Zeszyty Naukowe Politechniki Białostockiej

INFORMATYKA

Numer 9

Oficyna Wydawnicza Politechniki Białostockiej

Białystok 2012

REDAKTOR NACZELNY / EDITOR-IN-CHIEF:
Marek Krętowski (m.kretowski@pb.edu.pl, 85 746 90 95)

SEKRETARZE NAUKOWI / SCIENTIFIC EDITORS:
Magdalena Topczewska (m.topczewska@pb.edu.pl, 85 746 90 86)
Marek Parfieniuk (m.parfieniuk@pb.edu.pl, 85 746 91 08)

REDAKTOR STATYSTYCZNY / STATISTICAL EDITOR:
Dorota Mozyrska .(d.mozyrska@pb.edu.pl, 85 746 90 82)

SEKRETARZ TECHNICZNY / TECHNICAL EDITOR:
Tomasz Łukaszuk (t.lukaszuk@pb.edu.pl, 85 746 92 07)

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

SOLVING THE SUDOKU WITH THE DIFFERENTIAL EVOLUTION

Urszula Boryczka, Przemysław Juszczuk

Institute of Computer Science, Silesian University, ul. Bedzinska 39, Sosnowiec, Poland

Abstract: In this paper, we present the application of the Differential Evolution (DE) algorithm to solving the combinatorial problem. The advantage of the DE algorithm is its capability of avoiding so-called „local minima" within the considered search space. Thanks to the special operator of the adaptive mutation, it is possible to direct the searching process within the solution space. The DE algorithm applies the selection operator that selects from the child population only the offspring with the greater value of the fitness function in comparison to their parents. An algorithm applied to a combinatorial optimization problem: Sudoku puzzle is presented. Sudoku consists of a nine by nine grid, divided into nine three by three boxes. Each of the eighty-one squares should be filled in with a number between one and nine. In this article we show, that the mutation schema has significant impact on the quality of created solution.

Keywords: differential evolution, sudoku, discrete optimization

1. Introduction

This paper presents the Differential Evolution algorithm for solving a combinatorial problem - the Sudoku puzzle. Sudoku is a Japanese logical game which achieved international popularity in 2005. A classical Sudoku puzzle consists of 9 by 9 matrix partitioned into nine 3 by 3 sub-matrices. The rules for completing Sudoku are very simple:

– Every subgrid contains all 9 digits.
– Every row contains all 9 digits.
– Every column contains all 9 digits.

At the beginning of game the grid only contains some of the „start values" - selected cells from 9 by 9 matrix.

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 9, pp. 5-16, 2012.

			3					
4	3	5	7	6				
	6	9		2			4	
	8			9	6			2
1			2	4			7	
				1		6	3	
				3	2	8	1	5
					5			

Fig. 1. Example of a Sudoku puzzle

These are then used to fulfill remaining empty cells. Also set of start squares is determined in such way, that it provides a unique solution to the Sudoku. Rules of this game are very simple and the objective is clear. But in the elementary 9x9 puzzle version of the game there are about $6.671x10^{21}$ valid grids, and only one appropriate solution for some given cells is satisfied. Moreover the general case of the problem has been proven to be the NP-Complete [21], which makes it one of the most interesting problems, that can be solved by approximate algorithms.

When generating these puzzles it is important that the set of start squares ensure a unique solution, so in general, puzzles are created by calculating a valid solution, and then removing a subset of the grid values, leaving just enough that the configuration has a unique solution. In general a puzzle require at least 17 start squares to maintain its uniqueness, although this problem remains formally unsolved [15].

The standard Sudoku puzzle is 9 by 9, but any n by n grid can be made into Sudoku puzzle. For example rather often found in literature problem consist of 16 by 16 cells. In 16 by 16 puzzle the rows, columns and boxes are filled with the numbers 1 to 16. Such a puzzle will be referred to as size four in reference to the value of n. The standard Sudoku will be referred to as a size three puzzle. Similarly a 4 by 4 puzzle will be called a size two puzzle.

Main objective of this paper is to test if the Differential Evolution is an efficient method for solving combinatorial problems - especially the Sudoku puzzle. Proposed algorithm will be evaluated on different sets of example problems and different difficulty puzzle levels. Also authors will try to evaluate difficulty of puzzles depending on number of iterations needed to solve the problem and then estimate the puzzle rating.

The paper is structured as follows: first, authors discuss some related works in section 2 and give description of the differential evolution algorithm in section 3. In section 4 authors describe the proposed method - especially an evaluation function and a mutation schema. Next section provides experimental results and finally we end with some conclusions.

2. Related Works

Mathematical foundations of sudoku are well discused in [2,3]. For example, one of aims is to compute the number of valid Sudoku grids. The sudoku puzzle were deeply studied as a constraint programming problem [16,6]. Other proposed method is a transformation of the sudoku puzzle to the SAT problem [8,20].

Sudoku is a combinatorial optimization problem [7], It is obviously related to the ancient magic square problem (Latin square), where different size of squares must be filled in way that the sum of each column and row are equal to 45. The magic square problem has been solved by GAs [1].

Evolutionary methods are not popular methods for solving Sudoku puzzle. There are only a few articles which presents this approach: genetic algoritms were so far used to generate optimal solutions [10]. In [4] authors focus on using the genetic algorithm for generating new instances of the Sudoku puzzle. Other population based methods are ant systems [15] and the Particle Swarm Optimization [12,11,13].

3. The Differential Evolution Algorithm

The Differential evolution is a stochastic technique which was developed by K. Price and R. Storn in 1995 [17]. It is a population–based optimization method which can be used for example to numerical optimization [19], neural network training [9], filter design [18] or image analysis [5]. DE is conceptually similar to the evolutionary algorithm, but there are also quite big differences. First of all, the mutation is the major genetic operator. It is not a trivial process and it also provides the algorithm's convergence. Moreover, the mutation is performed before the crossover process.

The pseudocode of the general DE algorithm:

```
1. Create the initial population of genotypes $P_0 = \{X_{1,0}, X_{2,0}, ..., X_{n,0}\}$,
2. Set the generation number $g = 0$
3. Until the stop criterion is not met:
```

(a) Compute the fitness function for every genotype in the population $\{f(X_{1,g}), f(X_{2,g}), ..., f(X_{n,g})\}$
(b) Create the population of trial genotypes V_g based on P_g
(c) Make crossover of genotypes from the population P_g and V_g to create population U_g
(d) Choose the genotypes with the highest fitness function from the population U_g and P_g for the next population
(e) $generation = generation + 1$, go to step a.

The DE algorithm begins with the initialization of the population $P(0)$ which consist of n_X individuals. Mutation is a process that adds randomly-generated values to the selected genes. Each increment moves selected individuals towards the global optimum. Note that the mutation is used to every individual in the population. Trial individual $u_i(t)$ is created as follows:

$$u_i(t) = x_{i_1}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$$

Individual $x_i(t)$ is called the target vector. $(x_{i_2}(t) - x_{i_3}(t))$ is a differential vector created from the two random individuals $x_{i_2}(t)$ and $x_{i_3}(t)$.

Crossover uses both the genotype from the population P_g and the trial genotype (population V_g). This operator will be described deeply in next section. After the crossover process the offspring is compared with its parent. Next, the better one of these individuals is added to the new population. The last step of the algorithm is the increment of the generation counter t. The best individual from the last generation is the result of the DE algorithm.

4. The Proposed Methodology for Solving the Sudoku Puzzle

Sudoku is a combinatorial problem, where each row, column and also each of nine 3 by 3 small square must have number digit from set $\{1,2...9\}$ exactly once. Our method assumes, that the single individual genotype is the one dimension vector, where every gene holds value from set $\{1,2,\ldots,9\}$. The array is divided into nine sub blocks (building blocks) where each block consists of nine digits.

Proposed method of transforming Sudoku square to linear vector implies some conditions. First we assume that every digit may occur only once in every of 9 genotype parts (each part is 9 genes long and corresponds to small Sudoku square).

