in \mathcal{W}'_q$:

$$a' + m\Pi'_q = \frac{p_{q+1} - 3}{2} + y'p_{q+1}$$

so we will let $c = a' - \frac{p_{q+1}-3}{2}$.

Proof: Because Π'_q and p_{q+1} are coprime, existence and unicity of the solution are well-known. In [9] we introduced the concept of normalizer of such a Diophantine equation, and have shown its additive and multiplicative property.

Clearly if $c \equiv d\ [p_{q+1}]$ then $(m_c - m_d)\Pi'_q \equiv 0\ [p_{q+1}]$ and as Π'_q and p_{q+1} are coprime, $m_c \equiv m_d\ [p_{q+1}]$.

Also, because $0 + 0.\Pi'_q = 0.p_{q+1}$ we deduce that $m_0 = 0$.

Then from the fact that $c_1 + \Pi'_q \in p_{q+1}\mathbb{Z}$ we get that $m_{c_1} = 1$.

Furthermore, for all c, by multiplicative property:

$$m_{m_c c_1} \equiv m_c.m_{c_1} \equiv m_c\ [p_{q+1}]$$

Thus, $c \equiv -m_c\Pi'_q \equiv -m_{m_c c_1}\Pi'_q \equiv m_c c_1\ [p_{q+1}]$.

This proposition gives us an effective way of building all couples (c, m_c) modulo p_{q+1}: start from $(c_1, 1)$ and add it to itself (modulo p_{q+1}) up to $p_{q+1} - 1$ times (the last time we will get the couple $(0,0 = m_0)$).

Corollary 4-2-2: $\mathcal{W}_q$ and $\mathcal{W}'_q$ have $\prod_{k=1}^{q}(p_k - 1)$ elements.

Proof: Let us proceed by induction on q. The property is true for $q = 0$. Assume it is true for a given $q \in \mathbb{N}$. From proposition 4-2-1,

$$\mathcal{W}'_{q+1} = \left(\bigcup_{m=0}^{p_{q+1}-1} (\mathcal{W}'_q + m\Pi'_q)\right) \setminus \left\{\frac{p_{q+1} - 3}{2} + y'p_{q+1}, y' \in [\![0, \Pi'_q - 1]\!]\right\}.$$

Thus $\bigcup_{m=0}^{p_{q+1}-1}(\mathcal{W}'_q + m\Pi'_q) = \bigcup_{c' \in \mathcal{W}'_q}(c' + \Pi'_q[\![0, p_{q+1} - 1]\!])$ has exactly $p_{q+1}\prod_{k=1}^{q}(p_k - 1)$ elements, from which we must remove the indices of multiples of p_{q+1}. For a given $c' \in \mathcal{W}'_q$, from proposition 4-2-2 there is exactly one couple (m, y) such that:

$$c' + m\Pi'_q = \frac{p_{q+1} - 3}{2} + y' p_{q+1}$$

i.e. there is only one element of $c' + \Pi'_q [\![0, p_{q+1} - 1]\!]$ in $\left\{\frac{p_{q+1}-3}{2} + y' p_{q+1}, y' \in [\![0, \Pi'_q - 1]\!]\right\}$. So in total there are exactly $\prod_{k=1}^{q}(p_k - 1)$ elements in $\left(\bigcup_{m=0}^{p_{q+1}-1}(\mathcal{W}'_q + m\Pi'_q)\right) \cap \left\{\frac{p_{q+1}-3}{2} + y' p_{q+1}, y' \in [\![0, \Pi'_q - 1]\!]\right\}$, thus $(p_{q+1} - 1)\prod_{k=1}^{q}(p_k - 1) = \prod_{k=1}^{q+1}(p_k - 1)$ elements in $\mathcal{W}'_{q+1}$.

<u>**Proposition 4-2-3:**</u> ${\mathcal{W}'_q}^{k_{Max}} = \mathcal{W}'_q \cap [\![0, k_{Max}]\!]$ verifies the following induction property.

For all $q \in \mathbb{N}, {\mathcal{W}'_{q+1}}^{k_{Max}}$ is equal to:

$$\left(\left(\bigcup_{m=0}^{\min\left(p_{q+1}-1,\left\lfloor\frac{k_{Max}}{\Pi'_q}\right\rfloor\right)}\left({\mathcal{W}'_q}^{k_{Max}} + m\Pi'_q\right)\right) \setminus \left\{\frac{p_{q+1} - 3}{2} + y' p_{q+1}, y' \in \left[\!\!\left[0, \min\left(\Pi'_q - 1, \left\lfloor\frac{2k_{Max} + 3}{2p_{q+1}} - \frac{1}{2}\right\rfloor\right)\right]\!\!\right]\right\}\right) \cap [\![0, k_{Max}]\!]$$

<u>*Proof*</u>: Let $x \in {\mathcal{W}'_{q+1}}^{k_{Max}}$. From proposition 4-2-1, there exists $c' \in \mathcal{W}'_q$, $m \in [\![0, p_{q+1} - 1]\!]$ such that $x = c' + m\Pi'_q$. But $x \le k_{Max}$ so $m \le \lfloor k_{Max}/\Pi'_q \rfloor$. Furthermore, $x \notin \left\{\frac{p_{q+1}-3}{2} + y' p_{q+1}, y' \in [\![0, \Pi'_q - 1]\!]\right\}$ so a fortiori:

$$x \notin \left\{\frac{p_{q+1} - 3}{2} + y' p_{q+1}, y' \in \left[\!\!\left[0, \min\left(\Pi'_q - 1, \left\lfloor\frac{2k_{Max} + 3}{2p_{q+1}} - \frac{1}{2}\right\rfloor\right)\right]\!\!\right]\right\}.$$

Conversely, let $x \in \left(\bigcup_{m=0}^{\min(p_{q+1}-1,\lfloor k_{Max}/\Pi'_q\rfloor)}\left({\mathcal{W}'_q}^{k_{Max}} + m\Pi'_q\right)\right) \cap [\![0, k_{Max}]\!]$ such that $x \notin \left\{\frac{p_{q+1}-3}{2} + y' p_{q+1}, y' \in \left[\!\!\left[0, \min\left(\Pi'_q - 1, \left\lfloor\frac{2k_{Max}-3}{2p_{q+1}} - \frac{1}{2}\right\rfloor\right)\right]\!\!\right]\right\}$. The first condition means that $x \in \mathcal{W}'_{q+1}$ if $x \notin \left\{\frac{p_{q+1}-3}{2} + y' p_{q+1}, y \in [\![0, \Pi'_q - 1]\!]\right\}$. But if that were the case, there would be $y' \in [\![1, \Pi'_q - 1]\!]$ such that $x = \frac{p_{q+1}-3}{2} + y' p_{q+1}$. Thus $y \le \frac{k_{Max} - (p_{q+1}-3)/2}{p_{q+1}} = \frac{2k_{Max}+3}{2p_{q+1}} - \frac{1}{2}$, which cannot happen because $x \notin \left\{\frac{p_{q+1}-3}{2} + y' p_{q+1}, y' \in \left[\!\!\left[0, \min\left(\Pi'_q - 1, \left\lfloor\frac{2k_{Max}+3}{2p_{q+1}} - \frac{1}{2}\right\rfloor\right)\right]\!\!\right]\right\}$.

