Zeszyty Naukowe
Politechniki Białostockiej
INFORMATYKA
Numer 6

Oficyna Wydawnicza Politechniki Białostockiej

Białystok 2010

Artykuły zamieszczone w *Zeszytach Naukowych Politechniki Białostockiej. Informatyka*
otrzymały pozytywne opinie recenzentów wyznaczonych przez Redaktora Naczelnego i Radę Naukową

The articles published in *Zeszyty Naukowe Politechniki Białostockiej. Informatyka*
have been given a favourable opinion by reviewers designated by Editor-In-Chief and Scientific Board

ISSN 1644-0331

ADRES DO KORESPONDENCJI/THE ADRESS FOR THE CORRESPONDENCE:

„*Zeszyty Naukowe Politechniki Białostockiej. Informatyka*"
Wydział Informatyki /Faculty of Computer Science
Politechnika Białostocka /Bialystok University of Technology
ul. Wiejska 45a, 15-351 Białystok
tel. 85 746 90 50, fax 85 746 97 22
e-mail: znpb@irys.wi.pb.edu.pl
http://irys.wi.pb.edu.pl/znpb

Druk: Oficyna Wydawnicza Politechniki Białostockiej
Nakład: 100 egzemplarzy

CONTENTS

ELEVATED ACTIVE CONTOUR WITH GLOBAL IMAGE ENERGY BASED ON ELECTROSTATIC FORCE

Cezary Bołdak[1]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In this article a new modification of the well known segmentation technique, namely active contour - snake, was proposed. This modification consists in a new formulation of its external force based on the electrostatics. However the base idea of giving electric charges to the image and the snake has been already presented in several works, none of them clearly adressed the problem where the charged snake took place of a charged pixel. In this situation the electrostatic force is not defined, since the distance between charges is zero. The snake proposed in this work evolves on a plane elevated above the image, what never allows this distance to become zero. The method was implemented and verified on real microscopic images of oocytes, proving its superiority on the classic snake.

Keywords: image segmentation, active contour - snake, electrostatic force

1. Introduction

The active contour (alias snake), since its introduction by Kass [7], has been one of the most powerful and most often used techniques in the image segmentation domain. Its ability to deal with moderated local noise and discontinuities in segmented objects has been widely exploited in segmentation of many classes of objects, biomedical images being one of them [8].

Despite its robustness, the original snake suffers from some problems, including its sensibility to the parameters choice and locality in finding its optimal position (having minimum energy). The former still existing in the majority of its formulations, the latter was tried to be solved (or at least improved) in many works, cited in secion 2.

This work proposes a new formulation of the snake external energy based on the electrostatic force, where the image and the snake are given electric charges of the oposite signs. Such the global energy allows the snake to “see further”, beyond

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 6, pp. 5-21, 2010.

its local neighbourhood, potentially containing noise and smaller, less relevant structures, and correctly reach the object boudaries across them. But this formulation needs a special treatement when the charged contour takes the position of the charged pixel, because the electrostatic force between them is not defined in this situation since the distance is equal to zero - in the nature two charges never can take the same position. This problem was addressed and solved in this work.

This article is organized as follows. Section 2. gives a review of the original Kass's snake and several works improving its ability to find more distant objects, including approaches using the electrostatic force. Section 3. introduces a new formulation of the snake external energy based on the electrostatic force. Section 4. presents experiments on real biomedical images showing how the new "electric" snake overcomes problems in finding objects surrounded by different, less intensive or smaller structures.

2. Background

In its original form [7], the snake is a 2D curve $v(s) = (x(s), y(s))$ (closed or open) evolving in an environment under forces to minimize its energy, composed of two forms: internal and external (the third form proposed by Kass, energy of constraints, has been used in practice very rarely):

$$E_{snake} = \int (E_{internal}(v(s)) + E_{external}(v(s)))ds. \tag{1}$$

The internal energy (or energies) controls the snake shape, namely its tensility and rigidity by terms of its first and second spatial derivatives. The external energy drives the snake to desired regions in its evironment and in segmentation it is mainly based on image intensity or image gradient. All the energies are weighted in order to allow steering the snake behaviour, e.g. to be more smooth or to better fit an object to extract. In applications, very often the curve goes to its discrete form as an ordered collection of points and the total energy becomes sum of individual energies of each point. The minimalization process consists in iterative deplacing each point separately in quest of a position with locally minimal energy. Two strategies exist here:

– examining every possible next location of each point in its local neighbourhood and choosing this one with the lowest energy (if lower than the current point energy); potential point locations should be discrete and usually are limited to pixels;

– calculation of a resultant force acting on each point and deplacing this point accordingly to this force; points can be situated on arbitrary positions, even between pixels (image intensities on inter-pixel locations are interpolated).

In every case crutial is initialization of the snake position. In the first strategy the local neighourhood should be small (order of pixels) to avoid too fast snake evolution, in the second one the external force, being usually the local image gradient, is calculated very localy (also order of pixels). All this causes that the snake will not "see" an object to segment if it is situated outside this limited scope. Taking into consideration a natural tendence of the snake to shrink [3], initially situated outside the object to segment, the snake can successfully reach it only if there is not other objects on its way. But real images, with noise and complex scenes, very rarely have this property and the correct segmentation depends very strongly on a close and precise initialization.

Many works had as their goals to widen the ability of the snake to "see" further. In virtually every its application to the segmentation, the original image is preprocessed by the Gaussian blur filter in order to expand zones where the image gradient is not null aroung edges. It improves the segmentation but this extension still remains of order of pixels and does not help in more distant initializations.

One of the first improvements to the original snake was the Cohen's balloon [3]. Additional force, pushing the snake points outside, simply inverts its natural tendence to shrink and makes it growing, like a balloon inflated with air. This force allows to initially place the contour inside an object to extract, what in many cases is more convenient than starting the evolution from its outside. Although this modification can help the snake in many cases to reach its goal, it still has drawbacks inherited from the original snake:

– the initial countour should be placed completely inside the object to extract; if the entire initial shape or its part is situated outside, it will grow and will not converge to the object;
– the method is still very sensitive to its parameters; moreover, there is one parameter more, i.e. the weight of the balloon force; its choise is crucial to good segmentation and very often depends on many factors (e.g. scale, object and image intensities): too low will prevent the snake to reach the object (it will stop on a noise/other smaller object or because of its natural shrinking tendence), too high will cause the snake to overgo the object.

Gunn and Nixon [5] propose an interesting method to extract the most distinct object in some region by placing two classical active contours: one completely inside and one completely outside the object to segment. They evolve independently

until they reach their local minima. Then, this one with higher total energy is pushed toward the better placed one (with lower energy) by adding for a moment a supplementary force, similar to the Cohen's balloon force, in order to get it out from the local minimum. The segmentation ends when two contours converge to the same position. This technique works well, but it demands to place two initial contours, what can be sometimes troublesome. Also comparing global snake energies (to decide which contour will be given the supplementary force) can push the snake out from locally well extracted regions, if it has higher total energy.

A very eficient method of extending the snake ability to see distant objects can be found in the work of Xu and Prince [10], where the active contour evolves under a new force field - Gradient Vector Flow. This force replaces the original external force (the image gradient), and it is its extension to the image regions with the gradient magnitude close to zero. In an iterative pre-processing stage (consisting in solving the generalized diffusion equation) the gradient is propagated from the object boudaries and along its local directions to empty image parts. Thus the snake in every position is given a direction to an edge (high gradient region) and it can move to it. The most often it is the closest edge, but in concave forms this force leads to more distant, "internal" object fragments. This ability to correctly segment concave objects (inversely to the classical snake) is pointed out as the next main advantage of this technique. Also, the initial contour does not need to be placed completely inside or completely outside the object - it will always see the edges. However, the real images very rarely have the empty regions. Beside the object to extract, they contain other ones, as well as noice and artefacts. Even if less intensive, they will attract the being propagated gradient and consequently - the snake. To work well, the technique would demand some extra pre-processing, e.g. zeroing too small gradient by thresholding, what will introduce a new parameter - the threshold value.

The idea to give the snake and the image electric charges is not new. Jalba et al. [6] introduced the charged-particle model (CPM) composed of free particles positively charged and evolving independently in the electric field given by the negatively charged image pixels. The pixel charge values are based on the image gradient. The particles are not organized in an ordered collection, conversely to the classic snake, and each of them evolves in the electric field independently, however influenced (repulsed) accordingly to the electrostatic force by other equally charged particles. This repulsion plays role of the internal forces in the original snake and replaces them. Only after the convergence the continuous contour (or surface in 3D) is reconstructed. That model has ability allowing it to be placed in almost arbitrary initial position, e.g. outside and beside the object or even in the form of regular mesh evenly distributed on the entire image. Once the object boudary is reached,

in at least one its fragment, the particles are evenly distributed along the edges (not encountering obstacles from the external electric field) by the repulsion force to cover the entire shape. In the electrostatic force calculation the pixel (charge) from the position occupied by the being simulated particle is simply temporarily removed to avoid this partial force to be undefined (distance zero). This action is a deviation from the physical model, but apparently it was not reported as a problem in the whole process.

The CPM behaves worse, as pointed out Yand et al. [11], when some part of the object egdes is weaker or blurred, resulting in a non-continuous final contour. They blame the internal nature of the CPM, where not ordered particles constantly leave those regions attracted by close more distinct edges. To solve this problem they proposed to incorporate the electrostatic force in the frame of the standard active contour (however, under form of the geodesic one), which always guarantees the continuous result. Their electric field, driving the contour, is also dynamic, changing locally when some part of the contour reached strong edges: the bondary competition force starts to repulse other its fragments pushing them to other undiscovered regions. The geodesic formulation of the contour as the zero level set of the higher dimension function (instead of the discrete snake) allows not to address the problem where the charged snake takes the same position with the charged pixel.

Chang and Valentino [12] proposed an electrostatic deformable model as a charged fluid model to segment medical images. A propagating front is composed of fluid elements (cells), and each of them is filled with elementary electric charges moving freely between these cells (so only on this front) and defining the electric field. This field is summed with the image gradient field and the resulting one drives the whole front evolution. Thus there is not direct interaction charged particles-charged pixels, no need to calculate the electrostatic force and no problem with the zero distance.

3. Elevated electric snake

Global external force

This work proposes a new formulation of the external electric force, which similarly to the GVF replaces its original form based on the image gradient. This new force is based on the electrostatic force, attracting two point electric charges of opposite signs proportionally to the product of their values and inversely proportionally to the square distance between them:

$$F_{electrostatic} \sim \frac{q_1 q_2}{r^2} \tag{2}$$

where q_1, q_2 - values of two electric charges, r - distance between them.

The same relation is also given by the gravitational force, but the latter is limited to attracion, while the former describes two directions of the influence: attraction (when two charges are of opposite signs) and repulsion (charges of the same signs).

In the proposed technique the contour is discrete - composed of N points: $p(i) = (x_p(i), y_p(i))$, $i = 0..N-1$, constituting a closed curve. Each such point is given a unitary electric charge of the same sign, let's say positive one: $q_p(i) = 1$. These points evolve in the electric field defined by the image in the following manner: each pixel $I(x,y)$ of the image I is given the negative electric charge $q_I(x,y)$ with the value corresponding to the gradient magnitude in this position:

$$q_I(x,y) = -|\nabla I(x,y)|. \tag{3}$$

Each such the fixed charge $q_I(x,y)$ attracts every single snake point $p(i)$ with the force $\overrightarrow{f}(i,x,y)$ of magnitude proportional to the product of these two charges ($q_I(x,y)$ and $q_p(i)$) and inversely proportional to the square distance between the pixel and the snake point. Since the snake point charge is unitary, only the pixel charge remains in the numerator:

$$f(i,x,y) = \frac{|\nabla I(x,y)|}{\|\overrightarrow{p(i) - (x,y)}\|^1}. \tag{4}$$

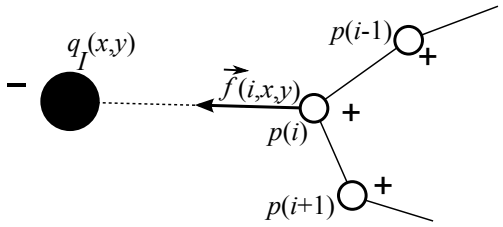


Fig. 1. Force attracting a snake point to a single pixel in the image.

Each snake point is attracted simultaneously by all the charges in the images. The vector sum of all these single forces defines the external force driving the snake in this point:

$$\overrightarrow{F}_{external}(i) = \sum_{x,y} \overrightarrow{f}(i,x,y). \tag{5}$$

Such the formulation allows the snake to see, and be attracted by, distant high gradient regions and to neglect close, low gradient obstacles. But the electrostatic

force is defined only for two charges separated by some distance. The closer they are each to other, the stronger this force becomes , tending to infinity while the distance tends to zero. The goal of each snake point is the highest gradient region, what means the situation when two charges will take the same position, where the electrostatic force between them would be not defined. To resolve this contradiction, two surfaces: the image itself and the snake evolution plane, are separated by some distance h by elevating the latter above the image (Fig. 2). Thus the snake points can move only on this elevated plane, parallel to the image, and will never touch the image pixels. The external force (Equations 4 and 5) becomes in this way 3D.

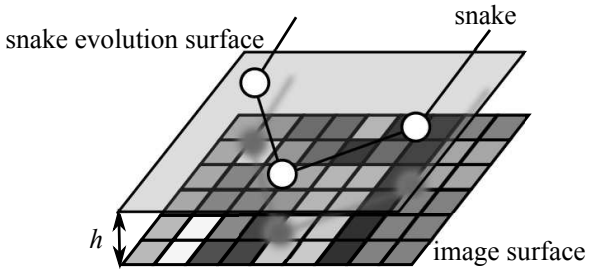


Fig. 2. Two surfaces: image and the snake evolution plane, are separated by distance h.

Snake energies and evolution

The snake evolution here consists in searching in the local neighbourhood of every of its points in order to find a new position with lower energy. Thus, the snake points locations are limited to the image pixels and every point should be given its energy instead of force.

The internal energies from the Equation 1 are responsible for the contour form: its regularity and smoothness. In this work the original Kass's formulations (based on spatial derivatives) were replaced by invariant to scale ones. Marking a vector from $p(i)$ to $p(i-1)$ by $\overrightarrow{v}_{i,i-1}$:

– the point regularity energy is expressed as normalized difference between actual distance to the previous point and the mean inter-point distance d in the whole contour:

$$E_{regularity}(i) = \frac{|d - \|\overrightarrow{v}_{i,i-1}\|\,|}{d}; \tag{6}$$

– the point smoothness energy is equal to cosinus of the angle between two vectors going to the neighbour points (in practice it is calculated as scalar product of

these normalized vectors):

$$E_{smoothness}(i) = \cos(\angle(\overrightarrow{v}_{i,i-1}, \overrightarrow{v}_{i,i+1})) = \frac{\overrightarrow{v}_{i,i-1}}{\|\overrightarrow{v}_{i,i-1}\|} \cdot \frac{\overrightarrow{v}_{i,i+1}}{\|\overrightarrow{v}_{i,i+1}\|}. \quad (7)$$

The external point energy is based on the "electrostatic" resultant force (Equation 5). For a single, stationary snake point it is assumed to be zero and only its deplacement can cause incrementing or decrementing it. This relative energy change is calculated as a negative scalar product of the electrostatic force and the vector of the examined point potencial deplacement $\overrightarrow{(\Delta x, \Delta y)}$ (Figure 3):

$$\Delta E_{external}(i, \Delta x, \Delta y) = -\overrightarrow{F}_{external}(i) \cdot \overrightarrow{(\Delta x, \Delta y)}. \quad (8)$$

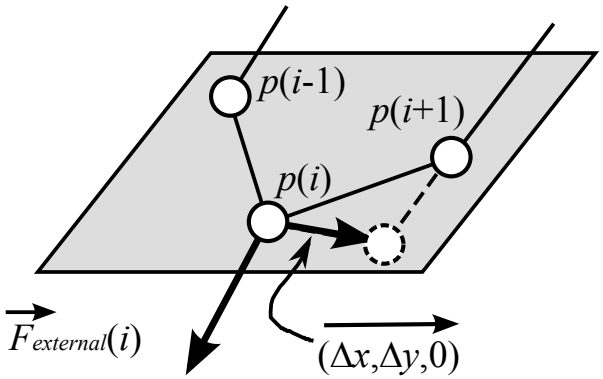


Fig. 3. Calculating the external energy change from the "electrostatic" force and the snake point potential deplacement.

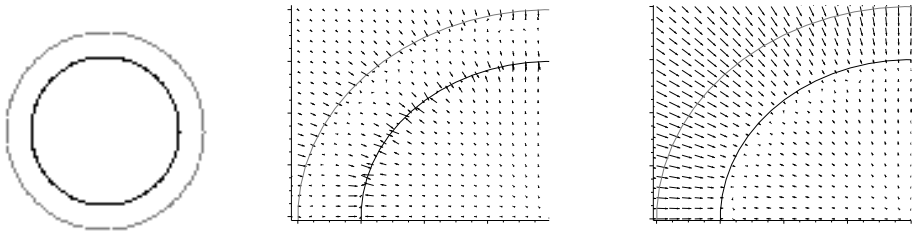
If the point deplacement is close to the force direction, then the product will be positive and the energy change negative, so the deplacement (in the energy minimalization process) will be probably accepted (if no better one is found). If it is going in the opposite direction, the energy change will be positive and it will be certainly rejected. Finally, if the force is completely vertical, all examined deplacements will be perpendicular to it and they will give the energy change equal to zero, so no better than the current position.

The overall segmentation procedure is organized as follows:

1. The initial countour is placed in the image by an operator. It does not need to be very close to the object to segment, but on its way to this object it can not find bigger objects having higher gradient and it should contain the object to segment in its inside (see Section 4.).

2. In every iteration step, each snake point is examined separately: if its deplacement in its local neighbourhood gives the energy decrementation, then this point is moved to this new position.
3. If in the current iteration no points are deplaced, the procedure can be finished.

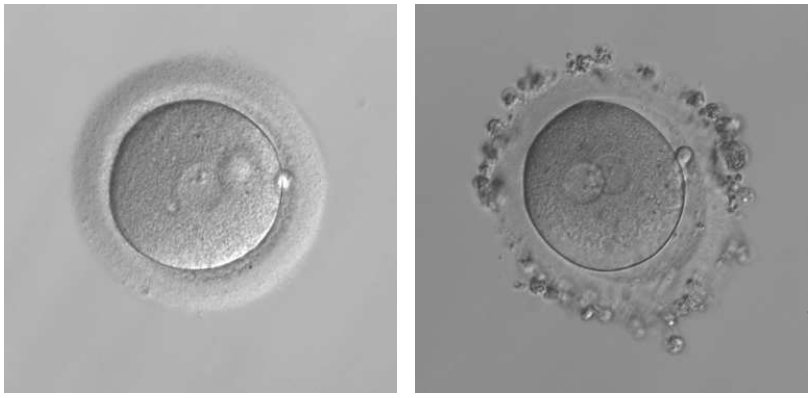
Snake elevation h



(a) (b) (c)

Fig. 4. Snake elevation steers its scope of view: (a) - the darker internal circle represents an object to segment while the brighter external one - an obstacle to cross; (b) - zoom on the upper-left part - the electric field on a lower level ($h = 1$, only x and y components shown) points toward the outer circle from both its sides; (c) - zoom on the upper-left part - the electric field on an upper level ($h = 9$, only x and y components shown) points everywhere toward the inner circle.

Besides the standard active contour parameters (mainly weights of the energies terms), the proposed technique adds one more: elevation h of the snake evolution surface above the image (Figure 2). In spite of complicating the whole model it can be also used to steer the snake scope of view. The higher is snake lifted, the farther it can look beyond smaller local obstacles. This feature is visualized in Figure 4, where the inner, darker circle plays role of an object to segment and the outside one, brigther is an obstacle (Figure 4a). If the elevation h of the snake is small ($h = 1$), the electric



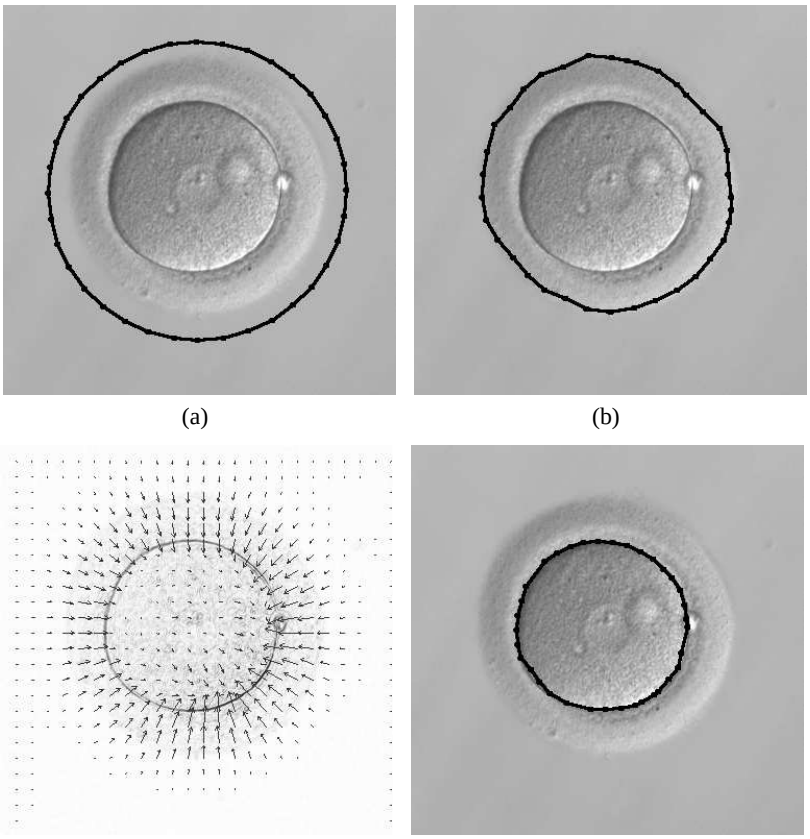
(a) (b)

Fig. 5. Microscopic images of oocytes in different stage of their evolution.

field from the outside of this configuration leads the active contour only to the outer shape (Figure 4b shows only x and y components of the 3D field) - it can not cross the outer circle because the field in its inside points toward it and not toward the inner, stronger circle. When the elevations is higher ($h = 9$, Figure 4c), the electric field, even between circles, points toward the inner, stronger one.

4. Experiments

The proposed method, implemented in Java, was verified on real microscopic images of oocytes. These images are especially well suited to show its advantages since they

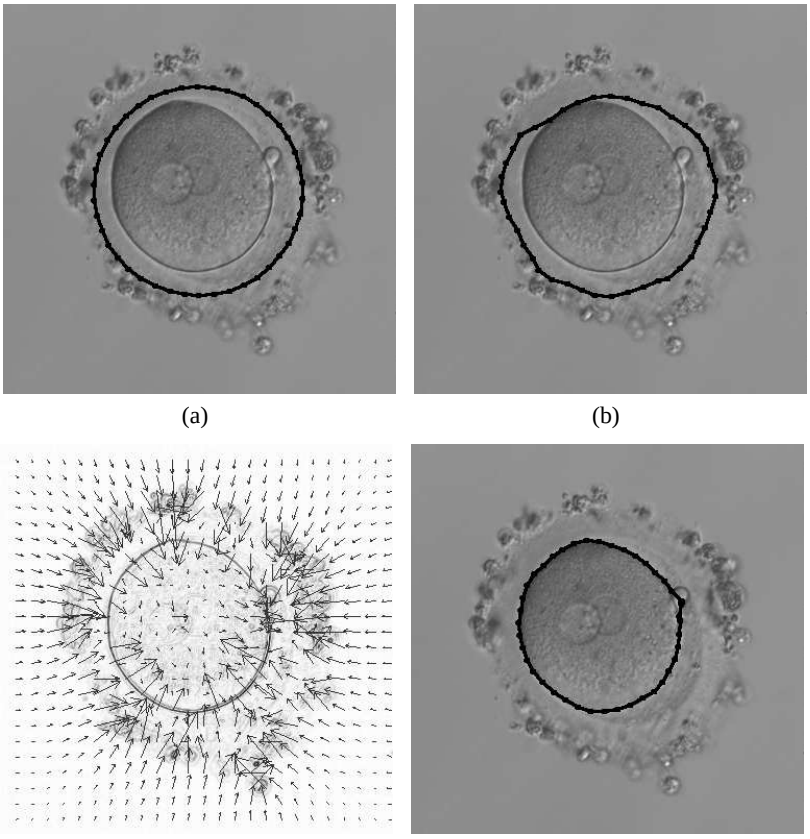


(a) (b)

(c) (d)

Fig. 6. Experiments with image 5a: (a) - initial contour (in black); (b) - result of the classic snake segmentation (in black); (c) - "electrostatic" force vector field superimposed on the image gradient; (d) - result of the "electric" snake (in black).

they contain one well distinguished shape surrounded by less distinct structures. In other works the oocytes (and other reproductive cells in general) were segmented using various techniques, including deformables models. Pastorinho et al. [9] used two deformable models: Active Shape Model to detect the nucleus and GVF snake to detect the whole zooplankton gonad. Alén et al. [1] applied and compared two techniques to segment and count fish oocytes form histological images: region growing and edge-based one, where unstructured detected edges are modelled with ellipses. Giusti et al. [4] segmented zygote (fertilized ovum - evolved from an oocyte) in very interesting manner, exploiting artefacts from the optical imaging to improve segmentation. The original zygote image, converted to the polar coordinated, defines a directed acyclic graph with arcs values computed using the image characteristics. The minimum-cost path in this graph traces the zygote limits. The same approach is



(a) (b)

(c) (d)

Fig. 7. Experiments with image 5b: (a) - initial contour situated between the oocyte and adjacent structures (in black); (b) - result of the classic snake segmentation (in black); (c) - “electrostatic” force vector field superimposed on the image gradient; (d) - result of the “electric” snake (in black).

also used to segment pronuclei inside the zygote. Basile et al. [2] used morphological operators in microscopic images to segment a single cell of an arbitrary shape and the Hough transform to identify its circular cytoplasm-nucleus boundary before applying a texture analysis to the segmented regions.

In the first image (Figure 5a) the oocyte is much more distinct than the surrounding structure. The initial contour was initialized outisde both the cell and the structure (Figure 6a). Despite lower intensity of that structure, the classic Kass's snake stopped its evolution on it and did not reach the cell itself (Figure 6b). The "electrostatic" force vector field (Figure 6c) gives correct direction toward the cell boundaries through adjacent structure from the outside of the cell. Thus, the "electric" snake easily crossed that structure and correctly segmented the cell (Figure 6d).

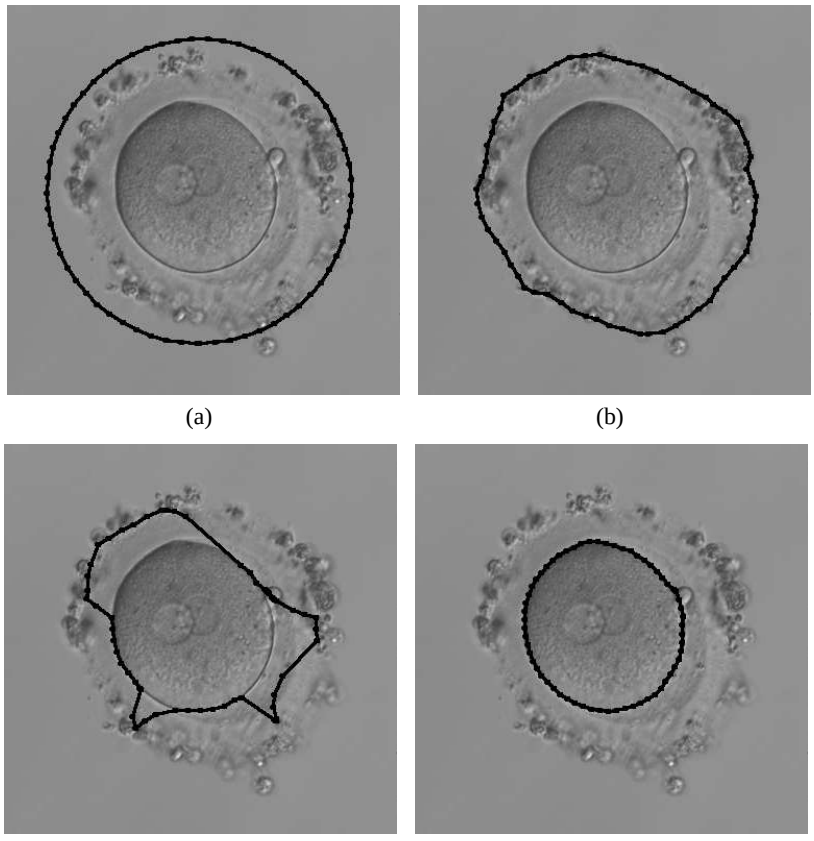
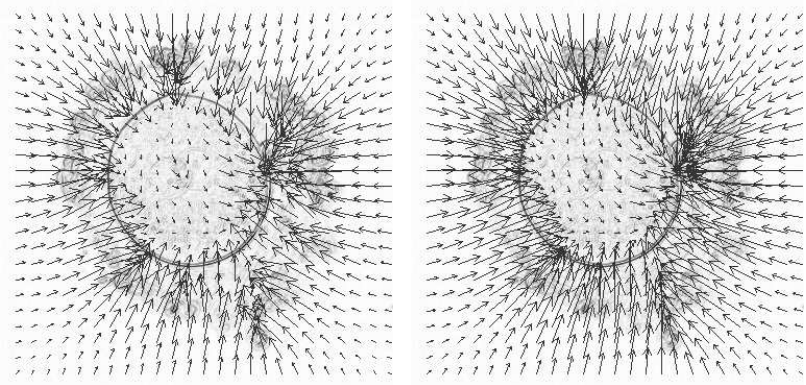


Fig. 8. Experiments with image 5b: (a) - initial contour situated outisde the oocyte and adjacent structures (in black); (b) - result of the classic snake segmentation (in black); (c) - result of the "electric" snake (in black), h=1; (d) - result of the "electric" snake (in black), h=2.