Two mutation schemas are presented in this paper. First mutation schema applied for our algorithm is based on standard swap operator used in combinatorial problems. The mutations are applied only inside a sub block. We are using the mutation strategy

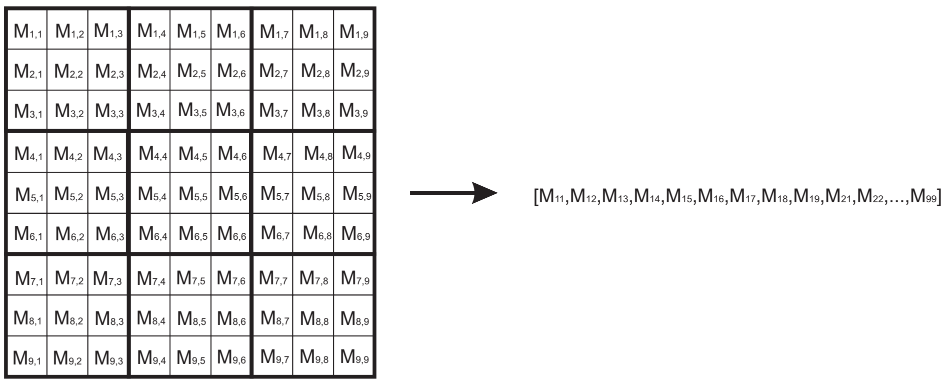


Fig. 2. Transforming Sudoku into the genotype

that is common in the combinatorial optimization - swap mutation (Fig. 3). Of course some genes in genotype are given from the beginning of the algorithm. This positions are related to the Sudoku puzzle. As it was said in section 1 each puzzle requires at least 17 start squares to maintain its uniqueness. Mutation schema allows to swap this position, but strong penalty function is used while evaluating each individual.

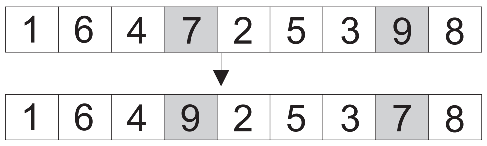


Fig. 3. Swap mutation

Other mutation schema is close to approach used in [14] and it is reffered as the geometric operator [14,13].

$$diffvector = \begin{bmatrix} x_{11} \; x_{12} \; \dots \; x_{1p} \\ x_{21} \; x_{22} \; \dots \; x_{2p} \\ \dots \; \dots \; \dots \dots \\ x_{n1} \; x_{n2} \; \dots \; x_{np} \end{bmatrix} - \begin{bmatrix} y_{11} \; y_{12} \; \dots \; y_{1p} \\ y_{21} \; y_{22} \; \dots \; y_{2p} \\ \dots \; \dots \; \dots \dots \\ y_{n1} \; y_{n2} \; \dots \; y_{np} \end{bmatrix}$$

where x and y are individuals from population P, n is number of rows in Sudoku (equal to 9) and p is number of columns in Sudoku (also equal to 9). In the next step, only a few randomly selected cells are copied to the trial vector.

The first interpretation of the mutation schema is close to the general one used in the Differential Evolution algorithm for the continuous optimization. Unfortunately, as we show in the next section, this approach gives worse effect than the second option.

The crossover operation is applied so that it exchanges whole sub blocks of nine numbers between individuals. Thus the crossover point cannot be inside a building block.

The core of this algorithm is the fitness function. It is required that the each row x_i and column x_j must be equal to set $A = \{1,2,...,9\}$. So we can calculate number of missing digits in each row x_i and column x_j set:

$$f_1 = |A - x_i|$$

$$f_2 = |A - x_j|$$

where $|\cdot|$ denotes for cardinality of a set. Also:

$$f_3 = f_3 + 100 \text{ if } (fixed_{i,j} \neq 0) and (fixed_{i,j} \neq x_{i,j})$$

where $x_{i,j}$ denotes for cell at the intersection of row i and column j, and $fixed$ denotes for matrix of given cells (matrix cell is filled only if cell is given, otherwise cell value is equal to zero). So the fitness function is equal to:

$$fit(x) = f_1 + f_2 + f_3 \tag{1}$$

The fitness function is computed for each offspring from the population U_g, and the comparison of the computed fitness function with the value of the fitness function of the parent (from the population P_g) follows. The genotype with the higher fitness function value is transferred to the next population. Our proposed method is based on two simple rules which allows properly evaluate population:

– rule 1 is derived from the set theory. It is required that the each row and column of Sudoku must be equal to set $A = \{1,2,...,9\}$.
– second rule is linked with mutation operator. If mutation affects on cell that is considered as constant value in puzzle - the fitness function is strongly decreased (this case explicitly rejects individual as unsatisfactory).

A selection schema used in this algorithm guarantee that individuals promoted to the next generation are not worse than individuals in actual generation.

$$X_{i,t+1} = \begin{cases} U_{i,t} \text{ when } f(U_{i,t}) \leq f(X_{i,t}), \\ X_{i,t} \text{ else.} \end{cases}$$

5. Experimental results

The aim of this research work is to determine if the Differential Evolution algorithm is capable to solve an example of the difficult combinatorial problem - the Sudoku puzzle. For all experiments described in this section we assume the following settings:

– size of the population $P = 200$,
– crossover $CR = 0.5$,
– swap mutation,
– the length of the gene was 81,
– the maximum number of iterations was set to 40000.

For our experiments we used a group of Sudoku puzzle divided in dependence of their difficulty. Every puzzle was run 100 times. In the Table 1 we put information about the given number of cells at the beginning of the algorithm, minimum, maximum and average number of iterations needed to achive solution, median and standard deviation. We also used the Differential Evolution to generate Sudoku puzzle from scratch. Afterwards newly generated puzzle is solved by our algorithm and rated. Unfortunately first mutation schema based on geometric operators provides far worse solutions and the algorithm is capable to solve within a reasonable time only easy puzzle (less than 20000 iterations), results shown below concern only the Differential Evolution algorithm with the swap mutation. Also, as a comparison, in the Table 2 we present results for solving Sudoku puzzle with the Genetic Algorithm [10]. We used the same Sudoku problems, and the same number of individuals in the population. As it may be seen, it is difficult to clearly rate newly generated puzzle. Four sets generated from the scrach, had 30 cells filled. It is rather unclear, if the newly generated puzzles should be assigned to the „easy” or „medium” classes. The Differential Evolution seems to be far more effective in solving the Sudoku problems. Most likely, cause of this is the selection schema. In the Differential Evolution only the best adapted individuals are moved to the next generation. In the Genetic Algorithm we used the proportional selection, which allows worse adapted individuals to move into the next generation. Premature convergence in the Sudoku is not the problem, and the more greedy schema is better. Interesting is fact, that both algorithms seems to be sometimes insensitive to the Sudoku difficulty level. It may be seen in the GA results, where average number of iterations needed to find solution i similar for the Medium and Hard problems. We also present distribution of population in subsequent iterations for selected „easy”, „medium” and „hard” puzzle.

Two interesting population features may be seen at Figures 4, 5 and 6. In all three cases, at the first step of the algorithm significant improvement of fitness function can

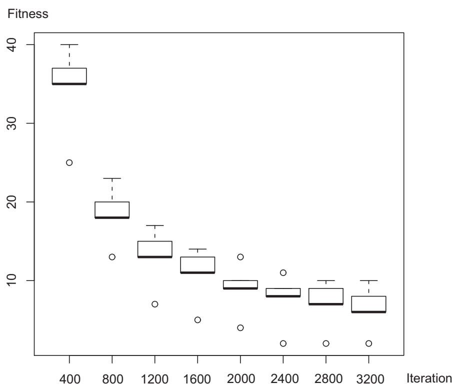


Fig. 4. Distribution of population in subsequent iterations for „easy” puzzle

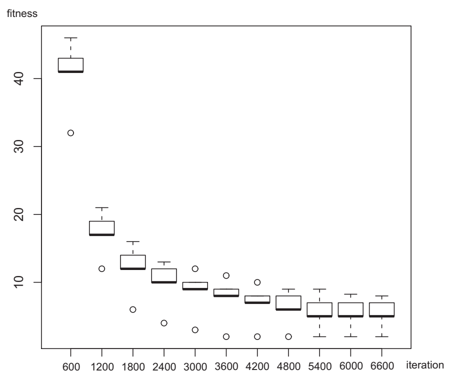


Fig. 5. Distribution of population in subsequent iterations for „medium” puzzle

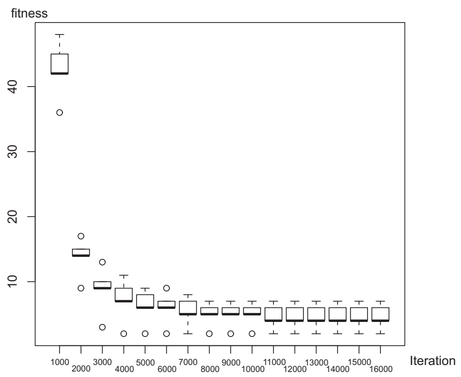


Fig. 6. Distribution of population in subsequent iterations for „hard” puzzle

Table 1. The comparison of how effectively DE finds solutions for the Sudoku puzzles with different difficulty ratings.