4.3 Wheel sieve algorithms

As per sections 4.1 and 4.2, the wheel sieve algorithms will consist in two steps:

(A) A first step where the wheel will always grow, as long as $\Pi_q < N_{Max}$, or:

$$\Pi'_q - 1 < k_{Max},$$

(B) A second step where we will no longer grow the wheel, but will have to keep eliminating composite numbers, as long as $p_q^2 < N_{Max}$, or:

$$\frac{p_q^2 - 3}{2} < k_{Max.}$$

This is equivalent to saying that we replace $\mathcal{W}_{q+1}$ by $\mathcal{W}_{q+1}^{N_{Max}}$ and similarly $\mathcal{W}'_{q+1}$ by ${\mathcal{W}'_{q+1}}^{k_{Max}}$. During step (B) we do not add new pseudo-primes, only remove those that we rule out as multiples of the next prime. Because Π_q grows exponentially, there will generally be more iterations in step (B) than in step (A).

<u>Quick description of the steps of the index wheel sieve algorithm (see appendix for full algorithm)</u>:

As for the previous algorithms, we note L_p the list of primes and i_l its number of elements. IL_p represents the list of indices of odd primes, and SIL_p the list of indices of squared odd primes. At step q, L_p will contain all primes up to p_q^2, coming from the wheel $\mathcal{W}'_q$, IL_p and SIL_p being filled with the corresponding indices.

1- Intialisation of the sieve for $q = 1$: $L_p = \{2,3,5,7\}$, $i_l = 4$ $IL_p = \{0,1,2\}$, $SIL_p = \{3,11,23\}$ and $\mathcal{W}'_1 = \{1, 2\}$ with $\Pi'_1 = 3$.
2- While $\Pi'_q < k_{max}$ (step A):
   a. We take p_{q+1} from L_p (or equivalently the first element of $\mathcal{W}_q$). The list of pairs (c, m_c) such that $c + m_c\Pi'_q$ has to be eliminated is then computed, according to proposition 4-2-2. Then we build the wheel $\mathcal{W}'_{q+1}$.
   b. Once this is done primes in the interval $[\![p_{i_l-1} + 2, p_{q+1}^2 - 2]\!]$ are added to L_p and i_l, IL_p and SIL_p are updated accordingly. Indices of the primes to add are those in $\mathcal{W}'_{q+1} \cap [\![IL_p(i_l - 2) + 1, SIL_p(q) - 1]\!]$.
3- While $SIL_p(q) < k_{max}$ (step B):
   a. Remove indices of multiples of p_{q+1} from ${\mathcal{W}'_q}^{k_{Max}}$ to get ${\mathcal{W}'_{q+1}}^{k_{Max}}$.
   b. Once this is done primes in the interval $[\![p_{i_l-1} + 2, p_{q+1}^2 - 2]\!]$ are added to L_p and i_l, IL_p and SIL_p are updated accordingly. Indices of the primes to add are those in $\mathcal{W}'_{q+1} \cap [\![IL_p(i_l - 2) + 1, SIL_p(q) - 1]\!]$.

Remark 4-3-1: Let k_1 and k_2 be the indices of two odd numbers, respectively n_1 and n_2, such as $n_2 - n_1 > 0$. Let $\alpha = k_2 - k_1$. The difference between the indices n_1^2 and n_2^2 is:

$$\beta = 2\alpha^2 + 2\alpha n_1.$$

Furthermore, if m is another integer, the difference between the indices of $n_1 m$ and $n_2 m$ is:

$$\gamma = \alpha m.$$

Proof: Note that $n_2 - n_1 = 2\alpha$ and thus:

$$\frac{n_2^2 - 3}{2} - \frac{n_1^2 - 3}{2} = \frac{1}{2}(n_2 - n_1)(n_2 + n_1) = \alpha(n_2 + n_1) = \alpha(2n_1 + 2\alpha) = \beta.$$

Similarly:

$$\frac{n_2 m - 3}{2} - \frac{n_1 m - 3}{2} = \alpha m = \gamma.$$

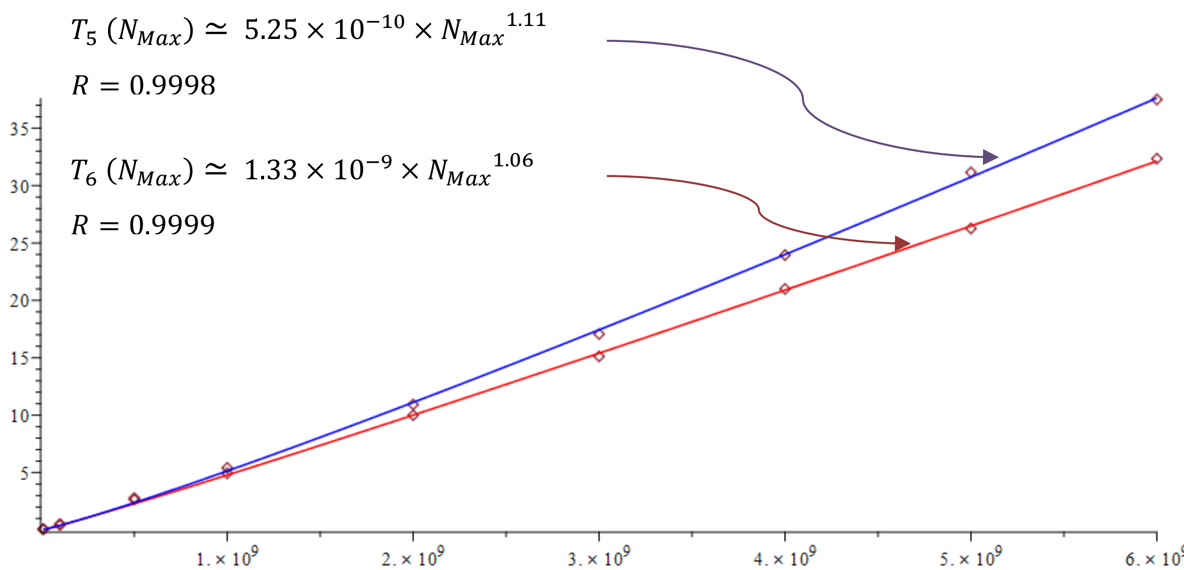
This last remark is used in steps 2-b. and 3-b. to fill SIL_p and to perform step 3-a.

