**Spis treści** • *Contents*

# Numerical Methods in Bioelectromagnetics Analysis

*Metody numeryczne w analizach bioelektromagnetycznych*

**Tomasz Długosz** [1]

**Abstract** — The paper is devoted to short characteristic of numerical methods using in electromagnetics investigations. Three main methods were presented: Method of Moments, Finite Element Method and Finite Difference Time Domain method. FEM and FDTD were used in computer simulations to show the interaction between the exposure system and tested object.

**Index Terms** — : electromagnetic fields, bioelectromagnetics, numerical methods, exposure system, TEM cell, accuracy of biomedical studies

**Streszczenie** - artykuł poświęcony jest krótkiej charakterystyce metod numerycznych wykorzystywanych w badaniach bioelektromagnetycznych. Omówiono trzy podstawowe metody: metodę momentów, metodę elementów skończonych i metoda różnic skończonych w dziedzinie czasu. Metody elementów skończonych i różnic skończonych zostały wykorzystane w symulacjach komputerowych na potrzeby wyznaczenia wzajemnego oddziaływania między układem ekspozycyjnym, a badanym obiektem.

**Słowa kluczowe** - pole elektromagnetyczne, bioelektromagnetyzm, metody numeryczne, układy ekspozycyjne, komora TEM, dokładność badań biomedycznych.

## INTRODUCTION

It is well known that the primary tool for quantitative research is hands-on experimentation and measurements. Unfortunately, the tests are not always possible due to high complexity of the studied objects, lack of appropriate sensors or their inaccuracy. This is especially important in the measurement of electromagnetic field (EMF). It is worth mentioning that any physical quantity measured (i.e.: frequency) are performed with $10^{-10\%}$ accuracy, whereas the error in creating a standard EMF equals 5% ÷ 10%. That influences the test tools' accuracy whose error can't exceed the one of creating EMF. Further appears the question of ethics of such tests. Experiments examining EMF's influence on human body are acceptable with person's consent, but still controversial. The same applies to the use of animals for this type of research. Above arguments show that bioelectromagnetic testing is a challenge, and is often impossible to perform. This is where use of mathematical models and computer programs based on numeric methods comes in handy. These tools give us some insight on the expected results. Similar results from different numerical methods can be considered as exemplary and reliable.

## Thermal And Non-Thermal Effects Of EMF

The first effects of the EMF on the human body has been identified in the nineteenth century by Jacques Arsene d'Arsonval, who was engaged in, among others, study of electrical phenomena occurring in the muscles. It was him, who introduced the current diathermic treatment. He was a pioneer in electrotherapy. He observed both, thermal and non-thermal effects of EMF [1][2].

The result of thermal impact on the body is the temperature rise of tissues and body fluids causing pathological changes and physiological responses caused by the tissues increased temperature. The increase in frequency causes an increase in the threshold current density. It also raises the intensity of energy absorption and causes thermal effects, which dominate at frequencies above 10 MHz. The temperature raise depends on many factors, such as: field strength, frequency, and individual characteristics of the person undergoing the exposure.

When it comes to non-thermal effects, the changes in the

1. Faculty of Computer Science, Wroclaw of Information Technology, ul Wejherowska 28, 54-239 Wrocław, Poland, tdlugosz@horyzont.eu

body are caused without raising the temperature of the tissues. These effects occur at low field intensities and at different levels of biological organization. They dominate at low frequencies (up to 100 kHz), where the absorption of electromagnetic energy by the body is weak. The most important non-thermal effects included are [3]-[8]:

- membrane stimulation of excitable tissues, due to the presence of induced,
- transport abnormalities of ions of calcium, potassium and sodium,
- memory disturbances,
- changes in blood smears and EEG records,
- sleepiness,
- fatigue,
- functional disorders of the nervous system.

The question of existence of non-thermal effects is still a subject of dispute among scientists and has so far not been clearly resolved. This leads to increased research in this area, which is shown in numerous publications. In addition, we observe the non-specific effects. They are linked to the fact that some people associate their health problems with increased sensitivity to EMF [9]. Some of the non-specific effects may include:

- headache,
- dizziness,
- sleep disturbances,
- sensation of heat,
- abnormal heart rate,
- shortness of breath.

The non-specific symptoms are reported, but they cannot be confirmed experimentally. According to the Scientific Committee on Toxicity, Ecotoxicity and the Environment, there is the possibility of a hypersensitivity to EMF in a small group of people. It is necessary to confirm this fact, though the report of this type of research cannot be relied on in conducting the limits of EMF exposure in the regulatory laws [5].

Besides the negative effects, electromagnetic field can also have a positive impact on living organisms. Those effects are used for diagnostic purposes and have various applications in medicine. Some of them are [10]-[12]:

- breast cancer detection,
- skin cancer treatment,
- treatment of depression,
- analgesia,
- orthopedics,
- degenerative and rheumatic diseases,
- inflammation of the joints and muscles.

As it can be seen from the above examples, the omnipresence of EMF is not indifferent to living matter, and under certain conditions can cause various biological and health effects [13].

It all points to the necessity of studying subjects related to the EMF's effect on living organisms and defining measures of these effects, as well as field measurements and their accuracy and exposure systems used during the production of the EMF. The best, fast and safe methods for thouse analysis are numerical methods.

## MATERIALS AND METHOD

Computational methods for electromagnetism are dated back to the sixties of the twentieth century. Today's software is based on electromagnetic field theory, and takes into account both physical phenomena, and numerical methods. Some of the most important ones are: Moment Method (MoM), Finite Element Method (FEM) and Finite Difference Time Domain method (FDTD).

### *A. Moment Method*

Moment method was created by Harrington in 1968 [14]. It is mainly used for solving operator-based equations like [15][16]:

$$Lf = g \tag{1}$$

where:
$L$ – linear operator,
$g$ – known excitation function ( $g = g(X)$),
$f$ – sought function, $f = f(Y)$,
$X$, $Y$- coordinates in a multidimensional space,

Sought approximate solution is presented in a form of a sum:

$$f(Y)=\sum_{i=1}^{n} a_i f_i(Y) \tag{2}$$

where:
$f_i(Y)$ – known set of linearly independent functions, which in the case of $n \to \infty$ must provide an exact solution [17],
$a_i$ – unknown coefficients.

Substituting equation (2) into (1) we get:

$$\sum_{i=1}^{n} a_i Lf_i = g \tag{3}$$

In order to determine the coefficients $a_i$, one should choose a set of stable, weighted functions $w_j(X)$, $j = 1,...,n$ and multiply equation (3) by a weighted scalar. This will result in a set of linear equations in the following form:

$$\sum_{i=1}^{n} a_i \langle w_j, Lf_i = w_j, g \quad , j = 1,...,n \tag{4}$$

Obtained solution (4) may be exact or approximate, and the accuracy depends on the choice of functions $f_i$ and $w_j$ [14].

### *B. Finite Element Method*

The history of FEM dates back to the fifties of the twentieth century where it was used for calculations in structural mechanics [18]. It was not until thirty years later that

the method was applied to the EMF's boundary value problems [19].

The basis of the FEM method is to divide the examined area into the elements, depending on the type of the problem [20]:

- one-dimensional - sections,
- two-dimensional - triangles or quadrangles,
- three-dimensional - tetrahedral.

Each element has its nodes which are linked to sought field sizes. In order to share nodes (and their field sizes) between neighboring elements, nodes are usually located on the element's sides or corners. Some types of elements have interpolated nodes located in the middle of the element.

Inhomogeneous Helmholtz equation (5) is a typical wave equation commonly seen in electromagnetism:

$$\nabla^2\phi + k^2\phi = g \tag{5}$$

where:
$\phi$ – sought field size,
$g$ – source function,
$k$ – propagation constant ($k = \omega\sqrt{\varepsilon\mu}$ ).

The above equation is found in three different cases:

- $k = g = 0$ – Laplace equation,
- k = 0 – Poisson equation,
- g = 0 – Helmholtz homogeneous scalar equation.

Minimizing the functional is necessary in order to find a solution to the nonhomogeneous wave equation [19][20]

$$F(\phi) = \frac{1}{2}\iint_{\Omega}\left[\left|\nabla\phi\right|^2 - k^2\phi^2 + 2\phi g\right]d\Omega \tag{6}$$

where:
Ω - considered area.

Both, the potential $\Phi$ and the source function $g$ can be expressed by the triangular element (for 2D analysis). In that case:

$$\phi_e(x,y) = \sum_{i=1}^{3}\alpha_i\phi_{ei} \tag{7}$$

$$g_e(x,y) = \sum_{i=1}^{3}\alpha_i g_{ei} \tag{8}$$

where:
$\phi_{ei}$ , $g_{ei}$ – values of functions $\phi_{ei}$ and $g_{ei}$ in the nodal point $i$ of the element $e$.

Knowing the functional equation for a single element

$$F(\phi_e) = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\phi_{ei}\phi_{ej}\iint\nabla\alpha_i\cdot\nabla\alpha_j d\Omega -$$
$$\frac{k^2}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\phi_{ei}\phi_{ej}\iint\alpha_i\alpha_j d\Omega$$
$$+\sum_{i=1}^{3}\sum_{j=1}^{3}\phi_{ei}g_{ej}\iint\alpha_i\alpha_j d\Omega = \frac{1}{2}[\Phi_e]^t[C^{(e)}][\Phi_e] -$$
$$\frac{k^2}{2}[\Phi_e]^t[T^{(e)}][\Phi_e] + [\Phi_e]^t[T^{(e)}][G_e] \tag{9}$$

we can use them for all N elements in a selected region of solutions:

$$I(\phi) = \sum_{e=1}^{N} I(\phi_e) \tag{10}$$

Finally a solution to this particulair problem can by written as a matrix:

$$I(\Phi) = \frac{1}{2}[\Phi]^t[C][\Phi] -$$
$$\frac{k^2}{2}[\Phi]^t[T][\Phi] + [\Phi]^t[T][G] \tag{11}$$

where:

$$[\Phi] = [\phi_1, \phi_2, \ldots, \phi_N]^t,$$
$$[G] = [g_1, g_2, \ldots, g_N]^t,$$

[C], [T] – global matrices containing corresponding local matrices [$C^{(e)}$] and [$T^{(e)}$].

When using FEM, it is important to choose appropriate element sizes. Elements dividing the selected area must be smaller than the shortest wave that may occur.

*C. Finite Difference Time Domain Method*

Finite difference time domain method was proposed by Kane Yee in 1966. FDTD is based on direct solution of Maxwell's equations. Therefore the implication of boundary conditions is the main problem.

Using the designation of the Yee cells (Fig. 1), Maxwell equations and saving them in a rectangular coordinate system (x, y, z) we get following equations [21]-[23]:

$$\frac{\partial \mathrm{H}_x}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} - \rho H_x\right) \tag{12}$$

$$\frac{\partial \mathrm{H}_y}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial z} - \frac{\partial E_x}{\partial y} - \rho H_y\right) \tag{13}$$

$$\frac{\partial \mathrm{H}_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} - \rho H_x\right) \tag{14}$$

$$\frac{\partial E_x}{\partial t}=\frac{1}{\varepsilon}\left(\frac{\partial H_z}{\partial z}-\frac{\partial H_y}{\partial y}-\sigma E_x\right) \quad (15)$$

$$\frac{\partial E_y}{\partial t}=\frac{1}{\varepsilon}\left(\frac{\partial H_x}{\partial z}-\frac{\partial H_z}{\partial y}-\sigma E_y\right) \quad (16)$$

$$\frac{\partial E_z}{\partial t}=\frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial z}-\frac{\partial H_x}{\partial y}-\sigma E_z\right) \quad (17)$$

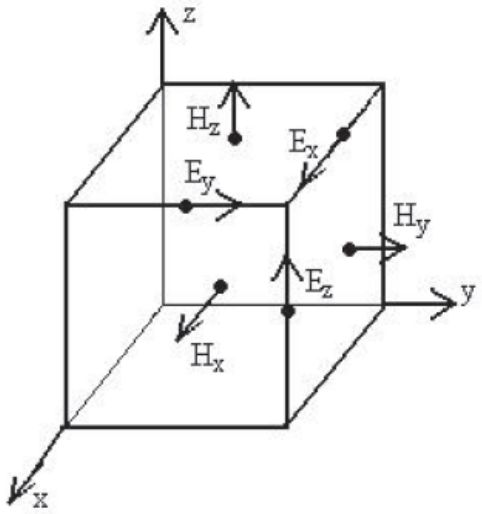



Fig. 1. Yee's cell

Using the notation introduced by Yee and marking the point in space as:

$$(i,j,k)=(i\delta,j\delta,k\delta) \quad (18)$$

and an arbitrary function describing the time and space in the following form

$$F^n(i,j,k)=F(i\delta,j\delta,k\delta,n\delta t) \quad (19)$$

where:

δ=δx=δy=δx – discretization step between consecutive points in space,

δt – time discretization step,

then the individual partial derivatives of function *F* can be written in the form of finite differences:

$$\frac{\partial F^n(i,j,k)}{\partial x}=$$

$$=\frac{F^n\left(i+\frac{1}{2},j,k\right)-F^n\left(i-\frac{1}{2},j,k\right)}{\delta}+0(\delta^2) \quad (20)$$

$$\frac{\partial F^n(i,j,k)}{\partial t}=$$

$$=\frac{F^{n+1/2}(i,j,k)-F^{n-1/2}(i,j,k)}{\delta t}+0(\delta t^2) \quad (21)$$

The determination of the individual components of the electric and magnetic fields is based on presented equations. Assuming that we know the value of such components of the magnetic field in the *n-th* moment of time and the value of the electric component at the time *n-1/2*, the only unknowns that remain in the calculation are the electrical components in the moments of *n* and *n+1/2*. The next step is to find the electrical component based on the values from previous step, and magnetic field values appointed for the half time step earlier (*n-1/2*).

All presented above numerical methods were implemented in many applications that allow to make EMF analysis.

## Numerical Methods In Accuracy Improvement In Exposure System

One of the most important problem in bioelectromagnetics studies is accuracy one. Bioelectromagnetic research is one of the least accurate and difficult to perform. In many cases the tests are performed when the EMF exposure is significantly different from the one to which objects are exposed to in real life. Estimates made by the author show that due to interaction between the tested objects and the exposure system and among objects themselves, errors may exceed even 100% [24]. These phenomena are the reason for significant differences in the results of research done in different research centres. Most bioelectromagnetic studies are dealing with basic principles. This means that the most important factor is the correlation of research conducted in the laboratory with the influence of EMF on the objects in real life, which typically resemble conditions of free space. It all points to the need of reinterpreting the results of biomedical research and detailed analysis of the subject.

Placing any object with conductivity different than zero in the EMF causes certain losses. If the values of the electric field intensity and conduction current density are known, then the power loss that is absorbed by the objects is described by the formula

$$P_{abs}=\int_V EJ\,dV \quad (22)$$

where:

$P_{abs}$ – absorbed power,

$E$ – electric field density vector,

$J$ – current density vector,

$V$ – volume of the object.

Substituting

$$J = \sigma E \tag{23}$$

and making simple transformations, we get the power absorbed by an object placed in the EMF, given by

$$P_{abs} = \sigma \int_V |E|^2 dV \tag{24}$$

A number of models were used for analysis and computer simulation with methods FEM and FDTD of interaction between the object and the exposure system. In this paper results for block model of a man are presented (Fig. 2). This model consists of a cube with 10cm long sides. Its electrical parameters equal ε = 80, σ = 0.84 S/m Dimensions of the conductive plates are 5 x 5 m.

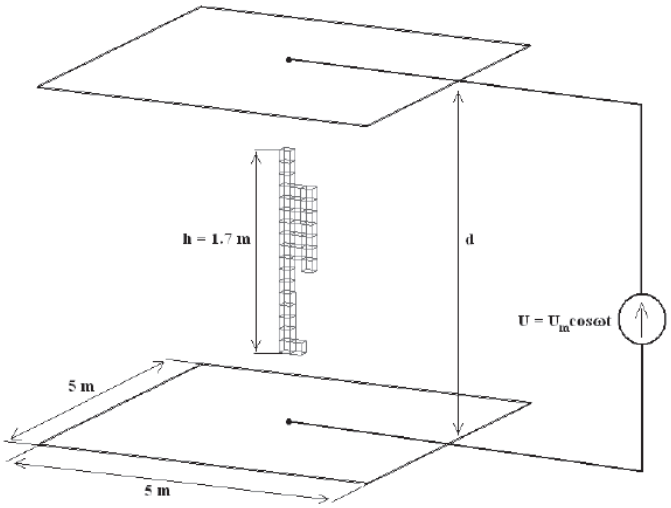



Fig. 2. A block model of a human in the TEM line

Simulations were performed for horizontal and vertical alignment of the model against metal plates. The line kept constant electric field of $E = 1$ *V/m*. After each time the plates were moved away (growth of *d/h* ratio) the voltage applied to the system was increased and the power absorbed by the object was calculated.

## Results And Discussion

It may be noticed that the size of the exposure system has a significant impact on the electric field distribution inside the tested object (Fig. 3). The field distribution shows that the field strength in the facility is greater when the plates are near the line of the test model, than if the plates are further away. Greatest intensity occurs in the hand and legs of the model, and on its outskirts you can see significant disturbances of the electric field.