The image shown on Figure 5b seems harder to segment because of more distinct structures around the oocyte - their intensities are comparable to the oocyte intensity. Initialized between them (Figure 7a), the classic snake was attracted by the closest edges: somewhere - the oocyte, elsewhere - the adjacent structures, and sometimes, initialized on noise, did not move at all (Figure 7b). The “electrostatic” force vector field (Figure 7c) points toward the actual cell boundaries, even through intensive local obstacles (their “charge” is smaller than the cumulative “charge” of the boundaries) and the “electric” snake correctly segmented the oocyte (Figure 7d).

Sight range

For the “electric” snake its sight range can be controlled by its elevation h above the image (Figure 2). The snake situated close to the image (smaller elevation h) will be more attracted by local pixels than by distant ones, even more intensive. Going up, it will acquire ability to look beyond local pixels to see (and reach) more intensive structures in its neighbourhood. Example of this behaviour can be observed on Figure 8. The initial snake was placed outside the oocyte and adjacent structures (Figure 8a). Certainly, the classic snake failed to extract the oocyte ((Figure 8b), but so did the “electric” snake evolving on the plane too close to the image (h=1, Figure 8c). Only after elevating it higher (h=2) it was able to correctly extract the cell passing over the adjacent intensive structures (Figure 8d). However, the “electric”



(a) (b)

Fig. 9. Influence of the “electric” snake elevation on the “electrostatic” force vector field: (a) - elevation h=10; (a) - elevation h=20.

snake elevated too high will become too global - it will focus on only few image locations with the highest gradient. Only to these regions will lead the “electrostatic”

force vector field, ignoring the rest of the object to segment (Figure 9) - all snake points will move there. So the elevation value should be adjusted correctly, what unfortunatly adds a new parameter to the model.

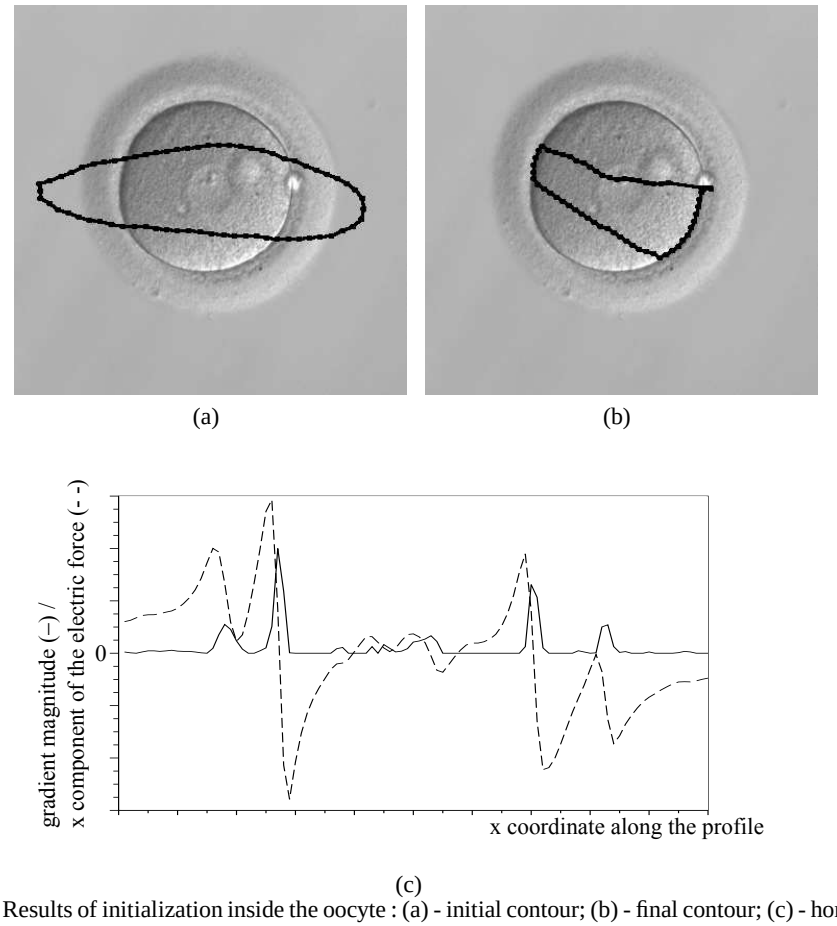


Fig. 10. Results of initialization inside the oocyte : (a) - initial contour; (b) - final contour; (c) - horizontal profiles (y=1/2 height) of gradient (solid line) and x-component of the electric force (dashed line).

Unfortunatly, the proposed model fails when initialized (even partialy) inside the oocyte (Figures 10a and 10b). Relatively high and in big quantity gradient inside the cell “hides” the border and the electric field points to the actual oocyte border only in its local neighbourhood. Figure 10c presents two profiles along horizontal line crossing the oocyte center (y=1/2 image height). The solid line marks the image gradient, with two higher inner peaks on the oocyte border an two lower outer ones on the adjacent structures border. The dashed line marks the x-component of the electric field: outside the peaks it points to the image center (positive on the left

side, negative on the right side), as well as between the higher and lower peeks pairs (between the oocyte and the adjacent structures), but inside the cell (the profil center) it is influenced by the cell interior and does not point to the border.

5. Conclusion

The proposed in this work elevated electric snake improves significantly ability of the active contour to segment objects when it is initialized in some distance from them (however - it can not be placed in an arbitrary position in the image, e.g. on one side of the object to segment). It differs from other deformables models using the electrostatics by taking into account that two electric charges can not be placed in the same position. It is done by elevating the contour above the image. Thanks to it, it can also pass over local obstacles (other structures, noise) even with comparable intensities (under condition that they are smaller, or in other words - their cumulative "charge" is smaller). This technique behaves worse when initialized inside circular objects filled with significant gradient (internal structures, noise). The GVF snake, also aiming to attract the snake by distant object edges, can not work with obstacles on its way - the "elevated electric snake" can do it, even if they are of comparable intensity.

Acknowledgmemts

First of all, I would like to thank prof. Jean-Louis Coatrieux from the LTSI laboratory, Université de Rennes 1, France, for the inspiration and initial idea of this work. The images of oocytes originate from the Bialystok Center for Reproductive Medicine "Kriobank", by courtesy of prof. Waldemar Kuczyński. The initial version of the software used in the experiments was developed in the frame of her Master thesis on the Faculty of Computer Science, Białystok University of Technology by Beata Suwała. This work was supported by the grant W/WI/1/2009 from the Bialystok University of Technology.

Bibliography

[1] Alén, S., Cernadas, E., Formella, A. Domínguez, R., Saborido-Rey, F.: Comparison of Region and Edge Segmentation Approaches to Recognize Fish Oocytes in Histological Images; ICIAR 2006, LNCS 4142, pp. 853-864, 2006.

[2] Basile, T.M.A., Caponetti, L., Castellano, G., Sforza, G.: A Texture-Based Image Processing Approach for the Description of Human Oocyte Cytoplasm; IEEE Transactions on Instrumentation and Measurement, vol. 59, no. 10, pp. 2591-2601, 2010.

[3] Cohen, L.D., Cohen, I. : Finite-Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images; IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 11, pp. 1131-1147, 1993.

[4] Giusti, A., Corani, G., Gambardella, L.M., Magli, C. and Gianaroli, L.: Lighting-Aware Segmentation of Microscopy Images for In Vitro Fertilization; ISVC 2009, LNCS 5875, pp. 576-585, 2009.

[5] Gunn, S.R., Nixon, M.S.: A Robust Snake Implementation: A Dual Active Contour IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 1, pp. 63-68,1997.

[6] Jalba. A.C., Wilkinson, M.H.F, Roerdink, J.B.T.M.: CPM: A Deformable Model for Shape Recovery and Segmentation Based on Charged Particles; IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 10, pp. 1320-1335,2004.

[7] Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models; International Journal of Computer Vision, vol. 1, no. 4, pp. 321-331, 1988.

[8] McInerney, T. Terzopoulos, D.: Deformable Models in Medical Image Analysis: A Survey; Medical Image Analysis, vol. 1, no. 2, pp. 91-108, 1996.

[9] Pastorinho, M.R., Guevara, M.A., Silva, A., Coelho, L., Morgado, F.: Development of a New Index to Evaluate Zooplanktons Gonads: An Approach Based on a Suitable Combination of Deformable Models; LNCS 3773, pp. 498-505, 2005.

[10] Xu, C., Prince, J. L.: Snakes, Shapes, and Gradient Vector Flow; IEEE Transactions on Image Processing, vol. 7, no. 3, pp. 359-369, 1998.

[11] Yang, R., Mirmehdi, M., Xie, X.: A Charged Active Contour Based on Electrostatics; ACIVS 2006, LNCS 4179, pp. 173-184, 2006.

[12] Chang, H.H., Valentino, D.J.: An Electrostatic Deformable Model for Medical Image Segmentation; Computerized Medical Imaging and Graphics, vol. 32, no. 1, pp. 22-35, 2008.

PODNIESIONY AKTYWNY KONTUR Z GLOBALNĄ ENERGIĄ OBRAZU OPARTĄ NA SILE ELEKTROSTATYCZNEJ

Streszczenie: W artykule tym zaprezentowana jest nowa modyfikacja techniki segmentacji znanej pod nazwą aktywnego konturu - węża. Polega ona na nowym sformułowaniu siły zewnętrznej opartej na sile elektrostatycznej. W istniejących pracach, w których obrazow i kontur posiadały ładunek elektryczny, omijano problem konturu zajmującego pozycję naładowanego piksela. W takiej sytuacji siła elektrostatyczna jest niezdefiniowana, gdyż odległość między ładunkami jest zerowa. Proponowany w tej pracy kontur operuje na płaszczyźnie wyniesionej ponad obraz, co sprawia, że odległość ta nigdy nie spada do zera. Metoda została zaimplementowana i zweryfikowana na rzeczywistych obrazach mikroskopowych oocytów, gdzie wykazała swoją wyższość nad klasyczną techniką węża.

Słowa kluczowe: segmentacja obrazów, aktywny kontur - wąż, siła elektrostatyczna

CHANGING PROBABILISTIC BELIEFS IN PERSUASION

Katarzyna Budzyńska[1], Magdalena Kacprzak[2]

[1]Institute of Philosophy, Cardinal Stefan Wyszynski University in Warsaw, Poland
[2]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: The aim of the paper is to extend our formal model of persuasion with an aspect of change of uncertainty interpreted probabilistically. The general goal of our research is to apply this model to design a logic and a software tool that allow for verification of persuasive multi-agent systems (MAS). To develop such a model, we analyze and then adopt the Probabilistic Dynamic Epistemic Logic introduced by B. Kooi. We show that the extensions proposed in this paper allow us to represent selected aspects of persuasion and apply the model in the resource re-allocation problem in multi-agent systems.

Keywords: persuasion, beliefs, probabilistic logic, formal verification

1. Introduction

Persuasion plays an important role in resolving different problems in multi-agent systems (MAS). It allows agents to cooperate and perform collaborative decisions and actions since it is a tool for resolution of conflicts amongst agents (see e.g. [10]).

The general goal of our research is to develop a robust model of persuasion that will allow us to describe different phenomena specific to persuasive multi-agent systems. We concentrate on application of persuasion to resolution of the resource re-allocation problem (RrAP). This is the problem of effectively reallocating the resources such that all the agents have the resources they need. The formal model that we elaborate is used to develop a formalism (Logic of Actions and Graded Beliefs $\mathcal{AG}_n$ [2]) and a software tool (the Perseus system [4]). The majority of existing work on agent persuasion considers protocols, which dictate what the possible legal next moves in persuasion are (e.g. [10]). We focus on verification of the persuasive systems for which protocols are already specified. The logic enables us to deductively test validity of formulas specifying agents participating in persuasion, as well as the

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 6, pp. 23-39, 2010.

properties of systems that can be expressed via our model. The software allows us to semantically verify satisfaction of the logic formulas, which describe properties under consideration in a given model, as well as to perform parametrical verification that enables search for answers to questions about such properties.

In this paper, we focus on enriching the formal model of persuasion with an account of changing agents' uncertainty, interpreted probabilistically (this interpretation was insightfully studied in e.g. [1,5]; in this paper, however, we do not focus on the issue of probabilistic beliefs, but on the change of such beliefs). This provides a key first step towards extension of $\mathcal{AG}_n$ into Probabilistic Logic of Actions and Graded Beliefs $\mathcal{PAG}_n$ and a further development of the Perseus system. As far as we are aware, there are no other formal or software tools that allow verification of formulas with modalities expressing updates of probabilistic beliefs induced by persuasion.

The aspect of the uncertainty change in persuasion is important when we want to examine not only the final outcome of a given persuasion, but also to track how the successive actions modify agents' uncertainty about exchanging resources at each stage of persuasion (after the first persuasive action, after the second, etc.) [3]. This allows us to check and evaluate agents' strategies and, as a result, to plan optimal ones. The $\mathcal{AG}_n$ logic enables expression of the uncertainty change in persuasion. The operator: $M!_i^{d_1,d_2}\alpha$, intuitively means that an agent i considers d_2 doxastic alternatives (i.e. possible scenarios of a current global state) and d_1 of them satisfy α. Further, the operator: $\Diamond(j:P)M!_i^{d_1,d_2}\alpha$, intuitively means that after executing actions P by agent j, agent i may believe α with degree $\frac{d_1}{d_2}$. The strength of the $\mathcal{AG}_n$ uncertainty operator is that it gives detailed information about local properties of a model we examine. For example, $M!_i^{1,2}\alpha$ provides information that i assumes that α holds in exactly one state, while for $M!_i^{2,4}\alpha$ the agent i assumes that α holds in two states. On the other hand, in pure $\mathcal{AG}_n$ it is difficult to explore the uncertainty in terms of a ratio. Suppose that we want to examine if i believes α with degree $\frac{1}{2}$. To this end, we have to verify the formulas $M!_i^{1,2}\alpha$, $M!_i^{2,4}\alpha$, $M!_i^{3,6}\alpha$ etc., since all of them describe the uncertainty ratio of $\frac{1}{2}$. A possible solution to this problem is to add the uncertainty operator interpreted probabilistically, since the probability is a natural way of expressing ratios. However, we must select a model, which would allow to describe not only the uncertainty, but also its change induced by persuasion.

In this paper, we examine a well-known framework proposed by Kooi [8]: Probabilistic Dynamic Epistemic Logic (PDEL). There are other logics that represent the change of degrees of beliefs, however, they do not refer to the probability in a direct manner. One such proposal is van Ditmarsch's model of graded beliefs

within Dynamic Epistemic Logic for Belief Revision [12]. In this framework, degrees of beliefs are related to agent's preferences which in turn correspond to a set of accessibility relations assigned to this agent. The other formalism is proposed by Laverny and Lang [9]. They define a graded version of the doxastic logic KD45 as the basis for the definition of belief-based programs and study the way the agents belief state is maintained when executing such programs.

Since our aim is to represent the change of probabilistic beliefs in persuasive MAS, the PDEL framework seems to be very promising. However, it has some serious limitations when directly applied to describe persuasion. A key contribution of this paper is that we not only identify those limitations but we also propose modifications that allow to avoid them.

The paper is organized as follows. Section 2 gives an overview of two frameworks that we explore in this paper: RrAP and PDEL. In Section 3, we propose the modifications to PDEL which are necessary if we want to apply it to the model of persuasion. In Section 4, we show how expressible the extended model is with respect to persuasion used in RrAP.

2. Background

In this section, we give a brief overview of the frameworks that we adopt to extend our model of persuasion. Moreover, we introduce an example that we use to illustrate our analysis in the next sections.

2.1 Resource re-allocation problem (RrAP)

The resource re-allocation problem can be intuitively described as the process of re-distributing a number of items (resources) amongst a number of agents. During the resource re-allocation process, agents may disagree in some respects. Persuasion can provide a solution to such problems, since it allows resolution of conflicts. As a result, persuasion enhances the exchange of resources. Observe that in RrAP scenarios, persuasion may be accompanied by negotiations (see e.g. [7] for a framework enriching RrAP with negotiations), since conflict of opinion and conflict of interests often coexists.[1] However, for the clarity of the paper we limit our considerations to persuasion.

Recall that the general aim of our research is to build a logic and a software tool which will allow to verify the persuasive MAS. In this manner, we will be able to

[1] See [13] for details of a specification for persuasion and negotiation.

examine agents' strategies for exchanging resources and evaluate the correctness and effectiveness of applied algorithms for persuasion.

Consider the simplified example of RrAP. Assume a system with two agents: John and Ann. Both agents know that in the world they exist there are five keys, two of which are needed to open a safe. Ann knows identifiers of the appropriate keys and knows that John owns them. Therefore she tries to exchange the keys persuading John that after the exchange he will have the appropriate keys. John does not know which keys open the safe. Does he consent to the exchange?

Suppose that keys are marked with identifiers: 1, 2, 3, 4, 5. At the beginning Ann has the keys with identifiers 1, 2, 4, while John has keys 3 and 5. The keys which open the safe are also 3 and 5. Ann offers to John an exchange of key 2 for key 3. She justifies an action's necessity with a statement, which is obviously false, that in order to open the safe one odd and one even key is necessary. The John's response is strongly determined by his attitude to Ann. If John trusts Ann and knows that she is a reliable source of information, then he will agree to the keys' exchange and will believe that the pair of odd/even keys opens the safe. If John does not trust Ann, then he can respond in different ways (again, for simplicity we assume only two possible responses). The one manner is that John agrees to the keys' exchange, but he doesn't reset his beliefs. The other way determines that John assumes that Ann is not a credible source of information. Therefore, John does not accept the exchange and begins to believe that the safe may be opened only with a pair of odd/odd or even/even keys. As a result in the next sections we examine three cases:

C1 John trusts Ann,
C2 John does not trust Ann and is indifferent to her,
C3 John does not trust Ann and believes the opposite of what she says.

2.2 Probabilistic Dynamic Epistemic Logic (PDEL)

In this section we show the syntax and semantics of PDEL introduced by Kooi [8]. Let $Agt = \{1,\ldots,n\}$ be a finite set of names of *agents* and V_0 be a countable set of *propositional variables*.

The set of all well-formed expressions of PDEL is given by the following Backus-Naur form (BNF):

$$\alpha ::= p|\neg\alpha|\alpha\wedge\alpha|\Box_i\alpha|[\alpha_1]\alpha_2|q_1\mathbf{P}_i(\alpha_1)+\ldots+q_k\mathbf{P}_i(\alpha_k)\geq q,$$

where $\alpha_1,\ldots,\alpha_k$ are formulas, $p\in V_0$, $i\in Agt$, and $q_1,\ldots,q_k$ and q are rationals. For $q_1\mathbf{P}_i(\alpha_1)+\ldots+q_k\mathbf{P}_i(\alpha_k)\geq q$, the abbreviation $\sum_{j=1}^{k} q_j\mathbf{P}_i(\alpha_j)\geq q$ is used. Formulas

$\mathbf{P}_i(\alpha) = q$, $\mathbf{P}_i(\alpha) < q$, $\mathbf{P}_i(\alpha) \leq q$, $\mathbf{P}_i(\alpha) > q$ are defined from $\mathbf{P}_i(\alpha) \geq q$ in the classical way, e.g. $(\mathbf{P}_i(\alpha) \leq q)$ for $(-\mathbf{P}_i(\alpha) \geq -q)$ or $\mathbf{P}_i(\alpha) = q$ for $(\mathbf{P}_i(\alpha) \geq q) \wedge (\mathbf{P}_i(\alpha) \leq q)$.

The non-graded belief formula, $\Box_i\alpha$, says that i believes that α. The probabilistic belief formula, $\mathbf{P}_i(\alpha) \geq q$, means that the probability, i assigns to α, is greater than or equal to q. A formula for updates, $[\alpha_1]\alpha_2$, says that α_2 is the case, after everyone simultaneously and commonly learns that α_1 is the case.

By a probabilistic epistemic model we mean a Kripke structure $\mathcal{M} = (S, R, v, P)$ where

- S is a non-empty set of states (possible worlds),
- $R : Agt \longrightarrow 2^{S \times S}$ assigns to each agent an accessibility relation,
- $v : V_0 \longrightarrow 2^S$ is a valuation function,
- P assigns a probability function to each agent at each state such that its domain is a non-empty subset of S
  $P : (Agt \times S) \longrightarrow (S \rightharpoonup [0,1])$ such that
  $\forall i \in Agt \forall s \in S \sum_{s' \in dom(P(i,s))} P(i,s)(s') = 1$,
  where $\rightharpoonup$ means that it is a partial function, i.e., some states may not be in the domain of the function.

The semantics of formulas of PDEL are defined by two interdependent definitions with respect to a Kripke structure $\mathcal{M}$. The first definition gives the semantics for the PDEL language, and the second, for updates.

Definition 1 For a given structure $\mathcal{M} = (S, R, v, P)$ and a given state $s \in S$ the Boolean value of the formula α is denoted by $\mathcal{M}, s \models \alpha$ and is defined inductively as follows:
$\mathcal{M}, s \models p$ iff $s \in v(p)$, for $p \in V_0$,
$\mathcal{M}, s \models \neg\alpha$ iff $\mathcal{M}, s \not\models \alpha$,
$\mathcal{M}, s \models \alpha \wedge \beta$ iff $\mathcal{M}, s \models \alpha$ and $\mathcal{M}, s \models \beta$,
$\mathcal{M}, s \models \Box_i\alpha$ iff $\mathcal{M}, s \models \alpha$ for all s' such that $(s, s') \in R(i)$,
$\mathcal{M}, s \models [\alpha]\beta$ iff $(\mathcal{M}_\alpha, s_\alpha) \models \beta$ (see Definition 2),
$\mathcal{M}, s \models \sum_{j=1}^{k} q_j \, \mathbf{P}_i(\alpha_j) \geq q$ iff $\sum_{j=1}^{k} q_j P(i,s)(\alpha_j) \geq q$,
where $P(i,s)(\alpha_j) = P(i,s)(\{s' \in dom(P(i,s)) | \mathcal{M}, s' \models \alpha_j\})$.

Definition 2 Let a model $\mathcal{M} = (S, R, v, P)$ and a state $s \in S$ be given. The updated model $\mathcal{M}_\alpha = (S_\alpha, R_\alpha, v_\alpha, P_\alpha)$ is defined as follows:
- $S_\alpha = S$,
- $R_\alpha(i) = \{(s, s') | \ (s, s') \in R(i)$ and $\mathcal{M}, s' \models \alpha\}$,
- $v_\alpha = v$,
- $dom(P_\alpha(i,s)) = dom(P(i,s))$ if $P(i,s)(\alpha) = 0$ and $dom(P_\alpha(i,s)) = \{s \in$

$dom(P(i,s)) : (\mathcal{M},s) \models \alpha\}$ otherwise,
- $P_\alpha(i,s)(s') = P(i,s)(s')$ if $P(i,s)(\alpha) = 0$ and $P_\alpha(i,s)(s') = \frac{P(i,s)(s')}{P(i,s)(\alpha)}$ otherwise.

The public announcement α updates the model by changing the accessibility relations and probability functions. The only states that remain accessible for each agent are the states where α holds. The probability functions work in a similar way. Their domains become limited to the states where α holds.

2.3 PDEL in RrAP-example

Let us analyze the initial step in the persuasion dialogue described in Section 2.1 with the use of PDEL. We need to define a probabilistic epistemic model $\mathcal{M}$. Assume that $Agt = \{John, Ann\}$ and $V_0 = \{p, even\_odd\}$, where p means that John has the correct set of keys, which enables him to open the safe and *even_odd* means that the combination of the pair of one even and one odd key opens the safe. The set of states is the set $S = \{(A,J,C) : A,J,C \subseteq \{1,2,3,4,5\},\ |C| = 2,\ A \cap J = \emptyset,\ A \cup J = \{1,2,3,4,5\}\}$. Thus a state $s = (A,J,C) \in S$ consists of three sets. The first one is a set of Ann's keys. The second is the set of John's keys. The intersection of the sets A and J is the empty set because the resources can not be shared. The union of these sets equals $\{1,2,3,4,5\}$ because Ann and John own all accessible resources. The third set, C, is a set of the keys which open the safe. Cardinality of C equals 2 since there are exactly two correct keys.

In this model there are two propositions: p and *even_odd*. Proposition p is true in every state in which the set of keys opening the safe is a subset of the set of keys owned by John, i.e., $v(p) = \{s \in S : s = (A,J,C)$ and $C \subseteq J\}$. Proposition *even_odd* is true in states in which even/odd combination of keys opens the safe, i.e., $v(even\_odd) = \{s \in S : s = (A,J,C)$ and $C = \{1,2\}$ *or* $C = \{1,4\}$ *or* $C = \{2,3\}$ *or* $C = \{2,5\}$ *or* $C = \{3,4\}$ *or* $C = \{4,5\}\}$.

Moreover assume that Ann has all the information about the actual state, i.e., when she is at state s she knows that she is at this state. As a result, her accessibility relation is defined as follows: $R(Ann) = \{(s,s') \in S^2 : s = s'\}$. John knows the keys he has and knows that Ann has the other keys, so his accessibility relation is $R(John) = \{(s,s') \in S^2 : s = (A,J,C),\ s' = (A',J',C'),\ J' = J,\ A' = A\}$.

Furthermore, say that the probability function P is as follows: $P(i,s)(s') = \frac{1}{|dom(P(i,s))|}$ for every $i \in Agt$, $s,s' \in S$ where $dom(P(i,s)) = \{s'' \in S : (s,s'') \in R(i)\}$. Notice that we propose to define the probability function in a classical way, i.e. we assume that $P(i,s)(s')$ is the quotient of 1 (one state s') and the number of all elements belonging to the domain of probability. This means that every state s' accessible

from s has assigned the same probability. Furthermore, we assume that the domain of probability function is a set of all accessible states. In this case there are no reasons to separate these sets. For example, if at state s John considers 10 accessible states s' then $P(John,s)(s') = \frac{1}{10}$ for every state s'.

At the beginning Ann has the keys 1, 2, and 4, John has the keys 3 and 5 and the same keys open the safe. So the initial state is $s_0 = (\{1,2,4\},\{3,5\},\{3,5\})$. John's accessibility relation and probability function for s_0 are depicted in Fig. 1.

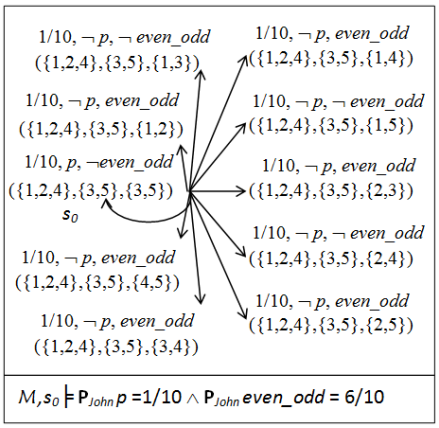


Fig. 1. John's accessibility relation and probability function before persuasion.