Difficulty rating	**Cells given**	**Minimum number of iterations**	**Maximum number of iterations**	**Average number of iterations**	**Median**	**Std dev**
New	0	855	1638	1258.714	1354	281.4378
Easy	34	2344	5344	3701.9	3377	1026.246
Easy	30	2270	5040	3801.9	3824	949.161
Medium	30	4571	12343	6833.9	5665	3537.016
Medium	26	5709	16743	10713.5	9946.5	3548.422
Hard	26	7276	29686	15206.4	12564.5	7361.502
Hard	24	6635	30105	19849.43	19701	7624.729
New generated 1	30	2461	6875	4976.571	5805	1642.706
New generated 2	30	2483	4164	3203.857	2923	652.475
New generated 3	30	3007	7447	4915.286	4238	1829.077
New generated 4	30	3480	9493	5583.571	4946	2153.169

be observed. But further, solutions have been improved very slowly. Second feature of the algorithm is clearly seen at Fig. 6. Solutions for the puzzle have been found in about 17 thousands iterations, but from 11th thousand iteration no improvements are reported. Minimal fitness value of population remains at 2 (0 is optimal solution).

6. Conclusions and future work

After the analysis of all test sets, the following conclusions may be reached:

- The Differential evolution is capable to solve a very difficult Sudoku puzzles. For example „hard" instance in 15 thousand iterations.
- The Proper mutation schema is very difficult to implement even with strong theoretic analysis of geometric operators.
- The Simple swap mutation brings very good results.
- The Algorithm is far better with finding solutions close to the optimal solution (for example the fitness function equal 2).

To summarize it should be observed as well that the Differential Evolution and other evolutionary algorithms are capable not only to find solutions for Sudoku puzzle but also to generate new puzzles from scratch. The new mutation schema seems to be very effective tool for solving dicrete problems. Our next goal is to adapt similar mutation into the continuous optimization problem. One of the most difficult things, will

Table 2. The comparison of how effectively GA finds solutions for the Sudoku puzzles with different difficulty ratings.

Difficulty rating	Cells given	Minimum number of iterations	Maximum number of iterations	Average number of iterations	Median	Std dev
New	0	6831	40802	20261.4	17462	11870.7
Easy	34	6251	15421	9111.2	7462	4397.7
Easy	30	4918	12572	6472.6	5150	4056.1
Medium	30	6253	26255	13921.4	7124	10559.5
Medium	26	7171	33498	19846.6	17463	9409.9
Hard	26	11563	21462	16312	16857	3987.2
Hard	24	10831	40802	20261.4	17462	11870.7
New generated 1	30	7513	24524	16598.4	16356	6578.3
New generated 2	30	8457	18436	11427	9135	4242.9
New generated 3	30	3476	31673	15062.6	12452	10409.1
New generated 4	30	4339	28463	13036	9687	9753.8

be descrption of the mutation schema, which will allow to modify only small part of the genotype even in the continuous optimization problems. The key issue should be the proper representation of the individual. We try to show, that appropriate problem transformation is one of the main problems for the described mutation schema.

References

[1] D. Ardel, TOPE and magic squares: a simple GA approach to combinatorial optimization, 1994, pp. 1–6.

[2] B. Felgenhauer, Jarvis F.,Mathematics of Sudoku I, 2006.

[3] B. Felgenhauer, Jarvis F., Mathematics of Sudoku II, 2006.

[4] M. Gold, Using Genetic Algorithms to come up with Sudoku Puzzles, 2005.

[5] K. U. Kasemir, K. Betzler., Detecting ellipses of limited eccentricity in images with high noise levels, Journal Image and Vision Computing, Vol. 21, 2003, pp. 221–227.

[6] J. Lauriere, A language and a program for stating and solving combinatorial problems, Journal of Artificial Intelligence, Vol. 10, 1978, pp. 29–127.

[7] E. Lawler, Kan A. Rinnooy, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, 1985.

[8] I. Lynce, J. Ouaknine, Sudoku as a SAT Problem, Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics, 2006.

[9] George D. Magoulas, Michael N. Vrahatis, George S. Androulakis, Effective backpropagation training with variable stepsize, Journal of Neural Networks, Vol. 10, No. 1, 1997, pp. 69–82.

[10] T. Mantere, J. Koljonen, Solving, rating and generating Sudoku puzzles with GA, IEEE Congress on Evolutionary Computation 2007, 2007, pp. 1382–1389.

[11] S. McGerty, Solving Sudoku Puzzles with Particle Swarm Optimisation, Final Report, Macquarie University 2009.

[12] A. Moraglio, C. Di Chio, J. Togelius and R. Poli, Geometric Particle Swarm Optimization - Research Article, Journal of Artificial Evolution and Applications, Vol. 2008, 2007.

[13] A. Moraglio, J. Togelius, Geometric Particle Swarm Optimization for the Sudoku Puzzle, GECCO 2007, Genetic and Evolutionary Computation Conference, 2007.

[14] A. Moraglio, J. Togelius, Geometric differential evolution, GECCO 2009, Genetic and Evolutionary Computation Conference, 2009, pp. 1705–1712.

[15] D. Mullaney, Using Ant Systems to Solve Sudoku Problems, University College Dublin.

[16] H. Simonis, Sudoku as a constraint problem, Proceedings 4th Int. Works. Modelling and Reformulating Constraint Satisfaction Problems, 2005, pp. 13–27

[17] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, Vol. 11, 1997, pp. 341–359.

[18] R. Storn, Differential evolution design of an IIR-filter, IEEE International Conference on Evolutionary Computation ICEC'96, 1996, pp. 268–273

[19] R. Storn, On the Usage of Differential Evolution for Function Optimization, AFIPS'96, 1996, pp. 519–523.

[20] T. Weber, A SAT-based Sudoku solver, 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2005, 2005, pp. 11–15.

[21] T. Yato, Complexity and Completeness of Finding Another Solution and its Application to Puzzles, IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. 5, 2003, pp. 1052–1060.

Urszula Boryczka, Przemysław Juszczuk

EWOLUCJA RÓŻNICOWA W ROZWIĄZYWANIU SUDOKU

Streszczenie: W artykule przedstawimy propozycję zastosowania algorytmu ewolucji różnicowej do rozwiązywania problemów kombinatorycznych. Przewagą ewolucji różnicowej jest zdolność do unikania optimów lokalnych w przestrzeni przeszukiwań. Specjalny operator mutacji pozwala ukierunkować proces poszukiwań rozwiązania. W ewolucji różnicowej stosowany jest operator selekcji, który promuje tylko najlepiej przystosowane osobniki z populacji rodziców i potomków. Przedstawimy zastosowanie opisanego algorytmu do problemu rozwiązywania Sudoku. Sudoku składa się z planszy 9 na 9, podzielonej na 9 sekcji - każda o rozmiarze 3 na 3 elementy. Każda z 81 kratek powinna zostać wypełniona wartością z przedziału 1 do 9. W artykule pokażemy, że ewolucja różnicowa pozwala na rozwiązywanie Sudoku.

Słowa kluczowe: ewolucja różnicowa, sudoku, optymalizacja dyskretna

AN ALGORITHM FOR GENERATING BINARY PSEUDO-RANDOM SEQUENCES

Wiktor Dańko

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In the paper it is presented an algorithm for generating pseudo-random binary sequences. There are formulated theorems concerning properties of the sequence generated by the algorithm. The sequence is not periodic. Moreover, for any natural number $n > 0$, the initial fragment of the generated sequence of the length $(2 \cdot n) \cdot 2^{(2 \cdot n)}$ contains all (binary) series of the length n.

Keywords: pseudo-random computer generators, computer simulations, probabilistic algorithms

1. Introduction

Computer modeling of real stochastic processes is mainly based on a pseudo-random computer generator.

In the present paper we shall be concerned with uniform binary pseudo-random generators, i.e., with the set $\{0,1\}$ of values.

We can restrict ourselves to the case of uniform binary pseudo-random generators because by means of such a binary generator, we can construct any uniform generator with the set of values of the form $\{0,1,...,m\}$, for arbitrary integer $m > 1$ (cf. [2], [5]). This construction essentially depends on the fact that the results of two successive random assignments

```
x:= random; x:= random;
```

realized by means of a pseudo-random generator, treated as random events, are independent.