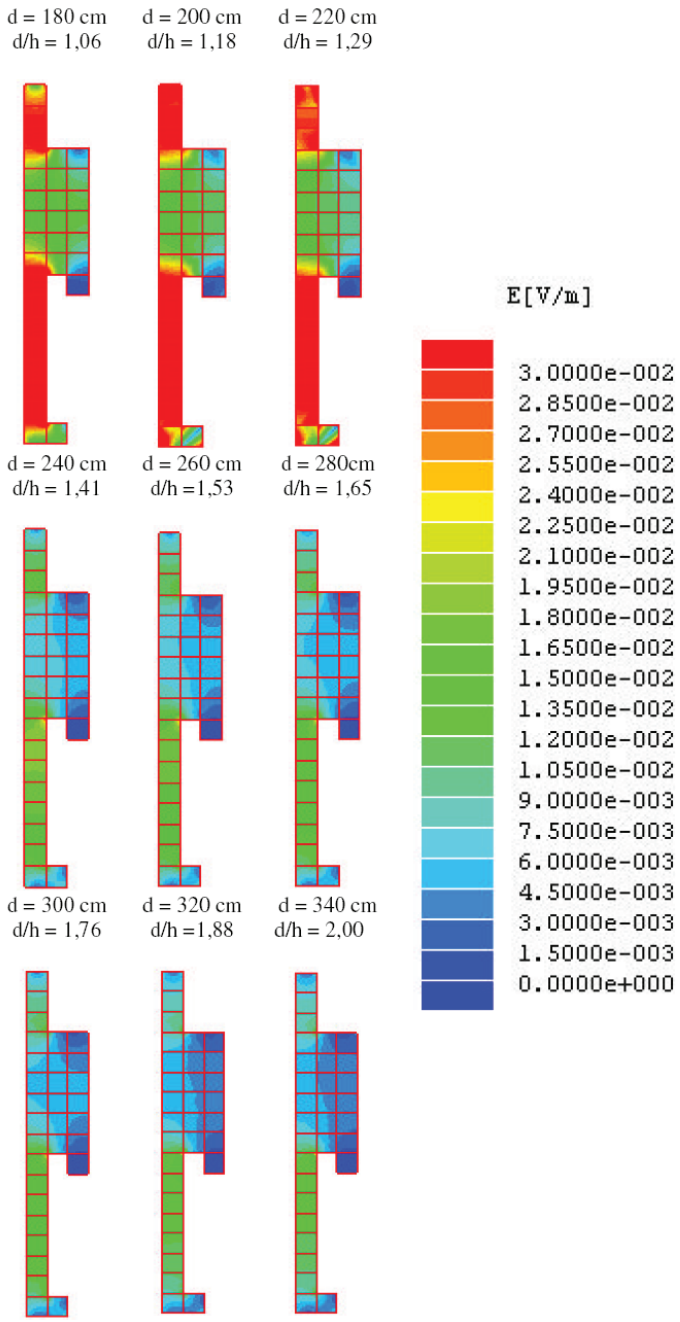



Fig. 3. Distribution of electric field inside a block model of a human placed perpendicular to the walls of the line for various distances between the plates, at a frequency f = 10 MHz

The results of changes in the power absorption as a function of the exposure and system's size are shown in Fig. 4. It is worth mentioning that when the plates of TEM line are close to the object the power absorbed is 30 times higher compared to the conditions of free space. Increase in the *d/h* ratio causes the absorbed power to decrease and approach asymptotically the value of absorbed power in free space, where the presence of metal plates is negligible. This condition is met for $d/h \approx 2$.

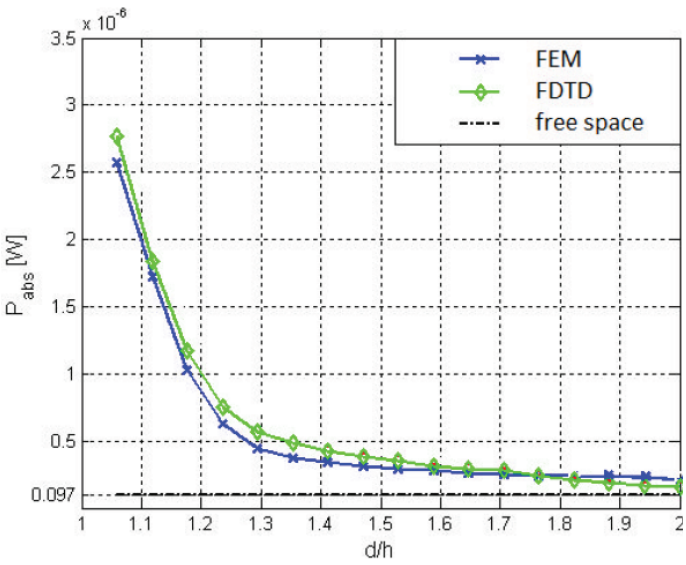



Fig. 4. The results of calculations of absorbed power by the block model of a human placed perpendicular to the plates of the walls

## Conclusion

The most important feature and the biggest advantage of computer simulations using numerical methods is their ability to predict the behavior of the actual object based on its mathematical model. It is much easier and faster to perform computer simulations, rather than perform the measurements in real life conditions. Computer simulations are also extremely useful when the experiments are too dangerous to perform, i.e.: when the researched EMF can cause health issues or death of tested objects. Major drawbacks of computer simulations are restraints of computing resources and long duration of the calculations.

## References

L.A. Geddes, "d'Arsonval, physician and inventor", *IEEE Engineering in Medicine and Biology Magazine*, Vol. 18, No. 4, pp. 118-122, 1999.

L.A. Geddes, "The first experiments on electrical safety, *IEEE Engineering in Medicine and Biology Magazine*, Vol. 25, No. 6, pp. 92-93, 2006.

I. Malikova, L. Janousek, "Non-thermal effects of low-frequency electromagnetic field on biological cells", . Proceedings of ELECTRO; 2014 May 19-20; Rajecke Teplice, Slovak Republic, pp. 509-602.

International Commission on Non-Ionizing Radiation Protection (ICNIRP)," ICNIRP Guidelines: Guidelines for limiting exposure to time-varying electric, magnetic and electromagnetic fields (up to 300 GHz)", *Health Physics*, Vol. 74, No. 4, pp. 494-522, 1998.

H. Hinrikus, "Changes caused by Microwave In Human EEG of individuals", *CD Proceeding of International Conference and COST 281 Workshop on Emerging EM-F-Technologies, Potential Sensitive Groups and Health*; April 20-21, Graz, Austria, 2006.

F.S. Prato,"Non-thermal extremely low frequency magnetic field effects on opioid related behaviors: Snail to humans, mechanisms to therapy", *Bioelectromagnetics*, Vol. 36, No. 5, pp. 333-348, 2015.

E.R. Adair, D.R. Black, "Thermoregulatory reponses to RF energy absorption", *Bioelectromagnetics*, 2003, Vol. 24, No. S6, pp. S17-S38, 2003.

A. Paffi, L. Micaela, F. Apollonio, A. Sheppard, Q. Balzano, "In vitro exposure: Linear and non-linear thermodynamic events in Petri Dishes", *Bioelectromagnetics*, Vol. 36, No. 7, pp. 527-537, 2015.

World Health Organization, "Environmental Health Criterias on electromagnetic Fields", available from: http://www.who.int/peh-emf/research/health_risk_assess/en/index2.html, May 2016.

M.Markov., C.F. Hazlewood, "Electromagnetic Field Dosimetry for Clinical Application", *Proceedings of 5th International Workshop on Biological Effects of Electromagnetic Fields*, 2008 , September 28th – October 2nd, Palermo, Italy.

L.I. Kheifets, "Electric and magnetic field exposure and brain cancer: A review", *Bioelectromagnetics*, Vol. 22, No. S5, pp. S120-S131, 2001.

Scientific Committee on Emerging Newly Identified Health Risks, "Opinion on potential health effects of exposure to electromagnetic fields", *Bioelectromagnetics*, Vol. 36, No. 6, pp. 480-484, 2015.

S. Szmigielski, E. Sobiczewska, „Naturalny rozwój zespołów chorobowych – rola czynników środowiskowych", *Proceedings of XXII autum School*, 20-24 October, Zakopane, Poland (in Polish), 2008.

R.F. Harrington, "Field Computations by Moment Methods", *MacMillan*, New York, 1968.

A.R. Djordjević, T.K. Sarkar, "A Theorem on the Moment Methods", *IEEE Transactions on Antennas and Propagation*, Vol. 35, No. 3, pp. 353 – 355, 1987.

J. Yang, L. Zhou, "Iterative algorithm of wavelet moments methods based on the multi-resolution analysis characteristics of wavelet", *Proceedings of Microwave technology and Computational Electromagnetics (ICMTCE)*, May 22-25; Beijing, China; IEEE 2011.

T.K. Sarkar, A. Djordjević, E. Arvas, "On the choice of expansion and weighting function in the method of moments", *IEEE Transactions on Antennas Propagation*, Vol. 33, No. 9, pp. 988-996, 1985.

M.J. Turner, R.W. Clough, H.C. Martin, L.P. Topp, "Stiffness and deflection analysis of complex structures", *J. Aeronautical Society*, No. 23, pp. 805-823, 1956.

P. Silvester, "A General High-Order Finite-Element Waveguide Analysis Program", *IEEE Transactions on Microwave Theory and Techniques*, Vol. 17, No. 4, pp. 204-210, 1969.

K.H. Huebner, E.A. Thornton, "The Finite Element Method for Engineers", *John Wiley and Sons*, 1982.

A. Taflove, "Computational Electrodynamics: The Finite-Difference Time-Domain Method", *Artech House Antennas and Propagation Library*, 1996.

A. Taflove, M.E. Brodwin, "Numerical Solution of Ste-

ady-State Electromagnetic Scattering Problems Using the Time-Dependet Maxwell's Equation", *IEEE Transaction Microwave Theory and Techniques*, Vol. 23, No. 4, pp. 623 – 630, 1975.

S-Y. Hyun, S-Y. Kim, "3-D thin-Wire FDTD Approach for Resistively Loaded Cylindrical Antennas Fed by Coaxial lines", *IEEE Transactions on Antennas and Propagation*, Vol. 58, No. 12, pp. 4095-4099.

T. Dlugosz, "Uncertainity Analysis of Selected Sources of Errors in Bioelectromagnetic Investigations", *Bio-Medical Materials and Engineering*, Vol. 24, No. 1, pp. 609-617, 2014.

# Aplikacja Maple do kryptograficznej ochrony folderów, utworzonych na pamięciach chmurowych, dyskach twardych i na nośnikach wymiennych

*Maple implementation of cryptographic protection of folders created in cloud storage, hard disks and portable memory devices*

**Czesław Kościelny**[1]

**Treść**: Opisano sposób realizacji aplikacji do szyfrowania i deszyfrowania folderów. Aplikację zrealizowano stosując program Maple 2016. Jest to program wyjątkowo łatwy w obsłudze, ponieważ użytkownik stosuje wyłącznie lewy klawisz myszy, wybierając folder, który będzie przetwarzany oraz jedną z operacji, czyli szyfrowanie lub deszyfrowanie folderu. Dlatego też aplikację można polecić specjalistom średnio biegłym w informatyce, jak filolodzy, ekonomiści, prawnicy, osoby duchowne, itp., którzy w swoich komputerach posiadają foldery z poufnymi dokumentami. Program posiada prosty graficzny interfejs użytkownika i przeznaczony jest głównie do kryptograficznej ochrony plików przechowywanych w folderach tworzonych w pamięciach chmurowych, (`cloud storage`) oraz na dyskach twardych i na nośnikach wymiennych. Aplikacja niezawodnie chroni pliki przed nieupoważnionym dostępem.

**Słowa kluczowe**: szyfr strumieniowy o strukturze bajtowej, szyfrowanie symetryczne plików.

**Abstract**. It has been shown how to implement a Maple worksheet which performs cryptographic protection of folders created in cloud storage, hard disks and portable memory devices by means of byte-oriented stream-cipher using the addition in GF(256).

**Keywords**: Maple byte-oriented stream-cipher, symmetric file encryption.

## 1. Wstęp

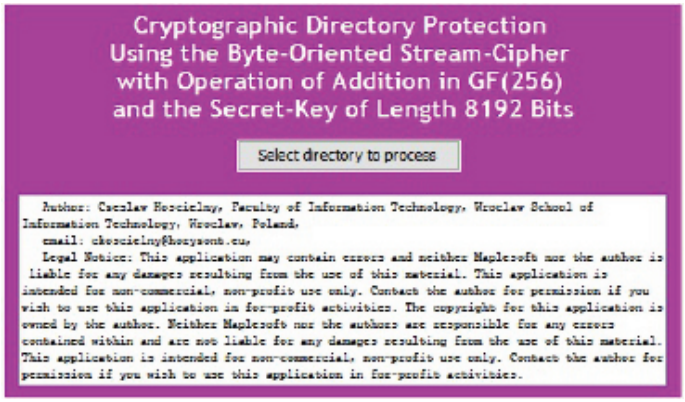
W niniejszym artykule opisano oryginalną aplikację w postaci programu typu `worksheet` o nazwie `a256k8192.mw` utworzonego w środowisku Maple, którego zadaniem jest ochrona kryptograficzna folderów tworzonych na pamięciach chmurowych, dyskach twardych i na nośnikach wymiennych. Jest oczywiste, że przechowywanie niezaszyfrowanych plików w pamięciach chmurowych jest wyjątkowo nierozsądne, dlatego też użytkownicy np. usługi `Microsoft OneDrive` powinni koniecznie stosować opisywany tu program `a256k8192.mw` lub jakiś inny program szyfrujący.
W aplikacji zastosowano narzędzie kryptograficzne w postaci szyfru strumieniowego o strukturze bajtowej, stosującego operację dodawania w 256-elementowym ciele Galois GF(256). Podstawowymi elementami klucza dla tego szyfru jest 1024-elementowa lista o nazwie `K`, zawierająca pseudolosową sekwencję bajtów o wartościach w zakresie 0 .. 255 oraz zmienna `seed` typu `posint`.
W zastosowanym szyfrze strumieniowym długość klucza szyfrującego musi być równa rozmiarowi pliku który jest przetwarzany, zatem sekwencja zawarta w liście `K` jest powtarzana odpowiednią liczbę razy. Dzięki takiej konstrukcji szyfru każdy zaszyfrowany plik ma postać pseudolosowego ciągu bajtów o wartościach od 0 do 255.
Można więc stwierdzić, że wartość tajnego klucza w zastosowanym szyfrze strumieniowym wynosi 8192 bity.
Oczywiście liczbę znaków w liście `K` można dowolnie zmieniać i tworzyć inne aplikacje, mniej lub bardziej odporne na rozszyfrowanie.
Użytkownik łatwo zauważy, że szyfrowana jest nie tylko zawartość pliku jawnego, ale też nazwa tego pliku.
Zaszyfrowana nazwa pliku jawnego nie ma żadnego rozszerzenia i składa się z wybranego zestawu dużych i małych liter alfabetu łacińskiego. W ten sposób ewentualny oponent nie ma żadnych wiadomości na temat struktury plików, zapamiętanych w zaszyfrowanym folderze.
Warto pamiętać, że tajny klucz jest zawarty w aplikacji, która powinna być zapamiętana na pendrajwie, pieczołowicie chronionym i przechowywanym w bezpiecznym miejscu.
Jeśli oponent chciałby podjąc próbę zdeszyfrowania zaszyfrowanego folderu, musiałby znać wartość zmiennej `seed` oraz wartości 1024 elementów listy `K`.
Można obliczyć, że dysponując najszybszym procesorem, wypróbowanie wszystkich wartości `K` oraz `seed` trwa-

1.Wydział Informatyki, Wrocławska Wyższa Szkoła Informatyki Stosowanej, ul. Wejherowska 28, 54-239 Wrocław, ckoscielny@horyzont.eu

łoby dłużej niż wiek ziemi (4 467 000 000 lat).
Zatem prezentowaną aplikację można uważać za bardzo efektywny i skuteczny sposób kryptograficznej ochrony plików przed nieupoważnionym dostępem.

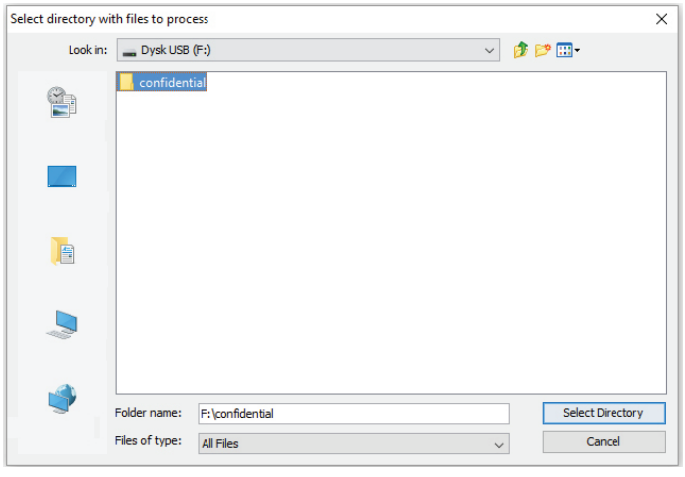
## 2. Graficzny interfejs użytkownika

Graficzny interfejs użytkownika pokazano na Rys. 1. Interfejs składa się z jednego elementu typu `table` w którym unieszczono elementy `Button0`, `TextArea0` oraz niewidoczny po otwarciu programu w sesji Maple element `ComboBox0`. Aby wybrać folder który ma być szyfrowany lub deszyfrowany użytkownik powinien kliknąć na nagłówku elementu `Button0` (`Select directory to process`).



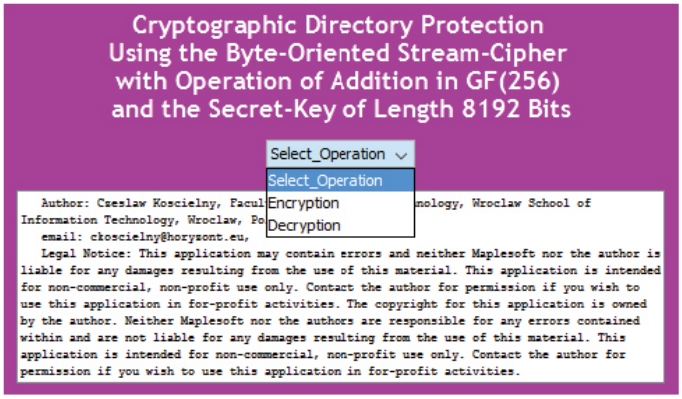

Rys. 1. Graficzny interfejs użytkownika

Po wykonaniu tej czynności pojawi się okno, umożliwiające wybór folderu do zaszyfrowania lub do odszyfrowania.