3. Adaptation of PDEL to persuasion model

In this section we analyze the limitations of PDEL with respect to representing the persuasion in RrAP. Moreover, we propose modifications that allow those limitations to be overcome.

In order to study a persuasive situation from the example we need to express the issues related to three stages of the persuasion:

– before the persuasion: John's attitude to the statement p – "John has the good (opening the safe) couple of keys". In other words, how strongly does he believe it to be true? Formally we can ask: to what degree does John believe that p holds in the initial state s_0?

– John's attitude to Ann's persuasive actions (e.g. proposal of the keys' exchange). In what manner might he react? What can he do and what will the result of John's behavior be?
– after the persuasion: John's attitude to the statement p. Does the degree of his belief change? If yes, how big a change is it?

The issues from the initial and the final stage can be successfully described in PDEL. Recall that John does not know the identifiers of keys which open the safe. Therefore at the beginning he considers all the possibilities (see Fig. 1). Since there are 10 possible situations and in only one of them p is true, in John's opinion the probability that in actual state he has good keys is $\frac{1}{10}$. Formally:

$$\mathcal{M}, s_0 \models \mathbf{P}_{John}(p) = \frac{1}{10}.$$

Similarly, we could compute the probability which John assigns to p when the persuasion is finished. However, we must first know and represent what happened in the intermediate stage of the persuasion. This section discusses the problems that we encounter when we want to express the issues belonging to that stage.

3.1 Trusting the persuader

Say that the first action that Ann performs is a public announcement that p is true. According to the PDEL definition of the satisfiability relation, it holds that $\mathcal{M}, s \models \mathbf{P}_{John}(p) = 1$ for a state s reachable from s_0 after execution of Ann's action. Observe that in PDEL semantics, the outcome of an action is not related to the performer of the action. That is, John's reaction will be exactly the same regardless of whether Ann or someone else says that p. Similarly, it is impossible to make John's behavior dependent on his attitude to Ann.

In agents' interactions (such as persuasions or negotiations), the reputation of the agent who performs the action can influence possibility of the action's success. In MAS, this problem is studied within the Reputation Management framework (see e.g. [11,14]). Say that an agent knows a persuader, since they exchanged resources before. If the agent evaluates those exchanges as beneficial and fair, then he will trust the persuader and be easily persuaded the next time they are going to exchange resources. On the other hand, if an announcement is executed by an agent which is unknown to other agents, then it may be disregarded. The first limitation of the PDEL expressivity is:

L1 The success of persuasion cannot be affected by reputation of persuader.

To avoid the limitation **L1**, we need to label every action with an agent who performs it. However this is not sufficient on its own. First of all we need to be able to express agents' attitudes to each other, i.e., which agent is perceived as credible by which agent. Therefore, we need to add a trust function T to the model, so now $\mathcal{M}^T = (S, R, v, P, T)$. A trust function assigns to every pair of agents one of the values 0, $\frac{1}{2}$, 1, i.e.,

$$T : Agt \times Agt \to \{0, \frac{1}{2}, 1\}.$$

With respect to the three cases from the example, the interpretation of T can be as follows:

C1 if $T(John, Ann) = 1$ then John trusts Ann and accepts everything she says,
C2 if $T(John, Ann) = \frac{1}{2}$ then John is indifferent to Ann, and as a result Ann's announcement does not influence John's beliefs,
C3 if $T(John, Ann) = 0$ then John does not trust Ann and what is more he is sure that she always tells lies.

In future work we plan to extend this approach and introduce more trust degrees and allow agents to adopt various attitudes to each other. In order to do this we intend to adopt the well-known solution from the Reputation Management framework proposed by Yu and Singh [14].

3.2 Public announcement as argument

Recall that in PDEL after Ann's announcement that p is true, it holds that $\mathcal{M}, s \models \mathbf{P}_{John}(p) = 1$. It means that after the action of announcing p, John *must* believe that p. Thereby we deprive John of deciding whether Ann is right or not. In persuasive scenarios it is a strong limitation, since it assumes that an audience will believe everything a persuader says. The only exception is when the audience believes the persuader's claim with probability 0. In other words it is impossible to express within this framework reactions of indifference (i.e. an audience is neutral with respect to a persuader) and other reactions of distrust (e.g., maximal distrust, i.e. when before a persuasion dialogue an audience believes p and – after the announcement that p – he begins to believe $\neg p$). So, the second limitation is:

L2 Audience must believe everything that a persuader claims, unless the claim is believed by audience with probability 0.

In order to avoid the limitation **L2**, the syntax of formula $[\alpha]\beta$ can be exchanged with $[j : \alpha]\beta$ where $j \in Agt$. Then

$$\mathcal{M}^T, s \models [j:\alpha]\beta \text{ iff } \mathcal{M}^T_{j,\alpha}, s_{j,\alpha} \models \beta$$

where $\mathcal{M}^T_{j,\alpha} = (S_{j,\alpha}; R_{j,\alpha}; v_{j,\alpha}; P_{j,\alpha}; T_{j,\alpha})$ is an updated model such that:

- if $T(i,j) = 1$ then $\mathcal{M}^T_{j,\alpha} = \mathcal{M}^T_{\alpha}$,
- if $T(i,j) = \frac{1}{2}$ then $\mathcal{M}^T_{j,\alpha} = \mathcal{M}^T$,
- if $T(i,j) = 0$ then $\mathcal{M}^T_{j,\alpha} = \mathcal{M}^T_{\neg\alpha}$,

where $\mathcal{M}^T_{\alpha}$ is a model $\mathcal{M}_{\alpha}$ (see Definition 2) extended with the trust function T.

In the running example:

C1 if $T(John, Ann) = 1$ then

$$\mathcal{M}^T, s_0 \models [Ann : even\_odd](\mathbf{P}_{John}(even\_odd) = 1),$$

i.e., if John trusts Ann then he agrees with everything she says (see Case 1 in Fig. 2 for John's probability function),

C2 if $T(John, Ann) = \frac{1}{2}$ then

$$\mathcal{M}^T, s_0 \models ([Ann : even\_odd]\mathbf{P}_{John}(even\_odd) = \frac{6}{10})$$

i.e., if John is indifferent to Ann then he does not change his beliefs (his probability function is the same as before the announcement – see Fig 1),

C3 if $T(John, Ann) = 0$ then

$$\mathcal{M}^T, s_0 \models [Ann : even\_odd](\mathbf{P}_{John}(even\_odd) = 0)$$

and

$$\mathcal{M}^T, s_0 \models [Ann : even\_odd](\mathbf{P}_{John}(\neg even\_odd) = 1),$$

i.e., if John does not trust Ann then he adopts the opposite of what she says (see Case 3 in Fig. 2).

3.3 Unpersuadable audience

Observe that, according to the semantics adapted from PDEL, for the formula $[j:\alpha]\beta$ it holds:

- if $\mathcal{M}^T, s \models (\mathbf{P}_i\beta = 1)$ then $\mathcal{M}^T, s \models [j:\alpha](\mathbf{P}_i\beta = 1)$ and
- if $\mathcal{M}^T, s \models (\mathbf{P}_i\beta = 0)$ then $\mathcal{M}^T, s \models [j:\alpha](\mathbf{P}_i\beta = 0)$,

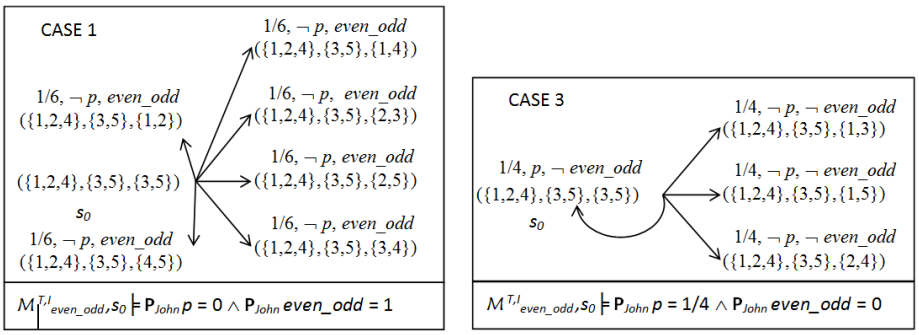


Fig. 2. John's probability function after Ann's public announcement that *even_odd* – cases 1 and 3.

for any formula α. Intuitively it means that if an agent i is sure that β is true then there is no way (no action which can be executed) to convince him that β is true with probability less than 1. A similar situation occurs when i is absolutely certain that β is false, i.e., if the probability of β is 0. In the context of persuasion scenarios, it is a serious limitation. For example, if an agent is sure that he needs some resources, then the other agent has no chance to persuade him to exchange it. So, the next problem with the PDEL expressivity is:

L3 A persuader has no chance to influence an audience about a claim in a case where it is absolutely sure that the claim is true or false.

The limitation **L3** is a consequence of an assumption that $dom(P_\alpha(i,s)) \subseteq dom(P(i,s))$. For instance, suppose that at state s_0 Ann says *even_odd*. Then $\mathcal{M}^T_{even\_odd}, s_0 \models \mathbf{P}_{John}(\neg even\_odd) = 0$. Next Ann says $\neg even\_odd$. Now, since John believes $\neg even\_odd$ with probability 0, both probability function and its domain are not changed (see Definition 2). As a result, John's beliefs remain unchanged, i.e. $\mathcal{M}^T_{\neg even\_odd}, s_0 \models \mathbf{P}_{John}(\neg even\_odd) = 0$.

In persuasion we must often deal with updates with information that has probability zero. The approach given in PDEL is simply to ignore the information. This is to ensure that one does not divide by zero. Moreover, the logic cannot deal well with updates with inconsistent information. Typically, the accessibility relation become empty after an inconsistent update. There is no philosophical reason for such a choice. However, this makes the system and its completeness proof relatively simple. There are some more advanced approaches in probability theory for updating sentences with probability 0 (see [6] for an overview). We propose to cope with this limitations in the way described below.

In order to resolve the problem **L3**, we can allow that after an update an agent may take into consideration a state which was not considered before. Formally, we assume that there exists a state s such that $s \in dom(P_\alpha(i,s))$ and $s \notin dom(P(i,s))$, i.e., $dom(P_\alpha(i,s)) \not\subseteq dom(P(i,s))$. Of course in the general case it may be a big challenge to establish which states with what probabilities can be added to a domain of function P. However, in some concrete applications it seems to be easy and natural. In our example it can work as follows. Let

$$dom(P(John,s_0)) = \{(\{1,2,4\},\{3,5\},C) \in S : \\ C \subseteq \{1,2,3,4,5\} \text{ and } |C| = 2\}$$

and in the updated model $\mathcal{M}^T_{even\_odd}$

$$dom(P_{even\_odd}(John,s_0)) = \\ \{(\{1,2,4\},\{3,5\},\{n_1,n_2\}) \in S : n_1 \text{ is an even and } n_2 \text{ is an odd number}\}.$$

Hence $\mathcal{M}^T_{even\_odd}, s_0 \models \mathbf{P}_{John}(\neg even\_odd) = 0$. Next if Ann says $\neg even\_odd$ then

$$dom(P_{\neg even\_odd}(John,s_0)) = \\ \{(\{1,2,4\},\{3,5\},\{n_1,n_2\}) : n_1,n_2 \text{ are even or odd numbers}\}$$

and John's beliefs are changed, i.e. $\mathcal{M}^T_{\neg even\_odd}, s_0 \models \mathbf{P}_{John}(\neg even\_odd) = 1$.

3.4 Nonverbal actions during the persuasion process

In persuasion, the proponent aims to change beliefs of the audience. The persuasion process begins with the first action of the proponent which has a given aim, and finishes with the last action with this aim. Yet, during persuasion agents can perform actions (with or without persuasive aims) which change not only beliefs of the audience, but also the environment of the agents. For example, during their persuasion dialogue, John and Ann can exchange the keys (i.e. before and after the action of exchange Ann performs some persuasive actions). Observe that this action can change the circumstances in which the persuasion process will continue. That is, the new circumstances can be favorable to Ann and her next persuasive action can make John believe her claim, while in the old circumstances such an effect may not be obtained. In other words, during the persuasion process some nonverbal actions influencing the environment can change a course (an effect) of persuasive actions performed after this nonverbal action.

The pure PDEL allows expression only of the actions of public announcement which do not influence the beliefs of an agent (a doxastic relation), no those which influence the environment (a state). In particular, it is not possible to describe such

situations in RrAP in which before a persuasion agents have some resources and after it they have other resources. Moreover, verbal actions (like public announcement) do not change values of propositions. This means that in this framework persuasive actions can not change the actual world (only the probabilistic beliefs may be modified). Thus the exchange of resources which is necessary to solve RrAP can not be described. Therefore the last limitation is:

L4 There is no possibility of expressing actions other than public announcements.

Since nonverbal actions are often applied in persuasion, we need to resolve the limitation **L4**. To this end, we propose to combine PDEL with our logic $\mathcal{AG}_n$. The strength of $\mathcal{AG}_n$ is that it is already adjusted to express persuasion. Assume a set Π_{nv} of nonverbal actions and enrich the model $\mathcal{M}^T$ with interpretation I of this actions where

$$I : \Pi_{nv} \to (Agt \to 2^{S\times S}).$$

Now we have a new model $\mathcal{M}^{T,I} = (S,R,v,P,T,I)$. After the execution of an action $a \in \Pi_{nv}$, a system reaches a state in which not only new accessibility relations can be assigned to agents but also new logical values may be assigned to propositions. Next, let formula $[j : a]\beta$ for $j \in Agt$ and $a \in \Pi_{nv}$ says that after the execution of action a by agent j the condition β holds. The semantics of this formula is as follows:

$$\mathcal{M}^{T,I}, s \models [j : a]\beta \text{ iff } \mathcal{M}^{T,I}, s' \models \beta$$
$$\text{for every state } s' \text{ such that } (s,s') \in I(a)(j).$$

In our example Ann intends to exchange the key 2 with the key 3. Let *ex* stand for the action of the keys exchange. The interpretation of *ex* is given below. If $s = (A,J,C)$ is a state such that $2 \in A$ and $3 \in J$ then for every state $s' = (A',J',C')$ it holds

$$(s,s') \in I(ex)(Ann) \text{ iff } A' = A\backslash\{2\} \cup \{3\},\ J' = J\backslash\{3\} \cup \{2\}, \text{ and } C' = C.$$

Otherwise $(s,s') \in I(ex)(Ann)$ iff $s = s'$.

4. Expessivity of extended model persuasion

Now we are ready to analyze the running example. At the beginning at state s_0 John assigns to the proposition p (the statement "John has the good keys") probability $\frac{1}{10}$ (see Fig. 1):

$$\mathcal{M}^{T,I}, s_0 \models (\mathbf{P}_{John} p = \frac{1}{10}).$$

Next Ann says that the combination of one even and one odd key opens the safe. Again consider the three cases:

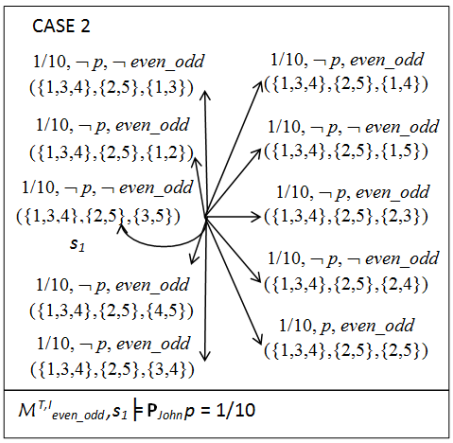


Fig. 3. John's probability function after the execution of the action *ex* – case 2.

C1 John trusts Ann and thinks that she is right. As a result, he removes from the domain of the probability function all states in which p is satisfied and thus assigns to p probability 0:

$$\mathcal{M}^{T,I}, s_0 \models [Ann : even\_odd](\mathbf{P}_{John} p = 0)$$

since (see Fig. 2): $\mathcal{M}^{T,I}_{even\_odd}, s_0 \models (\mathbf{P}_{John} p = 0)$. Then Ann and John exchange keys 2 and 3. It is easy to compute that after this action the probability of p will be $\frac{1}{6}$:

$$\mathcal{M}^{T,I}, s_0 \models [Ann : even\_odd][Ann : ex](\mathbf{P}_{John} p = \frac{1}{6})$$

since (see Fig. 4): $\mathcal{M}^{T,I}_{even\_odd}, s_1 \models (\mathbf{P}_{John} p = \frac{1}{6})$ where s_1 is a state reachable after the execution of the action *ex*. Therefore the replacement causes the growth of probability which John assigns to the statement that he has right keys.

C2 John preserves a neutral position with respect to Ann. Hence, he assigns to p the probability as at the start of persuasion dialogue:

$$\mathcal{M}^{T,I}, s_0 \models [Ann : even\_odd](\mathbf{P}_{John} p = \frac{1}{10}).$$

Then Ann and John exchange keys 2 and 3. After the action the probability of p remains unchanged:

$$\mathcal{M}^{T,I}, s_0 \models [Ann : even\_odd][Ann : ex](\mathbf{P}_{John} p = \frac{1}{10})$$

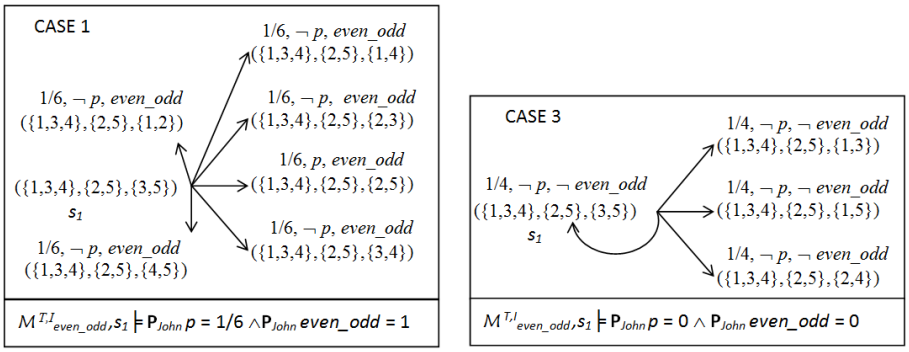


Fig. 4. John's probability function after the execution of the action *ex* – cases 1 and 3.

since (see Fig. 3): $\mathcal{M}^{T,I}_{even\_odd}, s_1 \models (\mathbf{P}_{John} p = \frac{1}{10})$. As a result, such an activity does not modify John's beliefs about whether he has right keys.

C3 John thinks that Ann lies. As a result he assigns to p degree $\frac{1}{4}$:

$$\mathcal{M}^{T,I}, s_0 \models [Ann : even\_odd](\mathbf{P}_{John} p = \frac{1}{4})$$

since (see Fig. 2) $\mathcal{M}^{T,I}_{even\_odd}, s_0 \models (\mathbf{P}_{John} p = \frac{1}{4})$.

Then Ann and John exchange keys 2 and 3. After the action the probability of p will be 0:

$$\mathcal{M}^{T,I}, s_0 \models [Ann : even\_odd][Ann : ex](\mathbf{P}_{John} p = 0)$$

since (see Fig. 4): $\mathcal{M}^{T,I}_{even\_odd}, s_1 \models (\mathbf{P}_{John} p = 0)$. For that reason the exchange results in John believing that he has the right keys with degree 0.

5. Conclusions

PDEL is a powerful tool which can be used for reasoning about update of an agent's uncertainty. In this paper, we analyze the possibility of applying this framework to represent change of probabilistic beliefs induced by persuasion and executed in the resource re-allocation scenarios. First, we indicate some limitations of PDEL, when it is directly interpreted in a persuasive MAS. Next, we propose how to avoid those limitations such that the advantages of the PDEL tool could be fully exploited to represent the persuasion in RrAP.

We discuss four limitations: **L1** requires that the success of persuasion cannot be affected by the reputation of the persuader, **L2** limits the audience to believe

everything that the persuader says, with the exception when **L3** holds, i.e., when the audience is absolutely sure that the claim is true or false, in which case it is impossible to change the audience's mind, and **L4** does not allow expression of actions other than public announcements. In order to solve **L1**, we apply elements of the Reputation Management framework. For **L2** we propose changing the syntax and semantics of the PDEL formulas which describe public announcements. For **L3** we suggest changing the specification for the domain of the probability function. Finally, to resolve **L4** we propose using elements of the $\mathcal{AG}_n$ logic.

The adaptation of PDEL to the persuasion model enriches the model expressivity with respect to change of probabilistic beliefs induced by persuasion. This provides a key first step towards creating $\mathcal{PAG}_n$, i.e., the Probabilistic Logic of Actions and Graded Beliefs, and extending the Perseus system designed to verify persuasive MAS. Moreover, in future work we are going to enrich the aspect of persuader's reputation e.g. by adding actions modifying trust. Such actions change neither accessibility relations nor values of propositions.

References

[1] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge*, MIT Press, Cambridge, 1990.

[2] K. Budzyńska and M. Kacprzak, 'A logic for reasoning about persuasion', *Fundamenta Informaticae*, **85**, 51–65, (2008).

[3] K. Budzyńska, M. Kacprzak, and P. Rembelski, 'Modeling persuasiveness: change of uncertainty through agents' interactions', in *Frontiers in Artificial Intelligence and Applications*. IOS Press, (2008).

[4] K. Budzynska, M. Kacprzak, and P. Rembelski, 'Perseus. software for analyzing persuasion process.', *Fundamenta Informaticae*, (91), (2009).

[5] J. Y. Halpern, 'An analysis of first-order logics of probability', *Artificial Intelligence*, **46**, 311–350, (1990).

[6] J.Y. Halpern, 'Lexicographic probability, conditional probability, and nonstandard probability', in *Proceedings of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge*, 17–30, (2001).

[7] A. Hussain and F. Toni, 'Bilateral agent negotiation with information-seeking', in *Proc. of EUMAS-2007*, (2007).

[8] B. Kooi, 'Probabilistic dynamic epistemic logic', *Journal of Logic, Language and Information*, **12**, 381?408, (2003).

[9] N. Laverny and J. Lang, 'From knowledge-based programs to graded belief-based programs part i: On-line reasoning', *Synthese*, (147), 277–321, (2005).

[10] H. Prakken, 'Formal systems for persuasion dialogue', *The Knowledge Engineering Review*, **21**, 163–188, (2006).
[11] S. D. Ramchurn, C. Mezzetti, A. Giovannucci, J. A. Rodriguez-Aguilar, R. K. Dash, and N.R. Jennings, 'Trust-based mechanisms for robust and efficient task allocation in the presence of execution uncertainty', *Journal of Artificial Intelligence Research*, (35), 119–159, (2009).
[12] H.P. van Ditmarsch, 'Prolegomena to dynamic logic for belief revision', *Knowledge, Rationality & Action (Synthese)*, **147**, 229–275, (2005).
[13] D. N. Walton and E. C. W. Krabbe, *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*, State Univ. of N.Y. Press, 1995.
[14] B. Yu and M.P. Singh, 'A social mechanism of reputation management in electronic communities', in *Proceedings of the Fourth International Workshop on Cooperative Information Agents*, (2000).

ZMIANA PROBABILISTYCZNYCH PRZEKONAŃ W PERSWAZJI

Streszczenie: Celem pracy jest rozszerzenie zaproponowanego przez nas formalnego modelu perswazji o aspekt zmiany niepewności przekonań agentów interpretowanych w teorii prawdopodobieństwa. Wzbogacony model jest podstawą do zdefiniowania logiki i zaprojektowania narzędzia, które umożliwia automatyczną weryfikację perswazyjnych systemów wieloagentowych. W celu realizacji tego zadania analizujemy i adaptujemy Probabilistyczną Dynamiczną Epistemiczną Logikę wprowadzoną przez B. Kooi. Zastosowanie zaproponowanego podejścia do analizowania wybranych aspektów perswazji omawiamy na przykładzie problemu alokacji zasobów w rozproszonych komputerowych systemach.

Słowa kluczowe: perswazja, przekonania, logika prawdopodobieństwa, formalna weryfikacja

LOAD BALANCING IN PARALLEL IMPLEMENTATION OF VASCULAR NETWORK MODELING

Krzysztof Jurczuk[1], Marek Kręţowski[1], Johanne Bézy-Wendling[2,3]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland
[2]INSERM, U642, Rennes, F-35000, France
[3]University of Rennes 1, LTSI, Rennes, F-35000, France

Abstract: In this paper, load balancing mechanisms in a parallel algorithm of vascular network development are investigated. The main attention is focused on the perfusion process (connection of new cells to vascular trees) as it is the most time demanding part of the vascular algorithm. We propose several techniques that aim at balancing load among processors, decreasing their idle time and reducing the communication overhead. The core solution is based on the centralized dynamic load balancing approach. The model behaviors are analyzed and a tradeoff between the different mechanisms is found. The proposed mechanisms are implemented on a computing cluster with the use of the message passing interface (MPI) standard. The experimental results show that the introduced improvements provide a more efficient solution and consequently further accelerate the simulation process.

Keywords: parallel algorithms, load balancing, cluster computing, computational modeling, vascular network

1. Introduction

The last decade has seen a revolution in high performance scientific computing [1]. This is mainly due to a tremendous development of parallel computers. Because of physical and economic limitations of processor frequency scaling (e.g. power consumption and consequently heat generation) both industry and science prefer to use many moderately fast processors, rather than a single high speed processing unit. Nowadays, computing clusters and multi-core/multi-processor computers are becoming widespread platforms [2]. As a results, many scientists have gained an easy access to parallel machines able to support an ever-rising demand for high-speed processing.

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 6, pp. 41-61, 2010.

In this paper, we focus on applying parallel computing to modeling and simulation in biomedical research on vascular networks. Vascular networks play a very important role in the detection process of various pathological anomalies since changes in their structure and function can be directly caused by diseases [3]. Moreover, when a contrast agent is administrated, these anatomical or functional modifications can appear in medical images. Therefore, the modeling of vascular systems can help to understand the mechanisms of dynamic image formation and support the development of methods to detect early disease indicators.

Nevertheless, one of the most important and simultaneously the most difficult challenges in model designing is to choose the level of details to include in the model [4]. A high quality vascular model has to take into account the most essential physiological and anatomical properties and to disregard those elements whose role is insignificant. Such a model should also be effective in practical cases, i.e. computational simulations must be performed in a reasonable time. Therefore, it seems to be very useful and desirable to take advantage of parallel computing in modeling of living organisms and particulary in the case of the vascular system modeling. Firstly, we are able to provide a significant increase in computational performance by splitting problem into parts that are performed by separate processors in parallel [5]. Secondly, using multiple processing units often allows us to provide a more precise solution or to solve a larger problem in a reasonable amount of time. Moreover, parallel computers are very useful when the same problem has to be evaluated multiple times, with different parameters for instance.

In parallel systems, computations are decomposed into tasks. In order to achieve an efficient solution, overheads of the parallel tasks have to be minimized [6]. One ought to strive to reduce the total amount of time some processors are idle while the others are still busy. Secondly, the amount of time spent for communication between processors has to be also minimized. These two objectives are often in conflict with each other, therefore one should find an optimal tradeoff between them and propose load balancing mechanisms able to spread the tasks evenly across the processors.

Load balancing techniques used in parallel algorithms can be broadly classified into two major categories: static and dynamic. In the former type, usually referred to as the mapping problem [7] or scheduling problem, tasks are distributed among processors before the execution of the algorithm based on a priori knowledge. Several techniques for static load balancing have been developed, e.g. round robin algorithm [1], simulated annealing (stochastic optimization algorithm) [8], [9] or real-coded genetic algorithms [10]. However, there exists a large class of applications that workloads of tasks are uneven and unpredictable and may change during the computation. Therefore, for these applications we are not able to spread the tasks

evenly across processors beforehand. In this case, dynamic load balancing (DLB) schemes are needed. In DLB, the decision on task arrangement is made during the execution of the program based on the current load status. Moreover, such an approach can be more appropriate in the case of heterogeneous parallel machines with additional sources of an external load. Due to a big and fast growing number of different dynamic load balancing techniques, we refer the reader to [11] for a detailed survey of DLB algorithms.

In our previous studies, we developed a two-level physiological model of vascularization [12], [13]. It consists of a macroscopic model able to simulate growth and pathological structural modifications of vascular network, and a microvascular model responsible for simulation of blood and contrast agent transport through capillary walls [14]. Initially, we made use of a sequential algorithm of vascular development to obtain the structure of the vascular network. The vascular development results from a progressive increasing number of cells and consequently a progressive increasing number of vessels that support blood supply for these cells. Subsequently, we introduced the basic [15] and improved [16] parallel solutions of vascular growth algorithm. These two parallel solutions were implemented on a computing cluster with the use of the message passing interface (MPI) standard [17].