Remark 4-3-2: The index wheel sieve involves operations with reduced input size compared with the number version. This is clear from remark 4-3-1 where β is exactly half of $n_2^2 - n_1^2$, for instance. Similarly Π'_q is half of Π_q so modulo operation input is also reduced.

4.4 Performance of algorithms

In this section, we present results from the previous algorithm of index wheel sieve, which we compare with a similar one on numbers (unspecified for to avoid a lengthy duplication). These results are similar to those obtained in the previous sections. As for the sieve of Atkin, we did not go for refinements that give a better time complexity, so theoretical complexity in terms of number of operations is $O(N)$ for both algorithms.

On the graph 4-4 below, we plot the computation time in seconds for both algorithms, for N_{Max} up to 6.10^9. The curve T_5 corresponds to the the algorithm ***WheelSieveReference*** and the curve T_6 corresponds to the the algorithm ***IndexWheelSieve***. The correlation coefficient R of each regression is given on the graph. Details of the Maple options used to get the regression are given in appendix 8.3. We notice that complexity of both algorithms again seems empirically slightly higher than linear.



Graph 4-4: computation time $T\ (N_{Max})$ in seconds for both algorithms (Wheel sieve)

Complexity is reduced by using indices, due to reduction of input size in the modulo and the multiplication operations (see Remark 4-3-2) and despite a higher number of operations with the algorithm **IndexWheelSieve**. Moreover, the amount of memory space used with indices is halved, due to the fact that we avoid even numbers completely.

5 Conclusion

In theory, indices are a way to work with odd numbers only by not representing even numbers. Most mathematical relations must be reformulated for indices, which lead to a higher number of (conversion) operations, but in return the input size of other operations is reduced. In this article, we have shown how this indexing translates into optimized algorithms in applied mathematics. From a basic primality test implementation, to the sieve of Atkin and Pritchard's wheel sieve, indices speeded up these algorithms, not by changing their complexity but by reducing the time cost by a constant factor, and generally also made them more efficient from a memory point of view.

Acknowledgments: We would like to thank François-Xavier VILLEMIN for his attentive comments and suggestions.

REFERENCES

[1]. René SCHOOF (2008), *Four primality testing algorithms*. Algorithmic Number Theory, MSRI Publications, Volume 44. http://www.math.leidenuniv.nl/~psh/ANTproc/05rene.pdf.

[2]. Manindra AGRAWAL, Neeraj KAYAL, Nitin SAXENA (2004), *PRIMES is in P*. Ann. of Math. (2) 160, No. 2, pp. 781-793. MR2123939 (2006a:11170). http://annals.math.princeton.edu/wp-content/uploads/annals-v160-n2-p12.pdf.

[3]. Paul PRITCHARD (1981), *A sublinear additive sieve for finding prime numbers*. Communications of the ACM 24(1), pp. 18-23. https://dl.acm.org/doi/pdf/10.1145/358527.358540?download=true.

[4]. Paul PRITCHARD (1982), *Explaining the Wheel Sieve*. Acta Informatica 17, pp. 477-485. http://fuuu.be/polytech/INFOF404/Doc/Explaining%20the%20wheel%20sieve.pdf.

[5]. Paul PRITCHARD (1994), *Improved Incremental Prime Number Sieves*. Algorithmic Number Theory Symposium. pp. 280–288. CiteSeerX 10.1.1.52.835. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.835&rep=rep1&type=pdf.

[6]. Arthur ATKIN AND Daniel BERNSTEIN (2003), *Prime sieves using binary quadratic forms*. Mathematics of Computation Volume 73, Number 246, pp. 1023-1030. https://www.ams.org/journals/mcom/2004-73-246/S0025-5718-03-01501-1/S0025-5718-03-01501-1.pdf.

[7]. Gabriel PAILLARD, Felipe FRANCA, Christian LAVAULT (2013), *A distributed wheel sieve algorithm using Scheduling by Multiple Edge Reversal*. HAL-00794389. https://hal.archives-ouvertes.fr/hal-00794389.

[8]. Marc WOLF, François WOLF (2018), *Representation theorem of composite odd numbers indices*. SCIREA Journal of Mathematics, Vol. 3, pp. 106-117. http://article.scirea.org/pdf/11066.pdf.

[9]. Marc WOLF, François WOLF, Corentin LE COZ (2018), *Calculation of extended gcd by normalization*. SCIREA Journal of Mathematics. Vol. 3, pp. 118-131. http://article.scirea.org/pdf/11067.pdf.

6 APPENDIX: ALGORITHM OF THE INDEX WHEEL SIEVE

This algorithm enumerates odd primes up to the limit N_{Max}. It is composed of a main function that is called ***IndexWheelSieve*** and the following auxilliary other functions:

7-2- **DiophantineSolutions**($Prime$,Π'_q)
7-3- **WheelTurn**($\mathcal{W}'_q, q, Prime, PrimeIndex, \Pi'_q, k_{Max}$)
7-4- **RemoveMultiples**($SquarePrimeIndex, Prime, \mathcal{W}'_q$)
7-5- **GetNewPrimes**($\mathcal{W}'_q, q, L_p, i_l, IL_p, SIL_p$)

Some marginal optimizations can still be performed, for instance modulo operations inside a loop can be replaced by substractions, and memory can be managed better. For the sake of readability we leave these optimizations out of scope.

Algorithm 6-1 ***IndexWheelSieve***(N_{Max}): $N_{Max} > 9$ is an odd integer.

This function returns the list of all prime numbers up to N_{Max}.

First step : intialisation of variables

$L_p \leftarrow \{2,3,5,7\}$ → Dynamic list of primes

$i_l \leftarrow 4$ → Number of primes in the list

$k_{Max} \leftarrow (N_{Max} - 3)/2$ → Index of N_{Max}

$IL_p \leftarrow \{0,1,2\}$

$SIL_p \leftarrow \{3,11,23\}$

$\mathcal{W}'_q \leftarrow \{1,2\}$

$\Pi'_q \leftarrow 3$

$q \leftarrow 1$

Second step : Wheel inflation.

Do

$Prime \leftarrow L_p(q+1)$

$PrimeIndex \leftarrow IL_p(q)$

$\Pi'_{q+1} \leftarrow \Pi'_q \times Prime$

→ Compute values of the new wheel from the previous one

$\mathcal{W}'_q \leftarrow$ ***WheelTurn***$(\mathcal{W}'_q, q, Prime, PrimeIndex, \Pi'_q, k_{Max})$

GetNewPrimes$(\mathcal{W}'_q, q, L_p, i_l, IL_p, SIL_p)$

$\Pi'_q \leftarrow \Pi'_{q+1}$

$q \leftarrow q+1$

While $k_{Max} > \Pi'_q$

Third step : Wheel deflation.