Rys. 2. Okno do wyboru folderu – wybrano folder `confidential` na pendrawie `F`

Okno pozwala wybrać dowolny folder dostępny w komputerze użytkownika. Po kliknięciu na przycisku `Select Directory` folder zostaje wybrany, element `Button0` staje się niewidoczny, a na interfejsie ukazuje się element `ComboBox0` z napisem `Select_Operation`. Teraz możliwy jest wybór jednej z dwóch opcji: `Encryption` lub `Decryption`. Warto pamiętać, że wybrany folder można szyfrować/deszyfrować dowolną liczbę razy.




Rys. 3. Opcje elementu `ComboBox0`

## 3. Oprogramowanie aplikacji

W obszarze `Edit Click Action` elementu `Button0` umieszczono instrukcje:

```
cd := dirsel();
use DocumentTools in
   SetProperty(ComboBox0,visible,true);
   SetProperty(Button0,visible,false)
end use;
```

Z kolei obszar kodu startowego aplikacji zawiera pięć procedur i siedem instrukcji:

```
#Generowanie tablicy dodawania w GF(256) i listy K
AEDK := proc(sd::posint)
local f, i, k;
global EcDc, K;
      randomize(sd);
      EcDc := Array(0 .. 255,0 .. 255);
      f := GF(2,8);
      for i from 0 to 255 do
        for k from 0 to 255 do
             EcDc[i,k] := f:-output(f:-`+`(f:-
             input(i),f:-input(k)))
        end do
      end do;
      K := [seq(rand(256)(),i = 1 .. 1024)]
end proc;

#Wybór folderu
dirsel := proc()
local d, i, db, dd;
      dd := Maplets:-Elements:-Maplet(Maplets:-
Elements:-FileDialog['DiDia'](('approvecap-
tion') = „Select Directory",fileselectionmode
= directoriesonly,('title') = „Select di-
rectory with files to process",('directory')
= „C:/",('onapprove') = Maplets:-Elements:-
Shutdown(['DiDia']),('oncancel') = Maplets:-
Elements:-Shutdown()));
    d := Maplets[Display](dd)[1];
      db := convert(d,bytes);
      for i to nops(db) do
```

```
        if db[i] = 92 then
          db[i] := 47
        end if
      end do;
      db := convert(db,bytes)
end proc;

#Szyfrowanie nazwy pliku
fne := proc(fn::string, sd::posint)
local a, l, ib2ob;
      randomize(sd);
      a := combinat:-randcomb([seq(i+65,i = 0..
      25), seq(i+97,i = 0 .. 25)],26);
      ib2ob := table([seq(i-1 = a[i],i = 1 ..
      26)]);
      l := convert(fn,bytes);
      l := convert(l,base,128,26);
      l := [seq(ib2ob[l[i]],i = 1 .. nops(l))];
      convert(l,bytes)
end proc;

#Deszyfrowanie zaszyfrowanej nazwy
pliku
fnd := proc(fn::string, sd::posint)
local a, l;
global ob2ib;
      randomize(sd);
      a := combinat:-randcomb([seq(i+65,i = 0..
      25), seq(i+97,i = 0 .. 25)],26);
      ob2ib := table([seq(a[i] = i-1,i = 1 ..
      26)]);
      l := convert(fn,bytes);
      l := [seq(ob2ib[l[i]],i = 1 .. nops(l))];
      convert(convert(l,base,26,128),bytes)
end proc;

#Usuwanie niepotrzebnego pliku
fr := proc(fn::string)
      if FileTools:-Exists(fn) then
      FileTools:-Remove(fn)
      end if
end proc;

#Ustalanie wartości elementów `TextArea0`,
`Button0`, `ComboBox0` use DocumentTools in
  SetProperty(TextArea0,value," Author: Czeslaw
Koscielny, Faculty of Information Technology,
Wroclaw School of Information Technology, Wroc-
law, Poland, \n  email: ckoscielny@horyzont.eu,
\n  Legal Notice: This application may contain
errors and neither Maplesoft nor the author is
liable for any damages resulting from the use of
this material. This application is intended for
non-commercial, non-profit use only. Contact the
author for permission if you wish to use this
application in for-profit activities. The copyri-
ght for this application is owned by the author.
Neither Maplesoft nor the authors are responsi-
ble for any errors contained within and are not
liable for any damages resulting from the use of
this material. This application is intended for
non-commercial, non-profit use only. Contact the
author for permission if you wish to use this ap-
plication in for-profit activities.");
  SetProperty(Button0,visible,true);
  SetProperty(Button0,caption,"Select directory
to process");
      SetProperty(ComboBox0,value,Select_Opera-
tion);
  SetProperty(ComboBox0,visible,false)
end use;

#Ustalanie wartości zmiennej `seed` i wywołanie
procedury `AEDK` z wybraną przez #użytkownika
wartością parametru `seed`
seed := 1234567654321;
AEDK(seed);
```

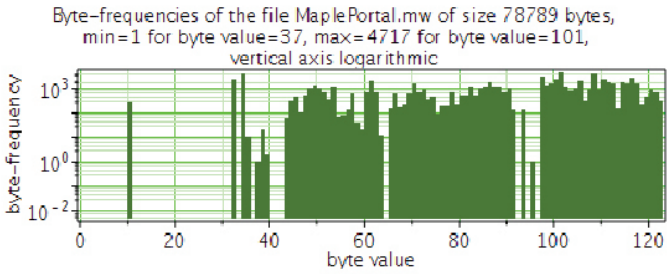
Pozostałą część kodu aplikacji umieszczono w obszarze `ComboBox0 Select Action`.

```
ed := DocumentTools:-GetProperty(ComboBox0,va-
lue);
fl := FileTools:-ListDirectory(cd);
nfl := nops(fl);
ss := {}:
for i to nfl do
   ss := ss union {FileTools:-
Size(cat(cd,"/",fl[i]))}
end do;
fsm := ss[nops(ss)];
K := [seq(K[1+`mod`(s-1,1024)],s=1 .. fsm)]:
t := time[real]();
tfs := 0;
if ed = „Encryption" then
  DocumentTools:-Do(%TextArea0 = „Encrypting .
. ."",refresh = true);
  for i to nfl do
    m := readbytes(cat(cd,"/",fl[i]),infinity);
    fs := nops(m);
    c := [seq(EcDc[m[n],K[n]],n = 1 .. fs)];
    efn := cat(cd,"/",fne(fl[i],seed));
    writebytes(efn,c);
    fclose(efn);
    tfs := tfs+fs;
  end do;
  t := time[real]()-t;
  mes := cat(convert(nfl,string)," files of total
size „,convert(tfs,string)," Bytes have been
encrypted in „,convert(t,string)," seconds. En-
cryption rate: „,convert(round(tfs/t),string),"
Bps. The input files have been removed. To use
the program once again click on the icon `Re-
start Maple server`. To finish select `File -
Exit` or press Alt and F4.");
  DocumentTools:-Do(%TextArea0 = mes,refresh =
true)
elif ed = „Decryption" then
  DocumentTools:-Do(%TextArea0 = „Decrypting .
. ."",refresh = true);
  for i to nfl do
    m := readbytes(cat(cd,"/",fl[i]),infinity);
    fs := nops(m);
    c := [seq(EcDc[m[n],K[n]],n = 1 .. fs)];
    efn := cat(cd,"/",fnd(fl[i],seed));
    writebytes(efn,c);
    fclose(efn);
    tfs := tfs+fs;
  end do;
  efl := FileTools:-ListDirectory(cd);
  t := time[real]()-t;
  mes := cat(convert(nfl,string)," files of to-
tal size „,convert(tfs,string)," have been de-
crypted in „,convert(t,string)," seconds. De-
cryption rate: „,convert(round(tfs/t),string),"
Bps. The input files have been removed. To use
the program once again click on the icon `Re-
start Maple server`. To finish select `File -
Exit` or press Alt and F4.");
  DocumentTools:-Do(%TextArea0 = mes,refresh =
```

```
true)
end if;
+for i to nfl do
    fr(cat(cd,"/",fl[i]))
end do:
DocumentTools:-SetProperty(ComboBox0,visible,f
alse);
```

## 4. Przykład

Zakładając, że w folderze pokazanym na Rys. 2 znajduje się plik `MaplePortal.mw` dostępny w instalacji programu Maple pod adresem `C:/Program Files/Maple 2016/examples`



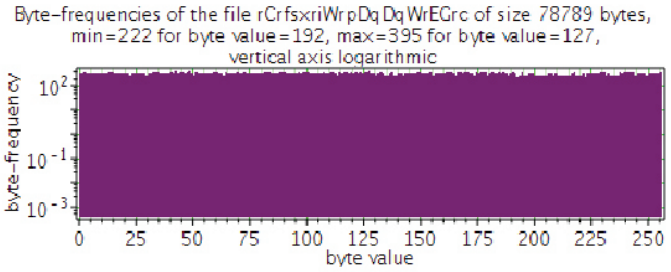

Rys. 4. Częstotliwości występowania bajtów w pliku niezaszyfrowanym o strukturze bajtowej pokazanej na Rys. 4 (strukturę bajtową pliku można obliczyć za pomocą aplikacji [8]).

W pliku `MaplePortal.mw` występują 83 wartości bajtowe podane niżej w postaci par

(wartość_bajtu liczba_wystąpień)

```
Byte-frequencies in the file `MaplePortal.mw`
(10 289),(32 2522),(34 3912),(35 11),(37 1),(38
21),(39 2),(43 63),(44 304),(45 494),(46
110),(47 535),(48 1100),(49 1444),(50 1071),(51
712),(52 342),(53 1163),(54 71),(55 86),(56
136),(57 596),(58 39),(59 21),(60 704),(61
1964),(62 704),(63 13),(65 150),(66 625),(67
174),(68 252),(69 623),(70 1588),(71 700),(72
911),(73 339),(74 483),(75 99),(76 218),(77
216),(78 684),(79 254),(80 561),(81 549),(82
1187),(83 573),(84 1027),(85 1258),(86
1906),(87 1295),(88 1267),(89 694),(90
1098),(91 142),(93 142),(95 1),(97 3203),(98
1316),(99 1616),(100 2438),(101 4717),(102
1224),(103 802),(104 1040),(105 2148),(106
315),(107 869),(108 4127),(109 983),(110
2352),(111 1815),(112 1636),(113 152),(114
1905),(115 1665),(116 2791),(117 1516),(118
255),(119 458),(120 989),(121 677),(122 334),
min. freq. = 1 for byte value = 37,
max. freq. = 4717 for byte value = 101.
```

Po zaszyfrowaniu plik zmienia nazwę na `rCrfsxriWrpDqDqWrEGrc`.
Rozmiar zaszyfrowanego pliku jest taki sam jak rozmiar pliku niezaszyfrowanego, ale w tym pliku występuje teraz 256 wartości bajtowych z zakresu 0 .. 255. Częstotliwość występowania wartości bajtowych jest dosyć równomierna, co ilustruje Rys. 5




Rys. 5. Częstotliwości występowania bajtów w pliku zaszyfrowanym

oraz zestaw par (wartość_bajtu liczba wystąpień).

```
Byte-frequencies in the file`rCrfsxriWrpDqDqWr-
EGrc`
(0 300),(1 321),(2 344),(3 314),(4 314),(5
294),(6 313),(7 286),(8 291),(9 298),(10
306),(11 300),(12 306),(13 336),(14 334),(15
292),(16 302),(17 298),(18 294),(19 289),(20
297),(21 310),(22 266),(23 300),(24 310),(25
304),(26 289),(27 316),(28 297),(29 267),(30
291),(31 309),(32 343),(33 355),(34 335),(35
294),(36 289),(37 308),(38 284),(39 318),(40
325),(41 317),(42 370),(43 309),(44 369),(45
335),(46 287),(47 324),(48 377),(49 302),(50
273),(51 359),(52 308),(53 283),(54 285),(55
313),(56 300),(57 311),(58 316),(59 353),(60
367),(61 322),(62 292),(63 246),(64 350),(65
293),(66 288),(67 309),(68 319),(69 342),(70
298),(71 300),(72 327),(73 312),(74 316),(75
302),(76 289),(77 304),(78 298),(79 294),(80
296),(81 339),(82 328),(83 328),(84 317),(85
307),(86 295),(87 312),(88 332),(89 344),(90
328),(91 303),(92 286),(93 303),(94 304),(95
312),(96 332),(97 297),(98 349),(99 346),(100
360),(101 300),(102 322),(103 314),(104
311),(105 299),(106 283),(107 337),(108
304),(109 315),(110 333),(111 363),(112
316),(113 262),(114 350),(115 284),(116
328),(117 355),(118 299),(119 354),(120
296),(121 364),(122 316),(123 327),(124
308),(125 328),(126 359),(127 395),(128
315),(129 293),(130 268),(131 296),(132
324),(133 315),(134 258),(135 274),(136
299),(137 325),(138 276),(139 268),(140
304),(141 298),(142 271),(143 309),(144
305),(145 303),(146 290),(147 305),(148
341),(149 339),(150 312),(151 312),(152
318),(153 301),(154 333),(155 268),(156
293),(157 318),(158 318),(159 318),(160
322),(161 339),(162 357),(163 334),(164
306),(165 278),(166 353),(167 347),(168
282),(169 325),(170 327),(171 309),(172
316),(173 267),(174 351),(175 312),(176
333),(177 335),(178 342),(179 279),(180
314),(181 343),(182 300),(183 319),(184
353),(185 316),(186 357),(187 314),(188
268),(189 314),(190 365),(191 319),(192
222),(193 272),(194 333),(195 252),(196
278),(197 301),(198 292),(199 302),(200
266),(201 275),(202 270),(203 279),(204
277),(205 276),(206 297),(207 285),(208
280),(209 284),(210 260),(211 271),(212
266),(213 260),(214 262),(215 269),(216
299),(217 346),(218 274),(219 250),(220
278),(221 287),(222 233),(223 294),(224
312),(225 284),(226 318),(227 291),(228
290),(229 332),(230 332),(231 279),(232
305),(233 258),(234 272),(235 277),(236
```

```
281),(237 332),(238 307),(239 335),(240
326),(241 329),(242 348),(243 333),(244
267),(245 311),(246 284),(247 283),(248
284),(249 272),(250 312),(251 303),(252
306),(253 286),(254 305),(255 344),
min. freq. = 222 for byte value = 192,
max. freq. = 395 for byte value = 127.
```

Inaczej mówiąc, zaszyfrowany plik ma postać pseudolosowej sekwencji bajtów z zakresu 0 .. 255. Podobną strukturę ma każdy plik zaszyfrowany.

## 5. Podsumowanie i wnioski

W artykule przedstawiono sposób tworzenia bardzo szybkiej, efektywnej aplikacji do kryptograficznej ochrony folderów przed nieupoważnionym dostępem. Aplikacja działa poprawnie pod warunkiem, że wybrany do przetwarzania folder nie posiada podfolderów, tzn. że w wybranym folderze zapisane są wyłącznie pliki i że w folderze zapamiętany jest co najmniej jeden plik.
Zaawansowany użytkownik Maple bez problemu utworzy plik `**a256k8192.mw**`, ale początkujący użytkownicy Maple mogą mieć pewne problemy z wykonaniem tego zadania. W takich przypadkach wystarczy pobrać plik `**a256k8192.mw**`, zapamiętany pod adresem http://wdawnictwo.horyzont.eu/podstrony/publikacje.html.

## 6. Literatura

[1] Czesław Kościelny, Mirosław Kurkowski, Marian Srebrny – Modern Cryptography Primer, Springer Verlag, 2013
[2] Czesław Kościelny - Program Maple do szyfrowania i deszyfrowania plików, zapisanych na dyskach i na nośnikach wymiennych, Biuletyn Naukowy Wrocławskiej Wyższej Szkoły Informatyki Stosowanej, Informatyka 2013.
[3] https://en.wikipedia.org/wiki/Stream_cipher.
[4] https://en.wikipedia.org/wiki/Cloud_storage
[5] How to Encrypt Files and Folders in Windows 10, http://windowsreport.com/encrypt-files-folders-windows-10/
[6] Czesław Kościelny, Cryptographic Protection of Definite PC Directory Using the AES Algorithm, http://fr.maplesoft.com/applications/view.aspx?SID=7086, 2009
[7] Dengguo Feng, Xiutao Feng, Wentao Zhang, Chuankun Wu, 1995, A Byte-Oriented Stream Cipher, https://www.researchgate.net/publication/220776034_Loiss_A_Byte-Oriented_Stream_Cipher
[8] Czesław Kościelny, Byte Frequency Analyzer, 2015, http://fr.maplesoft.com/applications/view.aspx?SID=153920

# Dekompozycja ciągu uczącego dla rozproszonej bazy danych

*Teaching sequence decomposition FOR DISTRIBUTED DATABASE*

**Swietłana Lebiediewa**[1]

**Abstract** - The problem of decomposition of a teaching sequence (TS) for distributed database is formulated. Three types of TS decomposition for a distributed database are formulated. Theorems concerning memory occupancy by the TS after decomposition are proved. The results of a calculation experiment are presented, that illustrate the memory occupancy depending upon the decomposition type, the redundancy of features in the tree and upon the type of the dispersion.