In this paper, we propose mechanisms that try to achieve balanced load among processors and reduce the communication overhead in the parallel modeling of the vascular network growth. Both static and dynamic algorithms are used. We consider a centralized model, in which tasks are generated at the central scheduler (master processor) and are allocated to slave processors. Workloads of the tasks are uneven and it is impossible to estimate their execution times because each particular job has an indeterminate number of steps to reach its solution. In addition, we have to deal with a dynamically changing structure of vascular trees. We analyze various model behaviors in a parallel environment and propose a tradeoff between the different load balancing strategies in order to provide a more efficient solution and consequently to further accelerate the simulation process.

The rest of the paper is organized as follows. In the next section, the vascular model is described and sequential and both parallel algorithms of vascular network development are recalled. In section 3 the load balancing mechanisms are presented. An experimental validation of the proposed mechanisms is performed in section 4. The last section contains the conclusion and future works.

2. Model Description

In the macroscopic part of the model we can distinguish two main elements: the tissue and the vascular network. The tissue is represented by a set of Macroscopic Functional Units (MFU) that are regularly (but randomly) distributed inside the specified, three-dimensional organ shape. The vascular network is composed of vessels supplying the MFUs. The microvascular part of the model is hidden in MFUs and is responsible for the propagation of an MRI contrast agents in the tissue. The five-compartments [18] and axially distributed Blood Tissue EXchange (BTEX) [14] contrast propagation approaches were proposed.

The most important and original part of the work presented here concerns the algorithms of vascular development on macroscopic level. Therefore, in the next part of this section, the macroscopic part of the model is described in more details followed by the presentation of sequential and parallel algorithms of vascular development.

2.1 Macroscopic model

Tissue modeling A MFU is a small, fixed size part of tissue to which a class is assigned that determines most of functional/structural (rhythm of mistosis/necrosis) and physiological features (e.g. blood flow rate, blood pressure). Several classes of MFUs can be defined to differentiate functional or pathological regions of tissue (e.g. tumoral, normal). Moreover, the MFU class can be changed over time, which makes it possible to simulate the evolution of a disease (e.g. from HepatoCellular Carcinoma to necrotic tissue or from benign nodule to malignant tumor). In order to introduce more natural variability, certain parameters (such as blood flow rate) are described by defined distributions.

Vascular Network Modeling Most of model features are not linked with any specific organ. However, it is very hard to model a vascular network without any kind of specialization. In our work, the model expresses the specificity of the liver. The liver plays a major role in the metabolism and has a number of functions in the body including protein synthesis, detoxification, glycogen storage, etc. [19]. Moreover, it stands out from other vital organs by its unique organization of vascular network that consists of three vessel trees. Hepatic arteries and portal veins deliver blood to cells, whereas, the hepatic venous tree is responsible for blood transport back to the heart.

In the model, each vascular tree is composed of vessels that can divide creating bifurcations (see Fig. 1a). A vessel segment (part of vessel between two consecutive

bifurcations) is represented by an ideal, rigid tube with fixed radius, wall thickness, length and position. The geometry of capillaries is not considered in the model. These smallest vessels are hidden in the MFUs (microvascular model). According to the morphometrical investigation dealing with bigger vessels, e.g. conducted by Zamir [20], it is assumed that a single vascular structure has a form of a binary tree. In effect, anastomoses (e.g. mutual vessel intersections) that may occur particulary in pathological situations or among vessels with very small radii are not taken into account.

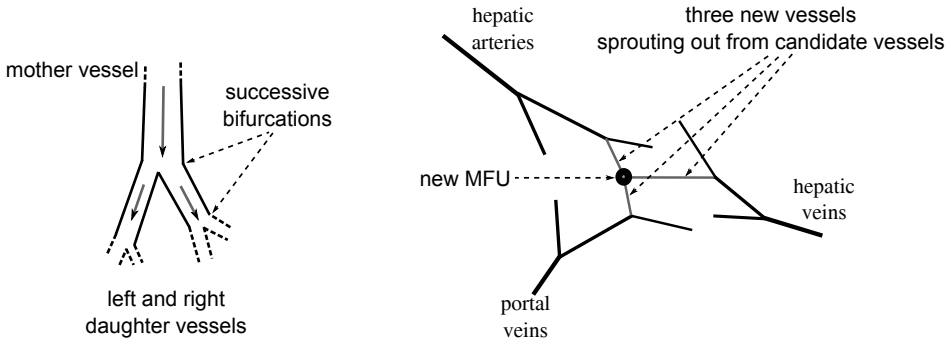


Fig. 1. Part of binary vascular trees: a) mother vessel and its two daughter vessels connected by a bifurcation, b) new MFU perfusion by three new vessels sprouting out from candidate vessels each from different vascular tree.

In the model, the blood is treated as a Newtonian fluid that is transferred from hepatic arteries and portal veins to the hepatic veins through MFUs. Its flow is modeled as a non-turbulent streamline flow in parallel layers (laminar flow) and governed by Poiseuille's law:

$$\Delta P = Q\frac{8\mu l}{\pi r^4}\ , \tag{1}$$

where l is the vessel length, r is its radius, Q is the blood flow and ΔP is the pressure difference between the two vessel extremities. Moreover, at each bifurcation the law of matter conservation has to be observed:

$$Q = Q_r + Q_l\ . \tag{2}$$

It says that the quantities of blood entering a bifurcation (blood flow in parent vessel Q) and leaving the bifurcation (blood flows in the right and left daughter branches

$Q_r + Q_l$) are equal. Another constraint deals with the decreasing vessel radii in the vascular trees when we move from proximal to distal segments of vascular network, creating/describing the relation between the mother vessel radius (r) and the radii of its two daughters (right r_r and left r_l):

$$r^\gamma = r_r^\gamma + r_l^\gamma \,, \tag{3}$$

where γ varies between 2 and 3 [21].

2.2 Sequential Vascular Network Growth Algorithm

An adult organ is obtained in a vascular development process that is modeled as a analogy to a hyperplasia process (progressive increasing number of cells [22]). The simulation starts with an organ whose size is a fraction of a mature one. After parameters' initialization, in discrete time moments (called cycles), the organ enlarges its size (growth phases). The relative positions of MFUs remain unchanged but distances between them are increased, leading to appearance of empty spaces. Subsequently, these spaces are filled by new MFUs in consecutive subcycles. In each subcycle, each MFU can divide and give birth to a new MFU of the same class (mitosis process) or die (necrosis process). Probabilities of mitosis and necrosis are sensitive to the time and they decrease exponentially with the age of the MFU. New cycle starts only when the current organ shape is totally filled by MFUs. The increasing number of MFUs induces the development of a vascular network which is responsible for the blood delivery.

New MFUs that appear during the mitosis process are initially ischemic, i.e. they are not perfused by the existing vascular network. Therefore, for each new macroscopic functional unit a fixed number of the nearest/candidate vessels is found. Then, each candidate vessel temporarily creates a bifurcation perfusing the MFU (one vessel is replaced by three vessels connected by a bifurcation point). The spatial position of the bifurcation is controlled by local minimization of the additional blood volume necessary to the MFU perfusion (Downhill Simplex algorithm [23]).

The above process can be regarded as a kind of competition because only one vessel in each tree can be finally designated to permanently perfuse the new macroscopic functional unit. Additionally, the problem of avoiding possible collisions between perfusing vessels is taken into account. The algorithms detects intersections between vessels coming from the same tree or from two different trees and rejects the related candidate vessels. Finally, from among the remaining candidate vessels, the combination (a single combination consists on one vessel from each tree) with the lowest sum of volumes is chosen to permanently perfuse the MFU

(see Fig. 1b). Afterwards, a recalculation of vessels' characteristics (i.e. pressure, radius etc.) in the vascular trees is performed. This step ensures the consistency of the characteristics according to the assumed physical and physiological laws.

After the reproduction process (i.e. mitosis and perfusion processes), comes the degeneration phase. At this step of the algorithm, few MFUs can die (necrosis process) and then all the vessels supplying these MFUs retract and disappear (retraction process). Next, the algorithm goes back to the reproduction process.

2.3 Parallel Vascular Network Growth Algorithm

In the presented sequential algorithm of vascular growth, all MFUs are connected to the vascular network one by one. Each MFU perfusion involves the necessity of creating and testing a number of temporary bifurcations. It requires a great number of calculations to face the imposed constraints to assure the consistency of vascular trees. A vascular tree is consistent if: i) it has the same blood pressure and fixed blood flow in all terminal vessels attached to MFUs and ii) the Poiseuille's law in each its vessel and the matter conservation and bifurcation laws in each its bifurcation are fulfilled. As a result, the perfusion process is the time dominant operation in the organ growth simulation. Profiling results (e.g. execution times of specific methods) showed us that it can generally consume around 70-90% of the total CPU time needed to develop an adult organ. Therefore, in order to accelerate the simulation process we proposed two parallel vascular growth algorithms [15], [16] that spread the most time consuming computations between processors and consequently are able to decrease the simulation time. Moreover, these implementations in a parallel environment can bring the model closer to reality where perfusion processes are inherently parallel.

The two previously proposed parallel algorithms are based on message passing paradigm [17] and therefore are perfectly suited for distributed memory architectures. Both algorithms use the master-slave model [5], it means that master/managing processor/node generates tasks and distributes them among slave/calculating processors.

The general scheme of the first algorithm [15] is presented in Fig. 2. It parallelizes the perfusion process. The remaining processes (i.e. necrosis, retraction and shape growth) are performed sequentially at the master node. In that case, before each perfusion phase, slave nodes do not possess the most current vascular system and tissue. Thus, at the beginning of each subcycle the master node has to broadcast the latest MFUs and vascular trees. Subsequently, after the sequential mitosis, the parallel perfusion is carried out. In comparison to its sequential version, here the managing node does not make any attempt to find candidate vessels and bifurcation points but instead it spreads these tasks over calculating nodes.

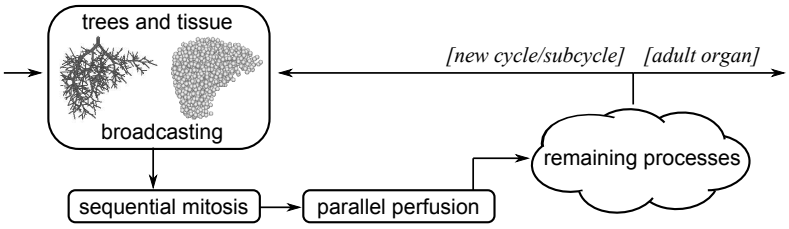


Fig. 2. The outline of the first parallel algorithm of vascular growth. At the beginning of each cycle/subcycle, the trees and tissue broadcasting is performed. Next, the sequential mitosis, parallel perfusion and remaining vascular processes (e.g. necrosis, retraction) are carried out in turn. The algorithm ends when the organ reaches its adult form.

When a computational node receives the message with several MFUs, it attempts to find the closest vessels and finally the optimal bifurcation points to perfuse these new tissue elements. Each time, when the search ends with success, the parameters of the optimal bifurcation are sent to the master node. Next, if there are any queued messages with permanent changes in vascular network sent by master node, the slave node applies these changes and continues to perform its remaining tasks.

The master node manages the perfusion process. It is responsible for gathering messages coming from the slave nodes and making decisions about the permanent perfusions. When it receives a message with optimal bifurcation parameters of one of the new MFUs, it has to check if this MFU can be connected to the current vascular network. A rejection is possible because vascular networks at individuals nodes (both at computational ones and managing one) can be slightly different (trees' nonuniformity) as a result of communication latency and independent work of slave nodes. Therefore, the master processor tries to find in its vasculature the vessels related with the proposed optimal bifurcation. If the processor cannot find at least one of these vessels, then the MFU is rejected. But in the other case, the new MFU is permanently connected to the vascular network and all organ changes related with the new tissue element are broadcasted to the slave processors.

However, we found that the efficiency of this algorithm can decrease in the case of a huge number of vessels (i.e. tens of thousands). The reason is related to the periodical broadcasting of the whole organ. We minimized the message size and only the parameters that cannot be reconstructed by slave nodes are sent. Moreover, many initial parameters are read from input files. As a result, the time to send the packed messages is insignificant. But, unfortunately, it turned out that the time needed to reconstruct vascular trees from the received packed messages by slave processors can be responsible for slowing down the algorithm.

Therefore, we also proposed an improved parallel algorithm [16]. Its general diagram is presented in Fig. 3. Each node during the whole simulation has its own copy of vascular trees and tissue. Thus, only at the beginning, the master node broadcasts the whole initial organ to ensure that all the nodes possess the same starting information. Each new subcycle starts with the sequential mitosis. Next, the perfusion process is carried out in parallel. Slave nodes attempt to find optimal bifurcations points, while the master node is responsible for managing the process of permanent perfusion and broadcasting changes.

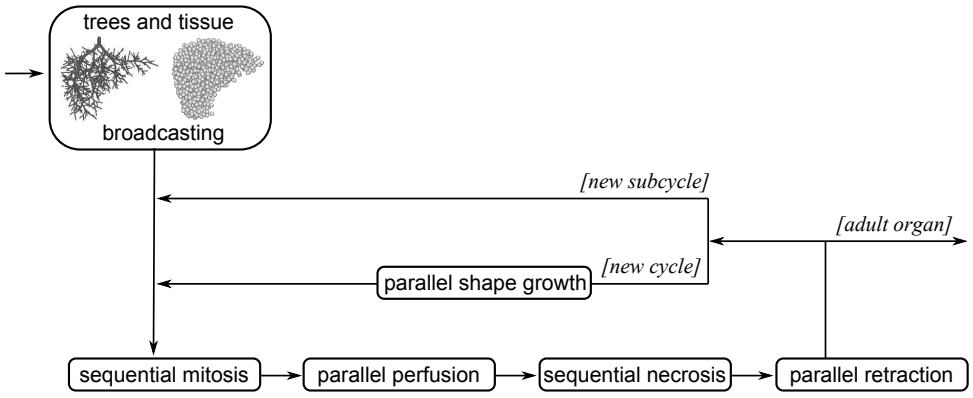


Fig. 3. The outline of the improved parallel algorithm of vascular growth. Only at the beginning, the trees and tissue broadcasting is performed. Next, the sequential mitosis, parallel perfusion, sequential necrosis and parallel retraction are carried out in turn. Then, in the case of new cycle, the parallel shape growth phase comes and algorithm returns to the sequential mitosis. In the case of new subcycle, the algorithm returns directly to the sequential mitosis. The algorithm ends when the organ reaches its adult form.

After the reproduction process, the degeneration phase follows. At the master node, the sequential necrosis is performed. Due to giving up broadcasting the whole organ in each subcycle, all the slave nodes have to be informed about possible necrosis changes. Therefore, the master node broadcasts to all other nodes information about the MFUs that have to be removed. The entire algorithm of retraction is performed at each node simultaneously. If the shape growth phase is needed, it is also carried out simultaneously at each node. The performance analysis showed that the time needed for these parts of the algorithm can be neglected, as it is very short in comparison to the perfusion time.

3. Load Balancing Mechanisms in Parallel Algorithm of Vascular Network Development

One of the most important issues in parallel computing is load balancing. It aims at roughly equal workload arrangement across processors and minimization of their idle time. Such an arrangement typically improves the performance and increases the efficiency of parallel applications, which reduces the run time of computations. Obviously, we can also improve the performance by increasing power of processors or by delivering more processors. Nevertheless, this expensive way of achieving the goal is not often able to increase the efficiency and usually should be used in the cases in which all processing units are overloaded or there is no possibility of an equal load distribution.

However, in many studies it has been shown that, even when the tasks are strongly linked with each other and their workloads are totally unpredictable, load balancing algorithms can be very useful [6]. On the other hand, one has to be careful to avoid that the cost of load balancing exceeds its possible benefits, which would decrease the overall performance.

In the next part of this section, we propose several load balancing mechanisms in the parallel algorithm of vascular network development. Firstly, the load balancing mechanisms across slave processors are presented and then we also describe how to efficiently load a master processor.

3.1 Load Balancing Across Slave Processors

In the proposed parallel algorithms we focus mainly on the perfusion process as it is the most time demanding phase of the vascular growth simulation. This process is decomposed into a set of tasks that solve the problem in parallel. A single task consists in finding a fixed number of candidate/nearest vessels and then optimal bifurcation points for a single MFU. In order to find the nearest vessels, the whole vascular network has to be searched. The time needed to perform this operation can differ for successive MFUs because of changes in vascular tree structures: new branches (i.e. vessel segments) can appear and old ones can disappear. Thus, it is very hard to estimate the time of this operation before the work distribution because one does not know a priori how the vascular system will look after each next permanent perfusion.

Subsequently, for all candidate vessels optimal bifurcation points are calculated. This operation is the most time consuming part of the perfusion process since it takes approximately 60-90% of the time needed to this process. In order to find

the position of bifurcation that minimizes a local volume of blood the Downhill Simplex algorithm is used [23], [24]. Again, we do not know how much time this searching can take because number of steps necessary to find the local minimum is hard to precise, even in the case of invariable structures of the vascular system. Moreover, for different MFUs the number of candidate vessels to be processed can be various since some of these vessels may be rejected due to their neighboring vascular structures preventing the creation of any bifurcation (e.g. lack of free space in the neighborhood).

The last stage of searching the optimal bifurcation points is the selection of one of the candidate vessels from each vascular tree in such a way that the global blood volume (i.e. for the whole vascular network) is minimal. Moreover, the algorithm detects all possible intersections between the perfusing vessels (vessels constituting the new bifurcation) in the same tree and two different trees (e.g. between arteries and veins) and rejects the related candidate vessels. Also in this case, it is impossible to predict the number of steps. This phase can end after checking the first combination of candidate vessels as well as after checking the last one.

To sum up the above general description of operations that have to be done in each task, we can state that the work required to find the optimal bifurcation points can vary for different MFUs. Moreover, it is impossible to approximate the time needed to perform each task before the work distribution as well as immediately before its execution. Thus, it is very hard to find any algorithm able to precisely decide when, where and how much work has to be assigned. In addition, we have to deal with the small grain parallelism (in one subcycle, the number of tasks can come to several thousands) and use of any sophisticated (i.e. computationally extensive) load balance algorithm can introduce an overhead that may exceed possible benefits. Therefore, we decided to propose a mechanism that is based on the basic centralized dynamic load balance algorithm [5].

After the sequential mitosis, the master node holds the collection of tasks, i.e. new MFUs to perfuse. At the beginning, a fixed part of these tasks is spread between processors (part A in Fig. 4 - 1st load balancing mechanism). Each slave processor receives approximately the same number of jobs. The master processor keeps the rest of new MFUs that will be assigned to slave nodes only on demand. When a slave node finishes its jobs, it sends a request to the master node in order to get more work (unbalance load detection). If the master node still has MFUs to be checked, it sends part of these MFUs to the under-loaded node (operation 5, part B in Fig. 4 - 1st load balancing mechanism). The number of MFUs to send is calculated according to the

following formula:

$$\frac{number\_of\_the\_remaining\_MFUs}{number\_of\_processors} + 1 \,. \tag{4}$$

This mechanism detects unbalance load dynamically and transfers tasks to idle processors. The decision on how many new MFUs are distributed immediately after the mitosis is made once at the beginning of the simulation. The great advantage of this mechanism is that it is simple for the master node to know when to terminate. In our case, the perfusion process ends when: the task queue is empty, all permanent perfusions are broadcasted across slaves nodes and all slave nodes have finished their jobs.

When a slave node finishes a single task, it sends the parameters of optimal bifurcation points to the master node (operation 2, part B in Fig. 4). Next, if there are any queued messages with permanent vascular tree changes broadcasted by the master node, the slave node applies these changes and continues to perform its remaining tasks. Such a solution decreases idle time of slave nodes because they can perform calculations without any break to wait for a response from the master node. On the other hand, due to continuous work of slave nodes (i.e. without waiting for a response whether the proposed optimal bifurcation points can be used for permanent perfusion) the trees' nonuniformity can increase. If the trees' nonuniformity increases then the possibility of MFU rejection by the master node also rises, which can cause that the time of simulation is longer. As a result, a tradeoff between these two approaches has to be found. We decided that if the number of computational nodes is quite small, the nodes work continuously. Otherwise, the nodes try to minimize the trees' nonuniformity and wait for a response (part B in Fig. 4 - 2nd load balancing mechanism).

Another crucial point in the algorithm is sending/broadcasting permanent vascular changes by the master node (operation 3, part B in Fig. 4). If one wants to minimize the time of communication between nodes, then the best solution is to send a set of changes (not a single change each time that this change appears). However, such an approach can increase the trees' nonuniformity. Thus, we made a decision to synchronize this mechanism with the mechanism described above. When computational nodes work continuously between successive MFUs (non-blocking receive of changes) then the changes are collected by the master node and sent as a set of changes (part B in Fig. 4 - 3rd load balancing mechanism). On the other hand, when the computational nodes wait for a response after each MFU (after each task) then the master node broadcasts changes as quick as possible (i.e. when they appear).

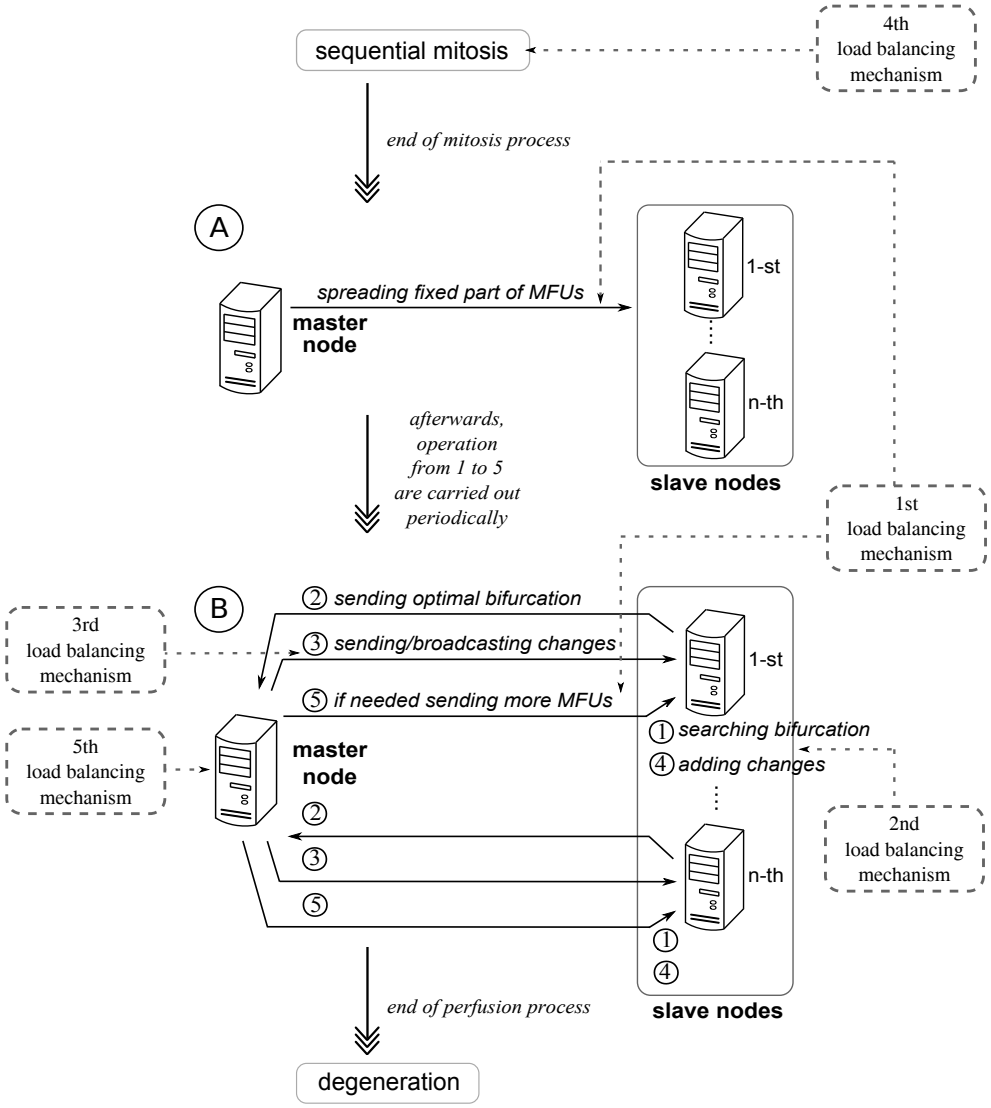


Fig. 4. Load balancing mechanisms in the parallel algorithm of vascular growth. The first load balancing mechanism is related with spreading only a fixed part of new MFUs and sending the remaining MFUs on demand. The second and third load balancing mechanisms are responsible for an appropriate organization of sending and receiving permanent vascular changes. The fourth one enables us to start the perfusion process even before the end of mitosis process. The fifth one tries to efficiently load the master processor in the case of a small number of slave nodes. Part A concerns the work distribution before the perfusion process, while part B illustrates the perfusion process during which slave nodes search optimal bifurcations and the master node manages permanent perfusions and broadcasts changes.

The next load balancing mechanism that we want to bring in concerns the mitosis process (see Fig. 4 - 4th load balancing mechanism). When the master node performs the algorithm of new MFU creation all slave nodes are idle. Therefore, we introduced the possibility that some MFUs can be spread even before the end of the mitosis process. In this case, slave nodes can start their work quicker.

3.2 Efficient Load of Master Processor

In the standard centralized dynamic load balancing algorithm [1], the master processor is responsible only for the managing of task distribution. However, in order to provide still more efficient solution, in the presented algorithm, the master processor can also perform calculations related to finding parameters of optimal bifurcations, i.e. the same as slave processors (see Fig. 4 - 5th load balancing mechanism). This mechanism can be particulary useful in the case of a small number of slave nodes since the master node can also have time to do additional calculations besides managing the perfusion process.

4. Experimental Validation

This section contains an experimental verification of the proposed load balancing mechanisms. The presented results were obtained in many experiments. We tested the behavior of the vascular model starting from small size configurations (about 1000 MFUs) and ending with large size configurations (about 50000 MFUs and consequently about 300000 vessel segments). In Fig. 5 a visualization of one of the obtained vascular network of a liver is presented. Typical physiological parameters of the hepatic vascular network were used [12]. At the beginning, the efficiency of proposed load balancing mechanisms is evaluated using the speedup and next the detailed results of the particular mechanisms are presented.

In the experiments a cluster of sixteen SMP servers running Linux 2.6 and connected by an Infiniband network was used. Each server was equipped with two 64-bit Xeon 3.2GHz CPUs with 2MB L2 cache, 2GB of RAM and an Infiniband 10GB/s HCA connected to a PCI-Express port. We used the MVAPICH version 0.9.5 [25] as the MPI standard implementation [17]. Moreover, we carried out the experiments on a similar cluster of sixteen SMP servers but each server was equipped with eight processing units and the MVAPICH version 1.0.3. In order to execute the performance analysis we used the Multi-Processing Environment (MPE) library with the graphical visualization tool Jumpshot-4 [17] and Tuning and Analysis Utilities (TAU) Performance System [26].

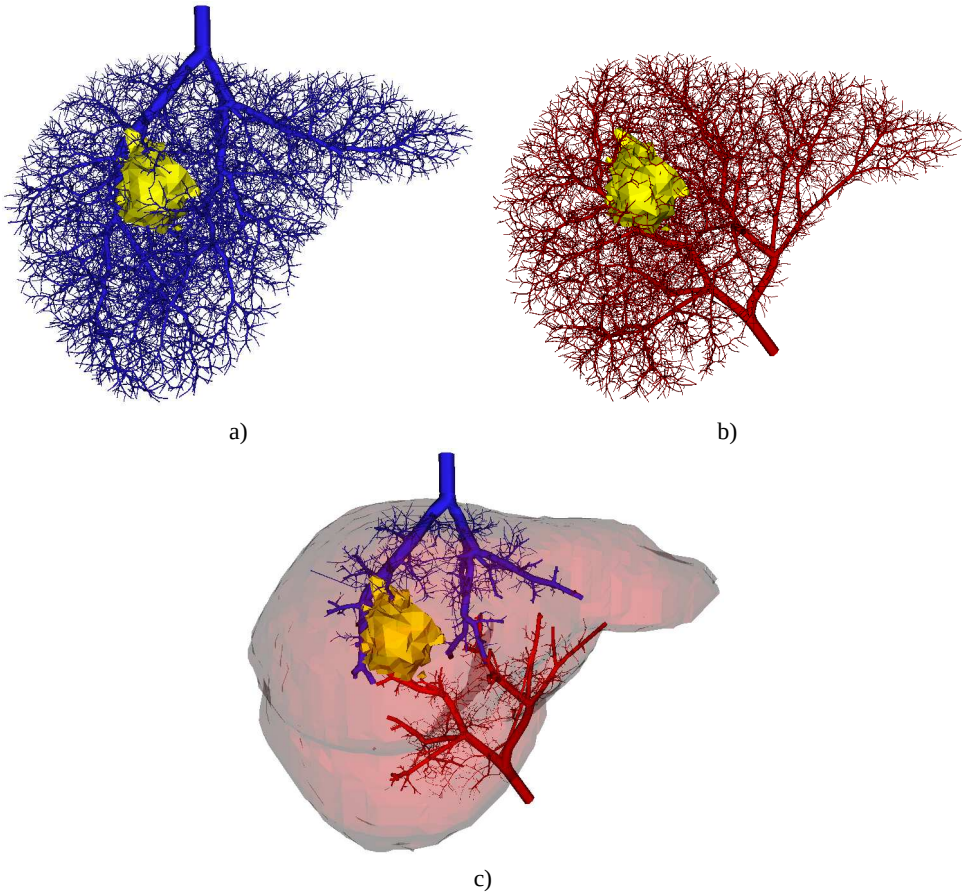


Fig. 5. Visualization of an adult liver (about 49000 MFUs and 300000 vessels): a) hepatic veins with a tumor shape, b) hepatic arteries with a tumor shape, c) main hepatic arteries, portal veins and hepatic veins with liver and tumor shapes.