Each uniform binary pseudo-random generator will be viewed in terms of an infinite sequence

$$b_0\, b_1\, b_2 \,...\, b_n \,...$$

of values b_n from $\{0,1\}$, $n = 0,1,2,...$, produced by the generator in succession, one after the other. We shall be interested in the case, where the sequence

$$\{b_n\}_{n=0,1,2,...}$$

is not periodic.

Typical information on a periodic pseudo-random computer generator concerns the set of obtained values and the period of the generator. Some users of computer generators believe, in an intuitive manner, that the credibility of results of computer simulations seems to essentially depend on the length of the period of the generator. But the dependence is not evident. Let us consider an example of a sequence produced by a generator with the following period sequence:

$$010011000111\ ...\ \underbrace{00...0}_{l}\underbrace{11...1}_{l}$$

of the length $p = l \cdot (l+1)$.
Speaking informally, since the frequencies of appearing of "0" and "1" are the same then this generator is uniform. On the other hand, it is easy to see that the sequence produced by generator does not contain any subsequence of the form

$$\underbrace{01...01}_{2k}$$

for $k \geq 2$. This means that this periodic sequence cannot be used as a straightforward (pseudo) random sequence, in spite of the fact that its period could be arbitrarily large. Other remarks related periodic pseudo-random generator will be formulated in the next section.

2. Remarks on periodic generators

Let us consider a fixed periodic binary pseudo-random generator producing a sequence

$$\{b_n\}_{n=0,1,2,...}\ .$$

We recall that this generator will be called periodic (cf. [13], [14]) if and only if there exist numbers q and $p > 0$ such that for $i \geq q$, $b_i = b_{i+j \cdot p}$ for $j = 0,1,2,....$ The sequence $b_q\, b_{q+1}\ ...\ b_{q+p-1}$ is called the period sequence of the sequence $\{b_n\}_{n=0,1,2,...}$.

We shall now introduce some definitions and notation.

By a series we shall understand any finite sequence c of the form

$$c_0\ c_1\ \dots\ c_{k-1}$$

where $c_i \in \{0,1\}$ for $i = 0,1,\dots,k-1$. The number k will be called the length of the series.

We shall say that a pseudo-random generator, producing a sequence $\{b_n\}_{n=0,1,2,\dots}$, realizes the series $c_0\ c_1\ \dots\ c_{k-1}$ if and only if there exists an index m, $m \geq 0$, such that the sequence $b_m\ b_{m+1}\ \dots\ b_{m+k-1}$ and the sequence $c_0\ c_1\ \dots\ c_{k-1}$ are identical. In the case, where such an index m does not exist, we shall say that the generator omits the series $c_0\ c_1\ \dots\ c_{k-1}$.

The following remark shows a shortcoming of periodic pseudo-random generators.

Remark 1.
Any periodic pseudo-random generator omits some series.

□

To argue this fact let us suppose that

$$b_q\ b_{q+1}\ \dots\ b_{q+p-1}$$

is the period sequence of a generator. It is easy to observe that the generator omits the sequence

$$b_q\ b_{q+1}\ \dots\ b_{q+p-1}\ (1-b_q)$$

(its length is length $p+1$). In reality, the minimal length of omitted series is essentially less than that period of the generator (cf. [5]).

Using the above remark we shall show that the use of periodic generators may lead to important differences between results of computer simulations and theoretically determined facts.

Let us consider a fixed binary pseudo-random generator producing a sequence $\{b_n\}_{n=0,1,2,\dots}$ and let $c_0\ c_1\ \dots\ c_{k-1}$ be a series omitted by the generator.

Let `P` denote the following program below, where the initial values of the variables `C`$_0$`,  C`$_1$`,  ...,  C`$_{k-1}$ are $c_0\ c_1\ \dots\ c_{k-1}$, respectively, and the realization of the instructions `x:=random{0,1}` consists in assigning to the variable `x` the values produced by the considered generator in succession, one after the other.

```
begin
     t:= 0;
     while (t = 0) do
        begin
           t:= 1;
           x:= random{0,1};
           if ((t=1)&(x=$C_0$)) then t:= 1 else t:= 0;
           x:= random{0,1};
           if ((t=1)&(x=$C_1$)) then t:= 1 else t:= 0;

                          .  .  .

           x:= random{0,1};
           if ((t=1)&(x=$C_{k-1}$)) then t:= 1 else t:= 0;
        end;
end;
```

It is easy to observe (cf. [3], [5]) that in the case, where the generator realizes the series $c_0\ c_1\ ...\ c_{k-1}$, the program P ends its computation and in the case, where the generator omits the series $c_0\ c_1\ ...\ c_{k-1}$ this program does not end its computation. This means that the results of computer simulations based on periodic pseudo-random generators may essentially differ from the result of computer simulations using straightforward random generators.

3. Grey codes

We start with the definition of Grey codes (cf. [1], [11]). First, we give examples of the sequences $C^{(l)} =< c_i^l,\ i = 0,1,...,2^l - 1 >$ of Grey codes of the length $l = 1,2,3$.

$$l = 1,\ C^{(1)} =< c_0^1, c_1^1 >=< 0,1 >,$$
$$l = 2,\ C^{(2)} =< c_0^2, c_1^2, c_2^2, c_3^2 >=< 00,10,11,01 >,$$
$$l = 3,\ C^{(3)} =< c_0^3, c_1^3, ..., c_7^3 >=< 000,100,110,010,011,111,101,001 > .$$

The induction step in defining the sequence $C^{(l+1)}$, provided that the sequence $C^{(l)}$ is defined, is the following:

$$\text{for } C^{(l)} =< c_0^{(l)}, c_1^{(l)}, ..., c_{2^l-2}^{(l)}, c_{2^l-1}^{(l)} >$$
$$\text{we define } C^{(l+1)} =< c_0^{(l)}0, c_1^{(l)}0, ..., c_{2^l-1}^{(l)}0, c_{2^l-1}^{(l)}1, ..., c_1^{(l)}1, c_0^{(l)}1 > .$$

Let us denote by $g(l,i,j)$ the j-th position of the i-th sequence $c_i^{(l)}$ of the sequence $C^{(l)}$, $i = 0,1,...,2^l-1$, $j = 0,1,...,l-1$. It is known that the binary values $g(l,i,j)$ can be defined inductively as follows (cf. [1], [11]):

$$g(1,0,0) = 0,\ g(1,1,0) = 1,$$
$$g(l+1,i,j) = \begin{cases} 0 & \text{for } i = 0,1,...,2^l-1 \text{ and } j = l+1 \\ g(l,i,j) & \text{for } i = 0,1,...,2^l-1 \text{ and } j = 0,1,...,l-1 \\ 1 & \text{for } i = 2^l, 2^l+1,...,2^{l+1}-1 \text{ and } j = l+1 \\ g(l,2^{l+1}-1-i,j) & \text{for } i = 2^l, 2^l+1,...,2^{l+1}-1 \\ & \quad\text{and } j = 0,1,...,l-1. \end{cases}$$

We shall now investigate whether the "infinite concatenation"

$$C^{(1)} \circ C^{(2)} \circ ... \circ C^{(l)} \circ ... \, .$$

of the sequences $C^{(l)}$, $l = 1,2,3,...$, could play the role of an uniform binary pseudo-random generator. Denote by

$$\{r_n\}_{n=0,1,2,...}$$

the sequence corresponding to the "infinite concatenation" of the sequences $C^{(l)}$, $l = 1,2,3,....$ Let us first note that the length $p(l)$ of the sequence

$$C^{(1)} \circ C^{(2)} \circ ... \circ C^{(l)}.$$

satisfies the inductive equation

$$p(0) = 0,\ p(1) = 2,$$
$$p(l+1) = p(l) + ((l+1)\cdot 2^{l+1}).$$

Let n be an arbitrary natural number. Denote by l the natural number such that

$$p(l) \le n < p(l+1)$$

and let

$$i = (n-p(l))\ div\ (2^l) \text{ and } j = (n-p(l))\ mod\ (2^l).$$

The sequence $\{r_n\}_{n=0,1,2,...}$ corresponding to the "infinite concatenation" of the sequences $C^{(l)}$ can be defined in the following way:

$$r_n = g(l,i,j)$$

where $l, g(l,i,j), p(l), i, j$ have the meaning as in the above. The following algorithm `R(n)` computes the elements of the sequence $\{r_n\}_{n=0,1,2,...}$. We shall assume that the program functions `P(l)`, `G(l,i,j)`, used in `R(n)`, have been previously programmed and compute the function $p(l)$, $g(l,i,j)$ defined above.

```
function R(n);
    begin
       if (n=0) then return(0)
       else if (n=1) then return(1)
            else
               begin
                  l:= 1;
                  while (P(l+1) <= n) do l:= l+1;
                  i:= (n-P(l))div(2^l);
                  j:= (n-P(l))mod(2^l);
                  return(G(l,i,j));
               end;
    end;
```

Remark 2.
The complexity of the algorithm `R(n)` *is* $O(n)$.