While $SIL_p(q-1) < k_{Max}$

$Prime \leftarrow L_p(q+1)$

$SquarePrimeIndex \leftarrow SIL_p(q)$

$\mathcal{W}'_q$ ←**RemoveMultiples**($SquarePrimeIndex, Prime, \mathcal{W}'_q$)

GetNewPrimes ($\mathcal{W}'_q, q, L_p, i_l, IL_p, SIL_p$)

$q \leftarrow q + 1$

End While

Return (L_p, i_l)

Algorithm 6-2 DiophantineSolutions($Prime$,Π'_q)

$c_1 \leftarrow Prime - \left(\Pi'_q \bmod Prime\right)$ → Solution such that $m = 1$

$c \leftarrow 0$

$Solutions \leftarrow \{0 \dots 0\}$ → Array of size $Prime$

For $m = 1$ **To** $Prime - 1$ **Do**

$c \leftarrow (c + c_1) \bmod Prime$

$Solutions(c) \leftarrow m$

End For

Return $Solutions$

Algorithm 6-3 WheelTurn($\mathcal{W}'_q, q, Prime, PrimeIndex, \Pi'_q, k_{Max}$)

This function computes $\mathcal{W}'_{q+1}$ by duplicating the wheel $\mathcal{W}'_q$ and removing indices of multiples of $Prime = p_{q+1}$.

First step : Compute all the pairs (c, m_c) *in the function* **DiophantineSolutions**

$Solutions \leftarrow$ **DiophantineSolutions**($Prime$,Π'_q)

Second step : Iteration

$WheelSize \leftarrow$ **Size**($\mathcal{W}'_q$)

$Table \leftarrow$ **Range**($\{\}, Prime$)

For $j = 0$ **To** $WheelSize - 1$ **Do**

$a' \leftarrow \mathcal{W}'_q(j)$

$c \leftarrow (a' - PrimeIndex) \bmod Prime$

$m \leftarrow Solutions(c)$

For $y = 0$ **To** $PrimeNumber - 1$ **Do**

$n \leftarrow a' + y\Pi'_q$

If $n > k_{max}$ **Do**

Break

End If

If $y \neq m$ **Do**

Append($Table(y)$,n)

End If

End For

End For

Third step : Build $\mathcal{W}'_{q+1}$ ***by concatenation***

$\mathcal{W}'_{q+1} \leftarrow \{\}$

For $y = 0$ **To** $PrimeNumber - 1$ **Do**

Concatenate($\mathcal{W}'_{q+1}$,$Table(y)$)

End For

Return $\mathcal{W}'_{q+1}$

Algorithm 6-4 RemoveMultiples($SquarePrimeIndex, Prime, \mathcal{W}'_q$)

$\mathcal{W}'_{q+1} \leftarrow \{\}$

$NextMultiple \leftarrow SquarePrimeIndex$

For $j = 1$ **To Size**($\mathcal{W}'_q$)-1 **Do**

If $\boldsymbol{\mathcal{W}}'_q(j) > NextMultiple$ **Do**

$NextMultiple \leftarrow NextMultiple + Prime$

$j \leftarrow j - 1$

Else If $\boldsymbol{\mathcal{W}}'_q(j) = NextMultiple$ **Do**

$NextMultiple \leftarrow NextMultiple + Prime$

Else

Append($\mathcal{W}'_{q+1}$,$\boldsymbol{\mathcal{W}}'_q(j)$)

End If

End For

Return $\mathcal{W}'_{q+1}$

Algorithm 6-5 GetNewPrimes($\mathcal{W}'_q, q, L_p, i_l, IL_p, SIL_p$)

This function adds new primes to the list L_p and updates i_l and the other lists IL_p and SIL_p (all passed by reference).

$SquareIndex \leftarrow SIL_p(q+1)$

$j \leftarrow i_l - q - 2$ → Offset to take into account already known primes

$NewPrimeIndex \leftarrow \mathcal{W}'_q(j)$

While $NewPrimeIndex < SquareIndex$ **Do**

$IL_p(i_l - 1) \leftarrow NewPrimeIndex$

$\alpha \leftarrow IL_p(i_l - 1) - IL_p(i_l - 2)$

$SIL_p(i_l - 1) \leftarrow SIL_p(i_l - 2) + 2\alpha^2 + 2\alpha L_p(i_l - 1)$

$L_p(i_l) \leftarrow L_p(i_l - 1) + 2\alpha$

$i_l \leftarrow i_l + 1$

$j \leftarrow j + 1$

$NewPrimeIndex \leftarrow \mathcal{W}'_q(j)$

End While

---

7 APPENDIXES: MAPLE REGRESSIONS

Here are numeric values obtained from our implementation (Visual Studio C++ 2012) of the algorithms presented in this article.

7.1 BASIC PRIMALITY TEST AND PRIMES ENUMERATION

In table 8.1, numeric values of $T_1(N_{Max})$ and $T_2(N_{Max})$ are obtained respectively from the ***PrimeEnumeration*** and ***IndexPrimeEnumeration*** algorithms.

Table 2: numeric values of $T_1(N_{Max})$ and $T_2(N_{Max})$ in seconds.

N_{Max}	10^7	10^8	5×10^8	10^9	2×10^9	3×10^9	4×10^9
$T_1(N_{Max})$	2.403	56.031	493.163	1306.884	3414.713	6271.249	8908.814
$T_2(N_{Max})$	2.375	54.725	487.568	1275.921	3329.573	6105.386	8664.438

To fit these observations, Maple's **NonlinearFit** function is used with the parameters below. Initial values for a and b were determined empirically.

NonlinearFit($a\times n^b/\ln(n)$, X, Y, n, initialvalues = [$a = 5.9\times10^{-9}$, $b = 1.41$], output = [leastsquaresfunction, residuals])

We get the following mathematical relationships:

$$T_1(N_{Max}) \simeq 5.79409775129480\times10^{-9}\times\frac{n^{1.40966993452829}}{\ln(n)},\ R = .99962000$$

$$T_2(N_{Max}) \simeq 6.10602965467609\times10^{-9}\times\frac{n^{1.406040046365699}}{ln(n)},\ R = .99962009$$

7.2 THE SIEVE OF ATKIN

In table 8.2, numeric values of $T_3(N_{Max})$ and $T_4(N_{Max})$ are obtained respectively from the ***SieveOfAtkin*** and ***IndexSieveOfAtkin*** algorithms.

Table 3: numeric values of $T_3(N_{Max})$ and $T_4(N_{Max})$ in seconds.