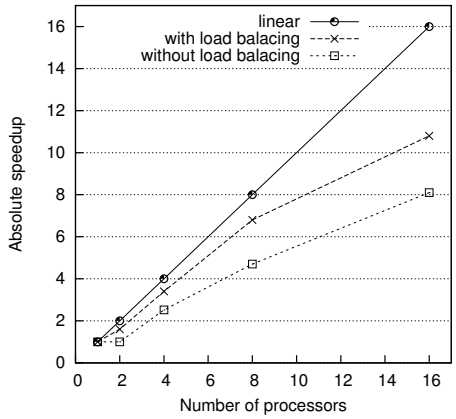


Fig. 6. Mean speedups of the parallel algorithm with and without load balancing mechanisms from experiments for different vascular network size (from several thousands to several hundreds thousands of vessel segments).

In Fig. 6 one can see the obtained average speedups of algorithms with and without load balancing mechanisms. It is clearly visible that the solution with load balancing mechanisms is from 20% to 25% quicker than the solution without this mechanisms. As a result, the simulation time on sixteen CPUs needed to receive the adult organ consisting of about 50000 MFUs and 300000 vessel segments equals approximately 2 hours with load balancing, instead of 3 hours without load balancing or 23 hours on a single processor machine (64-bit Xeon 3.2GHz with 2MB L2 cache, 2GB of RAM).

The following figures present in more detailed the influence of the load balancing algorithms. The mean time results from the simulation on eight CPUs (seven slave processors and one master processor) with large size vascular network (about 48000 MFUs) are shown in Fig. 7a . One can see the idle+communication time for particular slave processors with and without load balancing mechanisms. It is clearly visible that without load balancing mechanisms the processors waste more time for waiting and communication, which can be caused by a higher number of rejected MFUs by the master processor and consequently the necessity to test more new MFUs. Moreover, particular slave processors are unevenly loaded. On the other hand, we observe that with load balancing mechanisms the slave nodes waste approximately the same amount of time (i.e. are evenly loaded). In addition, the total idle time is shorter.

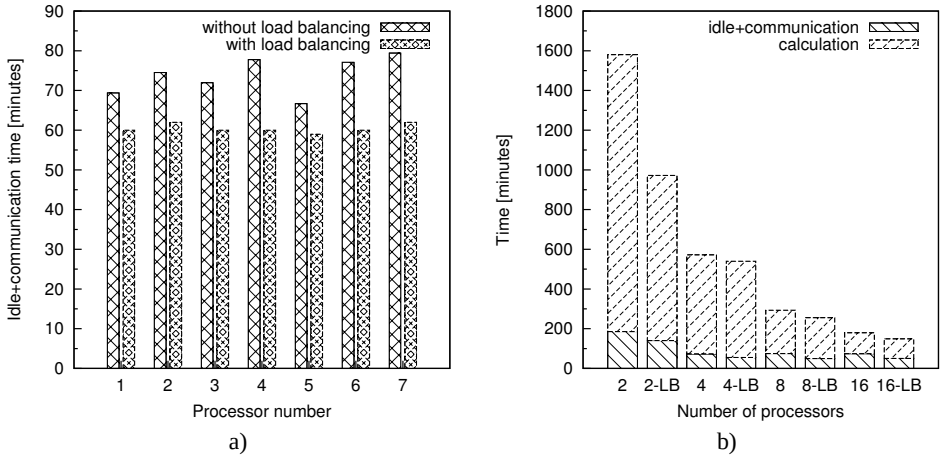


Fig. 7. The influence of load balancing mechanisms: a) mean idle+communication time for particular slave processors from experiments on eight CPUs with large size vascular network with and without load balancing mechanisms, b) mean idle+communication and calculation times for a different number of processors with large size vascular network with and without load balancing mechanisms.

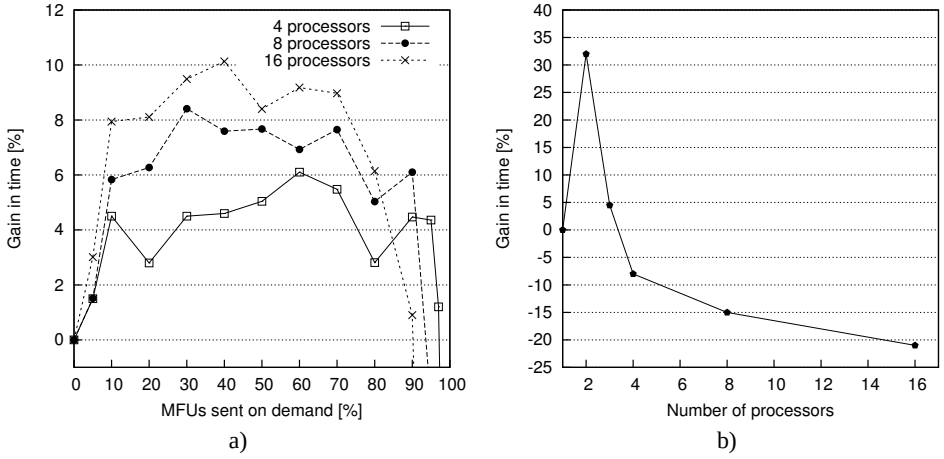


Fig. 8. The influence of load balancing mechanisms: a) related to a fixed part of MFUs that are kept by the master processor and sent only on demand to under-loaded processors, b) related to additional tasks that are assigned to the master processor.

Fig. 7b presents the mean idle+communication and calculation times obtained in experiments with large size vascular network (about 48000 MFUs) for a different number of processors without and with load balancing (LB) mechanisms. We can see that each time when the load balancing mechanisms are used both calculation and idle+communication times are shorter. The idle+communication time is shorter mainly because of keeping by the master node the fixed part of new MFUs that are sent on demand to under-loaded processors. While, the calculation time is shorter as a result of an appropriate organization of sending and receiving changes, which decreases the trees' nonuniformity. Based on many experiments, we suggest that if the number of processors is smaller than eight, the slave processors should work continuously between successive MFUs and the master node collects changes and sends them in groups. Otherwise, the slave processors wait for a response after each MFU and the master node broadcasts changes as quick as possible. Obviously, the found limit is suitable for the used clusters and may vary for other hardware.

Moreover, we thoroughly investigated the load balancing mechanism consisting in keeping by the master node a fixed part of new MFUs which are sent on demand to under-loaded processors (see Fig. 8a). From one point of view, we can see that it is very hard to choose one common value (number of MFUs) that gives the best gain in time for a different number of processors. On the other hand, it is visible that values higher than 90% can increase the simulation time. Finally, we suggest that any value within the range from 30% to 70% is acceptable.

Furthermore, we tested in which cases it is worth to involve the master processor also in calculations connected with finding optimal bifurcations points (see Fig. 8b). It is clearly visible that in the case of small number of processors (i.e. smaller than four) if the master node, besides managing, performs the same calculations as slave processors the simulation time can be reduced. On the other hand, i.e. the number of processor is bigger than three, we should not arrange any additional job to the master node.

5. Conclusion and Future Works

In this paper we propose several mechanisms with the aim to balance workload across processors and to reduce the communication overhead in the parallel algorithm of vascular network development. We consider a master-slave model in which tasks appear at the central scheduler (master processor) and are distributed between slave processors. Thus, the core mechanism is based on the centralized dynamic load balancing algorithm that detects an unbalance load dynamically and tries to send more job to under-loaded processors. Moreover, we investigate the influence of trees'

nonuniformity between particular calculation nodes and managing node on simulation time. In consequence, we found the tradeoff between communication overhead and processors' idle time. The proposed mechanisms were thoroughly validated in many experiments. The results have shown that the introduced improvements are able to further accelerate the process of vascular growth simulation. As a result, the simulation time even when we introduce more physiological details to the model or increase the number of MFUs can be done still in a reasonable period of time. In addition, it is easier to perform multiple experiments in order to calibrate model parameters.

In the future, we plan to pay more attention to parallel computing in shared-memory environments. We want to implement the vascular growth process in the framework of multi-platform shared-memory parallel programming (OpenMP) using a fine-grained parallelism. Moreover, our goal is to develop an hybrid solution able to take advantage of machines with both shared and distributed memory architectures (MPI+OpenMP implementation).

References

[1] Wilkinson, B., Allen, M.: Parallel Programming, Techniques and Applications Using Networked Workstation and Parallel Computers, Second Edition, Prentice Hall, 2005.

[2] Scott, L.R., Clark, T., Bagheri, B.: Scientific Parallel Computing, Princeton University Press, 2005.

[3] Maton, A.: Human Biology and Health, Third Edition, Pearson Prentice Hall, 1997.

[4] Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of Modeling and Simulation, Academic Press, 2000.

[5] Grama, A., Karypis, G., Kumar, V., Gupta, A.: Introduction to Parallel Computing, Addison-Wesley, 2003.

[6] Shirazi, B.A., Kavi, K.M., Hurson, A.R.: Scheduling and Load Balancing in Parallel and Distributed Systems, IEEE Computer Society Press, 1995.

[7] Bokhari, S. H.: On the mapping problem, IEEE Trans. Comput. 30(3), 1981, pp. 207-214.

[8] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing, Science 220, 1983, pp. 671-680.

[9] Lee, S.Y., Lee, K.G.: Synchronous and asynchronous parallel simulated annealing with multiple markov chains, IEEE Trans. Parallel and Distributed Systems 7, 1996, pp. 993-1008.

[10] Mani, V., Suresh, S., Kim, H.J.: Real-coded genetic algorithms for optimal static load balancing in distributed computing system with communication delays, Lecture Notes in Computer Science 3483, 2005, pp. 269-279.

[11] Osman, A., Ammar, H.: Dynamic load balancing strategies for parallel computers, Scientific Annals of Computer Science Journal of Cuza University 11, 2002, pp. 110-120.

[12] Krętowski, M., Rolland, Y., Bezy-Wendling, J., Coatrieux J.-L.: Physiologically based modeling for medical image analysis: application to 3D vascular networks and CT scan angiography. IEEE Trans. on Medical Imaging 22(2), 2003, pp. 248-257.

[13] Kretowski, M., Bezy-Wendling, J., Coupe, P.: Simulation of biphasic CT findings in hepatic cellular carcinoma by a two-level physiological model, IEEE Trans. Biomed. Eng. 54(3), 2007, pp. 538-542.

[14] Mescam, M., Kretowski, M., Bezy-Wendling, J.: Multiscale model of liver DCE-MRI towards a better understanding of tumor complexity, IEEE Trans. on Medical Imaging 29(3), 2010, pp. 699-707.

[15] Jurczuk, K., Kretowski M.: Parallel implementation of vascular network modeling, Lecture Notes in Computer Science 5101, 2008, pp. 679-688.

[16] Jurczuk, K., Kretowski M., Bezy-Wendling J.: Vascular network modeling – improved parallel implementation on computing cluster, Lecture Notes in Computer Science 6067, 2010, pp. 289-298.

[17] Pacheco, P.: Parallel Programming with MPI, Morgan Kaufmann Publishers, 1997.

[18] Mescam, M., Eliat, P.A., Fauvel, C., De Certaines, J.D., Bezy-Wendling, J.: A physiologically-based pharmacokinetic model of vascular-extravascular exchanges during liver carcinogenesis: Application to MRI contrast agents, Contrast Media Molecular Imag. 2(5), 2007, pp. 215–228.

[19] Sherlock, S., Dooley, J.: Diseases of the Liver and Biliary System, Blackwell Science, 2002.

[20] Zamir, M., Chee, H.: Branching characteristics of human coronary arteries, Can. J. Physiol. Pharmacol. 64, 1986, pp. 661-668.

[21] Kamiya, A., Togawa, T.: Optimal branching structure of the vascular trees, Bulletin of Mathematical Biophysics 34, 1972, pp. 431-438.

[22] Goss, R.J.: The strategy of growth in Control of Cellular Growth in Adult Organisms, Teir, H., Tapio R., Eds. Academic Press, 1967, pp. 3-27.

[23] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C. The art of scientific computing, Cambridge University Press, 1992.

[24] Kretowski, M., Rolland, Y., Bezy-Wendling, J., Coatrieux, J.-L.: Fast 3D modeling of vascular trees, Computer Methods and Programs in Biomedicine 70(2), 2003, pp. 129-136.
[25] MVAPICH: MPI over InfiniBand and iWARP, `http://mvapich.cse.ohio-state.edu/`
[26] Shende, S., Malony, A.D.: The TAU parallel performance system, International Journal of High Performance Computing Applications 20(2), 2006, pp. 287-311.

MECHANIZM ZRÓWNOWAŻENIA OBCIĄŻENIA W RÓWNOLEGŁEJ IMPLEMENTACJI ROZWOJU SIECI NACZYŃ KRWIONOŚNYCH

Streszczenie W artykule rozważane są mechanizmy zrównoważające obciążenie w równoległym algorytmie rozwoju sieci naczyń krwionośnych. Główną uwagę zwrócono na proces perfuzji (podłączanie nowych komórek do drzew krwionośnych) jako, że proces ten jest najbardziej czasochłonnym fragmentem rozpatrywanego algorytmu. Zaproponowane przez autorów rozwiązania mają na celu zrównoważenie obciążenia pomiędzy procesorami, skrócenie ich czasu bezczynności oraz zredukowanie narzutu komunikacyjnego. Jądro rozwiązania jest oparte na scentralizowanym dynamicznym podejściu równoważenia obciążenia. Zachowania modelu zostały przeanalizowane i kompromis pomiędzy różnymi technikami został zaproponowany. Przedstawione mechanizmy zostały zaimplementowane na klastrze obliczeniowym przy wykorzystaniu standardu MPI. Otrzymane rezultaty jednoznacznie pokazują iż wprowadzone usprawnienia zapewniają bardziej efektywne rozwiązanie co w konsekwencji pozwala na jeszcze większe przyśpieszenie procesu symulacji.

Słowa kluczowe: algorytmy równoległe, mechanizmy równoważenia obciążenia, klastry obliczeniowe, modelowanie komputerowe, system krwionośny

Artykuł zrealizowano w ramach pracy badawczej W/WI/3/2010.

IS MINIMAX REALLY AN OPTIMAL STRATEGY IN GAMES?

Katarzyna Kościuk[1]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In theory, the optimal strategy for all kinds of games against an intelligent opponent is the Minimax strategy. Minimax assumes a perfectly rational opponent, who also takes optimal actions. However, in practice, most human opponents depart from rationality. In this case, the best move at any given step may not be one that is indicated by Minimax and an algorithm that takes into consideration human imperfections will perform better.
In this paper, we show how modeling an opponent and subsequent modification of the Minimax strategy that takes into account that the opponent is not perfect, can improve a variant of the Tic-Tac-Toe game and and the game of Bridge. In Bridge we propose a simple model, in which we divide players into two classes: conservative and risk-seeking. We show that knowing which class the opponent belongs to improves the performance of the algorithm.

Keywords: Minimax, optimality, player modeling, bridge

1. Introduction

Typically, programs for game playing use the Minimax strategy [5], which assumes that the opponent is a perfectly rational agent, who always performs optimal actions. However, most humans depart from rationality [7]. In this case, at any given step, a move that is practically the best may not be one indicated by Minimax. If we know that the opponent plays defensively, for example, we can count on her not noticing or not performing moves that seem to be too daring. In order to consider a player's departures from rationality, we need to create her model and learn over time her strategies.

In this paper, we describe our attempt to model players in a variant of the Tic-Tac-Toe game and in the card game Bridge. The underlying assumption, supported by both anecdotal and empirical evidence, is that games can benefit from adapting to

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 6, pp. 63-75, 2010.

a particular opponent, rather than using the same general strategy against all players. We implement the programs with conventional Minimax strategy and the Minimax with Alpha-beta pruning. These algorithms search the game tree in order to choose the best move for the computer. We add an algorithms that makes decisions based on a model of opponent's weaknesses. The algorithm includes a learning module that observes the opponent carefully and learns his/her strategies. We test it against the Minimax strategy against human players.

This paper is organized as follows. Section 2. presents the Minimax algorithm. Section 3. describes our improvements in a variant of the Tic-Tac-Toe game. Section 4. describes the card game Bridge. Section 4.2 describes the classes of Bridge players and our experiments. Finally, Section 5. proposes some ideas of future work.

2. The Minimax Algorithm

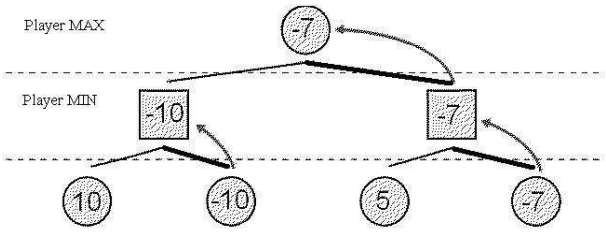


Fig. 1. Minimax algorithm

One way to pick the best move in a game is to search the game tree using the Minimax algorithm or its variants (Fig. 1). The game tree is a the directed graph in which nodes are positions in a game and edges are the moves. Minimax is used in zero-sum games (when we have two players A and B, zero-sum means that in any outcome of the game, player A's gains equal player B's losses). Each position or state in the game is evaluated using an evaluation function, that defines how good it would be for a player to achieve this position.

Minimax is the method that minimizes the maximum possible loss. At each step of the game the assumption is made that player A is trying to maximize the chances of A's winning, while player B is trying to minimize the chances of A's winning. We call the player A — MAX, the player B -– MIN, and we assume that MAX starts the game. The player MAX makes the move that maximizes the minimum value of the

position resulting from the possible next moves of the opponent and assigns a value to each of his or her legal moves. Minimax assumes that opponent always chooses the best move, but opponents are human and may depart from rationality – they can choose an inferior move (e.g., not know of or not notice a better move).

2.1 Bayesian Network

Because we use Bayesian networks for modeling the opponent in games, this section introduces them briefly.

A Bayesian network [6] is an acyclic directed graph, whose nodes represent random variables, such as observable quantities, latent variables, unknown parameters or hypotheses. The edges represent direct influences. Nodes that are not connected represent variables that are conditionally independent of each other. Each node is associated with a probability function that takes as input a particular set of values for the node's parent variables and gives the probability distribution over the variable represented by the node. An example of simple Bayesian network is shown in Fig. 2

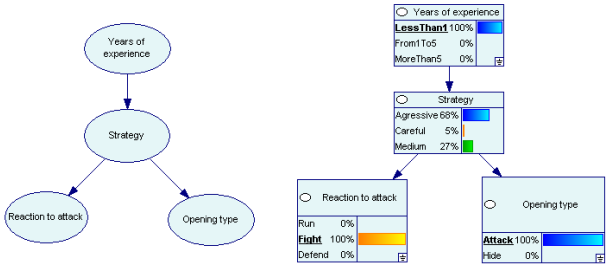


Fig. 2. Example Bayesian Network

The network has four nodes:

- The node Years_of_ experience represents the player's experience, it has three states: More_than_5, 1_5, Less_than_1.
- The node Strategy corresponds with the player's strategy, it has three states: agressive, careful, medium.
- The node Opening_type has two states: attack, hide.
- The node Reaction_to_attack has three states: run, fight, defend.

When the network is learned (for example, from the data collected during the games) we can set the evidence and check the player's strategy. For example, when

the player opens the game attacking, his reaction to attack is fight and he has less than 1 year experience, the probability that he is aggressive is 68%.

3. A simple game: Tic-Tac-Toe

3.1 Rules and Algorithm

We implement and compare several algorithms, such as Minimax with and without Alpha-beta pruning, in a variant of the Tic-Tac-Toe game [3].

We choose the board dimension 4x4. The winning position are three 'X' or three 'O' in the same line. For simplification we assume that computer plays 'O' and human player plays 'X'. We establish the following evaluation function F:

- F=0 for a draw,
- F=1 when computer wins,
- F=-1 when computer looses.

In such variant of the Tic-Tac-Toe the player who starts the game always wins, at least when she/he chooses optimal moves. We notice, as we assumed before, that people sometimes loose because they do not choose the best move.

Conventional Minimax evaluates every possible move, and when it calculates that 'O' looses (always when 'X' starts the game) it returns no move — surrenders, or it returns the first empty place on the board (depending on the implementation). It does not take into account that people often do not see the winning move.

We modify the algorithm. We count on opponent's weaknesses and continue the game even if it looks lost (evaluation function returns -1 for every possible move). For every possible move we calculate how many moves remains to the end of the game – end-depth value. Our Minimax choose the move that lets the computer play longer in order to wait for the opponent's mistake. In that case a lost game often turn out to be a winning game. When there are several moves with the evaluation function equal 1, our Minimax choose a move that lets the computer win fastest. When there are more than one move with the same F value and the same end-depth value, our algorithm returns random move. Conventional Minimax, depending on the implementation, always returns the first or the last move in such situations.

Experiments were carried out with two persons. Human player 'X' always started the game, because there were no chance of winning in another situations. Here are the average results:

- After 20 games our Minimax won in the 13 cases in 20 games, conventional Minimax won only in 1 case.

– After 60 games our Minimax won in the 27 games, conventional Minimax won in 2 games.

Experiments show that waiting for the opponent's mistake turns out to be a better strategy than surrendering. After about 30 games human player learnt where she/he should put the first move to win the game (for example in the middle of the board).

3.2 Modeling

In order to consider a player's weaknesses, it is necessary to model the player – learn and know his/her strategies. Our next step was adding an algorithm that makes decisions based on a model of opponent's weaknesses [4]. The algorithm includes a learning module that observes the opponent. We study the improvements of the algorithm over time and test it against the Minimax strategy. We build a Bayesian network to model the player. We learn the conditional probability tables in the network from data collected in the course of the game.

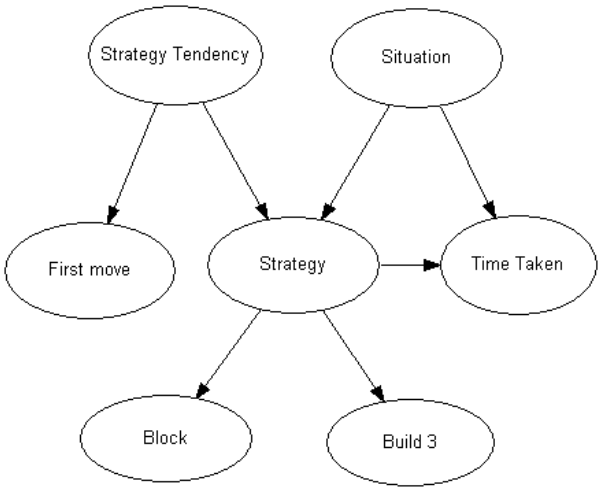


Fig. 3. Bayesian Network

Our program stores the following information about the opponent:

– the place where the opponent puts the first move,
– the opponent's long-term strategy,
– types of moves,

– situation on the board,
– how long the player thinks before a move.

We capture this information in the Bayesian network shown in Fig. 3.

– The node First_move represents the place where the player puts the first mark. It has three possible states: corner, side, and center. It might seem that there are 16 possible positions (corresponding to the 16 free places on the board). However, because of the symetry of the board, many of them are equivalent (e.g. placing the mark in any corner).
– The node Block has two states: yes and no. It assumes the state yes when the player puts a mark 'X' in order to block two 'O's in row, otherwise it assumes the state no.
– The node Build_3 has two states: yes where the player puts two 'X' in row, and has a opportunity to put the third 'X' in row, otherwise the state is no.
– The node Time_taken represents how long the player thinks before his/her move.
– The node Strategy corresponds with the player's strategy in the particular move:
  • Attack — the player attacks and sometimes does not see the necessity of defense.
  • Defense — the player blocks 'O's (even unnecessary, when he/she has a opportunity to win).
  • Attack_Defense — the player chooses the move that block 'O's and build three 'X' in row, or the move that does not change the situation on the board.
– The node Strategy_Tendency represents the player's general strategy:
  • Attack — the player has a tendency to attack more often than to defend.
  • Defense — the player has a tendency to block more often than to attack.
  • Attack_Defense — the player sometimes blocks, sometimes attacks
– The node Situation corresponds with the situation on the board. We distinguish seven situations:
  • UnderAttack -– two unblocked 'O's in row,
  • BeforeWin -– two unblocked 'X's in row,
  • Trap (fork) – an opportunity where 'O' can win in two ways,
  • Initial — before the first move,
  • Scarce — a lot of free places n the board,
  • Dense — a few free places on the board,
  • AttackedBeforeWin — two unblocked 'O's in row and two unblocked 'X's in row (player has an opportunity to win, but sometimes does not see it and blocks 'O's instead).

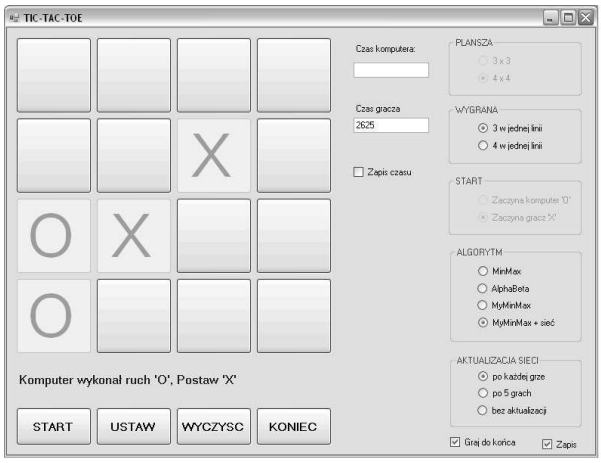


Fig. 4. A screen shot of the Tic-Tac-Toe program

When there are several moves with the same F value and the same end-depth value, our program (Fig. 4) consults the Bayesian network about the opponent's strategy.

The experiments were carried out with the 7-old boy. The Human player 'X' always started the game. We did not know before which strategy player prefers - offensive or defensive. The network was learnt during the games. Here are the average results:

- After 20 games Minimax + Bayesian Network won in the 14 cases, conventional Minimax won in 1 case.
- After 60 games Minimax + Bayesian Network won in the 25 games, conventional Minimax won in 1 game.

Experiments show that the gain from modeling the opponent in Tic-Tac-Toe is not large – our previous program, that waits for opponent's mistake, performs just as good. One of the reasons for this modest gain may be simplicity of the game therefore we start the experiments with more complex games, such as Bridge.

4. A Complex Game: Bridge

4.1 An Introduction to Bridge

Bridge is a card game played by four players with a standard deck of 52 cards. The players form two teams: North (N) — South (S) and East (E) — West (W).

The partners sit opposite each other. Bridge consists of two stages, (1) the auction (bidding), and (2) the play. During the auction, each player declares how many tricks she can probably take and what her best suit(s) is(are). The bidding ends with a contract — a declaration by one team that they take at least a stated number of tricks. They could agree on a specified suits as trump (clubs, diamonds, hearts, spades) or the game without trumps (no trump – NT). The partners that get the contract are called Declarer and Dummy, the other players are called the defenders. The first lead is made by the defender seated to the left of the Declarer. After the opening lead is played, the Dummy lays her hand face up on the table, i.e., visible to every player, including the opponents, and she does not take part in the game. Her cards are played by the Declarer. The play proceeds clockwise around the table.