□

To argue this remark it is sufficient to note that the complexity of computation of the functions $p(l)$ and $g(l,i,j)$ is of the order $O(l)$.

Now, we recall the well known property o Grey codes of the length l (cf. [1], [11]):

Let $C^{(l)} = < c_0^{(l)}, c_1^{(l)}, ..., c_{2^l-2}^{(l)}, c_{2^l-1}^{(l)} >$ *be the sequence of Grey codes of the length l. For each* $j = 1,2,...,2^l-1$, *the codes* $c_{j-1}^{(l)}$ *and* $c_j^{(l)}$ *differ on only one position.*

This property causes that the sequence $\{r_n\}_{n=0,1,2,...}$ cannot be used as a pseudo-random generator.

Remark 3.
The algorithm `R(n)` *is of any use in pseudo-random generating binary sequences.*

It can be used in testing whether a binary pseudo-random generator realizes or omits particular binary series. Moreover, since `R(n)` *produces all finite binary sequences, it can be used in algorithm testing.*

□

The main motive for the above detailed presentation of the algorithm `R(n)` is the fact, that the idea of the construction of the binary pseudo-random generator, presented in Section 5, is analogous, in a way, to the construction of `R(n)`. Namely, similarly to the case of `R(n)`, the result sequence of the proposed generator can be viewed as an "infinite" concatenation

$$S^{(1)} \circ S^{(2)} \circ S^{(3)} \circ \ldots \circ S^{(l)} \circ \ldots .$$

of sequences, where the length of $S^{(i)}$ is equal to 8, for $i = 1$ and $(i \cdot 2^i - \frac{1}{2} \cdot i \cdot 2^{l-1})$ for $i > 1$. We shall formulate (cf. Theorem 1, Section 5) the fact, that the initial fragment

$$S^{(1)} \circ S^{(2)} \circ S^{(3)} \circ \ldots \circ S^{(l)} \circ \ldots .$$

of the generated sequence, being of the length $l \cdot 2^l$, realizes all series of the length $\frac{1}{2} \cdot l$.

To describe precisely the algorithm we need to introduce a notation and formulate some auxiliary mathematical facts. This will be done in the next section.

4. Mathematical preliminaries

The main fact formulated in this section is a lemma concerning a manner of natural number representation.

To formulate this lemma we need a notation. By d we shall denote natural numbers of the form 2^u, where $u = 0, 1, 2, .$ W shall now define:

$$\begin{aligned} &l(d) = 2^d, \\ &w(d) = l(d) \cdot d, \\ &s(d) = 2 \cdot l(d) \cdot l(d) \cdot d, \end{aligned}$$

For example, we have:

$$\begin{array}{llll} d = 1, & l(1) = 2, & w(1) = 2, & s(1) = 8, \\ d = 2, & l(2) = 4, & w(2) = 8, & s(2) = 64, \\ d = 4, & l(4) = 16, & w(4) = 64, & s(4) = 2048, \\ d = 8, & l(8) = 256, & w(8) = 2048, & s(8) = 1048576. \end{array}$$

Using the above notation we shall formulate the main.

Lemma

Let n be an arbitrary natural number, greater than 7 and let d be the number of the form 2^u, where $u = 1, 2, \ldots$, such that

$$s(d/2) \leq n < s(d).$$

Then there exist natural numbers i, j, k, such that

$$n = i \cdot (2 \cdot w(d)) + j \cdot d + k$$

and such that

$$0 \leq i < l(d),\ 0 \leq j < 2 \cdot l(d),\ 0 \leq k < d.$$

The numbers i, j, k, are determined univocally.

□

The inductive proof of the lemma is rather technical and is omitted. We end this section with two examples illustrating the lemma.

Example 1.
$n = 47$.
For $d = 2$ we have:
$8 = s(1) \leq 47 < s(2) = 64$,
$2 \cdot w(2) = 2 \cdot 8 = 16$.
For the values $i = 2, j = 7, k = 1$, satisfying
$0 \leq i = 2 < l(d) = 4$, $0 \leq j = 7 < 2 \cdot l(d) = 8$, $0 \leq k = 1 < d = 2$,
we have
$47 = i \cdot (2 \cdot w(d)) + j \cdot d + k = i \cdot 16 + j \cdot 2 + k = 2 \cdot 16 + 7 \cdot 2 + 1$.

Example 2.
$n = 1831$.
For $d = 4$ we have:
$64 = s(2) \leq 1831 < s(4) = 2048$,
$2 \cdot w(4) = 2 \cdot 64 = 128$.
For the values $i = 14, j = 9, k = 3$, satisfying
$0 \leq i = 14 < l(d) = 16$, $0 \leq j = 9 < 2l(d) = 32$, $0 \leq k = 3 < d = 4$,
we have
$1831 = i \cdot (2 \cdot w(d)) + j \cdot d + k = i \cdot 128 + j \cdot 4 + k = 14 \cdot 128 + 9 \cdot 4 + 3$.

5. The algorithm

Using the notation introduced in the preceding section we can describe the proposed algorithm generating a binary pseudo-random sequence. For readability of inductive equations, describing the sequence generated by the algorithm, elements of the sequence will be denoted by

$$\{B(n)\}_{n=0,1,2,\ldots}.$$

The recursive definition of the sequence is as follows:

(A) Initial values:
We define the values $B(n)$, for n satisfying $n < s(1) = 8$, i.e., for $n = 0,1,2,3,4,5,6,7$:

$$B(0) = 1, B(1) = 1, B(2) = 0, B(3) = 1,$$
$$B(4) = 0, B(5) = 0, B(6) = 1, B(7) = 0.$$

(B) Induction step:
Assume that the elements $B(n)$ are defined for all values n satisfying

$$n < s(d/2).$$

We shall now define the values for n satisfying

$$s(d/2) \leq n < s(d),$$

i.e., for

$$n = i \cdot (2 \cdot w(d)) + j \cdot d + k,$$

where $i = 0,1,...,l(d)-1$, $j = 0,1,...,2 \cdot l(d)-1$, $k = 0,1,...,d-1$.

(B$_0$) We first define the values $B(n)$, for n satisfying

$$w(d) = s(d/2) \leq n < 2 \cdot w(d) :$$

Namely,

$$B(n) = B(w(d) + j \cdot d + k) = B([(l(d) - j - 1) mod l(d)] \cdot d + k)$$

for $j = 0,1,...,l(d)-1$, $k = 0,1,...,d-1$.
Thus we have defined the values $B(n)$, for $n = i \cdot (2 \cdot w(d)) + j \cdot d + k$, where $i = 0$ and $j = 0,1,...,2l(d)-1$, $k = 0,1,...,d-1$.

(B$_1$) Now, we define

$$B(n) = \begin{cases} B((i-1) \cdot 2 \cdot w(d) + j \cdot d + k) \\ \qquad\qquad \text{if } (i \text{ is even and } j \text{ is even}), \\ B((i-1) \cdot 2 \cdot w(d) + [(j+2) mod (2 \cdot l(d)] \cdot d + k) \\ \qquad\qquad \text{if } (i \text{ is even and } j \text{ is odd}), \\ B((i-1) \cdot 2 \cdot w(d) + [(j-2) mod (2 \cdot l(d)] \cdot d + k) \\ \qquad\qquad \text{if } (i \text{ is odd and } j \text{ is even}), \\ B((i-1) \cdot 2 \cdot w(d) + j \cdot d + k) \\ \qquad\qquad \text{if } (i \text{ is odd and } j \text{ is odd}) \end{cases}$$

for

$$n = i \cdot (2 \cdot w(d)) + j \cdot d + k,$$

where

$$\begin{aligned} & i = 1, 2, ..., l(d) - 1, \\ & j = 0, 1, ..., 2 \cdot l(d) - 1, \\ & k = 0, 1, ..., d - 1. \end{aligned}$$

This ends the induction step.

The above inductive definition of the sequence $\{B(n)\}_{n=0,1,2,...}$ is not convenient for computer implementation. However, it enables us to formulate theorems concerning properties of the sequence.

6. Properties of the sequence generated by the algorithm

In this section we formulate (without proofs) several facts about the sequence $\{B(n)\}_{n=0,1,2,...}$ that concern its probabilistic properties.