N_{Max}	10^8	5×10^8	10^9	1.5×10^9	1.6×10^9	2×10^9	3×10^9
$T_3(N_{Max})$	0.719	3.797	8.033	12.48	13.967	18.843	28.217
$T_4(N_{Max})$	0.727	3.921	8.225	12.152	12.953	16.507	25.342

N_{Max}	4×10^9	5×10^9	6×10^9	7×10^9	8×10^9	9×10^9	10^{10}
$T_3(N_{Max})$	41.534	54.871	72.044	84.511	100.727	116.093	133.184
$T_4(N_{Max})$	35.27	46.261	57.418	70.311	84.291	98.047	110.96

This time we used Maple's function **Fit** as below:

Fit($a\times n^2+b\times n$, X, Y, n, summarize = embed)

We get the following mathematical relationships:

$T_3(N_{Max}) \simeq 4.90268369826396\times10^{-19}\times {N_{Max}}^2 + 8.54576412559177\times10^{-9}\times N_{Max}$, $R =$ .999647

$T_4(N_{Max}) \simeq 3.78795281632082\times10^{-19}\times {N_{Max}}^2 + 7.39595089422000\times10^{-9}\times N_{Max}$,

$R = .999926$

7.3 WHEEL SIEVE WITH INDICES

In table 4, numeric values of $T_5(N_{Max})$ and $T_6(N_{Max})$ are obtained respectively from the ***WheelSieveReference*** and ***IndexWheelSieve*** algorithms.

Table 4 : numeric values of $T_5(N_{Max})$ and $T_6(N_{Max})$ in seconds.

N_{Max}	10^7	10^8	5×10^8	10^9	2×10^9	3×10^9	4×10^9	5×10^9	6×10^9
$T_5(N_{Max})$	0.071	0.496	2.783	5.407	10.931	17.070	23.944	31.150	37.501
$T_6(N_{Max})$	0.064	0.457	2.657	4.936	9.995	15.121	20.995	26.260	32.351

We used again **NonlinearFit** with empirically determined initial values a and b:

NonlinearFit($a \times n^b$, X, Y, n, initialvalues = [$a = 1.97461115539853 \times 10^{-6}$, $b = 1.1$], output = [leastsquaresfunction, residuals]).

We get the following mathematical relationships:

$$T_5(N_{Max}) \simeq 5.25118782575365 \times 10^{-10} \times n^{1.11016647384427}, R = .99982444$$

$$T_6(N_{Max}) \simeq 1.33020583039257 \times 10^{-9} \times n^{1.06187203820827}, R = .99986693$$

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 8 No 2
ISSN 2054-7390

COSM: Controlled Over-Sampling Method. A Methodological Proposal to Overcome the Class Imbalance Problem in Data Mining

Gaetano Zazzaro
CIRA, Italian Aerospace Research Centre, Capua (CE), Italy;
g.zazzaro@cira.it

ABSTRACT

The class imbalance problem is widespread in Data Mining and it can reduce the general performance of a classification model. Many techniques have been proposed in order to overcome it, thanks to which a model able to handling rare events can be trained. The methodology presented in this paper, called Controlled Over-Sampling Method (COSM), includes a controller model able to reject new synthetic elements for which there is no certainty of belonging to the minority class. It combines the common Machine Learning method for holdout with an oversampling algorithm, for example the classic SMOTE algorithm. The proposal explained and designed here represents a guideline for the application of oversampling algorithms, but also a brief overview on techniques for overcoming the problem of the unbalanced class in Data Mining.

***Keywords*:** Class imbalance problem; Data Mining; Holdout Method; Oversampling; Rare Class Mining; Undersampling.

1 Introduction

In many real application fields, the discovery and modeling of rare events is crucial for understanding complex phenomena [1]. For example, rare weather conditions, if not forecasted, can be very dangerous for the population, housing, air traffic, and so on; unauthorized and fraudulent use of a credit card must be detected as soon as possible; an unidentified cyberattack is very dangerous for companies, causing huge economic losses. Sometimes such events are so diluted in the database that the Data Mining algorithms used for training analytical models fail to characterize them: such events are exchanged as noise [2]; if these events constitute a class value (+), the trained model could always give the same answer (-), ignoring the minority class. The main problem with class imbalance states is that standard models are often biased towards the majority class.

In Data Mining this condition is called class imbalance problem and it occurs when one of the two classes (in the binary case) has many more samples than the other class. What “many more” means is not clearly quantifiable and depends on the case. Most of the time, being able to train a model capable of classifying rare events, in conditions of high class imbalance, is an impossible goal, unless ad hoc strategies are first adopted. This problem is one of the main problem that degrades the performance of classification models [3] [4]. Various techniques have been proposed in order to solve the problem of class imbalance, including

DOI: 10.14738/tmlai.82.7925
Publication Date: 02nd May, 2020
URL: http://dx.doi.org/10.14738/tmlai.82.7925

over-sampling of the minority class or under-sampling of the majority one. Another widely used approach is focused on the cost-sensitive learning techniques included in meta-learning approaches [5]. These techniques take the misclassification cost in its account by assigning higher cost of misclassifications to the minority class, penalizing the correct classifications to the majority class, and generating the model with lowest cost.

In this paper, a design of a methodology for oversampling is proposed. By using an oversampling technique, the minority class is oversampled by taking each minority class sample and adding new synthetic records by applying various strategies that are deepened and compared. The method is called Controlled Over-sampling Methodology (COSM) because a classification model – the controller – is created that can check if the new synthetic examples really belong to the minority class. The controller assists the entire sampling procedure, eventually rejecting the misclassified examples.

Various aspects of the proposed method are also considered, including its relationship with the holdout method.

2 Holdout Method

The holdout method [6] is a very common strategy in Data Mining, mainly aimed at providing a useful scheme for datasets split and design, in order to train a model and evaluate its performances.

2.1 Basic Holdout

The whole dataset is randomly partitioned into two disjoint sets, called training and test sets. It is common to hold out one-third of data for testing and use the remaining two-third for training, but other proportions are possible, depending on the amount of data and other factors discussed later in the paper.

This simple subdivision does not take into consideration the distribution of the target class. In spite of the random partitioning, the two subsets could have very different distributions of the target class. In order to overcome this unlikely event, the training and test sets must not only be obtained randomly but they must be also stratified, so that the class distribution of the records in each set is approximately the same as that in the initial dataset.

This subdivision scheme can be enriched by considering a further subset of the training set, called validation set. More in detail, the validation set is used to select the algorithm parameters and then choose the model with the best performance metrics. This step is essential to mitigate the overfitting problem [7] [8]. Also in this case, the subdivision is random, can be stratified, and the subdivision percentages can vary, or be the same as the first splitting.