If the winning team takes at least as many tricks as they have declared, they get the score [9]. Otherwise, the defenders get points for each trick that the Declarer lacks for making the contract. The score depends on:

- the trump ranked from the lowest to highest: clubs, diamonds, hearts, spades, no trumps;
- the number of tricks taken.

Bridge is an imperfect information game, as the players do not see each other hands, with the exception of the cards of the Dummy, visible by everybody. It is possible to predict the probability distribution of the unseen cards from the information gathered during the bidding.

The auction is an opportunity for all players to gather information about unseen cards. Very often, auction is based on the so called *basic natural system*, based on the High Card Points (this system is also called the Milton Work Point Count: Ace has 4 HCP, King 3 HCP, Queen 2 HCP and Jack 1 HCP). A hand that holds 12 HCP is considered sufficient to open. Opening, for example hearts, usually promises at least 4 or 5 cards in that suit. When the partners agree on hearts, we can assume that they have at least 8 cards in that suit. When a player passed, we can assume that she does not have more that 6 points (if her partner opened before) or 11 points (if she starts the bidding).

4.2 The SBA Program

The SBA (Simple Bridge Assistant) program (Fig. 5) uses the Monte Carlo technique, which is used by Ginsberg's [1] computer Bridge program (GIB). According to Ginsberg himself, GIB is currently the strongest computer Bridge program in the world. The Monte Carlo technique was originally proposed for card play by Levy

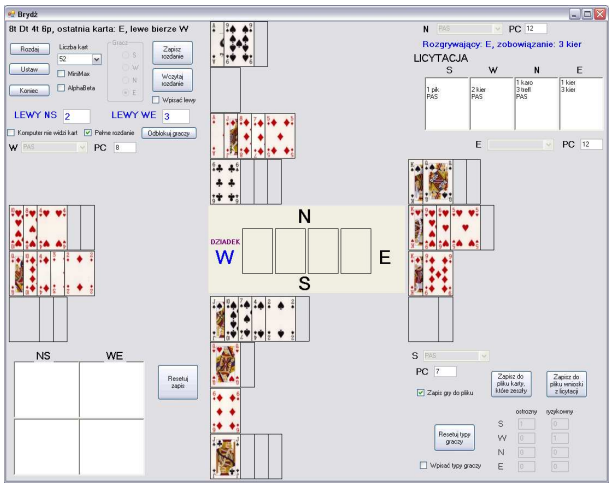


Fig. 5. A screen shot of the SBA

[2]. The unseen cards are dealt at random, using the information gathered during the auction and the cards played thus far. This perfect information variant of the game is called double dummy Bridge. All possible moves are evaluated in all generated deals. Algorithm adds the scores obtained by making every move and returns the move with the maximal sum. GIB uses brute force techniques, called partition search, to solve the double dummy game — we implemented the Minimax algorithm with Alpha-beta pruning. The game tree is very large because the average number of legal moves is four in any position. To deal with the computational complexity of computing all possible moves, we focused in our experiments on the end games consisting of the last seven tricks.

We also implement the player modeling, that recognizes the player's strategy. We divided the players into two classes — conservative and risk-taking. Our program recognizes and counts all risky and cautious steps for each player.

4.3 Classes of Bridge Players

Below we present examples taken from three real games.

The first example (Fig. 6) shows a conservative play. North is the Declarer, South is the Dummy, and the contract is for 4 Clubs. Let West play the Nine of Diamonds. If North is a careful player, she will play the Ace of Clubs (instead of Nine of Clubs, for example) being afraid that East has no diamonds and will beat a smaller trump.

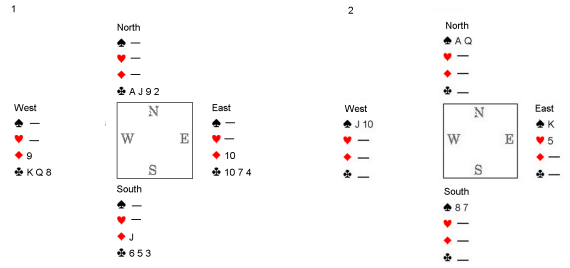


Fig. 6. Game examples: 1 – conservative player, 2 – finesse

A risky player would put a smaller trump, counting on that East has at least one Diamond.

The second example is a finesse (Fig. 6) in which a player can take advantage of the position of the particular cards. Assume that South is the Declarer, North is the Dummy, and the contract is 2 NT. A finesse is a technique that allows a player to promote tricks based on a favorable position of one or more cards in the hands of the opponents. A direct finesse is a finesse that gains a trick without losing one. Let South stars with a Seven of Spade and the Queen of Spade from Dummy's hand. If West had the King of Spade, North-South will win two tricks — one with the queen, second with the ace. The problem is that South does not know who, West or East, has the King. If he puts the Queen of Spades from Dummy, East will take two trumps.

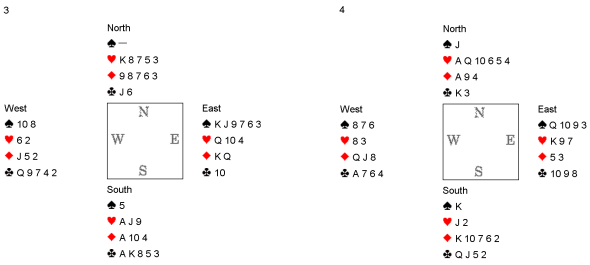


Fig. 7. Game examples: 3 – risky step, 4 – how we can use the knowledge about our opponent

Fig. 7 shows another example of a risky step. South is the Declarer, North is the Dummy, the contract is for 3 Diamonds. The first trick was taken by South. He is a risky player and now he plays the Jack of Hearts. If the Queen of Hearts were in West's hand, South would take the trick (if South makes that assumption, he will

play the Five of Hearts from Dummy's hand). West would not put the Queen of Hearts therefore he knows that the Dummy has the King of Hearts. South and North would take two tricks instead of one trick — first with Jack of Hearts, second with the King of Hearts. Unfortunately for South, West does not have the Queen of Hearts, and East will take the trick. Not to put the King of Heart from North's hand it is a risky step.

Fig. 7 also demonstrates how we can use the knowledge of our opponent's inclination to strategy. South is the Declarer, North is the Dummy, the contract is 2 NT. South took the first trick. Assume that he knows from the previous games that West is a very cautious player. South plays the Jack of Hearts. If West puts the Eight of Hearts, North will have to put the Ace of Hearts. If West had the King of Hearts, he would have put it instead. Therefore, with high probability, the Declarer knows that the King of Hearts is in East's hand.

The cautious players often puts the high cards, even it is not needed, because they afraid that someone else could take the trick.

4.4 Experiments

In our experiments we used the examples from the real game [8]. We conducted the experiments with 30 games. We compared:

- player modeling and Minimax with Monte Carlo sampling;
- the real game.

We recognized the player's category (4.3) by counting the risky and the cautious steps in 100 his last games. We chose two players that played about thousand games and strictly belongs to aforementioned categories (if the number of risky steps was greater more than 10 percent than the number of cautious steps we assumed that player is risky, accordingly we did for conservative player).

- the risky player: 123seksity;
- the conservative player: tutoo.

We chose 15 games for each chosen player and our system played the game instead of his opponents. The original moves was changed by the program only if the player's strategy could be used against him and then the rest of the game was playing by the Minimax for every player. The player modeling improved 6 scores: 4 in games with risky player, 2 in games with conservative player.

5. Discussion

Experiments show that games can benefit from adapting to a particular opponent, rather than using the same general strategy against all players. The gain from modeling the opponent in Bridge is more significant than in Tic-Tac-Toe. We plan to make more experiments in Bridge, inter alia add algorithms for other bidding systems.

References

[1] Ginsberg, M., L.: GIB: Imperfect Information in a Computationally Challenging Game. Journal of Artificial Intelligence Research 14, 2001, pp. 303–38.
[2] Levy, D.: The million pound bridge program, Heuristic Programming in Artificial Intelligence: The First Computer Olympiad. Chichester, UK: Ellis Horwood, 1989.
[3] Kościuk, K., Drużdżel, M.: Exploring opponent's weaknesses as an alternative to the Minimax strategy. XV Warsztaty Naukowe PTSK : Symulacja w badaniach i rozwoju, 2008.
[4] Kościuk, K., Drużdżel, M.: Player Modeling Using Bayesian Network. XVI Warsztaty Naukowe PTSK : Symulacja w badaniach i rozwoju, 2009.
[5] von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, 1944.
[6] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann Publishers, 1988.
[7] Simon, H., A.: A Behavioral Model of Rational Choice. The Quarterly Journal of Economics 69, No. 1, 1955, pp. 99–118.
[8] [http://www.kurnik.pl]
[9] [http://web2.acbl.org/laws/rlaws/lawofcontractbridgecombined_2004.pdf]

CZY MINIMAX JEST RZECZYWŚCIE OPTYMALNĄ STRATEGIĄ W GRACH?

Streszczenie Algorytmy grające w gry często używają strategii Minimax. Algorytm Minimax zakłada perfekcyjność przeciwnika, który wybiera zawsze najlepsze ruchy. Gracze jednakże mogą nie działać całkiem racjonalnie. Algorytm, który weźmie to pod uwagę

może dawać lepsze wyniki niż Minimax. W pracy przedstawiono jak modelowanie gracza i modyfikacje algorytmu Minimax mogą poprawić wyniki w grze kółko-krzyżyk i w brydżu. W brydżu zaproponowany został prosty model, dzielący graczy na dwie kategorie - konserwatywny i ryzykowny. Eksperymenty pokazały, że wiedza, do której klasy graczy należy przeciwnik, poprawia działanie algorytmu.

Słowa kluczowe: Minimax, optymalność, modelowanie gracza, brydż

Artykuł zrealizowano w ramach pracy badawczej W/WI/1/2009.

USING GPU TO IMPROVE PERFORMANCE OF CALCULATING RECURRENCE PLOT

Tomasz Rybak[1]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: Simulation and analysis of sophisticated systems require much computations. Moore's law, although still allows for increasing number of transistors on the die, does not lead to increase of performance of single chip — instead it leads to increased parallelism of entire system. This allows for improving performance of those algorithms that can be parallelised; recurrence plot is one of such algorithms. Graphical Processing Units (GPU) show the largest increase of parallel computations capabilities. At the same time they do not behave as traditional CPUs and require different style of programming to fully utilise their capabilities. Article shows techniques that can be used to increase performance of computing of recurrence plot on GPGPU.

Keywords: recurrence plot, non-linear analysis, fractal analysis, optimisation, parallel computations, GPGPU, CUDA

1. Introduction

Increase of performance of computers allows for using more sophisticated methods of analysis. Google's Page Rank algorithm or analysis of social networks done by Yahoo or FaceBook require enormous calculations to give results. Such algorithms perform so many computations and operate on such large amounts of data that single machines have problems with finishing calculations. Non-linear methods of analysis of systems, like recurrence plot described in this article, perform vast amounts of computations to provide user with characteristic of the system. Multi-core machines and distributed clusters allow for providing results faster as long as it is possible to parallelise computations.

While embedding many cores inside CPU is recent invention, producers of graphical cards have solved many problems with parallel systems that CPU manufacturers face. For many years even the cheapest GPUs were equipped with many

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 6, pp. 77-94, 2010.

units responsible for computing positions of points on the screen, fetching colour values from textures, drawing pixels after calculating their colours after analysing light present in 3D scene. Many scientists use GPUs for performing computations, using their enormous parallel capabilities. This requires changing description of scientific problem in the language of 3D graphics though. Article shows which libraries provided by GPU manufacturers can help with increasing performance of computations while requiring the least changes to algorithms.

This paper is organized as follows. The next section describes theory behind recurrence plots and its usage. Section 3. describes advances in hardware that allow for optimisations of calculations by using parallel capabilities of GPU. Section 4. describes details of implemented solution and achieved performance improvements using different methods. The last section presents summary and possibilities of improving program in the future.

2. Non-linear methods of analysis

Many systems show dynamic non-linear behaviour. Such systems are influenced by many parameters; values of those parameter change system in non-linear fashion. We assume that every system emits one dimensional signal (vector of samples); changes of values of those samples are caused by all variables describing the system.

Every analysis of non-linear system starts with determining dimensionality (size of phase space — the number of variables) of such a system. Finding dimensionality is done by using the loop. We begin with very small initial dimension d (1 or 2). Then, in the loop, we calculate dimension d_A of attractor reconstructed from such a system (described by our temporary dimension d). In the next iteration we increase dimension d and again calculate dimension d_A of reconstructed attractor. Loop is finished and final dimension d is known when reconstructed attractor's dimension d_A does not change after increasing d in the subsequent iterations.

Another variable influencing behaviour of non-linear system is the time delay τ. It is used to limit number of samples that are used to reconstruct attractor. Value of τ determines how many samples are used in analysis. Limiting number of points allows for avoiding clutter that would arise if all samples would be used to reconstruct attractor. Choosing proper value of τ influences validity of analysis of the system. If τ is to small, chosen points will be close to each other and system will appear stationary. When τ is too large, chosen points are far away and we lose information about trajectory of attractor in the phase space. To calculate proper value of τ one can use autocorrelation or mutual information (equation 1). The first minimum reached by the mutual information function becomes the value for delay τ.

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{1}$$

Recurrence plot can be defined as Haeviside function over difference of distance of points is space (over chosen metrics) and some threshold. Recurrence plot treats values from one-dimensional signal as properties of points in multi-dimensional phase space. It takes d samples separated by τ values and calculates distance between points created from such samples. Points are close in the phase space when distance between them is less than ε (threshold distance). Distances between all pairs of points are computed. If points P_i and P_j are closer then threshold distance ε, value in matrix on point Rp_{ij} is 1, otherwise it is 0. Matrix showing distances between all points is called recurrence plot. Matrix contains either ones or zeros for all pairs of points — depending on whether pair of points is close or far away according to chosen metrics (equation 2). Because distance calculation is symmetrical operation, recurrence plot matrix is also symmetrical. Details of calculating recurrence plot can be found in the previous papers, [14] and [15]. Many implementations of recurrence plot can be found on the web page http://recurrence-plot.tk/.

$$R_{ij} = \Theta(\|x_i - x_j\| - \varepsilon) \tag{2}$$

Single recurrence plot is 2D matrix of values from set of $\{0,1\}$. It can be plotted on the screen of paper. Black dot (value of 1) at coordinates (i,j) means that on system at time i and j was in similar state, because its attractor was represented as points that were close together (their distance was less than chosen threshold). This means that dot is plotted if two sequences coming from input data are similar (their inner product is larger than threshold). This allows for visually analysing similarity of signal at different scales. However this technique requires large amounts of memory and long processing. Similar techniques are used in analysis of gene sequences (FASTA, BLAST) to find similar gene sequences.

After computing recurrence plot we can perform visual analysis looking for sets of black dots (meaning points that are close together). Repetition of results requires however automation of analysis of data. According to theory horizontal (and vertical, as this matrix is symmetrical) lines mean that system was stationary or was changing in laminar way, very slowly. This is because each non-zero value in recurrence plot mean that two points are close together. Line means that sequence contains points that are close together. Diagonal lines mean that system was recurrent — points were returning to the same space after some time. Repetition of system is described by divergence — the instability of the system.

3. Modern Graphics Hardware

Graphical hardware performs many calculations which are necessary to generate photo-realistic image: 3D coordinates of all objects in the scene need to be transformed, non-visible points need to be eliminated, lightning must be computed, all 3D coordinates need to be transformed into 2D screen coordinates, textured need to be mapped to generated pixels, and finally the finishing effects (like bump-mapping) need to be computed for all the pixels. Every object in the displayed 3D world must come through such a path dozens of times per second to achieve sufficient fluency of animation ([8]). For this reason for many years companies producing graphical cards were making them faster and faster thanks to Moore's law. At the same time they were putting more functionality to graphical cards, to allow for game's programmers to perform more visual effects on graphical cards, without occupying already busy CPU.

It is impossible to increase performance of graphical cards indefinitely by increasing frequency of hardware clocks — similarly to limitations known from the CPUs. But from the early days producers of graphical cards had easier problem to solve. In the case of images, the same calculations are performed on large sets of points. This means that graphical problems are extremely easy to parallelise. Graphical cards, instead of using faster clocks, were equipped with more and more processing units; each of them had limited performance, but overall GPU was very efficient because it was able to compute value for hundreds of pixels at the same time. The big milestone in development of graphical hardware was invention of shaders, introduced in GeForce 3 in 2001. Shaders were small pieces of code that described transformations done on vertices (points in 3D space) and pixels (point in 2D space — on the screen). Introduction of shaders allowed for increasing performance of GPU without increasing its clock speed: tiny fragment of code was executed by dozens of processors, each of them operating on different input data. Such approach to calculations is called SIMD — Single Instruction Multiple Data principle.

This inherent parallelism present in graphical domain is the main reason why GPUs differ from CPUs. While the latter use large amount of cache to hide inevitable latency when accessing memory or peripherals, GPUs are using vast amounts of threads that are ready to be executed. When one thread is waiting for data to be transmitted, GPU switches to another thread that is ready to be executed. This is possible because of extremely fast thread context switching, provided by hardware architecture. CPU in the same situation must predict which part of memory will be needed for the next few instructions and fetch it while current instructions are still being executed. If this prediction is incorrect, entire CPU must wait for the missing

data to be read from the memory. For the same problem GPU just switches for another thread to be executed ([8]).

Shader is a function intended to operate on graphical primitives. This means that programmer is forced to describe problem in domain of 3D graphics: matrix has to become texture, calculations are changed into vertex transformation, etc. Shader languages do not have advanced control statements that would allow for easy implementation of scientific code. GPUs are built to focus on the raw calculations, not transistors needed for implementing jumps and loops. All those features of GPUs and shaders mean that using full potential present on graphical card was not easy.

3.1 CUDA

In 2008 NVIDIA introduced CUDA (Compute Unified Device Architecture), which is intended to ease writing general-purpose programs that use graphical hardware. CUDA allows to write programs for GPU as easy as for CPU by using C language. Programs written in CUDA consist of two parts: one is executed on CPU and one on GPU. Part executed on CPU (function "main") is responsible for managing data, calling code on GPU, transferring data to and from GPU, etc. This part runs as long as program is running — its end causes execution of other parts to finish. Parts that are executed on the GPU are grouped in functions called "kernels". CPU loads data and points GPU (by using CUDA library and driver functions) which kernel to call with which parameters. Each kernel is run in parallel, by as many processors as there is on the chip. Each kernel is executed by many threads and each thread is responsible for operating on fragment of input data. CUDA is responsible for mapping threads to different fragments of input. This mapping is done automatically, without need to manage it by programmer. This way hardware is almost always (depending on optimisations) fully used: on weaker GPU there will be 1000 threads running, on more powerful 16000 threads, so problem can be solved 16 times faster, but in both cases the same program will be run, and the same set of calculations will be performed. The only difference will be that in one case there will be more threads executing chosen kernel. To experience performance gains it is advised to run more threads than there is physical processors. Additional threads will be used to hide memory access latency; when GPU needs to access slow memory it switches to another thread and executes its code while waiting for data to arrive ([8]).

CUDA is not the only possible solution for using graphical hardware for calculations. ATI has its own solution called "Stream" that is using Radeon GPGPU for calculations ([2]). Intel is promising Larrabee chip which is supposed to be multi-core processor similar to the ones used by NVIDIA and ATI, but it is promised to be

compatible with x86 instruction set and offer strong cache coherency, meaning that it should be able to run existing programs without changes or recompilation. Another example of hardware offering parallel execution is Cell processor, made by Sony, Toshiba and IBM, used in PS3 and high-end computational hardware sold by IBM. The latest proposal in the world of parallel execution engines in OpenCL (Computing Language, [6]). As OpenGL allows for using the same code to run on different graphical cards, OpenCL allows for writing programs that can run computations on CPUs and GPUs without any source code changes. OpenCL programs use shader units on GPUs and cores on CPUs to provide programmer with access to parallel execution units in the same way, regardless of used hardware.

It is easy to see graphical roots of CUDA; threads are grouped in blocks, and blocks in grids. Blocks and grids have 3 dimensions and are placed in 3D space. This reflects hardware architecture — which first purpose is to operate on 3D and 2D graphical primitives. But this architecture is not limiting CUDA — one thread can perform execution on one element of matrix; this thread-block-grid placing make it rather easy to manage large amounts of threads and to know which thread is performing calculations on particular part of input data. Such an architecture is the compromise between abstraction and being close to the hardware. It allows to achieving as much performance as possible by managing threads and their groups..

To help with scientific problem solving CUDA is offering many built-in functions, like basic arithmetic functions, trigonometric functions, logarithms, etc. Full list of available functions and details of their implementation can be found in [5]. Because of hardware limitations many of currently available graphical cards do not offer fully hardware optimised and at the same time IEEE-754 compliant implementations of some functions. Programmer must often trade accuracy for speed. Situation changes quite rapidly though, as new hardware adds new mathematical functionality.

There are many levels of CUDA capabilities. New graphical cards add new features, like better thread management, support for double precision float point numbers, more mathematical functions implemented in the hardware, etc. Each graphical card presents programmer with version of offered functionalities, in the form of number; consumer-oriented GPUs have computing capabilities 1.1, scientific-oriented Tesla cards offer 1.3 capabilities. It is impossible to run program intended for CUDA 1.3 on card with capabilities 1.1, so it is important to know what is possible and which options of compiler are used. Details of all capabilities and matrix describing which chip and graphics card offers which capabilities can be found in the CUDA Programming Guide ([4]).

CUDA is not hiding different types of memory present in the GPU. Programmer can access all memory types that are available in the hardware. Each of those memory types has own advantages and disadvantages. Some of them are cached and some are not, some are private to threads or block of threads, other are shared among all executed kernels. While writing CUDA kernels and CPU code maintaining GPU code execution programmer must be aware of hardware limitations of memory access: cache coherency, cache conflicts, cost of accessing memory belonging to different banks. It is important to use different memory types to hide inefficiency of every one of them. For program to be executed with maximum possible performance, programmer must use combination of many of those memory classes. There is need to choose own trade-off between size, performance, and flexibility of used memory.

There are many research groups trying to increase performance of programs using GPUs. Gu et. al. [10] use different types of memory and different states of programs to increase performance of FFT. They change code to best use available hardware which allowed them to achieves better performance than libraries provided by NVIDIA. Another team trying to improve performance of FFT is lead by Nukada [13]. Bakkum et. al. use CUDA to speed up performance of database operations [1]. They were able to achieve performance gains from 20 to 70 times. Malony et. al. created sophisticated system that runs programs, analyses its trace, and then uses this information to try to increase performance [12].

3.2 PyCUDA

CUDA offers its functionalities to programs written in the C language. This choice means that it is available for many platforms, and it can be used from any language that can call C libraries.Calling CUDA kernels from C is not easy; there is much program state to manage, as one needs to manage memory on CPU and GPU, transfer data between them, create context in which kernels are called, manage number of threads, etc. That's the reason why many libraries were written to call CUDA from different languages. There are binding for CUDA in Java, .net, Python. Calling CUDA in high-level language makes it easier to use GPU to increase performance of computations. We avoid tedious resource management, and also does not get much performance hit — the most performance-dependent code is executed on GPU. Avoidance of resource management can also allow for focusing on better optimisation, and with some effort can even lead to CPU-GPU parallelism, as code on CPU and GPU are independent and can be called asynchronously.

PyCUDA is the CUDA library for Python — high-level interpreted language. Python offers many libraries for performing much of the most common programming

task, and provides many build-in structures, like lists, dictionaries. It is able to execute code in two modes: as full programs, which are passed to the interpreter, and in interactive mode, in which programmer is giving orders in the real time. The latter mode is great for testing new ideas; interactive mode of Python connected with PyCUDA is wonderful for experimenting and looking for possible optimisations ([11]).

```
import pycuda.driver as cuda
import pycuda.autoinit
from pycuda.compiler import SourceModule
import numpy

a = numpy.random.randn(4000).astype(numpy.float32)
b = numpy.random.randn(4000).astype(numpy.float32)
c = numpy.empty(4000).astype(numpy.float32)

m = SourceModule("""
__global__ void add(float *a, float *b, float *c, int N) {
int idx = blockIdx.x*blockDim.x + threadIdx.x;
if (idx < N)
c[idx] = a[idx] + b[idx];
}
""")
f = m.get_function("add")
f(cuda.In(a), cuda.In(b), cuda.Out(c), 4000, blocks = (256, 1, 1), grid=(16, 1))
```

Fig. 1. Program calculating sum of vectors using PyCUDA

PyCUDA's intention is to provide access for any functionality available in CUDA. Sample Python code calling kernel adding two vectors is shown on Figure 1; it it Python version of program shown in Section 1.3.4 of [3]. PyCUDA provides programmer with all advantages of Python such as Garbage Collector for all objects, including those stored on the GPU. This makes writing programs much easier. For managing resources that are scarce (like memory on the GPU) programmer may manually free any object, but this is not needed, as it will be freed during the next run of garbage collector. PyCUDA presents CUDA errors as exceptions to better integrate CPU and GPU code. PyCUDA accepts CUDA code as strings and calls compiler; NVIDIA compiler accepts the CUDA code and returns binary machine code. PyCUDA sends this binary to GPU; to avoid constant compilation PyCUDA caches compiled binaries and calls compiler only if code to be executed changed. This

means that it is possible to change code during program's execution, and PyCUDA will be able to deal with it. This CUDA code cache is hold on the hard drive so it can be used during subsequent program runs. This also means that for the first execution call will be slower as compiler will need to be run, but for all the next ones everything will be as fast as for original CUDA.

PyCUDA treats and manages CUDA programs as strings. This is the new implementation of old LISP idea "code is data". For PyCUDA program CUDA code is data (formally string inside the program) so it is possible to operate on it. By using templates it is possible to write skeleton of code and fill it with details during execution, when those values are known. This can be seen as analogous to implementing Strategy or Template Method patterns ([9]) but without subclassing and object-orientation.

At the same time writing multi-threaded programs that would be executed on CPU in Python does not bring much performance benefits. Python interpreter has Global Lock; it means that all instructions that are written in Python (even if they are executed in different threads) must wait for one another to be interpreted. Only one thread may be active in the interpreter at one time. This is not a problem in case of CUDA as all threads are executed on GPU and not on the CPU, but it disallows for improving program's performance by parallelising CPU code. CUDA threads are not interpreted by Python environment and need not wait one for one another under Global Interpreter Lock. This lock causes any CPU-threaded code in Python not to run faster than sequential code; it must be taken into consideration when analysing performance.

Python is not using machine code but internal representation of programs so it is able to provide programmer with more details regarding execution of code. For each kernel in CUDA programmer may access variables that point her how large the function is, how many registers it is using, how much shared memory is needed to execute particular kernel in one thread. This allows insight into hardware occupancy and can lead to optimisation; e.g. if function is using too much registers, it could be beneficial to move some input parameters to the constant memory, and allow compiler to use freed registers for optimising execution of kernel function. Detailed time information regarding execution is available via timers, which offer sub-millisecond accuracy and allow for timing of any events is GPU ([3]).

4. Optimising computations of recurrence plot

I wrote program in Python and program in C that calculate recurrence plot. The C program computes recurrence plot using CPU; the Python program contains two sets

of functions, one to perform computations on CPU and one that used GPU. This way program can be run on machines equipped with capable graphics card and on ones that do not contain NVIDIA-based GPU — appropriate functions are called depending on presence of hardware detected at the start of the program. All those program are used to compare results obtained by using CPU and GPU. Programs in C and Python were also used to test influence of interpreting of Python code on performance.

CPU code is single-threaded and loops over all points in the resulting recurrence plot; it is just implementation of equations shown in Section 2.. For the GPU code I followed advice from NVIDIA Best Practices ([3]) and decided to use one GPU thread to calculate value of one point in the recurrence plot matrix.

Creation of highly parallel code was possible because in recurrence plot all output points depend on the shared input vector, but do not depend on values of other output points. This means that values calculated by each of the threads are independent and no thread need to wait for any other thread. All threads are thus independent and may run at the same time, reducing the time of calculations as many times as there is threads running.

Figure 2 shows comparison of time needed by GPU and GPU to compute recurrence plot. Times are shown using logarithmic scale. GPU requires much less time to compute recurrence plot that CPU. Until there is still space of GPU to run more threads, increase of size of input vector is not visible. It means that it takes the same time to compute recurrence plot as long as not all cores in GPU are used.

In case of PyCUDA the first run of code is always slow because of need to compile code. Later executions are faster as PyCUDA caches already compiled code to save time needed to run it. As long as source code does not change only first run is slow which is visible in Table 4.. Table shows time (in seconds) that it took to perform computations. It shows that for CPU performing computations 10 times took 10 times longer. For GPU performing computations 10 times took only slightly longer time — most of the time was taken by compilation of code. This is reinforced by the last column of Table 4. which shows time needed to perform computations on GPU, without compiling code.