Theorem 1.
For any natural number $n > 0$, the initial fragment of the generated sequence of the length $(2 \cdot n) \cdot 2^{(2 \cdot n)}$ contains all (binary) series of the length n.

□

Corollary 1.
The sequence $\{B(n)\}_{n=0,1,2,...}$ is not periodic.

□

Let a denote a fixed series $a_0 a_1 ... a_{d-1}$ of the length d being of the form $d = 2^u$, where $u = 0, 1, 2, ...$.

By $N_a(n)$ we shall denote the number of occurrences of the series a in the initial fragment $B(0)B(1)...B(n)$.

Let us also define

$$F_a(n) = \frac{N_a(n)}{n}.$$

The intuitive meaning of $F_a(n)$ is the frequency of appearing of the series a in the initial fragment $B(0)B(1)...B(n)$.

Theorem 2.
Let a and b denote fixed series of the same length d.
Then

$$\lim_{n\to\infty} \frac{N_a(n)}{N_b(n)} = 1$$

$\square$

Let us treat the results of successive execution of two random assignments

```
x:= random; x:= random;
```

realized by means of a pseudo-random generator, as random events.
The following theorem concerns, in a way, the independence of such events.

Theorem 3.
Let a and b denote fixed series of the same length d.
Then

$$\lim_{n\to\infty} n \cdot \frac{N_{a\circ b}(n)}{N_a(n)\cdot N_b(n)} = \lim_{n\to\infty} \frac{F_{a\circ b}(n)}{F_a(n)\cdot F_b(n)} = 1$$

$\square$

The independence of results of random assignments based on a pseudo-random generator is of great importance for modeling real stochastic processes (cf. [3]).

7. Final remarks

The proofs of mathematical facts formulated in the paper are too long for presentation in this publication and therefore are omitted here. They will be presented in separate papers.

The question of nonrecursive implementation of the inductive definition of the sequence $\{B(n)\}_{n=0,1,2,...}$ will be discussed in a separate paper. This paper will also contain an analysis of the complexity of such an implementation.

References

[1] Cormen T., Leiserson C., Riverst R., Stein C., Introduction to Algorithms, Massachusetts Institute of Technology, 2001.

[2] Dańko A., Dańko W., Improving Pseudo-Random Generators, International Conference on Biometrics and Kansei Engineering, 24-28 June Cieszyn, Poland, pp. 163-166, 2009.

[3] Dańko W., The Set of Probabilistic Algorithmic Formulas Valid in a Finite Structure is Decidable with Respect to Its Diagram, Fundamenta Informaticae, vol. 19 (3-4), pp. (417-431), 1993.

[4] Dańko W., Remarks on Computer Simulations, Biometrics, Computer Security Systems & Artificial Intelligence Applications, red.: Khalid Saeed, Jerzy Pejaś, Romuald Mosdorf, Springer, Science+Business Media, LLC, New York, pp. 197-206, 2006.

[5] Dańko W., Algorithmic Models of Probabilistic Processes, Politechnika Białostocka, (in Polish, to appear).

[6] Feller W., An Introduction to Probability Theory and Its Applications, John Wiley and Sons, Inc., New York, London, 1961.

[7] Gentle J.E., Random Number Generation and Monte Carlo Methods, Springer, 2003.

[8] Jessa M., Designing Security for Number Sequences Generated by Means of The Sawtooth Chaotic Map, IEEE Transactions on Circuits and Systems - I: Regular Papers, vol. 53 (5), pp. 1140-1150, 2006.

[9] L'Ecuyer P., Random Numbers for Simulation, Comm. ACM 33 (10), pp. 85-97.

[10] L'Ecuyer P., Tezuka S., Structural Properties for Two Classes of Combined Random Number Generators, Math. Comput. 57, pp. l35-746, 1991.

[11] Lipski W., Combinatorics for Computer Scientists, WNT, Warszawa, 1985, (in Polish).

[12] Zieliński R., Random Numbers Generators, WNT, Warszawa, 1972, (in Polish), Transaction on Circuit and Systems - I:Regular Papers, vol. 53, 5, pp. 1140-1150.

[13] Zieliński R, Wieczorkowski R., Computer Random Numbers Generators, WNT, Warszawa, 1997, (in Polish).

[14] Tezuka S., A Unified View of Large-Period Random Number Generators, J. Oper. Res. Soc. Japan. 37, pp. 211-227.

ALGORYTM GENEROWANIA PSEUDOLOSOWYCH CIĄGÓW BINARNYCH

Streszczenie: W pracy został przedstawiony algorytm generowania pseudo-losowego ciągu binarnego. Sformułowane zostały twierdzenia dotyczące własności otrzymanego ciągu. Nie jest to ciąg okresowy i dla dowolnego $n > 0$, początkowy odcinek ciągu o długości $(2 \cdot n) \cdot 2^{(2 \cdot n)}$ zawiera wszystkie serie binarne o długości n.

Słowa kluczowe: pseudolosowy generator komputerowy, symulacja komputerowa, algorytm probabilistyczny

Artykuł zrealizowano w ramach pracy badawczej S/WI/1/2011.

FINITE STATE MACHINES POWER DISSIPATION CLASSIFICATION

Tomasz Grześ

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: Reduction of the power consumption of digital system can be obtained in many ways. Integrated circuits fabricated in CMOS technology consume power when the state of the output of logic element (gate or flip-flop) changes into opposite. Therefore minimizing the number of such changes lead to a reduction of the power consumption. In this paper is presented research of dependence the power dissipation in finite state machines (FSMs) on both probabilities of ones on input lines and probabilities of changes in the input value. The classification scheme for graphs obtained for those dependencies is also proposed. This classification can be used for testing the results of the power reduction process as well as testing the behavior of finite state machine while changing the statistical properties of input signals. Proposed classification can also be used for developing new methods and algorithms of reducing the power dissipation in finite state machines.

Keywords: power dissipation, FSM, power classification

1. Introduction

The problem of reducing the power consumption of digital systems has been an interest of many scientists in the past few years. The special incentive for intensification the research was a huge increase in the popularity of mobile devices (mobile phones, portable computers, including laptops, PDAs and netbooks). It was also important for the global trend to develop energy-efficient, and therefore environment friendly systems. Power reduction can also increase the performance and reliability of the system.

Value of the power dissipated in CMOS digital circuit can be obtained using the following formula [3]

$$P_a = \frac{1}{2} V_{DD}^2 f C_{out} N_a \tag{1}$$

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 9, pp. 31-44, 2012.

where: P_a - is power dissipated by element a; V_{DD} - is the power supply voltage; f - is the operating frequency; C_{out} - is the output capacitance; N_a - is the switching activity (the number of output value transitions is one cycle of f).

According to equation (1) reduction of the power dissipation can be obtained through decreasing value of factors V_{DD}, f, C_{out}, or N_a. But only switching activity can be reduced using algorithmic solutions. Examples of such approach are: using the specific state assignment methods ([2], [5], [9], [12]), decomposition of large circuits [10], using embedded memory blocks (in FPGA) [6]. Review the most known methods of state assignment can be found in [14].

One of the stages of low-power design and power-aware design is to estimate the power consumption of the digital system. Classical techniques for power estimation of finite state machine (FSM) were presented in [3], and [4]. In order to calculate the power consumed by FSM it is necessary to know either the transitions' graph or transitions' table (or list). Additionally the knowledge about the properties of the input sequence is needed. Mainly these properties are probabilities of input vector, which depends on probability of one (or zero) on input line [13].

The aim of the research presented in this paper is to determine the effect of power dissipation depending on the probability of the ones depending on the input and the frequency of changes in the value of the inputs of finite state machine (FSM).

2. Dependence of power on probabilities

2.1 Research methodology

The power value of the FSM is proportional to the switching activity (the frequency of switching the value in output) of the flip-flops carrying a memory function. Switching activity estimation process consists of three steps [7]:

- calculate the static probabilities of each state of the finite state machine (FSM) using the Chapman-Kolmogorov equations; static probability is a probability that the FSM is in a particular state at time t;
- calculate the transition probabilities ie. probabilities that FSM changes its state from one state to another;
- calculate the switching activity of flip-flops.

Based on the calculated flip-flop switching activity can be calculated power dissipated by each flip-flop, then the total power consumed by the system.

Assuming equal probability of ones on all inputs of FSM the dependence of the power dissipation on the probability of ones can be determined. The value of

$P(x_i = 1)$ varies in the range [0, 1], where $P(x_i = 1)$ is a probability that input line x_i is carrying logic one.