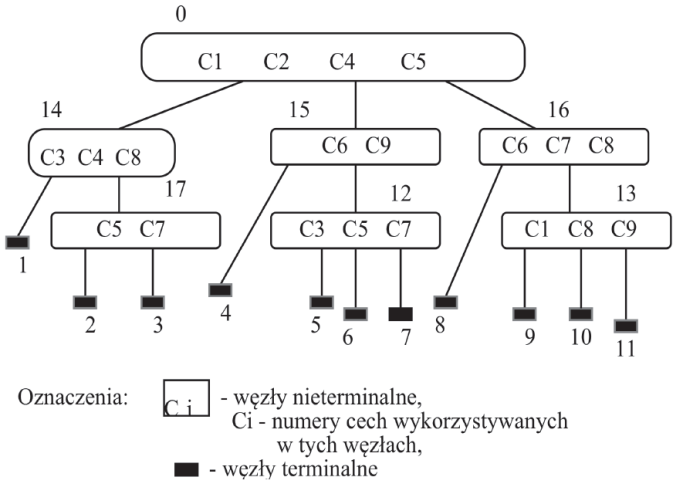
**Streszczenie** - Sformułowano problem dekompozycji ciągu uczącego (CU) dla rozproszonej bazy danych. Zdefiniowano trzy rodzaje dekompozycji. Przedstawiono twierdzenia dotyczące zajętości pamięci przez zdekomponowany ciąg uczący. Zaprezentowano wyniki badań symulacyjnych ilustrujących zajętość pamięci w zależności od typu rozproszenia i rodzaju dekompozycji.

**Słowa kluczowe**: rozproszona baza danych, rozpoznawanie wieloetapowe, ciąg uczący, dekompozycja

## 1. Sformułowanie problemu

Rozpoznawanie wielostopniowe polega na *dekompozycji* problemu decyzyjnego, czyli zastępowaniu jednorazowego rozpoznawania sekwencją tzw. rozpoznawań lokalnych przeprowadzanych w poszczególnych węzłach zgodne z zadaną konstrukcją drzewa decyzyjnego (DD), rys. 1. Żeby zaklasyfikować rozpoznawany obiekt do jednej z klas, należy przejść ścieżką w DD startując od korzenia. W każdym napotykanym węźle wewnętrznym (**decyzyjnym**) należy podjąć decyzję o dalszej drodze w grafie, aż osiągnięty zostanie węzeł terminalny. Klasa z nim związana reprezentuje końcowy wynik rozpoznawania wielostopniowego. Tak więc w trakcie klasyfikacji wielostopniowej rozpoznawany obiekt podlega sekwencji decyzji na ścieżce od korzenia do węzła terminalnego.

Algorytmy rozpoznawania z uczeniem korzystają z ciągu uczącego (CU) [1,2,3,4]. CU jest ciągiem poprawnie zaklasyfikowanych obiektów, elementami CU są pary postaci ($x_k$, $k$), gdzie $x$ jest wektorem wartości cech obiektu, $k$ – numerem klasy. Drzewo decyzyjne (DD) przedstawiono na rys.1, a odpowiadający mu CU – na rys. 2.




Rys. 1. Drzewo decyzyjne NR1a
*Figure 1. Decision tree No. 1a*

W CU przedstawionym na rys. 2 w miejscu wartości niektórych cech występuje symbol *, symbol ten oznacza, że dana cecha jest cechą nieistotną dla rozpoznawanego obiektu i nie jest brana pod uwagę przez algorytm rozpoznawania. Tak więc w procesie rozpoznawania wielostopniowego na różnych etapach rozpoznawania wykorzystuje się tylko niektóre elementy wektora cech obiektów, a pewne cechy wcale nie występują na poszczególnych ścieżkach.

1.Wrocławska Wyższa Szkoła Informatyki Stosowanej „Horyzont” (‘Horyzont’ Wroclaw School of Information Technology) swietlana@lebiediewa.com

| C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | NR KLASY |
|---|---|---|---|---|---|---|---|---|---|
| 16 | 3,44 | 3 | 6,55 | 40,0 | 48 | 116 | * | 14,9 | 7 |
| 15 | 3,45 | 2 | 8,45 | 40,0 | * | * | 120 | * | 1 |
| 14 | 2,4 | 1 | 8,33 | 38,1 | * | 110 | * | * | 2 |
| 18 | 3,0 | * | 7,5 | 35,0 | 40 | * | * | 15,0 | 4 |
| 15 | 2,7 | * | 6,0 | 20,0 | 35 | 110 | 112 | * | 8 |

Rys.2. Fragment ciągu uczącego dla DD NR1a
Figure 2. A fragment of the teaching sequence for DT No. 1a

W celu oszczędności pamięci proponuje się dekompozycję CU. Z algorytmów rozpoznawania przedstawionych w [2,3] widać, że czas pracy AR w każdym z węzłów składa się z czasu potrzebnego na utworzenie segmentu danych oraz czasu pracy algorytmów rozpoznawania. Odpowiednia postać CU może skrócić czas potrzebny na utworzenie segmentu danych. Celem dekompozycji jest rozbicie CU na takie podciągi, które zarówno minimalizują zajętość pamięci przez CU jak i zmniejszają czas potrzebny rozpoznawania obiektów [5,6]. W pewnych przypadkach wydaje się uzasadnione rozbicie procesu rozpoznawania na kilka lokalizacji, w których znajdują się lokalne bazy danych (LBD), które połączone siecią komunikacyjną stanowią rozproszoną bazę danych (RBD). Model konceptualny scentralizowanej bazy danych (SBD) przedstawiono w [5, 7]. Model konceptualny BD zawiera informacje o rozpoznawanym obiekcie a także informację o CU SEQUENCE i relacje opisujące strukturę DD KLASY, WĘZŁY i CECHY. W relacji SEQUENCE znajduje się informacja o CU. Relacja KLASY opisuje zależność pomiędzy numerem klasy a numerem węzła będącego bezpośrednim poprzednikiem tej klasy. dla każdej klasy wskazany jest jej poprzednik. Relacja WĘZŁY zawiera informacje o numerach każdego węzła, o bezpośrednich następnikach i poprzednikach węzła. W relacji CECHY zawarta jest informacja, w których węzłach wykorzystywane są poszczególne cechy [5, 6]. Model konceptualny RBD różni się od modelu konceptualnego SBD tym tylko, że w opisie relacji WĘZŁY występuje dodatkowy atrybut LOKALIZACJA, wskazujący na lokalizację węzła nieterminalnego [7]. Dekompozycję CU dla SBD przedstawiono w [6]. Przedstawimy dekompozycję CU dla RBD.

## 2. Dekompozycja ciągu uczącego dla rozproszonej bazy danych

Dla rozproszonej bazy danych (RBD) możemy wyróżnić trzy rodzaje dekompozycji ciągu uczącego: dekompozycję *naturalną*, zmodyfikowaną dekompozycję 1. rodzaju i dekompozycję 2. rodzaju. Ze względu na wysoki koszt transmisji danych wydaje się zasadnym przyjąć założenie, iż CU jest dekomponowany w taki sposób, ażeby algorytm rozpoznawania nie musiał korzystać z informacji zawartej w CU znajdującym się w innej lokalizacji. Można przyjąć, że w komputerze głównym pozostawia się cały CU zdekomponowany lub nie, ze względu na to, że w węźle będącym korzeniem wykorzystywane są cechy występujące we wszystkich elementach CU, można też założyć całkowite rozproszenie CU. W dalszym ciągu zakładamy całkowite rozproszenie CU (jeżeli założyć zachowanie CU w głównej lokalizacji, zajętość pamięci będzie nie mniejsza niż przy całkowitym rozproszeniu). Aby umożliwić ponowne otrzymanie niezdekomponowanego CU, do każdego wiersza CU zdekomponowanego dodajemy identyfikator.

Dekompozycja naturalna polega na tym, że z CU usuwa się wiersze, które w kolumnie NR KLASY nie zawierają numerów klas nieosiągalnych z węzłów nieterminalnych lokalnej bazy danych (LBD), następnie usuwa się kolumny zawierające wartości cech, nie wykorzystywanych w danej lokalizacji. Metoda dekompozycji 1. rodzaju dla RBD jest tak zmodyfikowana, że każdy podciąg (lub jego fragment) odpowiadający węzłowi pośredniemu znajdującemu się poza główną lokalizacją, zawiera tylko wartości cech wykorzystywanych w tej lokalizacji. Podciąg ciągu uczącego budujemy dla każdego węzła nieterminalnego, który jest bądź bezpośrednim poprzednikiem węzła nieterminalnego (klasy), bądź jest bezpośrednim poprzednikiem węzła nieterminalnego znajdującego się w innej lokalizacji. W [6] przedstawiono metodę dekompozycji 1. rodzaju dla SBD i metodę dekompozycji 2. rodzaju dla SBD.

Algorytmy dekompozycji korzystają z informacji o CU i strukturze DD zawartej w relacjach SEQUENCE, KLASY, WĘZŁY, CECHY modelu konceptualnego BD.

ALGORYTM 1. (algorytm naturalnej dekompozycji CU dla RBD)

Dane: model konceptualny RBD - relacje SEQUENCE, WĘZŁY, CECHY.

Wyznaczyć: CU zawierający wartości cech, wykorzystywanych w danej lokalizacji do rozpoznawania klas osiągalnych z węzłów nieterminalnych należących do tej lokalizacji.

KROK 0. Do relacji przechowującej CU dodaj kolumnę zawierającą identyfikatory elementów CU.

KROK 1. Dla każdej LBD z relacji WĘZŁY pobierz numery węzłów nieterminalnych znajdujących się w LBD.

KROK 2. Z relacji NASTĘPNIKI wybierz numery klas osiągalnych z węzłów nieterminalnych należących do LBD.

KROK 3. Dla każdej LBD z relacji przechowującej CU wybierz wiersze, dla których

NR KLASY = "klasa osiągalna z węzła nieterminalnego należącego do tej lokalizacji";

KROK 4. Z otrzymanej relacji usuń atrybuty zawierające wartości cech, które nie są wykorzystywane w tej lokalizacji. **STOP**.

Oznaczmy przez *ZPDN* zajętość pamięci przez CU otrzymany w wyniku dekompozycji naturalnej, przez $i$ – numer LBD, przez $LC_i$ – liczbę cech wykorzystywanych w lokalizacji $i$, przez $k_i$ – liczbę klas osiągalnych z węzłów nieterminalnych należących do lokalizacji $i$, przez $L$ – liczbę LBD. Zajętość pamięci przez CU dla całej RBD wyraża wzór:

$$ZPDN = \sum_{i=1}^{L} (LC_i + 2) * K_i \qquad (1)$$

ALGORYTM 2. (zmodyfikowany algorytm dekompozycji pierwszego rodzaju CU dla RBD)
Dane: model konceptualny RBD - relacje SEQUENCE, WĘZŁY, CECHY.

Wyznaczyć: dla każdego węzła nieterminalnego będącego bądź poprzednikiem klas, bądź poprzednikiem węzłów nieterminalnych znajdujących się w innej lokalizacji wyznaczyć taki podciąg ciągu uczącego, który zawiera wartości cech wykorzystywanych na ścieżce od "korzenia" poddrzewa do tego węzła.

KROK 0. Do relacji przechowującej CU dodaj kolumnę zawierającą identyfikatory elementów CU.

KROK 1. Z relacji KLASY i WĘZŁY pobierz numery węzłów należących do LBD, które są bądź bezpośrednimi poprzednikami węzłów terminalnych bądź bezpośrednimi poprzednikami węzłów nieterminalnych należących do innej lokalizacji.

KROK 2. Dla każdego takiego węzła utwórz relację zawierającą podciąg uczący dla wszystkich klas bezpośrednio osiągalnych z tego węzła i zawierający wartości cech wykorzystywanych w tej lokalizacji (na ścieżce od "korzenia" poddrzewa do tego węzła).

KROK 3. Jeżeli węzeł jest bezpośrednim poprzednikiem węzła nieterminalnego znajdującego się w innej lokalizacji, utwórz podciąg CU zawierający wartości cech wykorzystywanych w tej lokalizacji do rozpoznawania klas będących następnikami tego węzła, otrzymany podciąg dodaj do podciągu otrzymanego w kroku drugim.

**STOP**.

Niech $L$, jak wyżej, oznacza liczbę LBD, a $i$ – numer lokalizacji. Oznaczmy przez $J_i$ liczbę węzłów nieterminalnych w lokalizacji $i$, przez $LC_j$– liczbę cech wykorzystywanych w danej lokalizacji przez algorytmy rozpoznawania na ścieżce do węzła $j$, przez $K_j$ – łączną liczbę klas bądź będących bezpośrednimi następnikami węzła $j$, bądź osiągalnych z węzła – bezpośredniego następnika $j$ należącego w innej lokalizacji. Zajętość pamięci przez CU otrzymane w wyniku dekompozycji pierwszego rodzaju (RZPD1) dla RBD wyraża wzór:

$$RZPD1 = \sum_{i=1}^{L} \sum_{j=1}^{J_I} \left(LC_j + 2\right) * k_j \qquad (2)$$

Algorytm dekompozycji drugiego rodzaju dla RBD nie różni się od algorytmu dekompozycji drugiego rodzaju dla SBD:

ALGORYTM 3. (algorytm dekompozycji 2. rodzaju)
Dane: model konceptualny RBD – relacje SEQUENCE, WĘZŁY, CECHY.
Wyznaczyć: dla każdego węzła nieterminalnego utworzyć taki podciąg ciągu uczącego, który zawiera wartości tylko tych cech, które są wykorzystywane w tym węźle.

KROK 1. Do relacji SEQUENCE przechowującej CU dodaj kolumnę zawierającą identyfi katory elementów CU.

KROK 2. Z relacji WĘZŁY pobierz numery wszystkich węzłów nieterminalnych.

KROK 3. Dla każdego węzła nieterminalnego utwórz podciąg uczący (z CU wybierz wiersze, dla których NR KLASY = "klasa osiągalna z danego węzła", z otrzymanej relacji usuń atrybuty zawierające wartości cech nie wykorzystywanych w tym węźle). **STOP**.

Niech $J$ oznacza liczbę wszystkich węzłów nieterminalnych w bazie danych, $j$ – kolejny numer węzła, $LC_j$ – liczbę cech wykorzystywanych w węźle $j$, $K_j$ – liczbę klas osiągalnych z węzła $j$. Wzór na zajętość pamięci przy dekompozycji drugiego rodzaju dla RBD (*RZPD2*) ma postać:

$$RZPD2 = \sum_{j=1}^{J} (LC_j + 1) * K_j \qquad (3)$$

**Wniosek** Dla zmodyfikowanej dekompozycji 1. rodzaju dla RBD redundancja cech występujących na ścieżce nie ma wpływu na zajętość pamięci, jeżeli występuje w węzłach znajdujących się w jednej lokalizacji, natomiast zwiększa zajętość pamięci, jeżeli występuje w węzłach leżących w różnych lokalizacjach.

Z Algorytmu 2 wynika

**Twierdzenie 1**. Zajętość pamięci przy dekompozycji drugiego rodzaju nie zależy od sposobu rozproszenia.

Mają miejsce następujące lematy:

**Lemat 1**. Przy całkowitym rozproszeniu (każdy z węzłów nieterminalnych znajduje się w innej LBD) ma miejsce następujący wzór:

$$ZPDN = RZPD1 = RZPD2$$

**Lemat 2**. Jeżeli w LBD żadna para węzłów nieterminalnych nie znajduje się w relacji poprzednik–następnik, ma miejsce równość:

$$RZPD1 = RZPD2,$$

w przeciwnym przypadku

$$RZPD1 < RZPD2$$

Z lematów 1. i 2. wynika

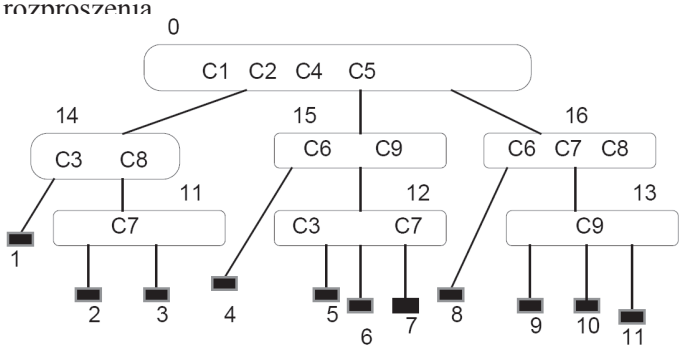
**Twierdzenie 2**. Dla RBD ma miejsce zależność:

$$RZPD1 \leq RZPD2$$

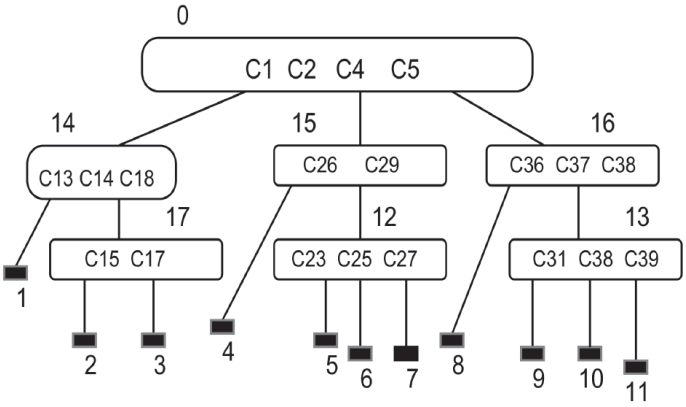
## 3.Eksperyment obliczeniowy

Przedstawimy wyniki eksperymentu obliczeniowego ilustrującego zajętość pamięci CU otrzymanych w wyniku zmodyfikowanej dekompozycji pierwszego i drugiego oraz

dekompozycji naturalnej dla rozproszonej bazy danych. Rozpatrujemy zajętość pamięci dla drzew 1a (Rys. 1), 1b (Rys 3.), i 1c (Rys 4.) w zależności od rodzaju dekompozycji i typu rozproszenia.