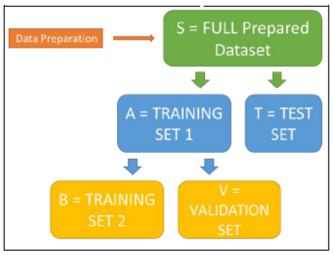


Figure 1. The general holdout method schema

Figure. 1 shows the framework of the complete holdout method. To recap:

- S is the dataset with all the records; it can be subpart of a much larger database; this set has undergone all the preparation steps: for example, possible selection, cleaning and normalization of data, selection and extraction of features if useful, and any other operations on the data.
- A is the intermediate Training set, starting from which the final training is obtained.
- B is the final Training set; a model, for example a classifier, is built on this set by applying mainly a statistical or a Machine Learning algorithm.
- V is the Validation set; it is useful to tune and select the parameters (or hyper-parameters) of the algorithm chosen for training the model. In other words, V is used to compare the performances of all the trained models and decide which one to take.
- T is the Test set; the model is then tested on this set; T is used to obtain the performance metrics, such as accuracy, sensitivity, specificity, AUC, F-measure, and so on; moreover, T is useful to understand if the model is overfitted. Generally, $T \cup A = S$.

The holdout method is not recommended when working with small datasets. In these cases, some variations may be applied to avoid that the dataset subdivisions can further reduce the number of data for the training and test phases.

2.2 Some Variations on the Theme: k-Fold Cross-Validation

The holdout method is the simplest kind of cross-validation method; the latter represents a more general method. In this approach, also called k-fold cross-validation, the original dataset is randomly partitioned into k (generally $k=10$) equal sized subsamples. For each of k experiments, $k-1$ folds are used for the training phase and the remaining one for testing phase. This procedure is repeated k times. The error estimates are averaged to yield an overall error estimate, as well as the other performance metrics. k-fold cross-validation seems to give better approximations of generalization than the holdout method [6] [8].

In some uses of the method described, such as when the multi-division of holdout reduces the number of records in the training set too much, a mixed approach between holdout and k-fold cross-validation can be applied. In few words, the training set is not further subdivided into validation (V), but the model is trained on Training set (A) with the k-fold cross-validation method: we can say that the model is cross-validated. Finally, as usual, it is tested on the test set (T) in order to calculate the performances of model.

3 Class Imbalance Problem

The class imbalance problem consists in a skewed distribution of instances that belong to different classes; because class distribution plays a key role in Data Mining and Machine Learning classification task, this problem can compromise the training phase and the performance of the classification model.

3.1 Examples in Real Datasets

In many real world applications, datasets suffer the problem of the class imbalance. In these situations, discovering instances of rare class is “akin to finding a needle in a haystack”. Furthermore, a model able to describe the minority class tends to be highly specialized, and this condition is not desired because a good model is a model that is able to generalize, otherwise the model goes into overfitting. However, most Data Mining algorithms do not work very well with imbalanced datasets [8].

Briefly, a dataset is affected by the unbalanced class problem when one of the classes has many more samples than the other ones. The most of machine learning algorithms is more focusing on classification of samples belonging to the majority class while ignoring or misclassifying samples of the minority one.

For example, if the target class has only two values (binary class case) "0" and "1", and if the distribution is 99.9% of "1" and 0.01% of "0", a classifier that always says "1" is a very accurate model, because it never exhibits a false "0" and has a very low percentage (precisely the 0.01%) of false "1".

There exist many case studies that do not have a balanced data set. Some examples are:

- Discovery of fake news;
- Distinction among earthquakes, nuclear and non-nuclear explosions;
- Document selection and filtering;
- Forecast of extreme weather conditions;
- Recognition of fraudulent telephone calls.

3.2 Strategies to Handle Imbalanced Datasets

As mentioned above, imbalance datasets can degrade the performance of a model that has been trained by applying a data driven technique; the Machine Learning algorithms lead to misclassifying the minority class records or treated them as noise. Even if the evaluation metric is changed, it is hard for the model to be accurate on the minority class, or that the chosen metric has a good result.

Many techniques have been proposed in order to overcome the problem of learning models on an unbalanced class. They can be categorized into three main categories: re-sampling [8], cost-sensitive learning [9] [10], and ensemble-based methods [11]. Some of them are summarized in Table 1. Nevertheless, the choice of the strategy to be followed strictly depends on the data and on the learning algorithm, and there is no absolute advantageous technique.

Strategies for overcoming the problem of the unbalanced class can be natively incorporated into the learning algorithm for training a classifier.

Table1. Techniques for Imbalanced Problem

Category	Technique	Algorithm	Reference
Re-Sampling	Under-sampling	Random Undersampling	[8]
		Clustering-based	[12]
	Over-sampling	SMOTE	[13]
		ADASYN	[14]
		Clustering-based	[15]
Cost Sensitive Learning	Direct	ICET	[16]
		CSDTree	[17]
	Meta-Learning	MetaCost	[18]
		CSC	[19]
		CSnaiveBayes	[20]
		Empirical Thresholding	[21]
Ensemble-based	Bagging-based	SMOTEBaggings	[22]
		Asymetric Bagging	[23]
		Ensemble Variation	[24]
	Boosting-based	SMOTEBoost	[25]
		RUSBoost	[26]
	Hybrid	EasyEnsemble	[27]
		BalanceCascade	[27]

3.3 SMOTE and ADASYN

Synthetic Minority Over-sampling Technique (SMOTE) [13] is an oversampling method able to create new artificial examples of minority class based on the similarity among the existing elements. SMOTE is the most used algorithm for oversampling, and there are numerous variants of it [28] [29] [30].

Let x_i be a record belonging to the minority class, $\hat{x_i}$ one of the k-nearest neighbors of x_i, and δ_i a random number belonging to $[0,1]$. A new synthetic example of the minority class is calculated as: $x_{new}=x_i+(\hat{x_i}-x_i)\cdot\delta_i$

The new x belongs to the line between x_i and $\hat{x_i}$.

The main shortcoming of SMOTE is the problem of overgeneralization. SMOTE's algorithm does not regard to the majority class and, in the case of highly skewed class distributions, a harmful mixture of the classes is obtained.

However, SMOTE yields among the best results as far as re-sampling and modifying the probabilistic estimate techniques go [31].

Another very common oversampling algorithm is Adaptive Synthetic (ADASYN) sampling procedure [14]. Its key idea is, in few words, to automatically find the number of synthetic observations to be generated for each observation belonging to the rare class by using a density distribution function. The number of synthetic samples, generated for each observation of the minority class, is determined by the percentage of samples belonging to the majority class in its neighborhood. The steps of the ADASYN are:

- Calculate the ratio of minority to majority examples using $d=\frac{m_s}{m_l}$, where m_s and m_l are the number of minority and majority class examples respectively. d is the Degree of Imbalance.
- Calculate the total number of synthetic minority data to generate, by using $G=\beta\cdot(m_l-m_s)$; G is the total number of minority class data to generate. β is the ratio of minority: majority data desired after ADASYN. $\beta=1$ means a perfectly balance between two classes after ADASYN.
- For each x_i of the minority samples, find its k-nearest neighbors and calculate the ratio $r_i=\frac{\Delta_i}{k}$, where Δ_i is the number of majority class examples.
- Normalize the r_i values: $\hat{r_i}=\frac{r_i}{\sum r_i}$, and $\sum\hat{r_i}=1$.
- Calculate the amount of new synthetic examples to generate in each neighborhood: $G_i=G\hat{r_i}$.
- Generate G_i new data for each neighborhood, taking x_i. Select in random manner another minority example x_{zi} within the same neighborhood. The new synthetic example can be calculated by using:

$$x_{new}=x_i+(x_{zi}-x_i)\cdot\delta$$

where δ is a random number belonging to $[0,1]$, x_i and x_{zi} are two minority examples within a same neighborhood.