As can be seen in Table 4. time needed to perform computations on CPU is proportional to the size of the input data for both C and Python version. The largest size of input data is smaller for GPU (10000 samples) than for CPU (16384 samples) because of lack of space in graphics memory for larger input data. Table 4. shows that for the small size of data using GPU is not beneficial. Any gains in performance achieved when using GPU for performing computations are lost because of cost of compiling GPU code and transfer of data between host and device. But as soon as

input data is large enough (512 samples in case of described hardware and recurrence plot for Python code, 2048 for C code) the time of performing computations on GPU is shorter than on CPU, even including management tasks. For the largest size of input data (8192 samples) time needed for compute recurrence plots on GPU is 210 times smaller than time used by CPU.

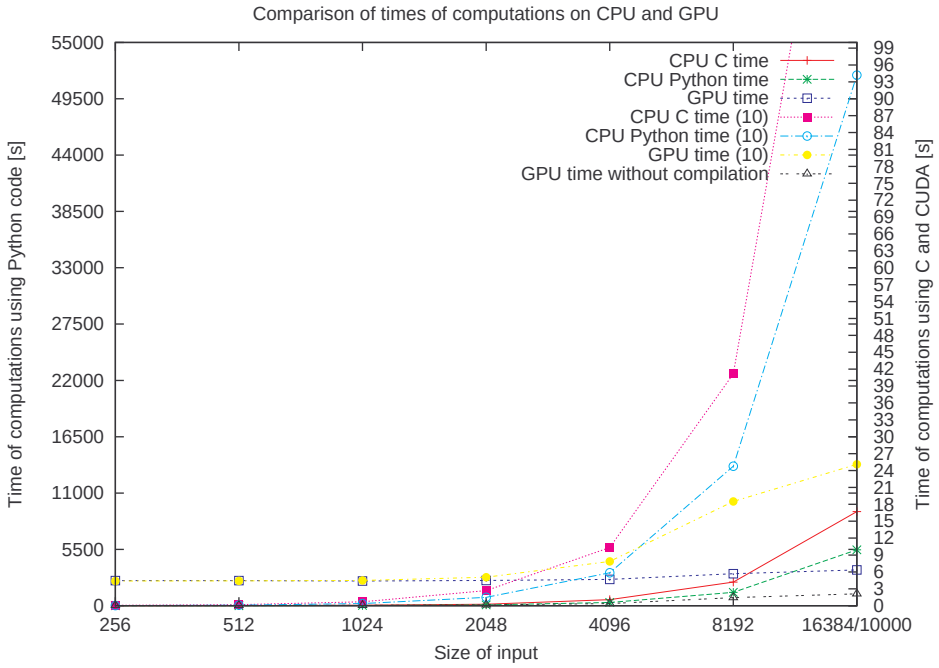


Fig. 2. Time needed for CPU and GPU to calculate recurrence plot.

To detect trends in the signal it is useful to calculate not only one recurrence plot but many of them, each for part of the signal starting at different moment. This procedure will generate not 2D matrix (recurrence plot) but series of them. This can be seen as creation of 3D matrix — which can take advantage of 3D block of threads provided by CUDA GPU. It poses problem with managing third dimension (Z axis) of the threads. In all of the currently available hardware Z dimension offers the smallest range of values. Also block also have the limited number of threads in it ([4]) so one must decide which axis will be given the most threads, and which will need to use CPU loop to manually call all kernels with appropriate parameters. The best choice

Table 1. Time needed for CPU and GPU to calculate recurrence plot.

Size	CPU-C (1)	CPU-Python (1)	GPU (1)	CPU-C (10)	CPU-Python (10)	GPU (10)	GPU compiled
256	0.007	1.28	4.47	0.06	12.21	4.40	0.05
512	0.032	5.08	4.45	0.19	49.65	4.42	0.06
1024	0.091	20.63	4.40	0.69	210.77	4.49	0.07
2048	0.267	85.32	4.47	2.68	815.60	5.06	0.14
4096	1.072	322.68	4.67	10.33	3219.37	7.86	0.39
8192	4.186	1307.67	5.68	41.17	13622.35	18.50	1.43
16384/10000	16.705	5462.22	6.36	165.63	51804.87	25.11	2.10

is to use the largest number of threads in the one plane with the same Z value ([3]). This means that it is impossible to perform all calculations at the same time and it is necessary to call kernel many times for the different values of Z. But the speedup achieved because of caching and avoiding repeated transferring the same input vector over the bus makes it worth it even for the price of more sophisticated code.

The code analysing recurrence plots uses one thread to calculate lengths of lines in one row, column or diagonal line. Because histogram is generated for the entire matrix there is a risk of inter-thread conflicts in histogram. To avoid this clash I use atomic functions that were introduced in CUDA 1.1. This means that this code can be no longer executed on low-level CUDA hardware like GeForce 8xxx but only on GeForce 9400 and better. Currently program does not check capabilities of CUDA hardware and assumes that if there is CUDA-capable GPU it can run any code regardless of required capabilities.

4.1 Increasing the performance by using hardware capabilities

Usage of GPU allows for great improvements of performance. At the same time CUDA does not hide implementation details, but makes it possible to see hardware that runs kernels, for example different types of memory. Proper usage of different types of memory allows for gaining even more performance. At the same time improper usage of memory types might cause code to be executed inefficiently and destroy all performance gains.

GPUArray present in the PyCUDA stores data in global memory of graphics card. This is the largest memory but it is also the slowest one. It is not cached which means that consecutive reads are as slow as the first one [5]. Among different types of memory texture memory is the best suited for computing recurrent plot. This is read-only memory that is not very fast but it is cached in the spatial way. It is used for

texturing in graphics programming; spacial caching means that not only read value but also its neighbours (their number depending on dimensionality of texture being read) are read ([3]). Texture cache assumes that when one texture fragment is used its neighbours will soon be needed to draw neighbour pixels on the screen. In the case of recurrence plot the situation is very similar — close threads read very close values from the input vector (see description of recurrence plot in the Section 2.). Texture memory was then natural choice for speeding up the program.

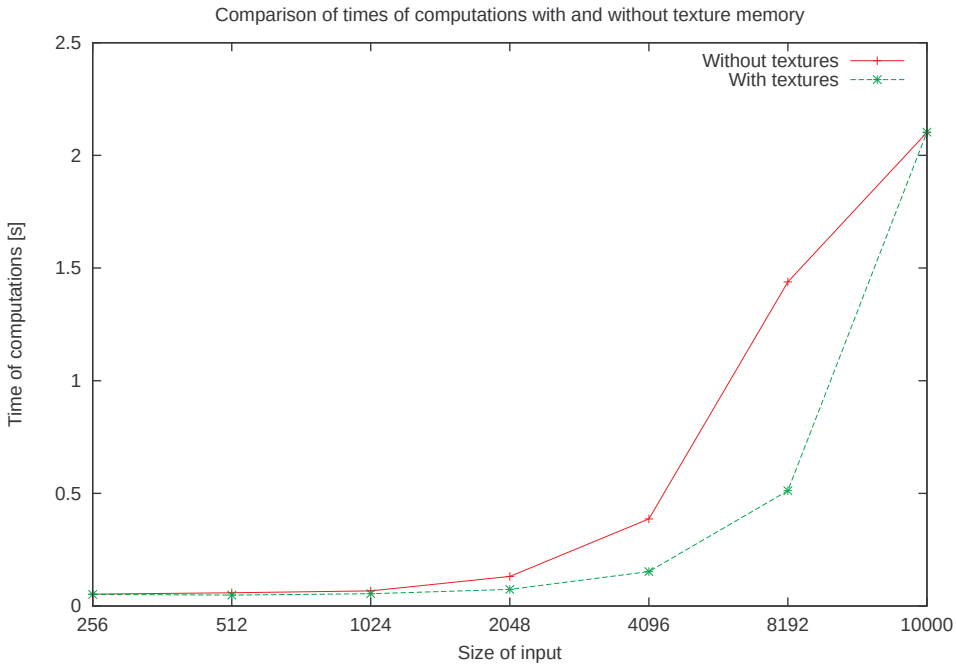


Fig. 3. Time needed to calculate recurrence plot with and without texturing

The problem with texture memory is that it is limited in size. Depending on the type of values stored in the texture and the dimensionality (1D, 2D, 3D), the size of texture in one dimension is limited to $2^{12}(4096)$ or $2^{13}(8192)$ elements. This means that it is not possible to use texture memory to perform calculations on large input vectors.

To allow usage of texture memory when it is possible program was changed that it compares size of input signal with maximum texture size. If it is smaller, texture

memory is used; if not, global GPU memory is used to store the input data. Check of size is performed on run-time and is checked for each signal independently. It means that no input is punished because of size of previously computed data.

Figure 3 shows time needed for calculate recurrence plot with and without usage of the texture memory. The same data is shown in Table 4.1. Usage of texture memory improves performance. For small size of input data difference is insignificant but as data size grows difference in performance gets significant. As can also be seen performance is the same for both cases when data is too large to fit into texture memory. As noted earlier instead of failing program notices that it cannot use texture memory and switches to the main memory. It prevents computations from failing, even though they are slower is such a case.

Table 2. Time needed to calculate recurrence plot with and without using texture memory

Size	Not using textures	Using textures
256	0.05	0.05
512	0.06	0.05
1024	0.07	0.05
2048	0.13	0.07
4096	0.39	0.15
8192	1.44	0.51
10000	2.10	2.10

It should be possible to use shared memory as described in “CUDA Best Practices Guide” [3] in case in which input data is too large to be able to use texture memory. Another possibility of optimisation is different usage of threads. Currently one thread performs very limited computations — only few additions and multiplications, for calculating one point in matrix. This is not the best solution according to “Best practices”. The key for achieving the best performance is to implement code and instrument it; this will lead to the rich library of functions and ability to choose one basing on available hardware and performance needs.

4.2 Metaprogramming

According to Andreas Kloeckner [11] it is possible to achieve great performance by checking all possible choices for better performance (like unrolling loops, changing values of parameters as constants), generating code, and checking which variant gives

the best performance. PyCUDA with its ability to generate code allows for easy experimentation to achieve great performance.

As noted in Section 3.2 PyCUDA stores kernels as text which allows for easy manipulation of those functions. This ability was used to generate programs on runtime, depending on the signal and parameters of execution; it allowed to avoid performance bottlenecks present in GPU hardware. GPU does not gives good performance for code that uses loops and conditional instructions. If there is divergence in code paths executed by different threads all other threads are stopped and only threads that contain particular jump conditions are executed ([3]).

Thanks to textual representation of CUDA code I was able to use template engine Cheetah to generate code in Python to CUDA. The most important change done was removal of the internal loop that depends on the dimension of calculated recurrence plot d (see Section 2.). Instead of looping program generates as many operations as there would be loop resolutions. This means that code executed on each of the threads is the same and there is no risk of divergent execution. This also allows for removing all looping variables. Because all threads in the block share registers, the more variables is used by thread, the less threads can be run simultaneously. Decreasing the number of variables used by threads allows for running more threads at the same time.

Usage of generated code allows for reducing number of parameters passed to the kernel. In the canonical case all variables (τ, dimension d, threshold ε) must be passed as parameters. This limits number of available registers and makes source code more sophisticated. All threads share pool of available registers, so every operation that increases number of available register allows for running more threads at the time. Switching to the generating code that includes values of parameters as constants allows CUDA to store this values in constant memory. This is special memory that is cached and shared among threads. In many cases read of all threads of the same position in constant memory is as fast as reading of one thread (see [4]) and it reduces number of used registers. Although the change of any value of parameters (dimension, tau, or threshold) forces recompilation of kernel code, the same change allows for unrolling loops — increasing overall GPU code performance.

Generating code simplifies situation with texture and global memory. Instead of two variants of each of the functions (one for global one for the texture memory) parameter was added to the Cheetah engine to choose which kind of memory should be used. When code is generated, size of input vector determines which kind of memory is used and appropriate variant of the used function is generated. This means that code calling the kernel function must know which variant was generated, but this

problem can be solved with usage of "with" statement present in Python[1]. The "with" statement allows for writing code surrounding particular block and is executed at the beginning and at the end of the block. This is the variant of the Aspect-Oriented Programming (AOP).

The final method of optimisation is using compiler options. Very often compiler does not have information about our code as is not able to change algorithm to use different type of memory or different hierarchy of threads. At the same time compiler can be used to optimise low-level details — remembering that people are better with high-level image of entire situation. Different options of compiler optimisations can be used after all described optimisation techniques are used. For example when entire loop is unrolled and can be executed without even one jump, there is no need to try to detect and resolve thread divergences and compiler may use other optimisations.

5. Summary

Usage of highly-parallel hardware on GPU via CUDA gave enormous performance improvements. It was done on sub-100EUR NVIDIA GeForce 9500GT. NVIDIA offers full range of hardware, from such cheap cards to Tesla cards intended to be used in high-end computational clusters.

Article used approach successfully applied by other researchers ([10], [12]). I was also able to achieve similar performance gains, from 20 to 100 times ([1]). Although created system is not sophisticated as described by Malony ([12]) there are many possible extensions to program described in this article.

Program does not check capabilities of CUDA hardware and assumes that if there is CUDA-capable GPU it can run any CUDA code. This should change in the future versions of the program.

The biggest chance to improve overall performance is avoiding transferring data between host and device. Currently each kernel call requires transferring data to GPU, performing computations, and transferring results back to CPU memory. Ability to hold working set on the device could decrease time needed to perform analysis. At the same time it would reduce amount of memory available for computations. Another disadvantage of such solution is risk of divergence of data stored in memory belonging to CPU and GPU. This might lead to different results depending on which device executes the code — especially as CUDA does not offer full IEEE-754 compatibility yet.

[1] Described in PEP 343, http://www.python.org/dev/peps/pep-0343/

In the middle of 2010 NVIDIA introduced new GPU chip family called Fermi. It is posed to replace Tesla chips, and offers vast improvements that can lead to better performance: integrated address space, better cache hierarchy, different management of threads, and so on ([7]). This means that even more decisions regarding code that might influence performance of computations are needed. Because of financial limitations author does not have access to Fermi chip and is not able to compare results between Tesla and Fermi, At the same time problems described in this article (and solutions to them) are even more important as hardware offers more features.

References

[1] Peter Bakkum and Kevin Skadron. Accelerating sql database operations on a gpu with cuda. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, GPGPU '10, pages 94–103, New York, NY, USA, 2010. ACM.

[2] ATI Corporation. *ATI Stream Computing OpenCL Programming Guide*, 08 2010. http://developer.amd.com/gpu/ATIStreamSDK/assets/ATI_Stream_SDK_OpenCL_Programming_Guide.pdf Accessed at 2010-10-04.

[3] NVIDIA Corporation. *NVIDIA CUDA C Best Practices Guide Version 3.2*, 08 2010. http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUDA_C_Best_Practices_Guide.pdf Accessed at 2010-10-04.

[4] NVIDIA Corporation. *NVIDIA CUDA C Programming Guide Version 3.2*, 09 2010. http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUDA_C_Programming_Guide.pdf Accessed at 2010-10-04.

[5] NVIDIA Corporation. *NVIDIA CUDA Reference Manual Version 3.2*, 08 2010. http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUDA_Toolkit_Reference_Manual.pdf Accessed at 2010-10-04.

[6] NVIDIA Corporation. *NVIDIA OpenCL Programming Guide for the CUDA Architecture Version 3.2*, 08 2010. http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/OpenCL_Programming_Guide.pdf Accessed at 2010-10-04.

[7] NVIDIA Corporation. *Tuning CUDA Applications for Fermi Version 1.3*, 08 2010. http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/Fermi_Tuning_Guide.pdf Accessed at 2010-10-04.

[8] Kayvon Fatahalian and Mike Houston. A closer look at gpus. *Communications of ACM*, 51(10):50–57, 2008.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object–Oriented Software*. Addison–Wesley, 1995.

[10] Liang Gu, Xiaoming Li, and Jakob Siegel. An empirically tuned 2d and 3d fft library on cuda gpu. In *Proceedings of the 24th ACM International Conference on Supercomputing*, ICS '10, pages 305–314, New York, NY, USA, 2010. ACM.

[11] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, and Ahmed Fasih. Pycuda: Gpu run-time code generation for high-performance computing.

[12] Allen D. Malony, Scott Biersdorff, Wyatt Spear, and Shangkar Mayanglambam. An experimental approach to performance measurement of heterogeneous parallel applications using cuda. In *Proceedings of the 24th ACM International Conference on Supercomputing*, ICS '10, pages 127–136, New York, NY, USA, 2010. ACM.

[13] Akira Nukada and Satoshi Matsuoka. Auto-tuning 3-d fft library for cuda gpus. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 30:1–30:10, New York, NY, USA, 2009. ACM.

[14] Tomasz Rybak and Romuald Mosdorf. Computer users activity analysis using recurrence plot. In *International Conference on Biometrics and Kansei Engineering*, Cieszyn, Poland, 2009. AGH.

[15] Tomasz Rybak and Romuald Mosdorf. User activity detection in computer systems by means of recurrence plot analysis. *Zeszyty Naukowe Politechniki Białostockiej. Informatyka Zeszyt 5*, pages 67–86, 2010.

UŻYCIE GPU W CELU ZWIĘKSZENIA WYDAJNOŚCI OBLICZANIA RECURRENCE PLOT

Streszczenie: Analiza skomplikowanych systemów wymaga przeprowadzenia wielu obliczeń. Prawo Moore'a, choć wciąż pozostaje w mocy, nie pozwala na zwiększanie wydajności pojedynczego procesora, ale pomaga w tworzeniu wydajnych równoległych systemów. Pozwala to na zwiększanie wydajności dla algorytmów które można zrównoleglić; recurrence plot należy do takich algorytmów. Procesory graficzne (GPU) oferują największą ilość równoległych jednostek obliczeniowych, jednocześnie jednak ich wydajne wykorzystanie wymaga innego podejścia programistycznego. Artykuł opisuje w jaki sposób wykorzystano technologię CUDA do przyspieszania obliczania recurrence plot.

Słowa kluczowe: recurrence plot, analiza fraktalna, optymalizacja, obliczenia równoległe, GPGPU, CUDA

Artykuł zrealizowano w ramach pracy badawczej S/WI/2/09.

QRS COMPLEX DETECTION IN NOISY HOLTER ECG BASED ON WAVELET SINGULARITY ANALYSIS

Paweł Tadejko[1], Waldemar Rakowski[1]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In this paper, we propose a QRS complex detector based on the Mallat and Hwang singularity analysis algorithm which uses dyadic wavelet transform. We design a spline wavelet that is suitable for QRS detection. The scales of this decomposition are chosen based on the spectral characteristics of electrocardiogram records. By proceeding with the multiscale analysis we can find the location of a rapid change of a signal, and hence the location of the QRS complex. The performance of the algorithm was tested using the records of the MIT-BIH Arrhythmia Database. The method is less sensitive to time-varying QRS complex morphology, minimizes the problems associated with baseline drift, motion artifacts and muscular noise, and allows R waves to be differentiated from large T and P waves. We propose an original, new approach to adaptive threshold algorithm that exploits statistical properties of the observed signal and additional heuristic. The threshold is independent for each successive ECG signal window and the algorithm uses the properties of a series of distribution with a compartments class. The noise sensitivity of the new proposed adaptive thresholding QRS detector was also tested using clinical Holter ECG records from the Medical University of Bialystok. We illustrate the performance of the wavelet-based QRS detector by considering problematic ECG signals from a Holter device. We have compared this algorithm with the commercial Holter system - Del Mar's Reynolds Pathfinder on the special episodes selected by cardiologist.

Keywords: ECG, heartbeat detection, QRS complex, wavelet singularity analysis, modulus maxima, noisy ECG, Holter recordings, adaptive threshold, dyadic wavelet transform

1. Introduction

Very often, QRS detection is difficult, not only because of the morphological variability of the QRS complexes, but also because of the various types of artifacts that can be present in ECG signals. Artifact sources include muscle noise, artifacts due to electrode motion, power-line interference, baseline wander, and high-frequency P or T waves similar to QRS complexes.

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 6, pp. 95-111, 2010.

However, as a matter of fact, QRS detection is a first step to be used in automatic analysis. It is necessary to determine the heart rate, and as reference for heartbeat type recognition and arrhythmia classification. A wide variety of algorithms for QRS detection have been proposed in the literature [21], [9], [10], [14], [22], [3], [1], [4], [19], [6], [2]. An extensive review of the approaches can be found in [13], [8].

High detection accuracy is often difficult to achieve, since various sources of noise are frequently encountered. Furthermore, morphological differences in an ECG waveform increase the complexity of QRS detection, due to the high degree of heterogeneity in QRS waveform and the difficulty in differentiating the QRS complex from tall peaked P or T waves.

Analysis of the local signal properties is of great importance to the signal processing, because the so-called *singularity* very often carries information important for further processing. According to the power spectra of the ECG signal, the frequency width of singularity overlaps the frequency width of normal QRS complex. A great deal of work's related to the QRS detection based on wavelet transform uses an analysis of the singularities proposed by Mallat and Hwang [16]. The most interesting works are Strang et al. [25], Burrus et al. [5], Bahoura et al. [3], Kadambe et al. [12], Li et al. [14], Martinez et al. [19], Sahambi et al. [23], Szilagyi et al. [26].

The algorithm we presented for the QRS detection in the ECG signal uses the Mallat and Hwang [16] wavelet singularity analysis. Using both, theory presented by Mallat and Hwang and our own experiments, a QRS detector has been built. Our solution contains additional original elements presented in this paper. One of them is a new approach to compute an adaptive threshold for QRS detection. The second, that is a set of rules connected with multiscale analysis. The heuristics allow to detect duplicate overlooked the QRS and determine the final set of QRS complexes from a wider set of candidates.

2. Wavelet transform and singularity analysis

The fundamentals of singularity detection in signal using the wavelet transform have been shown by Stéphane Mallat in [15], [16], [17].

Signal $f(t)$ smoothed by the function $\theta(t)$ can be represented as the result of the convolution

$$f(t) = f \star \overline{\theta}(u), \tag{1}$$

where

$$\overline{\theta}(t) = \theta(-t). \tag{2}$$

The derivative of the smoothed signal is equal to the convolution of the signal $f(t)$ with the derivative of smoothing function $\overline{\theta}'(t)$

$$\frac{d}{du}(f \star \overline{\theta})(u) = (f \star \overline{\theta}')(u) \tag{3}$$

where

$$\overline{\theta}'(t) = \frac{d\overline{\theta}(t)}{dt}. \tag{4}$$

The smoothing function and its argument can be scaled

$$\theta_s(t) = \frac{1}{\sqrt{s}}\,\theta\left(\frac{t}{s}\right), \tag{5}$$

where s is the factor of the scale. For $s > 1$ the function $\theta(t)$ is dilated and average of $f(t)$ is performed on a wider range of the independent variable t.

The derivative of the smoothed signal is given by the convolution of the signal with the scaled derivative of the smoothing function

$$\frac{d}{du}(f \star \overline{\theta}_s)(u) = f \star \overline{\theta}'_s(u), \tag{6}$$

where

$$\overline{\theta}'_s(u) = \frac{d\overline{\theta}_s(u)}{du}. \tag{7}$$

In order to show the connection between the *Continuous Wavelet Transform* (CWT) and the derivative of the smoothed signal, we define wavelet $\psi(t)$ in the form of the derivative smoothing function with the changed sign

$$\psi(t) = -\frac{d\theta(t)}{dt}. \tag{8}$$

The result is

$$\overline{\psi}(t) = \frac{d\overline{\theta}(t)}{dt} \tag{9}$$

and

$$\overline{\psi}\left(\frac{t}{s}\right) = s\,\frac{d}{dt}\left[\overline{\theta}\left(\frac{t}{s}\right)\right],$$

which means that

$$\overline{\psi}_s(t) = s\,\overline{\theta}'_s(t). \tag{10}$$

The CWT is determined and defined as follows [18]

$$Wf(u,s) = f \star \overline{\psi}_s(u). \tag{11}$$

By substituting in (11) with $\overline{\psi}_s(t)$ the right side of (10), we get

$$Wf(u,s) = s\,(f \star \overline{\theta}'_s)(u). \tag{12}$$

Based on the above equations and (6), we get

$$Wf(u,s) = s\,\frac{d}{du}\left(f \star \overline{\theta}_s\right)(u). \tag{13}$$

To differentiate the signal smoothing by the function $\theta(t)$, it is enough to calculate the CWT of the signal with the wavelet defined by the equation (8).

The domain of CWT is plane $0us$. For calculation purposes, it is necessary to discretize variables u and s. In the following, it is assumed that the signal $f(t)$ is analyzed in the range $[0,\ N-1]$.

2.1 Dyadic wavelet representation of a signal

For the purpose of singularity analysis, a *dyadic wavelet transform* of the signal $f(t)$ can be used for with the scale s takes dyadic values, i.e. $s = 2^k,\ \ k \in Z$.

The digital signal

$$a_0 = (a_0[n])_{n\in Z}, \tag{14}$$

should be calculated by the formula

$$a_0[n] = \int_{-\infty}^{+\infty} f(t)\ \phi(t-n)\ dt, \quad n \in Z. \tag{15}$$

where $\phi(t)$ is the scaling function of the wavelet analysis.
Scaling ϕ by dyadic step we can get a series of digital representation

$$(a_k)_{k\in Z}, \tag{16}$$

where

$$a_k[n] = \int_{-\infty}^{+\infty} f(t)\ \frac{1}{\sqrt{2^k}}\ \phi\left(\frac{t-n}{2^k}\right) dt, \quad n \in Z. \tag{17}$$

Averaging of the signal is performed at a distance proportional to the scale 2^k.

Detailed signals d_k are calculated by analogous to the signals a_k.

$$(d_k)_{k\in Z}, \tag{18}$$

where

$$d_k[n] = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{2^k}}\ \psi\left(\frac{t-n}{2^k}\right) dt, \quad n \in Z. \tag{19}$$

Starting from the digital signal a_0 given by (15), we calculate the *dyadic discrete wavelet transform* of a signal as:

$$\{(d_k)_{1\leq k\leq K}, a_K\}, \tag{20}$$

where a_K is a coarse representation of the signal at scale 2^K, and d_k is a detail signal at scale 2^k, wherein $1 \leq k \leq K$. Most of the details are in the signal d_1, least in d_K.

For the purposes of the singular points detection in a signal, it is sufficient to perform only signal analysis, i.e. to find the dyadic discrete wavelet transform. If the coefficients $a_0[n]$ are zero for $n < 0$ and $n \geq N$ then the dyadic discrete wavelet transform corresponds to the grid points of K rows, $K \leq log_2 N$ for N points in each row. Detection of singular points is associated with searching modulus maxima of wavelet coefficients *centered* around the vertical lines of the grid corresponding to fixed values of u.

2.2 The à trous algorithm of dyadic wavelet decomposition

Coefficients defined by (17) and (19) can be calculated iteratively using the so-called *à trous algorithm* proposed by Holschneidera, Kronland-Martineta, Morleta and Tchamitchiana [11] [24].

After selecting the scaling function ϕ and wavelet ψ based on the wavelet equations there calculated two digital filters: the low pass filter h and high pass filter g. Let, h_k is filter obtained by inserting $2^k - 1$ zeros between each pair of impulse response filter coefficients h. The extension of the filter by inserting zeros creates *holes* (franc. trous) that are mentioned in the name of algorithm. By definition

$$\overline{h}_k[n] = h_k[-n]. \tag{21}$$

The same designations apply to the filters g and g_k.

Coefficients of the dyadic discrete wavelet transform of a signal a_0 can be calculated using the following iterative calculation scheme [18]

$$d_{k+1}[n] = a_k \star \overline{g}_k[n], \tag{22}$$

$$a_{k+1}[n] = a_k \star \overline{h}_k[n] \tag{23}$$

for $k = 0, 1, 2, \ldots$.
If the input signal $(a_0[n])_{n\in Z}$ has a finite length of N samples, then the operations (22) and (23) can be achieved by using circular convolution (recurring).

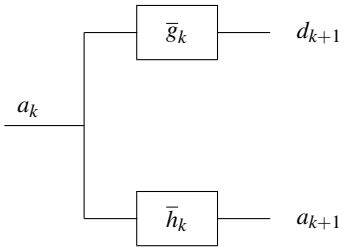


Fig. 1. Filter bank implementing one step of à trous algorithm

3. QRS detection based on singularity analysis approach

All of the wavelet-based peak detection methods mentioned in this paper [25], [5], [3], [12], [14], [19], [23], [26] are based on Mallat and Hwang's approach for singularity detection [16].