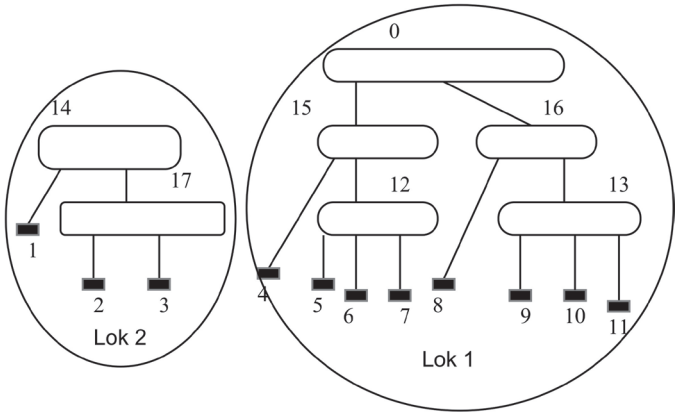

Rys. 3 Drzewo decyzyjne 1b



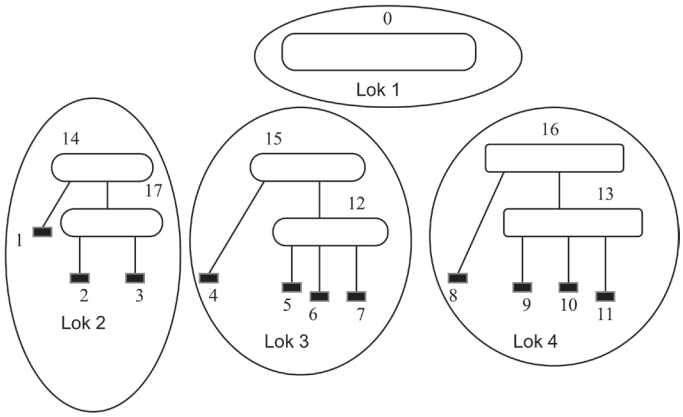

Rys. 4. Drzewo decyzyjne 1c
*Figure 4. Decision tree No. 1c*

Struktura drzew 1a, 1b i 1c jest identyczna, drzewo 1b różni się od drzewa 1a tym, że na poszczególnych ścieżkach nie ma redundancji cech, a w drzewie 1c w każdym z węzłów występują inne cechy.

Rozpatrzmy dwa rodzaje rozproszenia dla drzew 1a-1c: rozproszenie typu *A* i rozproszenie typu *B*, rys.5. W przypadku rozproszenia typu *A* rozproszona baza danych składa się z dwóch LBD. Do pierwszej LBD należą węzły 0, 16, 13, 15 i 12, do drugiej – węzły 14 i 17. W przypadku rozproszenia typu *B* rozproszona baza danych składa się z czterech LBD, do pierwszej LBD należy węzeł numer 0 (korzeń drzewa), do drugiej – węzły 14 i 17; do trzeciej – węzły 15 i 12; do czwartej – węzły 16 i 13. Rysunki 6 i 7 ilustrują zajętość pamięci przez CU dla drzew 1a-1c odpowiednio dla rozproszenia typu *A* i rozproszenia typu *B*. Jak widać różnice w zajętości pamięci przez CU otrzymywane w wyniku dekompozycji naturalnej, dekompozycji pierwszego rodzaju i dekompozycji drugiego rodzaju zmniejszają się w miarę wzrostu rozproszenia aż do zaniku różnic (por. Lemat 1).

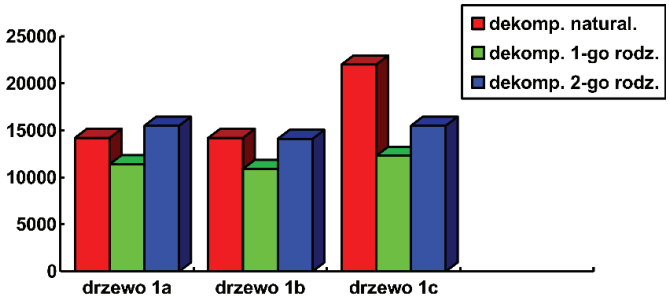



Rozproszenie typu A

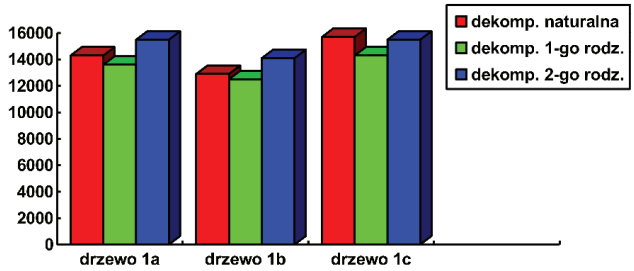



Rozproszenie typu B

Rys. 5. Rozproszenie typu A i B dla drzew 1a – 1c.
*Figure 5. The dispersion of type A and B for the trees 1a - 1c.*




Rys. 6. Wykres zajętości pamięci przez CU otrzymane w rezultacie dekompozycji pierwszego i drugiego rodzaju oraz dekompozycji naturalnej dla drzew 1a-1c, rozproszenie typu A
*Figure 6. Chart memory use by the CU obtained as a result of decomposition of the first and second kind, and for the decomposition of natural trees 1a-1c, the dispersion of type A*




Rys. 7. Wykres zajętości pamięci przez CU otrzymane w rezultacie dekompozycji pierwszego i drugiego rodzaju oraz dekompozycji naturalnej dla drzew 1a-1c, rozproszenie typu B
*Figure 7. Chart memory use by the CU obtained as a result of decomposition of the first and second kind, and for the decomposition of natural trees 1a-1c, the dispersion of type B*

Najlepsze rezultaty daje dekompozycja 1-go rodzaju dla rozproszenia typu A. Dla CU otrzymanych w wyniku dekompozycji 1-go rodzaju wzrost zajętości pamięci następuje wraz ze wzrostem rozproszenia i wzrostem redundancji cech. Zajętość pamięci przez CU otrzymane w wyniku dekompozycji 2-go rodzaju wzrasta wraz ze wzrostem redundancji. Rodzaj rozproszenia nie ma wpływu na zajętość pamięci przez CU otrzymane w wyniku dekompozycji 2-go rodzaju. Dekompozycja naturalna daje najlepsze rezultaty w przypadku braku redundancji i rozproszenia typu C. Najgorsze rezultaty daje dekompozycja naturalna w przypadku rozproszenia typu B (większa zajętość pamięci niż w przypadku CU nie zdekomponowanego).
Dla każdej dekompozycji korzystny wpływ na zajętość CU ma rozpoznanie części klas na wcześniejszych etapach.

## Wnioski:

Dla RBD zmodyfikowana dekompozycja 1. rodzaju daje wyraźnie lepsze wyniki, gdy liczba etapów jest duża i w poszczególnych lokalizacjach znajdują się węzły należące do długich ścieżek. Im większe rozproszenie, tym mniejsza różnica między rodzajami rozproszenia. Przy całkowitym rozproszeniu każda z dekompozycji daje ten sam rezultat. Dekompozycja 2. rodzaju nie zawsze daje oszczędność pamięci. Dekompozycja 2. rodzaju na ogół potrzebuje więcej pamięci niż dekompozycja 1. rodzaju, a w niektórych przypadkach – także więcej niż dekompozycja naturalna. Wraz z wzrostem rozproszenia (wzrostem liczby lokalizacji) maleje różnica w zajętości pamięci przez CU otrzymywane w wyniku dekompozycji naturalnej, dekompozycji pierwszego i drugiego rodzaju. Przy dużym rozproszeniu najbardziej korzystna wydaje się być dekompozycja 2-go rodzaju, ponieważ CU otrzymane w wyniku dekompozycji 2. rodzaju są identyczne z fragmentami CU wykorzystywanych przez algorytmy rozpoznawania w węźle, dlatego w przypadku stosowania dekompozycji 2. rodzaju nie ma potrzeby tworzenia specjalnej relacji *Fragment CU w węzle* przy tworzeniu modelu zewnętrznego, a różnica w zajętości pamięci przez CU otrzymane w wyniku dekompozycji 1-go i 2-go rodzajów jest nieznaczna lub żadna w przypadku całkowitego rozproszenia.

## References

[1] Bubnicki, Z. ‘Knowledge-Based Approach as a Generalization of Pattern Recognition Problems and Methods’. Systems Science Vol. 19, No 2 (1993), pp. 5–21
[2] Flasiński, M. Wstęp do sztucznej inteligencji. Warsaw: PWN, 2011
[3] Kurzyński, M. Algorytmy rozpoznawania wieloetapowego oraz ich zastosowania medyczne i techniczne. Wrocław: Wyd. PWr., 1987
[4] Józefczyk, J. ‘Rozpoznawanie i zastosowania biomedyczne’ [in:] Problemy automatyki i informatyki, Wrocław: Wyd. Ossolineum, 1998, pp. 45–58
[5] Lebiediewa, S, ‘System informatyczny dla wieloetapowego rozpoznawania obiektów’. Biuletyn Naukowy WWIS. Informatyka, 2014
[6] Lebiediewa, S, ‘Training sequence decomposition’. Biuletyn Naukowy WWIS. Informatyka, 2015
[7] Lebiediewa, S. Metodologia projektowania problemowo zorientowanych baz danych do systemów wielostopniowego podejmowania decyzji. Wrocław: Wyd. Pwr., 1998

# Wielkoskalowe i zautomatyzowane testowanie przypuszczenia Beal'a

*Bigscale and automatized testing of Beal's Conjecture*

**Łukasz Świerczewski**[1]

**Abstrakt :** Praca prezentuje aspekt adaptacji oraz wykorzystywania algorytmów zaprezentowanych w publikacji [1] na platformie do obliczeń rozproszonych BOINC [2]. Dodatkowo wykonano testy skalowalności przyśpieszenia oprogramowania na takich platformach jak Intel Xeon Phi 5110P [3] oraz platformie wykorzystującej Versatile SMP Foundation Advanced Platform firmy ScaleMP (rozwiązanie klasy vSMP [4][5]). Dzięki długotrwałym obliczeniom udało się znaleźć 47 rozwiązań przystających prawidłowo modulo 264. Żadne z uzyskanych rozwiązań nie jest jednak prawidłowe w przestrzeni całego zbioru liczb naturalnych, a co za tym idzie nie odnaleziono poprawnego kontrprzykładu dla przypuszczenia Beal'a.

**Słowa kluczowe**: *Przypuszczenie Beal'a, BOINC, obliczenia równoległe*

**Abstract** : This paper presents adaptation aspect and use of algorithms presented in publication [1] on distributed computing platform BOINC. What is more, there were made some test of software acceleration scalability on such platforms like Intel Xeon Phi 5110P and platform that uses Versatile SMP Foundation Advanced Platform made by ScaleMP. Thanks to long-lasting computation 47 solutions correctly congruent modulo $2^{64}$ were found. None of the solutions obtained is not correct in the space around the set of natural numbers and what's connected to that, any correct counterexample for Beal's Conjecture was not found.

**Keywords**: *Beal's conjecture, BOINC, parallel computing*

## Wprowadzenie

Przypuszczenie Beal'a jest nierozwiązanym do dnia dzisiejszego problemem matematycznym z gałęzi teorii liczb i mówi ono, że jeśli:

$$x^m + y^n = z^r$$

gdzie *x, y, z, m, n* oraz *r* są dodatnimi liczbami całkowitymi, *a m, n, r* > 2, to *x, y,* z mają wspólny dzielnik będący liczbą pierwszą.
Pomimo wielu usilnych prób (zarówno analitycznych jak i obliczeniowych) nie udało się tego twierdzenia obalić – znaleźć kontrprzykładu dla liczb z wartościami *x, y, z*, które są parami względnie pierwsze.
Aktualnie za rozwiązanie problemu jest przewidziana nagroda $1 000 000.

## Cel Pracy

Celem pracy jest komputerowa weryfikacja poprawności przypuszczenia Beal'a w zadanych przedziałach. Szczegółowo tezę pracy można postawić następująco:
Przypuszczenie Beal'a jest prawidłowe dla $0 < \boldsymbol{i} < 5\,000$[2] oraz $9\,999 < \boldsymbol{i} < 10\,601$, gdzie zmienna ***i*** definiuje zweryfikowany przedział w jakim mogą znajdować się wartości podstaw ***x, y*** i ***z*** jako:

$$x, y, z \in [\, i \cdot 2000 ; (i + 1) \cdot 2000 \,)$$

Wartości wykładników zawsze są zdefiniowane następująco:

$$m, n, r \in [\, 3 ; 1000 \,]$$

1 Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży, Instytut Automatyki i Robotyki

2.Przy i < 10 000 nie zastosowano dwukrotnej weryfikacji danych (zakresy przeliczono tylko raz). W przypadku gdy obliczenia z powodów technicznych zakończyły się błędem (błąd sprzętowy węzła) mogło dojść z pewnym prawdopodobieństwem do pominięcia możliwych rozwiązań, pomimo że teoretycznie po błędzie zadanie powinno zostać jeszcze raz, na nowo załadowane do systemu kolejkowania. Na klastrze zdefiniowany był przez administratorów ICM UW limit 7 dni na wykonanie pojedynczego zadania – sporadyczna część zadań nie zmieściła się w tych ograniczeniach czasowych, chociaż były także i zadania liczące w granicach dwóch dni. Bez podwójnej weryfikacji nie można mieć całkowitej pewności czy nie istnieje w tych zakresach kontrprzykład dla przypuszczenia Beal'a.

| i | Ograniczenie dolne przedziału dla podstaw (włącznie) | Ograniczenie górne przedziału dla podstaw (bez) | Ograniczenie dolne przedziału dla wykładników (włącznie) | Ograniczenie górne przedziału dla wykładników (włącznie) |
|---|---|---|---|---|
| 0 | 0 | 2 000 | 3 | 1 000 |
| 1 | 2 000 | 4 000 | | |
| 2 | 4 000 | 6 000 | | |
| 3 | 6 000 | 8 000 | | |
| … (włącznie) | | | | |
| 5 000 | 10 000 000 | 10 002 000 | 3 | 1 000 |
| … (bez) | | | | |
| 10 000 | 20 000 000 | 20 002 000 | 3 | 1 000 |
| … (włącznie) | | | | |
| 10 600 | 21 200 000 | 21 202 000 | 3 | 1 000 |

Tab. 1. Tabela przedstawiająca zestawienie zweryfikowanych przedziałów.
*Tab. 1. Table showing comparison of verified divisions.*

Pierwszych 5 000 przedziałów zbadano z wykorzystaniem zasobów superkomputerowych ICM UW[3] (zajęło to w przybliżeniu 600 000 godzin obliczeniowych[4]). Ostatnich 601 przedziałów ($9\,999 < i < 10\,601$) sprawdzono wykorzystując zarówno platformę BOINC, jak i podłączone do niej zasoby superkomputerowe[5] za pomocą specjalnie skonfigurowanego środowiska. Daje to w przybliżeniu kolejnych 21 636 godzin obliczeniowych.

## Obliczenia rozproszone i platforma BOINC (Berkeley Open Infrastructure for Network Computing)

Obliczenia rozproszone umożliwiają współdzielenie zasobów, które są często rozproszone nawet pod względem geograficznym. O tego typu obliczeniach będziemy mówić jednak nawet wtedy, gdy zasoby nie będą rozproszone pod względem geograficznym, a gdy posiadają architekturę heterogeniczną. Wśród Internautów dużą popularnością cieszą się projekty rozproszone, które umożliwiają bezpłatne udostępnienie mocy obliczeniowych domowych komputerów na potrzeby rozwiązywania bardzo złożonych problemów naukowych. Jedną z takich platform jest BOINC.

### Obliczenia rozproszone

Obliczenia rozproszone są pewnym podzbiorem obliczeń równoległych. Nie uruchomimy więc efektywnie wszystkich algorytmów równoległych w systemie rozproszonym. Jak się jednak okazuje takie rozwiązanie umożliwia nam dobrą analizę dużej ilości problemów. Systemy rozproszone bazują zazwyczaj na architekturze klient – serwer, której zobrazowanie przedstawiono na Ryc. 1.

Ryc. 1. Architektura klient – serwer. W centrum umieszczony serwer, do którego są podłączeni klienci. Źródło: Opracowanie własne.
*Fig. 1. Client-Server architecture. Server placed in center and clients connected to it. Source: Own data.*

Architektura klient – serwer, a co za tym idzie także obliczenia rozproszone mają swoje wady. Jedną z nich jest to, że przesyłanie informacji możliwe jest głównie jedynie na liniach klient – serwer oraz w drugą stronę (serwer – klient). Nie ma możliwości samodzielnej łączności pomiędzy klientami, co czasem może bardzo ograniczać programistę.
Do zalet systemów rozproszonych należy jednak to, że umożliwiają one na dostęp do relatywnie dużych zasobów obliczeniowych (w porównaniu do klastrów komputerowych i komputerów z pamięcią wspólną).

### Platforma BOINC

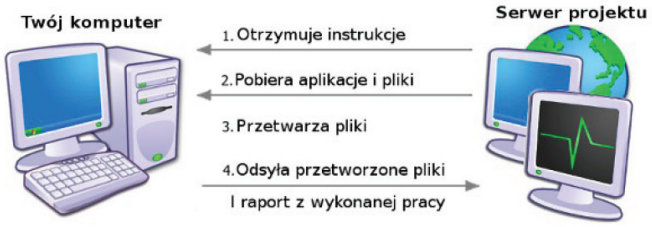
Początkowo platforma BOINC była wykorzystywana jedynie przez projekt poszukiwania cywilizacji pozaziemskich SETI@Home. Z czasem jednak zaczęło z niej korzystać wiele ośrodków naukowych na całym świecie. W Tab. 2 zaprezentowano zestawienie przykładowych projektów BOINC. Istniało oraz istnieje także wiele polskich projektów BOINC (m.in. zaprezentowany w pracy [6] [7]).