For different values of the probability $P(x_i = 1)$ were obtained different values of power dissipation. The following probabilities' values were taken into account: 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 0.99. The results formed curves that for a variety of FSMs arranged in a characteristic way. When analyzing the generated graphs of power consumption of a FSM the following classification scheme was proposed [8], [11]:

1. Class 1 (Gaussian-like graphs, Figure 1a), which is divided into sub-classes:
   - 1 - symmetric Gaussian distribution graph for which the maximum value falls in the middle of the graph (probability value equal to 0.5);
   - 1L - asymmetric Gaussian distribution graph for which the maximum value falls on the left half of the graph (the probability value of less than 0.5);
   - 1R - asymmetric Gaussian distribution graph for which the maximum value falls on the right half of the graph (for values greater than 0.5 probability).
2. Class 2 (graphs a constant value, Figure 1b): the power value does not depend on the value of probabilities of the ones on the input line.
3. Class 3 (linear graphs, Figure 1c), which is divided into sub-classes:
   - 3R - graph linearly rising;
   - 3F - graph linearly falling;
4. Class 4 (logarithmic graphs, Figure 1d), which is divided into sub-classes:
   - 4R - graph logarithmically rising;
   - 4F - graph logarithmically falling;
5. Class 5 (exponential graphs, Figure 1e), which is divided into sub-classes:
   - 5R - graph exponentially rising;
   - 5F - graph exponentially falling.
6. Class 6 (parabolic graphs, Figure 1f), which is divided into sub-classes:
   - 6 - symmetric parabolic graph for which the minimum value falls in the middle of the graph (probability value equal to 0.5);
   - 6L - asymmetric parabolic graph for which the minimum value falls on the left half of the graph (the probability value of less than 0.5);
   - 6R - asymmetric parabolic graph for which the minimum value falls on the right half of the graph (for values greater than 0.5 probability).

In some cases, the graph cannot be classified exactly in the class due to slight inaccuracies. In such cases, the graph was indicated by the prefix "~", which has formed additional classes: ~1, ~1L, ~1R, ~2, ~3R, ~3F, ~4R, ~4F, ~5R, ~5F, ~6, ~6L, ~6R.

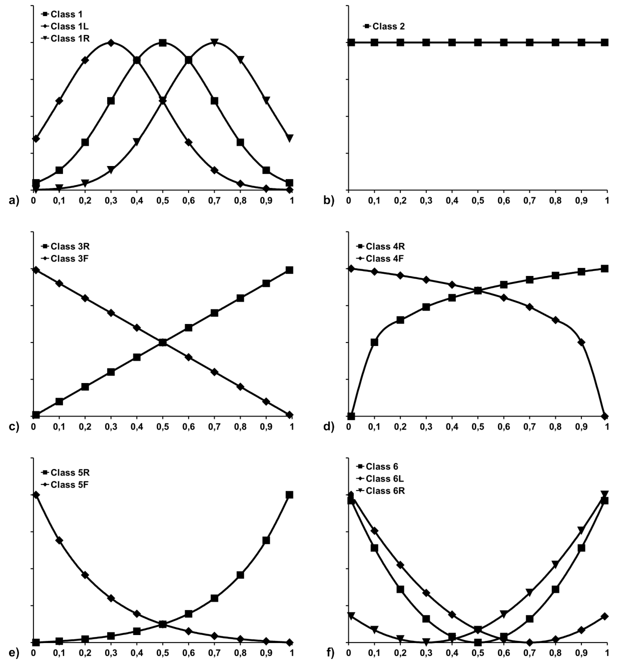


Fig. 1. Graphs for classes 1, 1L, 1R (a), 2 (b), 3R, 3F (c), 4R, 4F (d), 5R, 5F (e), 6, 6L, 6R (f)

3. Experimental results

Table 1 shows the classification of graphs created for standard benchmark circuits [1]. In the column "Benchmark" is placed the name of the benchmark. Column "Class" contains the name of the class to which the graph was assigned, the column "Binary" contains the results for the FSM encoded using binary encoding (the code of the state corresponds to the number of binary encoded number of state), while the column "one-hot" contains the results for the FSM encoded using one-hot encoding (number of state code in a one-hot encoding).

In addition in columns "Binary" and "One-hot" is indicated that the used encodings gave a higher maximum score (using the "+") or whether the results are identical or nearly identical (with "*"). Next columns describe the differences in the classification of the FSM encoded using binary and one-hot encoding. The column "Exact" specifies that in both cases the results are in the same class, while the column "Approximate" indicates that in both cases the results are in the same approximate class (preceded by "~"). The last column "Mixed" contains information about the classification of the mixed power graphs for FSMs encoded using binary encoding and one-hot encoding, the "W" means that both graphs are classified into different subclasses of one class, while "S" - when the graphs fall into two different classes.

Table 2 shows the allocation of power dependence graph of benchmark for each class. The column "Class" indicates the name of the class, while the column "Benchmark" indicates the benchmark name assigned to the class. If the graph is classified only for benchmark which has been encoded with binary code, the name has an extension ".b" added to the end of its name, and when the graph is classified only for benchmark that has been encoded with one-hot code, the name has an extension ".o". For some classes there were no benchmarks assigned, e.g. class 6, class 6L, and class 6R.

For the calculation 51 examples were used, and all of them have been encoded with two codes what in result gave 102 graphs. Most of the graphs were assigned to classes 1, 1L and 1R (57 charts, which gives 56%). In addition, one graph (s27.o) belongs to the class ~1, so that the total number of graphs belonging to different variants of the class 1 is equal to 58 (57%). The four graphs (dk27.o, and both charts dk512.o example tav) belonged to class 2, while five (ex1.b, planet.o, planet11.o, and both graphs of benchmark S510) to the class ~2 (4% and 5% respectively). Class 3R, 4R and 5R was represented by 7 graphs (dk27.b, ex2.o, mark1.b and ex4 and modulo12 at both encoding methods), 4 graphs (dk16, dk17) and 4 graphs (s1488, s1494) respectively, (7% , 4%, and 4%), and the classes 3F, 4F and 5F was represented by 1 graph (ex3.o), 2 graphs (keyb.o, s298.o) and 3 graphs (keyb.b, planet.b, planet11.b) respectively (1%, 2% and 3%). To class ~3R there were assigned 10 charts, 2 graphs belong to the class ~4R, and to the class ~3F - 2 charts. None graphs were assigned to the following classes: 6, 6L, 6R, ~1L, ~1R, ~4F, ~5R, ~5F, ~6, ~6L, ~6R.

In 16 of 51 cases (31%) curves for FSMs encoded with a binary code were assigned for other classes than the same FSMs encoded with one-hot code, wherein in 7 cases there were different subclasses within a single class (beecount, donfile, ex2, lion, lion9, mark1 and S27), and 9 were the different classes (dk27, dk512, ex1, ex3, EX7, keyb, planets, planet11 and s298).

Qualification for more than half the graphs to classes 1, 1L and 1R partially confirms the hypothesis that the graph of the power of the probabilities of ones on the inputs for FSM corresponds to a Gaussian distribution.

Over 30% of the graphs were classified into different classes and subclasses, depending on the encoding method used. Therefore it can be concluded that the encoding of internal states of FSM has a very large impact on the power consumption characteristics. This is particularly evident in the examples ex3 and ex7, which resulted in the classification of the binary encoding to the class ~3R (where the power increases with an increase in the probabilities of ones), while the one-hot encoding

Table 1. Classification of the dependence of power dissipation of the standard benchmarks on probabilities of ones on inputs

Benchmark	Class		Exact	Approximate	Mixed
	Binary	One-hot			
bbara	1R	1R+	+		
bbsse	1R*	1R*	+		
bbtas	1L	1L+	+		
bbtas2a)	1R	1R+	+		
beecount	1	1L+			W
cse	1L	1L+	+		
dk14	1	1	+		
dk15	~3R	~3R+		+	
dk16	4R+	4R	+		
dk17	4R	4R+	+		
dk27	3R	2			S
dk512	~3R	2			S
donfile	1R+	1			W
ex1	~2+	~3R			S
ex2	~3R+	3R			W
ex3	~3R*	3F*			S
ex4	3R	3R+	+		
ex5	~4R*	~4R*		+	
ex6	1R	1R+	+		
ex7	~3R*	~3F*			S
keyb	5F	4F+			S
lion	1	1R+			W
lion9	1L+	1R			W
mark1	3R	~3R+			W
mc	~3R	~3R+		+	
modulo12	3R	3R+	+		
opus	1L	1L+	+		
planet	5F+	~2			S
planet11	5F+	~2			S
pma	1R+	1R	+		
s1	1L+	1L	+		
s1488	5R	5R+	+		
s1494	5R	5R+	+		
s1a	1L+	1L	+		
s208	1	1	+		
s27	1	~1+			W
s298	~3F+	4F			S
s386	1R	1R+	+		
s420	1	1	+		
s510	~2*	~2*		+	
s8	1L	1L+	+		
s820	1L	1L+	+		
s832	1L	1L+	+		
sand	1R+	1R	+		
shiftreg	1	1	+		
sse	1R*	1R*	+		
styr	1R	1R+	+		
tav	2	2	+		
tma	1	1	+		
train11	1L	1L+	+		
train4	1L	1L+	+		