Therein, the correspondence between singularities of a function $f(t)$ and local maxima in its wavelet transform $Wf(u,s)$ is investigated. The authors of these works have used the dyadic wavelet transform (DyWT) for detecting singular points. Analysis of the value that characterize the QRS complex based on the local modulus maxima of wavelet transform. The QRS complex and other singularities in the signal are represented by local maxima at successive levels of decomposition.

It is shown that singularities correspond to pairs of modulus maxima across several scales (Fig. 2). The figure clarifies the correspondence between a signal with singularities and its wavelet coefficients. Characteristic points are detected by comparing the coefficients of the discrete DyWT of selected scales against fixed thresholds. R-peaks are detected when the locations of modulus maxima of adjacent scales exceed a threshold that is calculated for every segment. For most R waves, their energies are concentrated at scales $|d_3[n]|$ and $|d_4[n]|$.

The developed QRS detector based on wavelet singularity analysis of a signal consists of three main blocks:

- filtering ECG signal by a filter bank of dyadic wavelet transform (DyWT) using a quadratic spline wavelet and a quadratic box spline scaling function,
- detection of candidates for QRS complexes using locations of modulus maxima on selected scales with adaptive thresholds,
- determining the final list of QRS complexes using heuristic methods which use additional decision rules for reducing the number of false-positive detections.

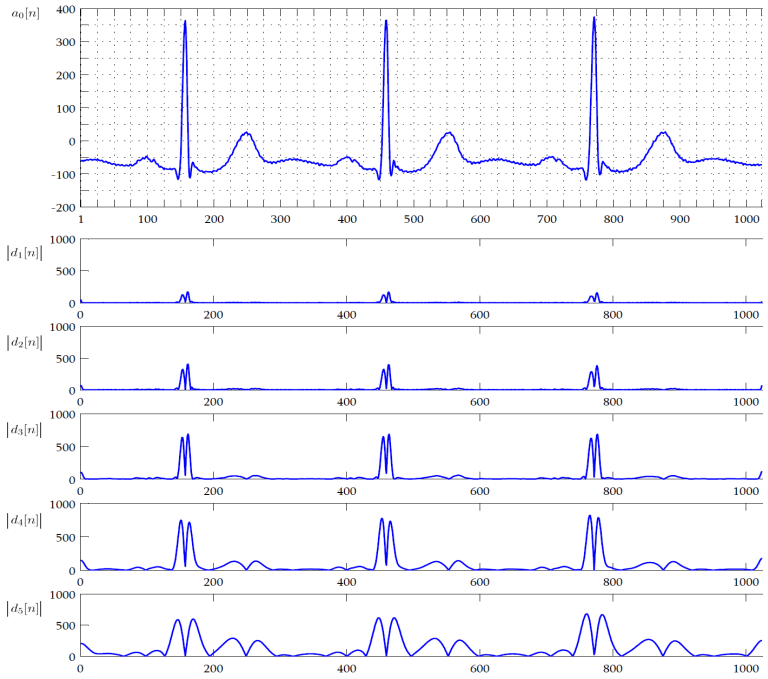


Fig. 2. The singularities in the ECG signal and its dyadic WT calculated by the *à trous* algorithm; $a_0[n]$ - ECG signal, $|d_1[n]| - |d_5[n]|$ - the modulus maxima of details coefficients in different scales.

The main part of the detection stage of the QRS complex is an original adaptive algorithm for the calculation of a QRS detection threshold.

4. An adaptive threshold for QRS complex detection

Problems in detection of the QRS complex have already been studied. The main problem is due to the presence of various types of noise (slow baseline drift, high frequency noise, impulsive noise). The great variability of patterns depends on the specific characteristics of the patient and change over time. The idea of a threshold for the detection algorithm presented in the paper uses the distribution of class intervals as one of several properties. The length of the analyzed window encloses the time interval of 16 cycles of ECG, which statistically should contain the average number of normal QRS.

Methods of determining the threshold presented in many other publications are based of empirical research on ECG signal. Therefore, complicated algorithms for

the QRS detection threshold [4], [6], [2] was built on experiments, in which linear combinations of the factors is result of fine tuning on particular set of data, such as the MIT-BIH database [1], [14], [19]. Empirically selected thresholds for the MIT-BIH database do not provide the same high accuracy, for other ECG databases, e.g. in clinical practice.

The adaptive method of threshold detection of the QRS complex that we present in this paper is independent for each successive ECG signal window. The threshold algorithm uses the properties of distribution with a compartments class.

Let the modulus maxima DyWT values of the processed ECG window

$$x_{k,1}, ..., x_{k,n_k} \tag{24}$$

will be n_k-element samples which contain the absolute modulus maxima values at each level k dyadic wavelet transform.

The *distance* of the *feature X* is the difference

$$R_k = x_{k,max} - x_{k,min}, \tag{25}$$

where $x_{k,max}$ and $x_{k,min}$ denotes the highest and lowest value in a sample.

The distance is thus the length of the shortest interval in which all values are within a sample. With a larger sample size (over 30), in which facilitates analysis, aggregated in *classes*. For simplicity, we assume that intervals is equal length. Let suppose that all values in a given class are identical to the measure class. There are several rules for determining the number of classes c_k depending on the cardinality n_k of the sample. In our research here, they are:

$$c_k \approx \sqrt{n_k} \quad or \quad c_k \approx 1 + 3.322 \log n_k, \tag{26}$$

If R_k is the range of the sample, the number of classes c_k, then b_k denotes the *length of class*, and we assume that $b_k \approx R_k/c_k$. The number of samples with values contained in the i-class is called cardinality (the size) i of the class, and we denote $n_{k,i}$ and take into account the k levels of dyadic wavelet decomposition $n_{k,i}$. The symbol $\bar{x}_{k,i}$ means the center of a subsequent class, whose values are mean values lower and upper limit for each interval i class. The result is a series of pairs of numbers: center of the class $\bar{x}_{k,i}$ and the cardinality $n_{k,i}$, called distribution with a compartments class.

Each windows are contain modulus maxima values of DyWT to be analyzed. The value of the threshold for every DyWT scale is calculated as follows:

1. Two classes are calculated that contains the local maxima with the largest values - $x_{k,c_{k-1}} = x_{k,max} - 3/2 \cdot R_k/c_k$ and $x_{k,c_k} = x_{k,max} - 1/2 \cdot R_k/c_k$, for the distribution $\bar{x}_{k,i}$ values of modulus maxima for all scales of decomposition k.

2. Thresholds for QRS detection are established at each level k as central values in the interval $[x_{k,c_{k-1}}, x_{k,c_k}]$ for all decomposition scales k.

The occurrence of a QRS complex is detected by comparing the QRS candidates for selected level of all scales of DyWT. If the locations of the local maxima exceed the threshold correlated across two consecutive scales $2^3, 2^4$ (Fig. 2), we assume that the locations of these maxima correspond to the location of QRS complexes.

Almost all algorithms use additional decision rules for reducing the number of false-positive detections e.g. search-back or eyeclosing strategy [1]. Besides this condition, the algorithm applies heuristic decision rules such as conditions on the timing of the peak occurrence within the different scales. For example, we chose one representative of the QRS when for a short time was detected a few candidates for the QRS complex, and then we remove the remaining values as a potential duplicate QRS. Due to the intrinsic refractory period of cardiac muscle, a valid QRS complex cannot occur within 200 *ms* of the previous QRS complex.

5. Results of QRS detection for the MIT-BIH Arrhythmia Database

The MIT-BIH Arrhythmia Database provided by MIT and Boston's Beth Israel Hospital was used for evaluating the proposed QRS detection algorithm. To evaluate the performance of the detection algorithm we use rates including false negative (FN), which means failing to detect a true beat (actual QRS), and false positive (FP), which represents a false beat detection. By using FN and FP the Sensitivity (SE), Positive Prediction $P+$ and Detection Error D_{err} can be calculated using the following equations respectively: $SE = TP/(TP+FN)$, $P+ = TP/(TP+FP)$ and $D_{err} = (FP+FN/totalQRS)$ where true positive (TP) is the total number of QRS correctly detected by the algorithm.

Obtained results of QRS detection for all MIT-BIH records gives sensitivity of 98.72% and a positive prediction of 99.12%. The quadratic spline wavelet with compact support and quadratic box spline scaling function were used [16].

In order to give an impression about difficulties in ECG analysis and QRS detection, the presented algorithms have been applied to selected signals from MIT-BIH database. They are shown in Table 1. These signals are the records 105, 108, 201, 208, 222. Generally, detection problems may occur for: record 105 (very high level of noise); record 108 (high P wave amplitude, often wrongly interpreted as R wave); record 201 (junctional escape beat occurs immediately after episodes of premature beat); record 208 (complexes which include among others premature ventricular contractions are grouped in 2- or 3-element blocks); and record 222 (noise and interference at higher frequencies is very similar to the QRS complex).

These signals are often part of a separate analysis of the publications: record 105 [14], [22], [3], [4]; record 108 [14], [22], [3], [1]; record 207 [22], [19], [6]; record 222 [21], [14], [3]. Their specificity means that they cause the most problems in automatic analysis. Our QRS detector performed well even in the presence of noise.

Table 1. Results QRS detection selected signals from MIT-BIH Arrhythmia Database.

	[record in MIT-BIH database] / [number of QRS complexes in record]														
	105 / 2572			108 / 1763			201 / 1963			208 / 2956			222 / 2484		
Publication	FP	FN	D_{err}	FP	FN	D_{err}	FP	FN	D_{err}	FP	FN	D_{err}	FP	FN	D_{err}
Pan & Tompkins et al., 1985 [21]	67	22	3.46	199	22	12.54	0	10	0.51	4	14	0.60	101	81	7.33
Hamilton & Tompkins, 1986 [9]	53	22	2.95	50	47	5.67	3	19	1.14	9	19	0.95	40	37	3.14
Li et al., 1995 [14]	15	13	1.09	13	15	1.59	1	12	0.66	0	4	0.14	1	9	0.40
Poli et al., 1995 [22]	86	5	3.53	143	25	9.52	0	45	2.29	15	18	1.12	4	10	0.56
Bahoura et al., 1997 [3]	27	15	0.63	20	29	2.78	7	24	1.07	2	6	0.27	12	27	1.57
Afonso et al., 1999 [1]	53	16	3.22	121	55	9.98	4	7	0.56	8	43	1.73	4	4	0.32
Chiarugi et al., 2007 [6]	37	17	2.10	34	5	2.21	0	65	3.31	11	19	1.02	1	3	0.16
this work	19	8	1.07	83	13	5.45	82	22	5.30	1	17	0.61	18	11	1.17

The processed signal window contains 2048 samples of MIT-BIH ECG recording with a sampling rate of 360 Hz. There should be about 6-10 potential candidates for QRS complexes. The set of DyWT modulus maxima values was divided into 15 ranges on each decomposition level. Detection threshold was as the central of the two average values of greatest ranges. Heuristic applied on the decision stage says that the QRS candidate must occur at least on two levels of the wavelet decomposition.

The results of the comparison are shown in Table 1. The interpretation of the results provides a partial overview and gives a good impression on which algorithms are potentially useful for a real clinical analysis systems. However, from an objective point of view, reported in many publication results are not truly reliable, because an algorithms almost always was tuned to perform perfectly on such pathological signals from MIT-BIH, but not on a clinical ECG recordings.

Unfortunately, a QRS detector which performs well for a given training database often fails when presented with different ECG's data sets. Better results could be achieved by extreme fine tuning of its parameters. Such an inconsistency in performance is a major limitation that prevents highly reliable ECG processing systems to be widely used in clinical practice. Because of this, the threshold value was updated using the formula [13], [1], [14], [19], [21], [9] which corresponds to a linear combination of constant factor specific for particular ECG data sets, e.g. MIT-BIH Database. In such cases good reported results might be difficult to reproduce.

In our algorithm, the threshold level is automatically calculated independently for each signal window, using the algorithm described in section 4. No fixed threshold level can be used, and the value must adapt to varying signal levels in order to remain at the same relative level for different statistical properties of ECG signal.

In the next section we show that the implemented method is able to detect well, wider and unusually shaped QRS complexes even when it's performed in the presence of noise or artifacts.

6. Evaluation algorithm of QRS detection on clinical data

The currently achievable detection rates often determine only the overall performance of the detectors. These numbers hide the problems that are still present in case of noisy or pathological signals. A satisfying solution to these problems still does not exist. For example, the leading suppliers of solutions for analyzing Holter records, e.g. Mortara Instrument [20] or Del Mar Reynolds Medical [7] declare that the rate sensitivity of their methods is greater than 99% for the MIT-BIH Arrhythmia Database.

Collaboration with cardiology doctors of Medical University of Bialystok allowed us to evaluate the QRS detector algorithm developed in this work. In particular, it gave us the opportunity to compare our algorithm with the world-class commercial solution: Del Mar's Reynolds Pathfinder - Holter analysis system [7]. Three cases presented in Fig. 3, 4, 5 have been analyzed with doctor in order to compare the results of the our detection algorithm and the Pathfinder software. The examples are part of a wider range of material developed during the expertise. The medical doctor was decided to study the specially selected records of two patients, six episodes of ECG for each of them.

Examples shown on figures (Fig. 3, 4, 5) are the cases of the most common disorders of the ECG recording. There are several sources of distortion and Pathfinder consequently shows shortcomings of the QRS complex detection. They may be additional components of a signal, e.g. due to work the chest muscles or the presence of strong electromagnetic field. Noise sources include muscle noise, artifacts due to electrode motion, power-line interference, baseline wander, and T waves with high-frequency characteristics similar to QRS complexes.

As we can see, the algorithm for QRS detection in the Pathfinder system works poorly for the noisy ECG signal in real conditions. The world-class automatic analysis system is not always be able to properly analyze *difficult* ECG signals.

The experiments were performed using our QRS detector with the defaults settings of the thresholding algorithm and the same quadratic spline wavelet with

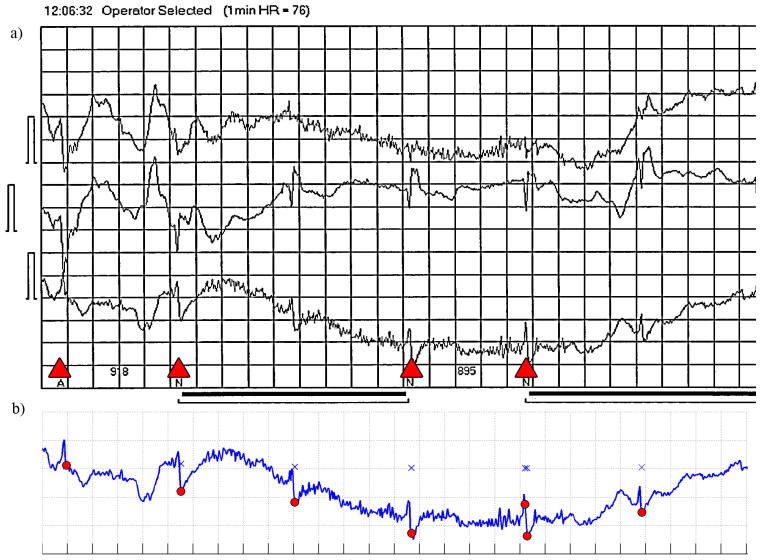


Fig. 3. The results of QRS detection for patient 1, episode 2: a) the Pathfinder has lost two QRS complexes, which are presented as dark sections of the timeline; QRS correctly detected are marked with triangle and letter "N", b) detector developed by the authors for that analyze the signal from 3th electrode has correctly detected all the QRS complexes, detected QRS are marked with dots.

compact support and a quadratic box spline scaling function. The results shows that our algorithm has detected the small number of QRS duplicates. This situation occurs because we intentionally deactivate heuristic rules for duplicates detector.

It should be noted that in the tests only one electrode was used to evaluating, although in some cases, see for example Fig. 5, where the signal from the other electrode was better quality in terms of detection.

7. Conclusions

In this paper, a QRS detection algorithm based on the Mallat and Hwang singularity analysis has been proposed. We have described the properties of the DyWT necessary for ECG signal processing. Our QRS detection algorithm results in a relatively low number of false-positives (FP) and false-negative (FN). Results obtained for full 48 recordings of MIT-BIH Arrhythmia Database are characterized by sensitivity of 98.72% and a positive prediction of 99.12%.

The conducted experiments show that the proposed algorithm may give very high efficiency in detection of QRS complex in the noisy ECG waveforms. Prelim-

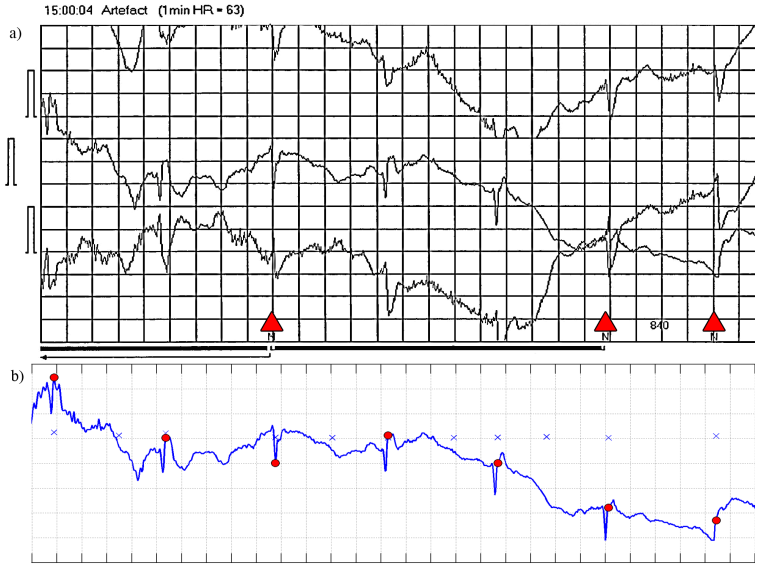


Fig. 4. The results of QRS detection for patient 1, episode 4: a) the Pathfinder has lost four QRS complexes, which are presented as dark sections of the timeline; QRS correctly detected are marked with triangle and letter "N", b) detector developed by the authors for that analyze the signal from 2nd electrode has correctly detected all the QRS complexes, detected QRS are marked with dots.

inary results obtained with clinical patient data, shows that the proposed algorithm correctly detects the QRS, even under the presence of noise, baseline drift, muscle noise and artifacts due to electrode motion. However, the large variation in the QRS complex waveforms as well as noise may still appear, so that further performance improvements are still an important goal of current research.

One of key advantages is that the QRS detection uses a reliable adaptive threshold algorithm. Calculation of the detection threshold was performed independently for each analyzed window of the signal. It makes a truly, highly adaptive algorithm. This means that the proposed solution has a great potential of clinical uses.

The algorithm has been also tested using ECG records from the Holter system at Medical University of Bialystok. In comparisons with the world-class commercial solution: Del Mar's Reynolds Pathfinder - Holter system. The proposed algorithm showed a better QRS detection results. This suggests that there is a real opportunity to create dedicated software for analyzing of Holter records that could be competitive for the currently available solutions system on the market.

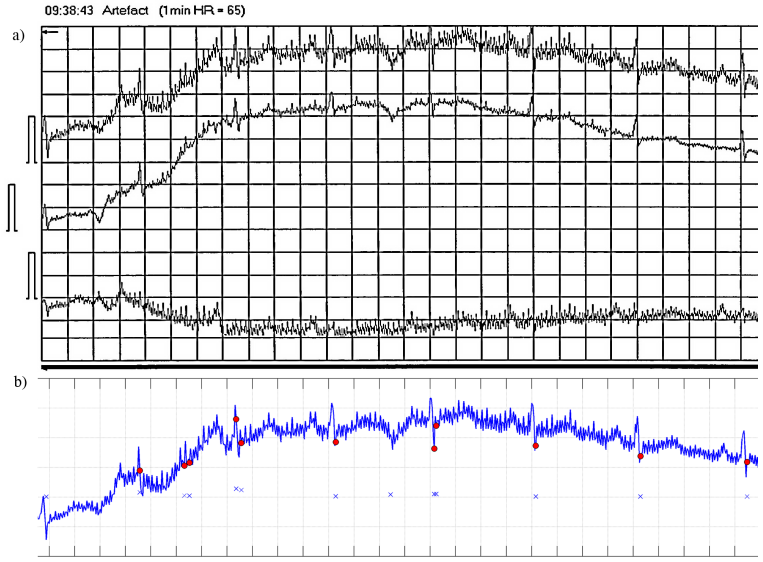


Fig. 5. The results of QRS detection for patient 2, episode 2: a) the Pathfinder has lost all QRS complexes, which are presented as dark sections of the timeline; QRS correctly detected are marked with triangle and letter "N", b) detector developed by the authors for that analyze the signal from 1st electrode has correctly detected all the QRS complexes, detected QRS are marked with dots. Three of the QRS complexes have been detected twice but the duplicate detection heuristic was inactive.

8. Acknowledgement

This paper was partially supported by grant of Faculty of Computer Science, Białystok University of Technology, Białystok, no. S/WI/4/08.

References

[1] Afonso V.X., Tompkins W.J., Nguyen T.Q., Luo S., *ECG beat detection using filter banks*, IEEE Transaction of Biomedical Engeering, vol. 46, s. 192-202, 1999

[2] Arzeno N.M., Deng Z-D., Poon C-S., *Analysis of First-Derivative Based QRS Detection Algorithms*, IEEE Transactions on Biomedical Engineering, vol. 55, s. 478-484, 2008

[3] Bahoura M., Hassani M., Hubin M., *DSP implementation of wavelet transform for real time ECG wave forms detection and heart rate analysis*, Computer Methods Programs Biomedicine, vol. 52, s. 35-44, 1997

[4] Benitez D.S., Gaydecki P.A., Zaidi A., Fitzpatrick A.P., *A new QRS detection algorithm based on the Hilbert transform*, Computers in Cardiology, s. 379-382, 2000

[5] Burrus C.S., Gopinath R.A., Guo H., *Introduction to Wavevlets and Wavelet Transforms*, Prentice Hall, 1998.

[6] Chiarugi F., Sakkalis V., Emmanouilidou D., Krontiris T., Varanini M., Tollis I., *Adaptive threshold QRS detector with best channel selection based on a noise rating system*, Computers in Cardiology, s. 157-160, 2007

[7] Del Mar Reynolds Medical, *Pathfinder Holter Analyzer*, http://www.spacelabshealthcare.com, 2009

[8] Duraj A., *QRS detection algorithms in the ECG signals from patients with implanted pacing system*, University of Zielona Góra, 2007

[9] Hamilton P.S., Tompkins W.J., *Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database*, IEEE Transaction of Biomedical Engineering, vol. 33, s. 1157-1165, 1986

[10] Hamilton P.S., Tompkins, W.J., *Adaptive matched filtering for QRS detection*, Engineering in Medicine and Biology Society, Proceedings of the Annual International Conference of the IEEE 4-7, s. 147-148, 1988

[11] Holschneider M., Kronland-Martinet R., Morlet J., Tchamitchian P., *Wavelets, Time-Frequency Methods and Phase Space*, chapter A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform, Springer-Verlag, s. 289-297, 1989

[12] Kadambe S., Murray R., Boudreaux - Bartels G.F., *Wavelet Transform - based QRS complex detector*, IEEE Transaction on Biomedical Engineering, vol. 46, s. 838-848, 1999

[13] Kohler B-U., Hennig C., Orglmeister R., *The Principles of Software QRS Detection*, IEEE Engineering in Medicine and Biology, vol. XX, s. 100-200, 2002

[14] Li C., Zheng C., Tai C., *Detection of ECG characteristic points using wavelet transforms*, IEEE Transaction of Biomedical Engeering, vol. 42, s. 21-28, 1995

[15] Mallat S., *Zero-crossings of a wavelet transform*, IEEE Transactions on Information Theory, vol. 37, no. 4, s. 1019-1033, 1991

[16] Mallat S., Hwang W.L., *Singularity detection and processing with wavelets*, IEEE Transaction on Information Theory, vol. 38, s. 617-643, 1992

[17] Mallat S., Zhong S., *Characterization of signals from multiscale edges*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 7, s. 710-732, 1992

[18] Mallat S., *A Wavelet Tour of Signal Processing, Second Edition*, Academic Press, 1999

[19] Martinez J.P., Almeida R., Olmos S., Rocha A.P., Laguna P., *A wavelet-based ECG delineator: evaluation on standard databases*, IEEE Transactions on Biomedical Engineering, vol. 51, s. 570-581, 2004
[20] Mortara Instrument *Diagnostic Cardiology Company*, WI and Los Altos, CA, http://www.mortara.com, 2009
[21] Pan J., Tompkins W.J., *A Real-Time QRS Detection Algorithm*, IEEE Transactions on Biomedical Engineering, vol. BME-32, s. 230-236, 1985
[22] Poli R., Cagnoni S., Valli G., *Genetic design of optimum linear and nonlinear QRS detectors*, IEEE Transaction of Biomedical Engeering, vol. 42, s. 1137-1141, 1995
[23] Sahambi J.S., Tandon S.N., Bhatt R.K.P., *Using wavelet transforms for ECG characterization. An on-line digital signal processing system*, IEEE Engineering in Medicine and Biology Magazine, vol. 16, s. 77-83, 1997
[24] Shensa M.J., *The Discrete Wavelet Transform: Wedding the À Trous and Mallat Algorithms*, IEEE Transactions on Signal Processing, vol. 40, no. 10, s. 2464-2482, 1992
[25] Strang G., Nguyen T., *Wavelets and Filter Banks*, Wellesley-Cambridge Press, 1996
[26] Szilagyi L., Benyo Z., Szilagyi S.M., Szlavecz A., Nagy L., *On-line QRS complex detection using wavelet filtering*, Engineering in Medicine and Biology Society, Proceedings of the 23rd Annual International Conference of the IEEE, vol. 2, s. 1872-1874, 2001

DETEKCJA ZESPOŁU QRS OPARTA NA FALKOWEJ ANALIZIE OSOBLIWOŚCI SYGNAŁU W ZAKŁÓCONYCH ZAPISACH EKG POCHODZĄCYCH Z URZĄDZENIA HOLTERA

Streszczenie Praca przedstawia algorytm detekcji zespołu QRS oparty na falkowej analizie osobliwości sygnału Mallata i Hwanga, wykorzystujący diadyczną transformatę falkową. Filtry cyfrowe analizy falkowej odpowiadają falce i funkcji skalującej w postaci tzw. spline'ów bramkowych drugiego stopnia o zwartym i krótkim nośniku. Dzięki temu podczas analizy sygnału i detekcji osobliwości możemy dokładniej kontrolować parametry procesu separacji wybranych częstotliwości. Dzięki analizie wieloskalowej możliwe jest zlokalizowanie miejsca gwałtownej zmiany sygnału, a tym samym lokalizacji zespołu QRS. Metoda posiada mniejszą wrażliwość na zmiany morfologii kolejnych zespołów QRS, minimalizuje problemy związane z występowaniem składowej wolnozmiennej, artefaktów ruchu

i napięcia mięśni oraz pozwala na łatwiejszą separację załamka R w stosunku do załamków P i T. W niniejszej pracy zaproponowano oryginalny, adaptacyjny sposób wyznaczania progu detekcji przy użyciu właściwości statystycznych obserwowanego sygnału oraz dodatkowych heurystyk. Metoda wyznaczania progu jest niezależna dla każdego kolejnego okna sygnału, składającego się z kilkunastu cykli EKG. Algorytm wyznacza wartość progu na podstawie analizy własności szeregu rozdzielczego z przedziałami klasowymi. Działanie algorytmu zostało przetestowane przy użyciu zapisów z bazy MIT-BIH Arytmia Database. Dodatkowo, wrażliwość na zakłócenia adaptacyjnego detektora QRS była przetestowana przy użyciu, specjalnie wyselekcjonowanych przez kardiologa, epizodów EKG z systemu Holtera z Uniwersytetu Medycznego w Białymstoku. Porównania wyników dokonano z komercyjnym systemem Pathfinder firmy Del Mar Reynolds.

Słowa kluczowe: EKG, detekcja uderzeń serca, zespół QRS, falkowa analiza osobliwości, modulus maxima, zakłócony EKG, zapisy Holtera, progowanie adaptacyjne, diadyczna transformata falkowa

Artykuł zrealizowano w ramach pracy badawczej Wydziału Informatyki Politechniki Białostockiej S/WI/4/08.