3.Do weryfikacji wykorzystano superkomputer HP BladeSystem/Actina – Hydra.
4. Wykorzystano część superkomputera Hydra zawierającą 120 węzłów złożonych z dwóch procesorów klasy Intel Xeon X5660 oraz 24 GB pamięci RAM. Obliczenia trwały dwa miesiące. W tym okresie jednak nie były wykorzystywane wszystkie zasoby danej części superkomputera (korzystały z niego także inni użytkownicy).
5. W tym etapie obliczeń wzięło udział maksymalnie jedynie 9 węzłów z procesorami Intel Xeon X5660.

| Nazwa | Kategoria | Dziedzina | Właściciel |
|---|---|---|---|
| Asteroids@Home [8] [9] | Nauki fizyczne | Astrofizyka | Charles University in Prague |
| ATLAS@Home [10] | Nauki fizyczne | Fizyka | CERN (European Organization for Nuclear Research) |
| Climateprediction.net [11] | Nauki o Ziemi | Klimatologia | Oxford University |
| Cosmology@Home | Nauki fizyczne | Astronomia | University of Illinois at Urbana-Champaign |
| Einstein@Home [12] [13] | Nauki fizyczne | Astrofizyka | Univ. of Wisconsin - Milwaukee, Max Planck Institute |
| FightNeglectedDiseases@Home | Biologia i medycyna | Poszukiwanie leków przeciw malarii | University College Dublin |
| GPUGrid.net | Biologia i medycyna | Modelowanie molekularne protein | Barcelona Biomedical Research Park (PRBB) |
| Malariacontrol.net [14] | Biologia i medycyna | Epidemiologia | The Swiss Tropical Institute |
| MindModeling@Home [15] | Nauki kognitywne i sztuczna inteligencja | Nauki kognitywne | University of Dayton and Wright State University |

Tab. 2. Przykładowe projekty działające na platformie BOINC. Źródło: Opracowanie własne w oparciu o stronę Uniwersytetu Kalifornijskiego w Berkeley.
*Tab. 2. Example projects working on BOINC platform. Source: Own elaboration based on UC Berkeley.*

Jak zaprezentowano w Tab. 2 możliwości BOINC wykorzystuje wiele wiodących instytucji (od CERN po Oxford). Idea tego systemu jest bardzo podobna do innych platform rozproszonych. Schemat działania każdego projektu BOINC zaprezentowano na Ryc. 2.




Ryc. 2. Przepływ danych/instrukcji pomiędzy serwerem BOINC, a klientem. Źródło: Strona zespołu BOINC@Poland.
*Fig. 2. Flow of data/instructions between BOINC server and client. Source: BOINC@Poland website.*

Pierwszym etapem w komunikacji jest połączenie się klienta z tak zwanym schedulerem projektu. To on na podstawie informacji o hoście (rodzaj systemu operacyjnego, typ procesora, ilość pamięci RAM itp.) może wysłać do niego aplikację/aplikacje liczące oraz dane wejściowe. Po pobraniu niezbędnych plików klient przechodzi do obliczeń. W momencie gdy skończy on wykonywać operacje dla danego zadania odsyła przetworzone pliki do serwera. Razem z wynikami wygenerowanymi przez aplikację wysyłany jest także m.in. plik *stderr.txt*, w którym znajduje się zapis wyjścia diagnostycznego aplikacji liczącej.
Należy także dodać, że aplikacje, które wykorzystują projekty BOINC można podzielić na te które używają intensywnie CPU i takie co nie zajmują czasu CPU praktycznie wcale. Do projektów *Non-CPU-intensive* należy m.in. WUProp@Home, który ma na celu analizę wykorzystania zasobów komputerów klientów przez inne projekty BOINC.

## Analiza wyników zrównoleglenia na platformie złożonej dodatkowo z akceleratora Intel Xeon Phi 5110P

Dość niedawno firma Intel wprowadziła na rynek nowe akceleratory Xeon Phi mające spełniać oczekiwania użytkowników wymagających bardzo dużych mocy obliczeniowych. Tego typu dwa akceleratory (5110P) zostały zainstalowane w klastrze Moss dostępnym w Poznańskim Centrum Superkomuterowo-Sieciowym. Do sprzętu tej klasy miałem dostęp w ramach narodowej inicjatywy PL-Grid. Specyfikację techniczną wybranych akceleratorów z rodziny Intel Xeon Phi przedstawiono w Tab. 3.

| Typ akceleratora: | 3120A | 3120P | 5110P | 5120D | 7120X | 7120P |
|---|---|---|---|---|---|---|
| TDP (WAT): | 300 | 300 | 225 | 245 | 300 | 300 |
| Ilość rdzeni: | 57 | 57 | 60 | 60 | 61 | 61 |
| Częstotliwość zegara (GHz): | 1,100 | 1,100 | 1,053 | 1,053 | 1,238 | 1,238 |
| Wydajność teoretyczna (GFLOPS): | 1003 | 1003 | 1011 | 1011 | 1208 | 1208 |
| Wydajność pamięci (GT/s): | 5,0 | 5,0 | 5,0 | 5,5 | 5,5 | 5,5 |
| Przepustowość pamięci: | 240 | 240 | 320 | 352 | 352 | 352 |
| Rozmiar pamięci (GB): | 6 | 6 | 8 | 8 | 16 | 16 |
| Rozmiar pamięci cache (MB): | 28,5 | 28,5 | 30,0 | 30,0 | 30,5 | 30,5 |
| Tryb Turbo: | NIE | NIE | NIE | NIE | TAK | TAK |
| Częstotliwość trybu Turbo (GHz): | - | - | - | - | 1,333 | 1,333 |

Tab. 3. Tabela przedstawiająca specyfikację techniczną wybranych akceleratorów z rodziny Intel Xeon Phi. Źródło: *Opracowanie własne na podstawie danych technicznych firmy Intel*.
*Tab. 3. Table showing technical specifications of selected accelerators from Intel Xeon Phi family. Source: Own elaboration based on Intel website.*

Na klastrze Moss dostępne są dwa akceleratory Intel Xeon Phi. Na potrzeby tej pracy przetestowano jednak możliwości zrównoleglenia aplikacji jedynie na jednym z nich (mic0). Istnieje oczywiście możliwość wykorzystania

dwóch z nich jednak należałoby w takim przypadku skorzystać ze środowiska MPI. Uzyskane czasy realizacji oraz przyśpieszenie dla trybu natywnego [16] przedstawiono w Tab. 4. Analogiczną tabelą dla trybu offload [16] (środowisko OpenMP) jest Tab. 5.

| Ilość wątków | Czas realizacji | Przyśpieszenie |
|---|---|---|
| 1 | 3729 | - |
| 2 | 2269 | 1,643 |
| 3 | 1530 | 2,436 |
| 4 | 1101 | 3,386 |
| 6 | 672 | 5,543 |
| 8 | 558 | 6,675 |
| 10 | 404 | 9,214 |
| 12 | 369 | 10,094 |
| 16 | 238 | 15,621 |
| 20 | 195 | 19,078 |
| 24 | 166 | 22,374 |
| 32 | 126 | 29,465 |
| 48 | 84 | 43,973 |
| 50 | 82 | 45,212 |
| 60 | 74 | 50,277 |

Tab. 4. Czasy realizacji oraz uzyskane przyśpieszenia rozwiązania zrównoleglonego w środowisku OpenMP (na akceleratorze Intel Xeon Phi 5110P w trybie natywnym) wykonywanego z parametrami max_base = 500, max_power = 500. Czasy mierzone w sekundach. Źródło: Opracowanie własne.

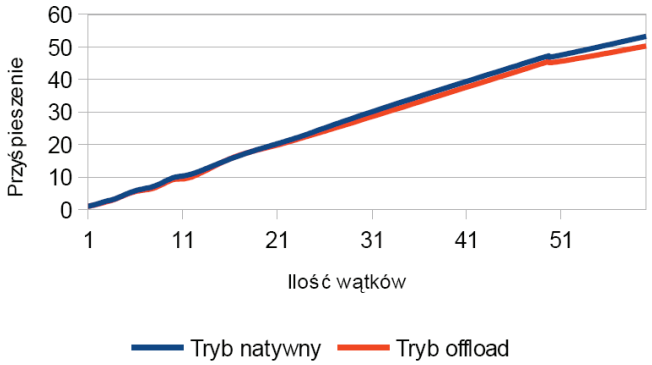
*Tab. 4. Time of realization and obtained accelerations of paralleled solution in OpenMP environment (on Intel Xeon Phi 5110P accelerator in native mode) performed with parameters max_base = 500, max_power = 500. Times measured in seconds. Source: Own data.*

| Ilość wątków | Czas realizacji | Przyśpieszenie |
|---|---|---|
| 1 | 3729 | - |
| 2 | 2269 | 1,643 |
| 3 | 1530 | 2,436 |
| 4 | 1101 | 3,386 |
| 6 | 672 | 5,543 |
| 8 | 558 | 6,675 |
| 10 | 404 | 9,214 |
| 12 | 369 | 10,094 |
| 16 | 238 | 15,621 |
| 20 | 195 | 19,078 |
| 24 | 166 | 22,374 |
| 32 | 126 | 29,465 |
| 48 | 84 | 43,973 |
| 50 | 82 | 45,212 |
| 60 | 74 | 50,277 |

Tab. 5. Czasy realizacji oraz uzyskane przyśpieszenia rozwiązania zrównoleglonego w środowisku OpenMP (na akceleratorze Intel Xeon Phi 5110P w trybie *offload*) wykonywanego z parametrami max_base = 500, max_power = 500. Czasy mierzone w sekundach. *Źródło:* Opracowanie własne.

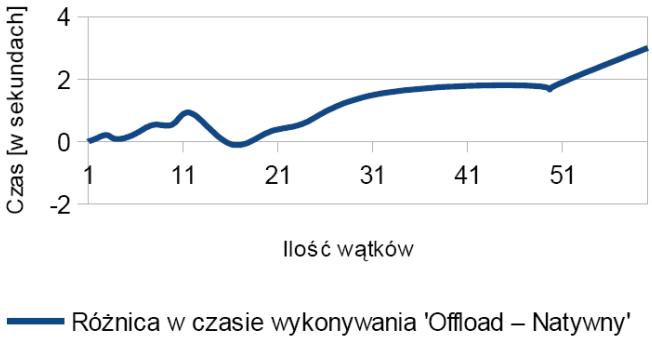
*Tab. 5. Time of realization and obtained accelerations of paralleled solution in OpenMP environment (on Intel Xeon Phi 5110P accelerator in offload mode) performed with parameters max_base = 500, max_power = 500. Times measured in seconds. Source: Own data.*

Jak widać w powyższych tabelach minimalnie lepsze rezultaty (tj. wyższe przyśpieszenia) uzyskano zgodnie z oczekiwaniami w trybie natywnym. Na Ryc. 3 oraz Ryc. 4 można zobaczyć graficzne zobrazowanie na wykresie tych wyników.




Ryc. 3. Porównanie przyśpieszeń uzyskanych na Intel Xeon Phi w trybie offload oraz w trybie natywnym. Źródło: Opracowanie własne.

*Fig. 3. Comparison of accelerations obtained on Intel Xeon Phi in offload mode and in native mode. Source: Own data.*




Ryc. 4. Przedstawienie graficzne różnicy w czasie wykonywania pomiędzy trybem offload, a natywnym. Źródło: Opracowanie własne.

*Fig. 4. Graphical presentation of difference in performance time between offload mode and native mode. Source: Own data.*

Na Ryc. 4 dobrze widać, że praktycznie zawsze (nie licząc jednego przypadku dla 16 wątków) realizacja natywna jest szybsza od tej w OpenMP (*offload*). Jeżeli więc mamy bezproblemową możliwość skorzystania z tego trybu to warto jego wykorzystać, chociaż różnica z *offload* nie jest wielka. Należy także zauważyć, że różnica pomiędzy skrajnymi punktami tego wykresu jest bardzo mała (wynosi niecałe 3,5 sekundy). Dlatego też ponowne wykonanie pomiarów może dać dość zasadniczo inny przebieg.

### Analiza wyników zrównoleglenia na platformie wykorzystującej Versatile SMP Foundation Advanced Platform firmy ScaleMP

Przetestowano możliwość zrównoleglenia oprogramowania z wykorzystaniem standardu OpenMP oraz platformy wykorzystującej Versatile SMP Foundation Advanced Platform firmy ScaleMP. Platforma ta pozwala agregować dziesiątki zwyczajnych serwerów w jedną dużą maszynę SMP w oparciu o szybką sieć InfniBand. Rozwiązanie tworzy komputer z współdzieloną pamięcią (dziesiątki TB pamięci operacyjnej) i setkami rdzeni obliczeniowych.

Specyfikacja dostępnego sprzętu:

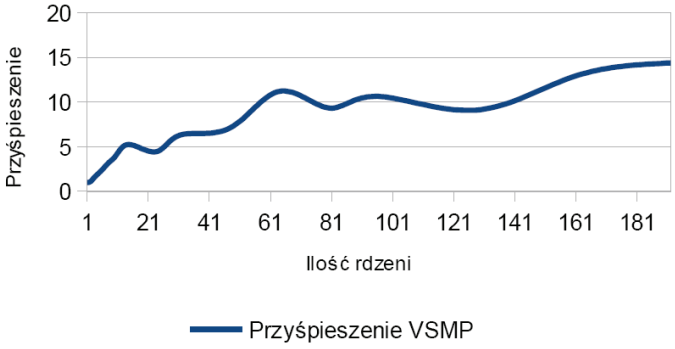
Obudowy typu blade HP C7000 10U, serwery: typu blade HP ProLiant BL490 G6 (16 płyt głównych na obudowę blade), procesory Intel Xeon X5670 @ 2.93GHz, 12MB Cache, architektura EM64T, liczba procesorów: 32, Liczba rdzeni obliczeniowych: 192, Moc obliczeniowa: 2,25 TFlops (całość), całkowita pamięć operacyjna: 1152 GB (16 nodów po 72 GB), Sieć InfiniBand (40Gb/s) oraz Gigabit Ethernet (10Gb/s) .

Wyniki analizy wydajnościowej przedstawiono w Tab. 6 oraz Ryc. 5.

| Ilość wątków | Czas realizacji | Przyśpieszenie |
|---|---|---|
| 1 | 2128 | 1,000 |
| 2 | 2003 | 1,062 |
| 3 | 1502 | 1,416 |
| 4 | 1195 | 1,780 |
| 6 | 870 | 2,445 |
| 8 | 678 | 3,185 |
| 10 | 557 | 3,820 |
| 12 | 488 | 4,750 |
| 16 | 412 | 5,165 |
| 24 | 476 | 4,470 |
| 32 | 335 | 6,352 |
| 48 | 298 | 7,140 |
| 64 | 190 | 11,200 |
| 80 | 228 | 9,333 |
| 96 | 200 | 10,640 |
| 128 | 234 | 9,094 |
| 160 | 166 | 12,819 |
| 192 | 148 | 14,378 |

Tab. 6. Wyniki czasowe uzyskane podczas analizy przypuszczenia Beal'a zrównoleglonego w środowisku OpenMP wykonywanego z parametrami max_base = 500, max_power = 500. Czasy mierzone w sekundach. Źródło: Opracowanie własne.

*Tab. 6. Time results recived during Beal's conjecture analysis of the parallel system based on OpenMP environment performer with parameters max_base = 500, max_power = 500. Times measured in seconds. Source: Own data.*




Ryc. 5. Przyśpieszenia uzyskane na platformie wykorzystującej Versatile SMP Foundation Advanced Platform firmy ScaleMP. Źródło: Opracowanie własne.

*Fig. 5. Accelerations received with use of Versatile SMP Foundation Advanced Platform, by ScaleMP. Source: Own data.*

Jak wynika z powyższej Tabeli oraz powyższego wykresu obliczenia na platformie vSMP wykorzystujące dużą ilość rdzeni są bardzo mało opłacalne. Wykorzystanie więcej niż 64 rdzeni nie daje już relatywnego przyśpieszenia. Jest to spowodowane najprawdopodobniej nie skalowaniem się w tym przypadku całej platformy i ograniczeniem przepustowości (lub też opóźnień) pamięci.

### Adaptacja oprogramowania na potrzeby projektu rozproszonego

Aby uruchomić własny system rozproszony należało dostosować do jego potrzeb zarówno samą aplikację generującą wyniki, jak i sam rdzeń BOINC. Ten rozdział skrótowo opisuje przeprowadzone prace.