4 COSM Framework

One of the main disadvantages of the methods for oversampling is that the synthetic examples may not have the minority label or that they could never occur in the real world.

COSM is a general framework for the application of oversampling algorithms. Its main strength is the controller model C, trained using an undersampling technique $\mathcal{M}$ and a machine learning algorithm; furthermore, C is tested on an independent set, which has the same class distribution as the original dataset, and which is also used to test the final classifier $\mathcal{F}$.

4.1 General Description

Figure 2 shows the framework of COSM. COSM can be entirely employed, for example, by using the operators, filters and algorithms of the WEKA open source software [32] [33].

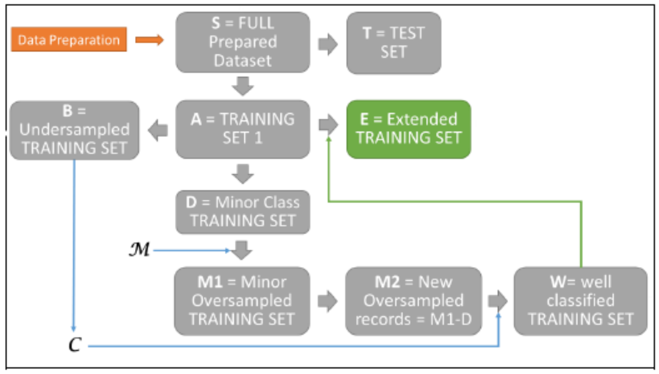


Figure 2. The complete schema of the subdivision of the dataset in COSM.

The Data Preparation step of CRISP-DM methodology [34] for the knowledge discovery in large database process covers all activities needed to build the final dataset from the raw data. After this phase, the full prepared and imbalanced dataset S is splitted into test set (T) and training set 1 (A), in accordance with the holdout method. T has the same distribution of the class target of S by applying a stratified filter, and T is, for example, the 34% of S.

The set B is obtained by randomly undersampling the set A. B has the same number of records tagged with (+) and (-)

Since the undersampling technique can lead to a loss of information, in a more advanced way, the random removal of minority class records (+) can be replaced by applying a “bootstrap” (“bagging”) approach [35]. In a nutshell, the set A is subdivided into N subsets, in which the elements of the minority class (+) are all fixed, while the records of the majority class (-) are randomly sampled with replacement in a number equal to records tagged with minority class (+). In this way, each of the N subsets is balanced, the elements tagged with “+” do not vary, and may have records tagged with “-” in common. In a formal way:

$B_i = D \cup R_i \; (i=1,\dots,N)$
where $D = \{x \in A : x \; has \; "+" \; class\}$ and $|D| = m$
$R_i = \{x \in A : x \; has \; "-" \; class\}, \; |R_i| = m, \forall i = 1,\dots,N$
$B = \bigcup B_{i \, i=1,\dots,N}$

The model C (Figure 3), called controller, is an ensemble [36] of N classifiers C_i $(i=1,\dots,N)$. Each C_i is a cross-validated classifier trained on a different balanced set B_i. Moreover, each C_i is trained by a different learning algorithm. Finally, the class can be obtained by taking a majority vote on the individual predictions of the C_i base classifier

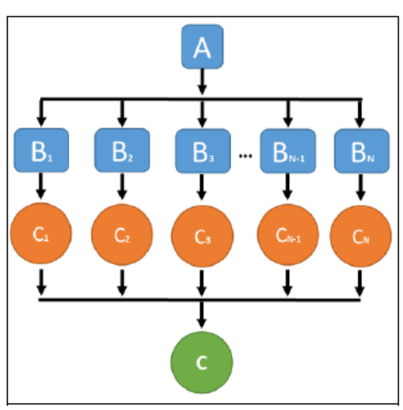


Figure 3. Combine classifiers in the ensemble schema of COSM

Additionally, C is tested on the set T, by calculating the metrics of the Table 2, based on confusion matrix [8].

The COSM procedure proceeds by considering the subset D of A consisting of only the m elements of the minority class (D={x∈A:x has "+" $class$}), and by applying an oversampling technique $\mathcal{M}$ to D in order to create a set (M_2) of new synthetic minority class examples.

M_1 is the set of all the minority class examples, including the new synthetic ones (M_2=M_1−D). The number of elements of M_2 depends on $\mathcal{M}$ and its parameters.

The controller model C is applied on the new set M_2 in order to reject the examples that are misclassified by C: these elements are false positives according to classifier C.

Table 2. Classification Performance Evaluation Metrics

Name	Formula	Description
Accuracy	Acc=(TP+TN)/ (TP+TN+FP+FN)	Fraction of correct predictions on the total number of predictions
True Positive Rate / Sensitivity	TPR=TP/(TP+FN)	Fraction of positive examples predicted correctly by the classifier
True Negative Rate / Specificity	TNR=TN/(TN+FP)	Fraction of negative examples predicted correctly by the classifier
False Positive Rate	FPR=FP/(TN+FP)	Fraction of negative examples predicted as a positive class
False Negative Rate	FNR=FN/(TP+FN)	Fraction of positive examples predicted as a negative class
Precision	Prc=TP/(TP+FP)	Fraction of examples classified as positive that are really positive
AUC		Area Under the ROC Curve

Finally, W is the subset of M_2 well classified by C: it is made up of non-rejected elements. And E=A∪W is the new extended training set, which is the training set for the final classification model $\mathcal{F}$ and which is tested on the test set T. The performances of $\mathcal{F}$ on T can be compared with the performances of C on T.

5 Conclusion

Overcoming the problem of the unbalanced class depends on numerous elements. It depends on complexity of the data, severity of class imbalance, size of data and classifier involved. The framework designed here can be applied independently of the Machine Learning algorithm or the selected oversampling technique.

COSM needs to be tried and tested, especially to define a strategy for selecting the following its parameters:

- the percentage of S to get the test set T;
- the algorithm to obtain the controller model C trained on set B;
- the oversampling technique $\mathcal{M}$;
- the algorithm to obtain the final classifier trained on the set E.