### Dostosowanie do potrzeb silnika analizującego przypuszczenie Beal'a

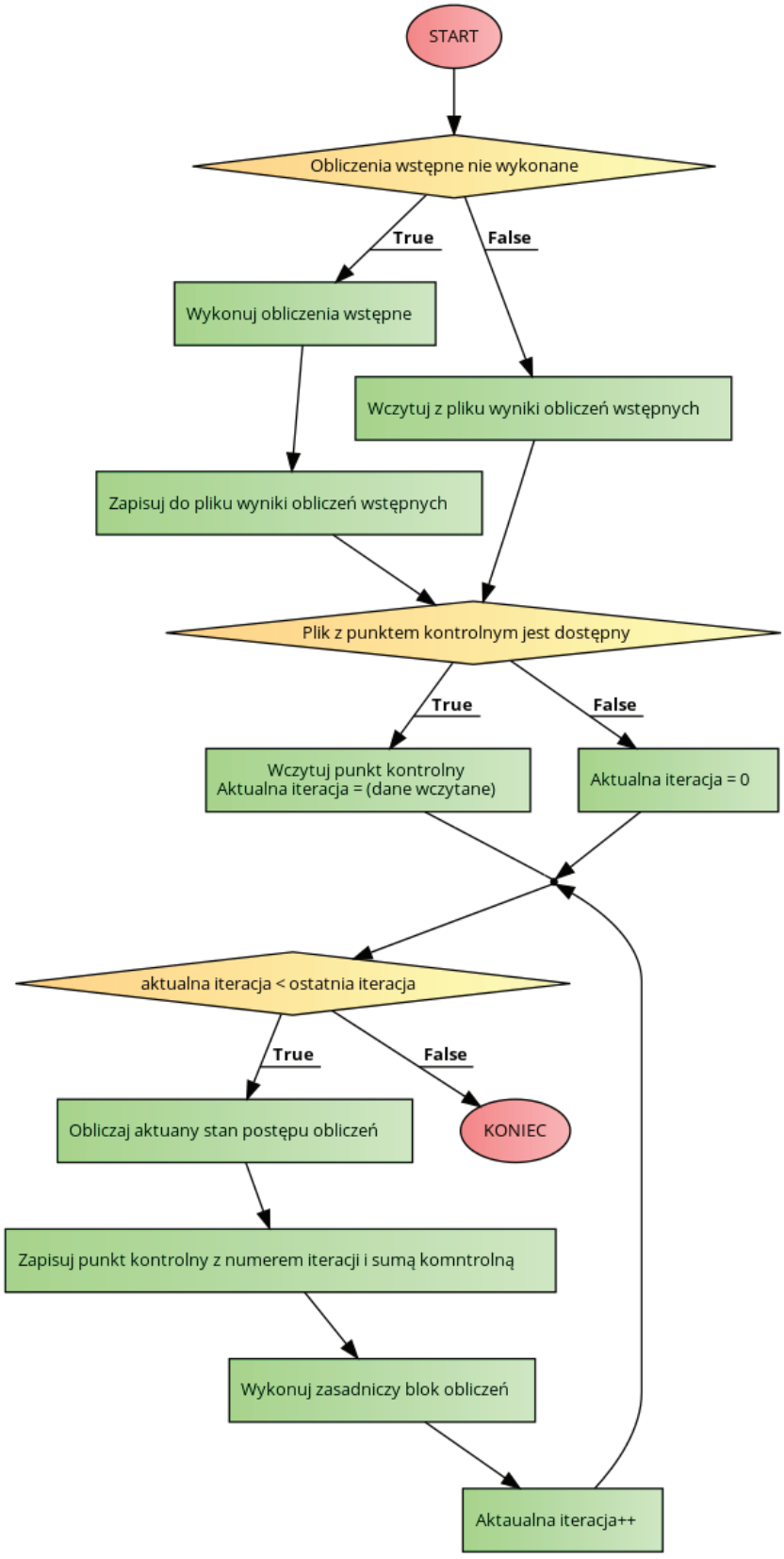
Praktycznie każdą uruchamianą niezależnie przez klienta BOINC aplikację należy dostosować do jego wymogów. Wyjątkiem tutaj mogą być programy obliczeniowe uruchamiane za pomocą tak zwanego *wrappera* (pośrednika). W projekcie BealF@Home zastosowano jednak bezpośrednie wykorzystanie BOINC API z pominięciem funkcjonalności jakie daje wrapper, gdyż jest on całkowicie zbędny[6]. Na Listingu 1 przedstawiono uproszczony schemat zastosowanego rozwiązania. Na Ryc. 6. zaprezentowano natomiast analogiczny schemat blokowy.

6.Wrapper ma sens np. gdy aplikacja licząca jest napisana w innym języku niż C/C++ i nie ma możliwości z jego poziomu wywoływania funkcji BOINC API.

```
START;

if(Obliczenia wstępne nie wykonane)
{
  Wykonuj obliczenia wstępne;
  Zapisuj do pliku wyniki obliczeń wstępnych;
}
else
{
  Wczytuj z pliku wyniki obliczeń wstępnych;
}

if(Plik z punktem kontrolnym jest dostępny)
{
  Wczytuj punkt kontrolny
  Aktualna iteracja = (dane wczytane);
}
else
{
  Aktualna iteracja = 0;
}

while(aktualna iteracja < ostatnia iteracja)
{
  Obliczaj aktuany stan postępu obliczeń;
  Zapisuj punkt kontrolny z numerem iteracji i sumą komntrolną;
  Wykonuj zasadniczy blok obliczeń;

  if(Znaleziono prawdopodobne rozwiązanie)
  {
    Zapisuj rozwiązanie do pliku;
  }

  Aktaualna iteracja++;
}

KONIEC;
```

Listing 1. Uproszczony pseudokod prezentujący możliwość adaptacji programu na potrzeby projektu BOINC. Źródło: Opracowanie własne.

*Listing 1. Simplified pseudocode presenting programs adaptation capabilities for BOINC project. Source: Own data.*




Ryc. 6. Uproszczony schemat blokowy prezentujący możliwość adaptacji programu na potrzeby projektu BOINC. Źródło: Opracowanie własne.
*Fig. 6. Simplified block diagram presenting programs adaptation capabilities for BOINC project. Source: Own data.*

Na początku programu oczywiście należy załączyć (*include*) plik "*boinc_api.h*". Następnie na na samym początku funkcji głównej main zostało zainicjowane środowisko BOINC API za pomocą funkcji *boinc_init()*. Jej obsługę zaprezentowano na Listingu 2.

```
int rc = boinc_init();
if (rc)
{
    fprintf(stderr, "APP: boinc_init() failed. rc=%d\n", rc);
    exit(rc);
}

boinc_fraction_done(0.0);
```

Listing 2. Obsługa funkcji boinc_init() oraz ustawienie postępu obliczeń na 0% za pomocą funkcji boinc_fraction_done(). Źródło: Opracowanie własne.

*Listing 2. Handling of boinc_init() function and setting of computing progres to 0% with use of boinc_fraction_done(). Source: Own data.*

Kolejnym etapem, który warto opisać są kroki zaznaczone na Listingu 4 i Ryc. 11 jako „*Obliczaj aktualny stan postępu obliczeń*" oraz „*Zapisuj punkt kontrolny z numerem iteracji i sumą kontrolną*". Są one realizowane już w funkcji głównej poszukującej kontrprzykładu dla przypuszczenia Beal'a. Operacje te przedstawiono na Listingu 3.

```
progress_value = (float) (xi - min_index) / (float) (max_index - min_index);

fp_progress=fopen("progress", "w");
fprintf (fp_progress, "%f", progress_value);
fclose(fp_progress);

boinc_fraction_done(progress_value);

fp_checkpoint=fopen("checkpoint", "w");
fprintf (fp_checkpoint, "%lld %lld", xi, checksum);
fclose(fp_checkpoint);

boinc_checkpoint_completed();
```

Listing 3. Kod realizujący obliczanie aktualnego stanu postępu obliczeń, aktualizowanie paska postępu oraz zapis do pliku punktu kontrolnego i sumy kontrolnej. Źródło: Opracowanie własne.

*Listing 3. Code realizing computation of actual computing state, actualization of progress bar and record to control point file and control sum. Source: Own data.*

Na Listingu 6 wykorzystano dwie funkcje z BOINC API: wspomnianą już wcześniej *boinc_fraction_done()*, która przyjmuje za argument ułamek określający postęp obliczeń (0 → 0%, 0.5 → 50%, 1.0 → 100%) oraz *boinc_checkpoint_completed()*, która jak nazwa informuje wysyła informację do BOINC API, że punkt kontrolny został zapisany poprawnie.

Obliczanie stanu postępu obliczeń zostało zaprezentowane w pierwszej linijce Listingu 6. Do obliczania stanu wykorzystywane są zmienne: *xi, min_index oraz max_index*. Określają one odpowiednio numer aktualnej iteracji, numer iteracji startowej oraz numer iteracji końcowej. Dla przykładu przy wartościach: *xi = 620, min_index = 600, max_index = 800* wartość wyświetlona przy odpowiednim zadaniu na pasku postępu będzie wynosiła *(620-600) / (800-600) = 20 / 200 = 0.1 => 10%.*

Zapisany punkt kontrolny oraz sumę kontrolną należy przy wznawianiu obliczeń odpowiednio wczytać z pliku. Kod odpowiedzialny za obsługę tego działania zaprezentowano na Listingu 4.

```
unsigned long long int checksum = 0;

if ( (fp_checkpoint=fopen("checkpoint", "r"))==NULL )
{
    xi = min_index;
}
else
{
    fscanf(fp_checkpoint, "%lld %lld", &xi, &checksum);
    fclose(fp_checkpoint);
}
```

Listing 4. Kod w funkcji poszukującej kontrprzykładu dla przypuszczenia Beal'a odpowiedzialny za wznawianie obliczeń z zapisanego punktu kontrolnego (wczytaniu podlega także zapisana suma kontrolna). Źródło: Opracowanie własne.

*Listing 4. Code of function searching for counterexample for Beal's conjecture responsible for resuming computation from stored control point (control sum is also loaded). Source: Own data.*

W tym miejscu warto także napisać czym jest wcześniej wspomniana suma kontrolna. Suma kontrolna jest zawsze dopisywana na końcu pliku „*result*" z ewentualnymi rezultatami. Znaczna część zadań (ponad 99%) kończy się nie odnalezieniem w danym przedziale nawet rozwiązania prawidłowego modulo $2^{64}$. Oznacza to, że gdyby nie było sumy kontrolnej zwracany byłby jedynie pusty plik. Nie byłoby więc możliwości zweryfikowania przez serwer, czy faktycznie dany komputer klienta wykonał wszystkie wymagane obliczenia. Można więc wyobrazić sobie sytuację, gdy użytkownik zatrzymuje aplikację liczącą, podmienia jedynie wartość w pliku gdzie zapisany jest ostatni punkt kontrolny na wyższą (bliżą końcu obliczeń) i wznawia obliczenia ale już od nowego, dalszego punktu kontrolnego. W ten sposób teoretycznie możliwe byłoby oszukanie systemu, co jednak zostało wyeliminowane przez zastosowanie sumy kontrolnej.

Suma kontrolna na samym starcie aplikacji wynosi 0 (Linia pierwsza w Listingu 7). Z czasem działania funkcji głównej wyszukującej kontrprzykłady jest zwiększana o wartość bezwzględną pewnej liczby zwróconej na danym kroku iteracji przez *binary_search()*. Jest to procedura całkowicie deterministyczna i na różnych komputerach, w różnym czasie obliczeń będzie dawała ten sam wynik końcowy (sumę kontrolną). Kod prezentujący obliczenie końcowe sumy kontrolnej oraz operację jej zapisu do pliku „*result*" przedstawiono na Listingu 5.

```
checksum = 1000 + checksum % 1000;

fp_results=fopen("result", "a");
fprintf (fp_results, "CHECKSUM=%ld", checksum);
fclose(fp_results);
```

Listing 5. Kod prezentujący obliczenie końcowe sumy kontrolnej oraz operację jej zapisu do pliku „result". Źródło: Opracowanie własne.

*Listing 5. Code presenting final computation of control sum and operation of it's record to the „result" file. Source: Own data.*

Jak widać na Listingu 8 suma kontrolna podlega ostatecznie pewnemu „okrojeniu". Dzięki operacji w linii pierwszej będzie miała ona jednak zawsze cztery cyfry (od 1000 do 1999). Jest to istotne z pewnych względów i aspekt ten został wykorzystany podczas programowania walidatora po stronie serwera projektu BOINC.

Na samym końcu programu można wywołać ostatecznie *boinc_fraction_done(1.0)* oraz funkcję *boinc_finish(0)*, która poinformuje BOINC API o zakończeniu obliczeń.

## Dostosowanie do potrzeb platformy BOINC

Pierwszą rzeczą jaką powinniśmy zrobić na zainstalowanym już serwerze BOINC jest dodanie aplikacji. Pierwszą rzeczą jaką powinniśmy wykonać jest dodanie nazwy naszej aplikacji do pliku *project.xml* znajdującego się w katalogu głównym projektu. Wpis ten będzie wyglądać następująco:

*<app>*
*<name>beal_engine</name>*
*<user_friendly_name>Beal Engine</user_friendly_name>*
*</app>*

Kolejnym etapem jest stworzenie odpowiedniej struktury katalogów (od ./apps) oraz umieszczenie w nim odpowiednich plików (w tym oczywiście także aplikacji liczącej). Przykładowo dla wersji aplikacji 1.00 i platformy Linux x86_64 *(x86_64-pc-linux-gnu*) będzie to:

*./apps/beal_engine/1.00/x86_64-pc-linux-gnu/*

Dla aplikacji beal_engine w wersji 1.01 i platformy Microsoft Windows x86_64 będzie to:

*./apps/beal_engine/1.01/windows_x86_64/*

Więcej etykiet określających różne platformy odnajdziemy w pliku project.xml. Należy zaznaczyć, że możemy zdefiniować aplikacje dla różnych, bardzo egzotycznych platform takich jak konsola do gier PlayStation 3 lub system Android działający na procesorach ARM.
W powyżej zdefiniowanych katalogach umieszczamy aplikację liczącą. W naszym przypadku dla wersji 1.00 i Linux x86_64 (*x86_64-pc-linux-gnu*) jej nazwa będzie wyglądać następująco:

*beal_engine_1.00_x86_64-pc-linux-gnu*

Zasadniczo można powiedzieć, że szablon nazwy aplikacji można zdefiniować jako:

*nazwa_aplikacji_number_wersji_platforma*

Dodatkowo w tym samym katalogu co aplikacja musimy umieścić także dwa pliki: *job.xml* oraz *version.xml*. Zawartość pliku *job.xml* przedstawiono na Listingu 6. Na Listingu 7 przedstawiono natomiast zawartość pliku *version.xml*.

```
<job_desc>
    <task>
        <application>beal_enigne</application>
    </task>
</job_desc>
```

Listing 6. Zawartość pliku job.xml. Źródło: Opracowanie własne.
*Listing 6. Contents of job.xml file. Source: Own data.*

```
<version>
   <file>
      <physical_name>beal_engine_1.00_x86_64-pc-linux-gnu</physical_name>
      <needs_network/>
      <copy_file/>
      <main_program/>
    </file>
    <file>
      <physical_name>job.xml</physical_name>
      <logical_name>job.xml</logical_name>
   </file>
</version>
```

Listing 7. Zawartość pliku version.xml. Źródło: Opracowanie własne.
*Listing 7. Contents of version.xml file. Source: Own data.*

Kolejnym krokiem jaki powinniśmy wykonać jest stworzenie odpowiednich szablonów i zapisanie ich w katalogu *./templates/*. Ja szablon wejściowy nazwałem *input.xml (./templates/input.xml)*, a szablon wyjściowy *output.xml (./templates/output.xml)*. Przykładowy szablon wejściowy przedstawiono na Listingu 8, a przykładowy szablon wyjściowy na Listingu 9.

```
<file_info>
        <number>0</number>
</file_info>
<workunit>
        <file_ref>
                <file_number>0</file_number>
                <open_name>input</open_name>
                <copy_file/>
        </file_ref>

        <target_nresults>2</target_nresults>
        <min_quorum>2</min_quorum>
        <rsc_fpops_bound>99999999999999999</rsc_fpops_bound>
        <rsc_fpops_est>575705506285715</rsc_fpops_est>
        <rsc_memory_bound>10000000.000000</rsc_memory_bound>
        <delay_bound>604800</delay_bound>
</workunit>
```

Listing 8 Przykładowy szablon wejściowy. Źródło: Opracowanie własne.
*Listing 8. Example input template. Source: Own data.*

Najbardziej rozwinięty jest w szablonie wejściowym tag *<workunit>*. To w nim są zawarte informacje niezbędne do wygenerowania nowego zadania na platformie BOINC. W tagu *<file_ref>* znajdują się informacje o pliku lub też plikach wejściowych, z których korzysta nasza aplikacja licząca. Tag *<delay_bound>* określa czas *deadline* – czas (wyrażony w sekundach) w jakim klient musi wykonać wszystkie obliczenia i odesłać je do serwera. Jeżeli w tym okresie serwer nie otrzyma odpowiedzi to przydzieli wykonanie określonej porcji obliczeń innemu klientowi. *<rsc_memory_bound>* określa górny limit pamięci (wyrażony w Bajtach), który może być wykorzystany przez aplikację liczącą. *<rsc_fpops_est>* to przeciętna ilość operacji zmiennoprzecinkowych, która jest wymagana do zakończenia zadania, a *<rsc_fpops_bound>* maksymalna, graniczna ilość operacji zmiennoprzecinkowych po przekroczeniu, które zadanie zostanie zabite.

```
<output_template>
    <file_info>
        <name><OUTFILE_0/></name>
        <generated_locally/>
        <upload_when_present/>
        <max_nbytes>32768</max_nbytes>
        <url><UPLOAD_URL/></url>
    </file_info>
    <result>
        <file_ref>
            <file_name><OUTFILE_0/></file_name>
            <open_name>result</open_name>
            <copy_file>1</copy_file>
            <no_delete/>
        </file_ref>
    </result>
</output_template>
```

Listing 9. Przykładowy szablon wyjściowy. Źródło: Opracowanie własne.
*Listing 9. Example output template. Source: Own data.*

Szablon wyjściowy może się wydawać trochę prostszy od wejściowego. Najistotniejsza informacja jest zawarta w tagu <*result*> i <*file_ref*> - jest to nazwa pliku wynikowego wygenerowanego przez aplikację liczącą – w tym przypadku *result*. Oczywiście plików wynikowych może być wiele. W przypadku plików, które mogą zajmować dużo miejsca należy także pamiętać o prawidłowym ustaleniu wartości w tagu <*max_nbytes*>.
Po stworzeniu odpowiedniej struktury katalogów w *./apps/* oraz umieszczeniu w niej niezbędnych plików i utworzeniu odpowiednich szablonów można przejść do właściwego dodawania aplikacji do serwera BOINC.
Najpierw wydajemy polecenie:

*./bin/xadd*

Dzięki temu zostanie dodana wstępnie aplikacja. Teraz musimy zaktualizować jej wersję wydając polecenie:

*./bin/update_versions*

Wynik wywołania tego polecenia przedstawiono na Listing. 10.

```
boincadm@beal:~/projects/bealf$ ./bin/update_versions

Found app version directory for: beal_engine 1.00 x86_64-pc-linux-gnu

    NOTICE: You have not provided a signature file for job_2.0.xml,
    and your project's code-signing private key is on your server.

    IF YOUR PROJECT IS PUBLICLY ACCESSABLE, THIS IS A SECURITY VULNERABILITY.
    PLEASE STOP YOUR PROJECT IMMEDIATELY AND READ:
    http://boinc.berkeley.edu/trac/wiki/CodeSigning

    Continue (y/n)? y
cp apps/beal_engine/1.00/x86_64-pc-linux-gnu/job_2.0.xml \
/home/boincadm/projects/bealf/download/job_2.0.xml
cp apps/beal_engine/1.00/x86_64-pc-linux-gnu/beal_engine_1.00_x86_64-pc-linux-gnu \
/home/boincadm/projects/bealf/download/beal_engine_1.00_x86_64-pc-linux-gnu
    Files:
        beal_engine_1.00_x86_64-pc-linux-gnu (main program)
        job_2.0.xml
    Flags:
        API version: 7.1.0
    Do you want to add this app version (y/n)? y
    App version added successfully; ID=1
```

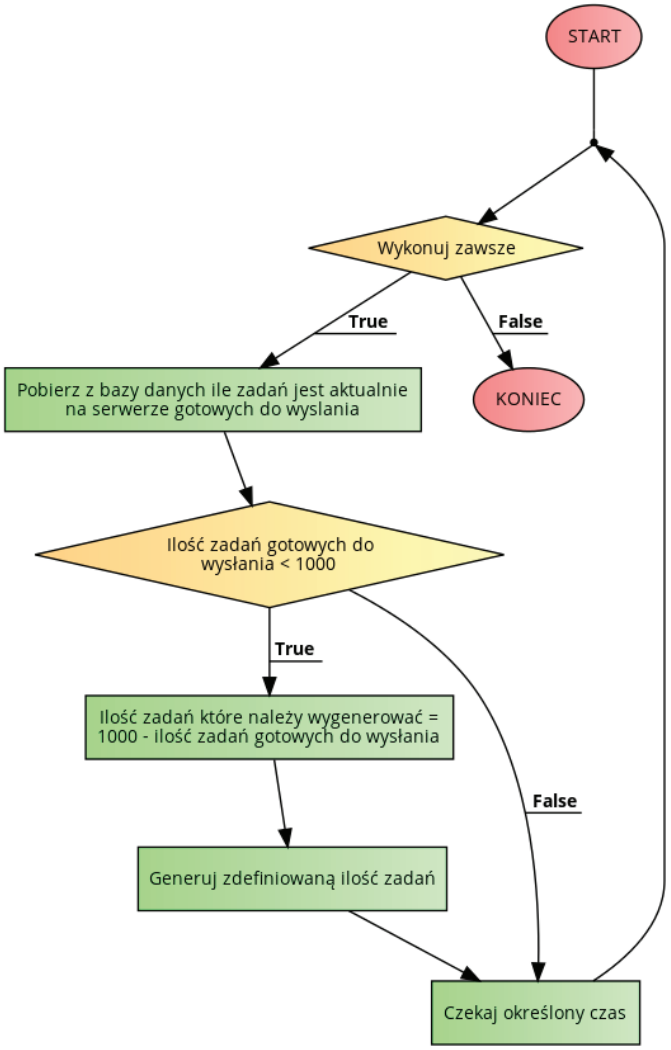
Listing. 10. Wywołanie polecenia ./bin/update_versions podczas pierwszej aktualizacji wersji aplikacji. Źródło: Opracowanie własne.
*Listing 10. Induction of ./bin/update_versions instruction during first actualization of application version. Source: Own data.*

Ostatnim już etapem będzie wygenerowanie WU *(ang. workunit*), które będą mogli pobrać klienci i rozpocząć obliczenia.
Możemy zrobić to wprowadzając:

*./bin/create_work -appname beal_engine -wu_name NAZWA-WU -wu_template templates/input.xml -result_template templates/output.xml plik_wejściowy*

gdzie *plik_wejściowy* jest nazwą pliku, który znajduje się w katalogu *./download/*. Po wykonaniu tej operacji możemy połączyć się BOINC Manager'em (lub samym BOINC klientem) z naszym projektem i pobrać pierwsze zadanie obliczeniowe.
Jako walidator zadań na serwerze BOINC zastosowano zmodyfikowany *sample_bitwise_validator*. Jedno takie samo zadanie jest zawsze wysyłane co najmniej do dwóch klientów (min_quorum = 2). Odsyłają oni wynik obliczeń, który powinien być taki sam. *Sample_bitwise_validator* porównuje ze sobą nadesłane wyniki. Jeżeli są identyczne to zatwierdza zadanie jako zrealizowane. Jeżeli natomiast jest pomiędzy nimi jakakolwiek różnica, to wysyła to samo zadanie do trzeciego klienta i na podstawie wyniku, który on zwróci waliduje ostatecznie daną porcję obliczeń.
Główną modyfikacją walidatora jest dodanie do jego możliwości wyłuskiwania z plików wynikowych potencjalnych rozwiązań. W prosty sposób wykorzystano fakt, że waga pliku, w którym nie ma żadnego rozwiązania wynosić będzie zawsze 13 Bajtów (jest to jedynie zapis sumy kontrolnej, która ma postać: *CHECKSUM=XXXX*). W przypadku gdy rozmiar ten jest większy to znaczy, że w pliku znajdują się potencjalne rozwiązania przypuszczenia Beal'a i zostają one umieszczone w dedykowanej do tego celu bazie danych.

Jednym z kluczowych zadań na platformie BOINC jest także odpowiednie generowanie masowych zadań (jak wygenerować jedno zadanie zaprezentowano wcześniej). Można oczywiście funkcjonalność tą realizować w sposób trywialny – np. za pomocą specjalnego skryptu Bash'a wywoływać BOINC'owy program *create_work* i w ten sposób generować kolejne próbki zadań. Można także opracować bardziej wyrafinowane metody. W projekcie BealF@Home początkowo wykorzystywano właśnie prosty skrypt Bash (dodany do pracy jako *załącznik D*). Na późniejszym etapie zastosowano bardziej zaawansowany generator napisany w języku C, który wywoływał polecenia systemowe i łączył się z bazą danych projektu (dodany do pracy jako *załącznik E*). Uproszczony schemat blokowy tego generatora zaprezentowano na Ryc. 7.




Ryc. 7. Uproszczony schemat blokowy bardziej zaawansowanego generatora zadań wykorzystanego na platformie BealF@Home. Źródło: Opracowanie własne.
*Fig. 7. Simplified block diagram of more advanced task generator used on BealF@Home platform. Source: Own data.*

Jak widać powyższy generator zadań jest wykonywany w nieskończonej pętli (może zostać przerwany tylko przez użytkownika). Przy każdej iteracji (które dzielą odpowiedni okres czasu) jest sprawdzana ilość zadań na serwerze, i jeżeli jest ona mniejsza niż 1000 to automatycznie generowane są nowe próbki. Dzięki takiemu rozwiązaniu zadania są generowane zawsze na bieżąco, bez chwilowego, znacznego obciążenia serwera jak to było w przypadku prostego rozwiązania napisanego w Bash'u.

## Podsumowanie

Opracowana na potrzeby tej pracy metoda testowania przypuszczenia Beal'a nie jest oczywiście doskonała. Analizy wykonane dla określonych przedziałów mają tą wadę, że nie znajdą rozwiązania, którego zmienne *x, y, z, n, m, r* leżą w różnych z nich. Dla przykładu weźmy analizowane dwa analizowane przedziały:

> Pierwszy z nich dla podstaw: [2000, 4000]; Dla wykładników: [3, 1000].
>
> Drugi z nich dla podstaw: [4000, 6000]; Dla wykładników: [3, 1000].

Algorytm nie odnajdzie rozwiązania jeżeli takowe znajduje się w sumie zbioru podstaw: [2000, 6000]. Niedogodność tą można rozwiązać przynajmniej na kilka sposobów, kosztem jednak znacznego zwiększenia zapotrzebowania na moc obliczeniową. Rozwiązanie tego problemu będzie jednak zagadnieniem początkowym dla przyszłych badań. Ostatecznie w ramach tej pracy nie udało się odnaleźć kontrprzykładu dla przypuszczenia Beal'a. Nie oznacza to oczywiście jak wcześniej zaznaczono, że takie rozwiązanie nie istnieje. Rezultatem przeprowadzonych obliczeń są rozwiązania prawidłowe modulo $2^{64}$. Takich przypadków do dnia 07.06.2015 r. odnaleziono w sumie 47 (zostały przedstawione w Załączniku).

## Podziękowania

Obliczenia wykonano w Interdyscyplinarnym Centrum Modelowania Matematycznego i Komputerowego (ICM) Uniwersytetu Warszawskiego w ramach grantu obliczeniowego nr G55-11.
Praca została wykonana z wykorzystaniem Infrastruktury PL-Grid.

## Literatura

1. Monika Kwiatkowska, Łukasz Świerczewski, „Testowanie przypuszczenia Beal'a z wykorzystaniem klasycznych procesorów”, Biuletyn Naukowy Wrocławskiej Wyższej Szkoły Informatyki Stosowanej. Informatyka, 2015 5.
2. Swierczewski, Lukasz. "The Distributed Computing Model Based on The Capabilities of The Internet." *arXiv preprint arXiv:1210.1593* (2012).
3. Liu, Tianyu, X. George Xu, and Christopher D. Carothers. "Comparison of two accelerators for Monte Carlo radiation transport calculations, Nvidia Tesla M2090 GPU and Intel Xeon Phi 5110p coprocessor: A case study for X-ray CT imaging dose calculation." *Annals of Nuclear Energy 82* (2015): 230-239.
4. Schmidl, Dirk, et al. "How to scale nested openmp applications on the scalemp VSMP architecture." *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*. IEEE, 2010.
5. Berr, Nicolas, et al. "Trajectory-Search on ScaleMP's vSMP Architecture." *PARCO*. 2011.
6. Świerczewski, Łukasz. "Symulacja funkcjonalnego systemu kwantowego na równoległych komputerach klasycznych IV generacji." (2013).
7. Свєрчевський, Лукаш, and Łukasz Świerczewski. "Polish BOINC projects." *Proceedings of the third international scientific and practical conference FOSS Lviv 2013*. 2013.
8. Ďurech, J., J. Hanuš, and R. Vančo. "Asteroids@ home—A BOINC distributed computing project for asteroid

shape reconstruction." *Astronomy and Computing 13* (2015): 80-84.
9. Durech, Josef, J. Hanus, and R. Vanco. „Asteroids@ Home." *AAS/Division for Planetary Sciences Meeting Abstracts*. Vol. 44. 2012.
10. Cameron, David. ATLAS@ *Home: Harnessing Volunteer Computing for HEP*. No. ATL-SOFT-SLIDE-2015-159. ATL-COM-SOFT-2015-018, 2015.
11. Stainforth, Dave, et al. „Climateprediction. net: Design Principles for Publicresource Modeling Research." *IASTED PDCS*. 2002.
12. Abbott, B. P., et al. „Einstein@ Home search for periodic gravitational waves in early S5 LIGO data." *Physical review d* 80.4 (2009): 042003.
13. Aasi, Junaid, et al. „Einstein@ Home all-sky search for periodic gravitational waves in LIGO S5 data." *Physical Review D* 87.4 (2013): 042001.
14. Krebs, Viola. „Motivations of cybervolunteers in an applied distributed computing environment: MalariaControl. net as an example." *First Monday* 15.2 (2010).
15. Harris, Jack, et al. *Mindmodeling@ Home... and Anywhere Else You Have Idle Processors*. AIR FORCE RESEARCH LAB MESA AZ WARFIGHTER READINESS RESEARCH DIVISION, 2009.
16. Ronald W Green (Intel), Native and Offload Programming Models, September 9, 2012, (Online), https://software.intel.com/en-us/articles/native-and-offload-programming-models.

## Załączniki

### 1. Tablica z odnalezionymi rozwiązaniami z wykorzystaniem jedynie zasobów superkomputerowych ICM UW prawidłowymi dla operacji modulo $2^{64}$

Równania w postaci: $x^m + y^n = z^r$ (*W kolejności odnalezienia*)

| Lp. | x | y | z | n | m | r | Wartość modulo |
|---|---|---|---|---|---|---|---|
| 1 | 676444 | 677849 | 676337 | 20 | 818 | 857 | 17784773319750998129 |
| 2 | 604942 | 605391 | 605965 | 35 | 246 | 456 | 2288372447511289633 |
| 3 | 1220330 | 1221513 | 1220387 | 38 | 908 | 600 | 4948160361256468193 |
| 4 | 1694050 | 1695039 | 1694513 | 28 | 108 | 432 | 12609605036839803457 |
| 5 | 2372032 | 2372959 | 2372289 | 7 | 688 | 552 | 680105010251968001 |
| 6 | 2172698 | 2173887 | 2172769 | 37 | 532 | 760 | 9706423526405764353 |
| 7 | 2128339 | 2128438 | 2129297 | 560 | 30 | 492 | 13199036057921569473 |
| 8 | 3957038 | 3957605 | 3956577 | 25 | 880 | 38 | 11882078048601687105 |
| 9 | 3859227 | 3859885 | 3859618 | 35 | 557 | 31 | 15501064842072031232 |
| 10 | 4018178 | 4018833 | 4019705 | 41 | 111 | 890 | 5477952416353461617 |
| 11 | 4708052 | 4708227 | 4708753 | 17 | 732 | 75 | 6724712973859386417 |
| 12 | 5240198 | 5240399 | 5240533 | 3 | 334 | 306 | 3866028622879127929 |
| 13 | 5304794 | 5305287 | 5304823 | 34 | 853 | 555 | 8893799672103127271 |
| 14 | 5501351 | 5501900 | 5501959 | 228 | 24 | 628 | 16487540361621293025 |
| 15 | 5574977 | 5575922 | 5574213 | 620 | 22 | 576 | 17465805828848740097 |
| 16 | 5607226 | 5607697 | 5607227 | 42 | 785 | 860 | 12384975864894544401 |
| 17 | 6030307 | 6031672 | 6030159 | 752 | 9 | 868 | 263824262007728833 |
| 18 | 6708793 | 6709351 | 6709566 | 567 | 203 | 49 | 15942179730938134528 |
| 19 | 6708818 | 6709351 | 6708793 | 50 | 58 | 162 | 569265639489243057 |
| 20 | 6709351 | 6709678 | 6708793 | 58 | 50 | 162 | 569265639489243057 |
| 21 | 6980534 | 6981047 | 6980681 | 30 | 292 | 36 | 14623819350573189025 |
| 22 | 7170340 | 7170971 | 7171143 | 27 | 612 | 906 | 113473291799894129 |
| 23 | 7170838 | 7170971 | 7171143 | 55 | 408 | 604 | 16563665043124141217 |
| 24 | 7170874 | 7170971 | 7171143 | 54 | 612 | 906 | 113473291799894129 |
| 25 | 7170971 | 7171143 | 7171454 | 306 | 453 | 53 | 13249590103723999232 |
| 26 | 7170971 | 7171270 | 7171143 | 612 | 54 | 906 | 113473291799894129 |
| 27 | 7170971 | 7171624 | 7171143 | 612 | 18 | 906 | 113473291799894129 |
| 28 | 7170971 | 7171862 | 7171143 | 408 | 55 | 604 | 16563665043124141217 |
| 29 | 7170971 | 7171898 | 7171143 | 612 | 54 | 906 | 113473291799894129 |
| 30 | 7171143 | 7171232 | 7170971 | 604 | 11 | 408 | 7232206615212473505 |
| 31 | 7171143 | 7171306 | 7170971 | 604 | 55 | 408 | 7232206615212473505 |
| 32 | 7504489 | 7504556 | 7505381 | 402 | 10 | 92 | 7432643896266708881 |
| 33 | 7718927 | 7719707 | 7719778 | 821 | 596 | 9 | 4756025264355303936 |
| 34 | 8234936 | 8235097 | 8234449 | 5 | 514 | 11 | 10930583924314857201 |
| 35 | 8524223 | 8524634 | 8524993 | 256 | 30 | 768 | 10111295053388726273 |
| 36 | 8401517 | 8401847 | 8400550 | 302 | 611 | 18 | 13584895879439187968 |
| 37 | 8738482 | 8738539 | 8738035 | 12 | 218 | 78 | 4609790393328797913 |
| 38 | 8846743 | 8847064 | 8846613 | 882 | 15 | 364 | 12111255234754506385 |
| 39 | 8962177 | 8963428 | 8963573 | 377 | 20 | 352 | 16483435336951299201 |

**2. Tablica z odnalezionymi rozwiązaniami z wykorzystaniem zarówno zasobów superkomputerowych ICM UW, jak i platformy do obliczeń rozproszonych BOINC prawidłowymi dla operacji modulo 264**

Równania w postaci: $x^m + y^n = z^r$ (*W kolejności odnalezienia*)

| Lp. | x | n | y | m | z | r | Wartość modulo | Odkrywca | Data odkrycia |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 20070431 | 770 | 20070463 | 113 | 20071398 | 25 | 14155650366 401675264 | G55-11* | Fri, 08 May 2015 23:19:00 GMT |
| 2 | 20200287 | 784 | 20201362 | 27 | 20200409 | 192 | 12393561479 426804225 | G55-11* | Mon, 18 May 2015 04:27:31 GMT |
| 3 | 20394767 | 544 | 20394958 | 50 | 20394071 | 448 | 71486657090 90860545 | Zombie67 [MM] | Thu, 21 May 2015 01:25:10 GMT |
| 4 | 20376694 | 33 | 20377193 | 762 | 20376099 | 964 | 13348672607 142987985 | Zombie67 [MM] | Thu, 21 May 2015 20:34:04 GMT |
| 5 | 20432342 | 18 | 20433535 | 530 | 20433547 | 576 | 32772966616 92875521 | Zombie67 [MM] | Fri, 22 May 2015 01:48:17 GMT |
| 6 | 20805104 | 7 | 20805295 | 936 | 20804583 | 752 | 47443809959 60797313 | Zombie67 [MM] | Sat, 23 May 2015 05:19:51 GMT |
| 7 | 20817022 | 33 | 20817127 | 16 | 20816479 | 44 | 15952473183 518859137 | Zombie67 [MM] | Sat, 23 May 2015 07:40:39 GMT |
| 8 | 21028767 | 608 | 21029887 | 815 | 21028166 | 32 | 88959050428 05030912 | G55-11* | Thu, 04 Jun 2015 22:02:35 GMT |