As mentioned above, COSM can be entirely employed, for example, by using WEKA software. The paper describes the design of the methodology, while its implementation with all the experimental tests will be addressed in a future work.

ACKNOWLEDGMENT

The author would mention the Big Data Facility and COND-IWT projects, both funded by the Italian PRORA.

REFERENCES

[1] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, *Classification with class imbalance problem: A review*, Int. J. Advance Soft Compu. Appl, vol. 7, no. 3, November 2015, pp. 176-204, ISSN: 2074-8523.

[2] S. M. A. Elrahman and A. Abraham, *A Review of Class Imbalance Problem*, Journal of Network and Innovative Computing, vol. 1, 2013, pp. 332-340, ISSN: 2160-2174.

[3] N. V. Chawla, N. Japkowicz, and A. Kotcz, *Editorial: special issue on learning from imbalanced data sets*, SIGKDD Explorations, vol. 6, no. 1, 2004, pp. 1-6.

[4] H. He and E. A. Garcia, *Learning from imbalanced data*, IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, September 2009, pp. 1263-1284.

[5] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: applications to data mining*, Springer-Verlag Berlin Heidelberg, 2009, doi: 10.1007/978-3-540-73263-1.

[6] S. Raschka, *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*, ArXiv abs/1811.12808, 2018.

[7] D. M. Hawkins, *The Problem of Overfitting*, Journal of Chemical Information and Computer Sciences 44(1):1-12, May 2004, doi: 10.1021/ci0342472.

[8] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Addison Wesley, 2005.

[9] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, C*ost-sensitive learning methods for imbalanced data*, in The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, 2010, pp. 1-8, doi: 10.1109/IJCNN.2010.5596486.

[10] C. Elkan, *The Foundations of Cost-Sensitive Learning*, in Proc. Of the 17th Intl. Joint Conf. on Artificial Intelligence, August 2001, pp. 973-978.

[11] M. Galar, A. Fernandez, E. Barrenechea, and H. Bustince, *A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches*, in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 4, pp. 463-484, 2012, doi: 10.1109/TSMCC.2011.2161285.

[12] W. C. Lin, C. F. Tsai, Y. H. Hu, and J. S. Jhang, *Clustering-based undersampling in class-imbalanced data*, Information Sciences, voll 409–410, October 2017, pp. 17-26, doi: doi.org/10.1016/j.ins.2017.05.008.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *SMOTE: Synthetic Minority Over-sampling Technique*, in Journal of Artificial Intelligence Research 16, 2002, pp. 321-357.

[14] H. He, Y. Bai, E. A. Garcia, and S. Li, *ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning*, 2008 International Joint Conference on Neural Networks (IJCNN 2008), Honk Hong, 2008, pp. 1322-1328, doi: 10.1109/IJCNN.2008.4633969.

[15] A. Sánchez, E. Morales, and J. Gonzalez, *Synthetic Oversampling of Instances Using Clustering*, International Journal of Artificial Intelligence Tools, 22, 2013, doi: 10.1142/S0218213013500085.

[16] P. D. Turney, *Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm*, in Jouirnal of Artificial Intelligence Research 2, 1995, pp. 23-26.

[17] C. X. Ling, Q. Yang, J. Wang, and S. Zhang, *Decision Trees with Minimal Costs*, in Proceedings of the 21st International Conference on Machine Learning (ICML), Banff, Canada, 2004.

[18] P. Domingos, *MetaCost: A General Method for Making Classifiers Cost-Sensitive*, in Proceedings of the fifth international conference on knowledge discovery and data mining, SIGKDD, San Diego, ACM, New York, 1999, pp. 155-164.

[19] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques with Java implementations*, Morgan Kaufmann, San Francisco, 2005.

[20] X. Chai, L. Deng, Q. Yang, and C. X. Ling, *Test-Cost Sensitive Naive Bayes Classification*, in Proceedings of the fourth IEEE International Conference on Data Mining, ICDM, 2004, doi: 10.1109/ICDM.2004.10092.

[21] V. S. Sheng and C. X. Ling, *Thresholding for Making Classifiers Cost-sensitive*, in Proceeedings of the 21st National Conference on Artificial Intelligence, 2006, pp. 476-481.

[22] S. Wang and X. Yao, *Diversity analysis on imbalanced data sets by using ensemble models*, in IEEE Symp. Comput. Intell. Data Mining, pp. 324-331, 2009.

[23] D. Tao, X. Tang, X. Li, X. Wu, *Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval*, IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 7, pp. 1088-1099, Jul. 2006.

[24] C. Li, *Classifying imbalanced data using a bagging ensemble variation (BEV)*, in Proc. 45th An. Southeast Reg. Conf., NY:ACM, pp. 203-208, 2007.

[25] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, *SMOTEBoost: Improving prediction of the minority class in boosting*, Proc. Knowl. Discov. DB, pp. 107-119, 2003.

[26] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, and A. Napolitano, *Rusboost: A hybrid approach to alleviating class imbalance*, IEEE Trans. Syst. Man Cybern. A Syst. Humans, vol. 40, no. 1, pp. 185-197, Jan. 2010.

[27] X. Y. Liu, J. Wu, and Z. H. Zhou, *Exploratory undersampling for class-imbalance learning*, IEEE Trans. Syst. Man Cyber. Appl. Rev, v. 39, n. 2, pp. 539-550, 2009.

[28] T. Maciejewski and J. Stefanowski, *Local Neighbourhood Extension of SMOTE for Mining Imbalanced Data*, in Proceeedings of 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Paris, pp. 104-111, 2011, doi: 10.1109/CIDM.2011.5949434.

[29] B. Santoso, H. Wijayanto, K. A. Notodiputro, and B. Sartono, *Synthetic Over Sampling Methods for Handling Class Imbalanced Problemas: A Review*, in IOP Conf. Series: Earth and Environmental Science 58, 2017, doi: 10.1088/1755-1315/58/1/012031.

[30] S. Hu, Y. Liang, L. Ma, and Y. He, *MSMOTE: Improving classification performance when training data is imbalanced*, in Proc. 2nd Int. Workshop Comput. Sci. Eng., vol. 2, 2009, pp. 13-17.

[31] N. V. Chawla, *C4.5 and Imbalanced Data sets: Investigating the Effect of Sampling Method*, Probabilistic Estimate, and Decision Tree Structure, in ICML Workshop on Learning from Imbalanced Data sets, Washington, DC, 2003.

[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten (2009), *The WEKA Data Mining Software: An Update*, SIGKDD Explorations, Volume 11, Issue 1.

[33] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.

[34] P. Chapman et al., *CRISP-DM 1.0. Data Mining guide*, 2000.

[35] L. Breiman, *Bagging predictors*, Machine Learning, vol. 24, no. 2, pp. 123–140, 1996, doi:10.1007/BF00058655.

[36] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley and Sons, Inc., 2004.