Table 2. Allocation of graphs for benchmarks according to classes

Class	Benchmarks
1	beecount.b, dk14, donfile.o, lion.b, s208, s27.b, s420, shiftreg, tma
1L	bbtas, beecount.o, cse, lion9.b, opus, s1, s1a, s8, s820, s832, train11, train4
1R	bbara, bbsse, bbtas2, donfile.b, ex6, lion.o, lion9.o, pma, s386, sand, sse, styr
2	dk27.o, dk512.o, tav
3R	dk27.b, ex2.o, ex4, mark1.b, modulo12
3F	ex3.o
4R	dk16, dk17
4F	keyb.o, s298.o
5R	s1488, s1494
5F	keyb.b, planet.b, planet11.b
6	
6L	
6R	
1	s27.o
1L	
1R	
2	ex1.b, planet.o, planet11.o, s510
3R	dk15, dk512.b, ex1.o, ex2.b, ex3.b, ex7.b, mark1.o, mc
3F	ex7.o, s298.b
4R	ex5
4F	
5R	
5F	
6	
6L	
6R	

- the class ~3F (where the power decreases with an increase in the probabilities of ones).

Therefore, it is possible to develop a method of encoding the FSM internal states, which would provide the desired characteristics of the power depending on the probabilities of the ones on the input lines.

4. Dependence of power on frequency

Described above dependence do not give a complete picture of the characteristics of the power consumption of a FSM. It is also appropriate to examine the dependence of the FSM power on the input signal change frequency (clock frequency must be greater than the frequency of the input signal changes). Therefore, it is a need to go on the characteristics of the probabilities of ones on inputs to the frequency of changes in the value of the input signal.

4.1 Research methodology

We can assume that the maximum frequency of the signal is obtained by $P(x_i = 1) = 0.5$, which is the equally likely changed from 1 to 0 and vice versa. By increasing $P(x_i = 1)$ from 0,5 to 1, and the reduction from 0,5 to 0 the frequency changes of the signal will be decreased. To calculate the probability of a change of the signal on input it is possible to use the following expression:

$$P(x_i) = 2 \times P(x_i = 0) \times P(x_i = 1) \tag{2}$$

To create a graph of the dependency of power dissipated in the FSM on probability of changing the input value, it should be noted that for the value of the probability of ones on input lines equal $P(x_i = 1) = p$ and $P(x_i = 1) = 1 - p$ (where $p \in (0; 0.5)$) the frequency of changes of the input signal is the same. Power value is therefore the sum of the power obtained for $P(x_i = 1) = p$ and $P(x_i = 1) = 1 - p$.

In other words, the graph of the dependency of power dissipated in the FSM on the probability of ones on input lines allowed to create new graphs based on the probability of changes in the input signal, by reflection axis $P(x_i = 1) = 0.5$, and adding up the curves obtained. In the resulting graph, the horizontal axis is the frequency of the signal. the discussed process is shown schematically in Figure 2.

Using a similar strategy, as in the previous section there was developed a FSM power graphs classification based on the resulting characteristics:

1. Class LR - the graphs are rising logarithmic curves (Figure 3a).

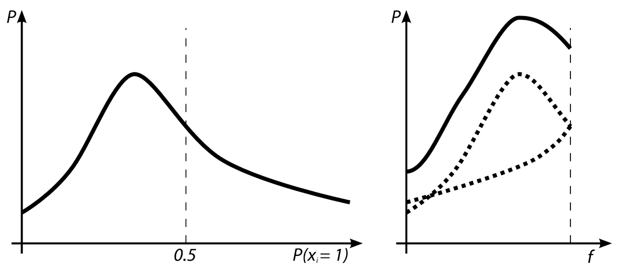


Fig. 2. Creating a graph based on the probability of changes in the input signal from a graph of the dependency of power dissipated in the FSM on the probability of ones on the input lines

2. Class LF - the graphs are falling logarithmic curves (Figure 3b).
3. Class C - the graphs are horizontal lines (Figure 3c).
4. Class IR - the graphs are rising in an irregular manner (Figure 3d).
5. Class IF - charts are falling in an irregular manner (Figure 3e).

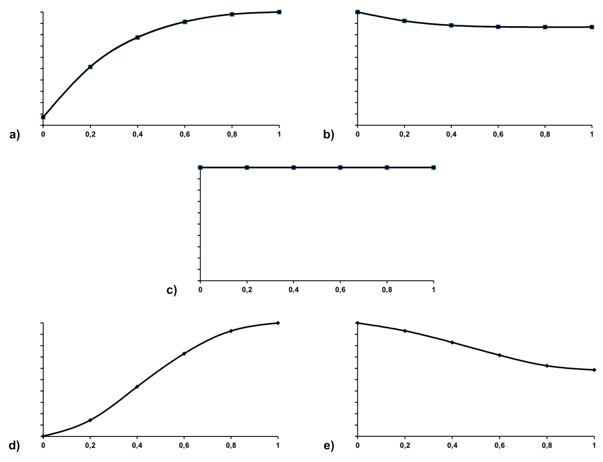


Fig. 3. Graphs for classes LR (a), LF (b), C (c), IR (d), and IF (e)

5. Experimental results

Table 3 shows the classification of the dependency of power dissipated by FSM on the frequency of the input signal changes for standard benchmarks [1]. In the column

"Benchmark" placed the name of the test. Column "Class" contains the name of the class to which a power graph was assigned, the column "Binary" contains the results for the FSM encoded using binary encoding, while the column "one-hot" contains the results for the FSM encoded using one-hot encoding. Next columns define the differences in the classification of the system encoded in binary and one-hot encoding. The column "Exact" specifies that in both cases the results are in the same class, while the column "Approximate" indicates that in both cases the results are in the same approximate class (preceded by "~"). The last column "Mixed" contains information about the mixed dependence graph classification for FSMs encoded using binary and one-hot encoding.

According to Table 3 it can be noticed that the most graphs are qualified to class LR (76 out of 102, giving 74.5%). To class LF there were qualified seven graphs (ex4, keyb.b, mc, planet.b and planet11.b) which gives 6.9%, while to class C - 8 graphs (dk27.o, ex2.o, planet.o, planet11.o and both graphs for modulo12 and tav) which gives 7.8%. To class IR 3 graphs were qualified (cse and keyb.o) which gives 2.9%, while the class IF - 4 graphs (s1488 and s1494) that gives 3.9%. To class C 4 were qualified (3.9%) graphs (dk27.b, dk512.o, mark1.b, s298.b). 7 graphs (13.8%) were classified into two different classes depending on the used encoding (dk512, ex2, keyb, mark1, planets, planet11, s298), and 1 benchmark (dk27) was assigned to the same but approximate class.

Variability in the classification of graphs of power depending on the frequency of the input signal changes to different classes when different methods of encoding FSM internal states were used, says the existence of the large impact of encoding methods to power. This is particularly evident for benchmark "keyb", where using the binary encoding leads to a qualification to Class LF, while using the one-hot encoding - to IR class. Also, in the examples "planet" and "planet11" it is visible the shift from class LF (binary coding) to class C (one-hot encoding). In these examples, the power dissipation is increasing with decreasing the frequency during the transition from the one-hot encoding to the binary encoding.

Some examples are classified to class LF (7 examples, 6.9%), and IF (4 examples, 3.9%), for which the frequency of the input signal changes is increasing while the power consumed by the FSM decreases. At the same time, for some benchmarks encoded using the different methods of encoding (binary and one-hot) the graphs of power consumption on the probability of changing input signals were classified into different classes. This means that it is possible to provide such a method of encoding the internal states of a FSM, which allows to obtain the desired relationship between power (corresponding to a specific class) and a FSM operating frequency (for exam-

Table 3. Classification of the dependence of power dissipation of the standard benchmarks on changes of frequency of input signals

Benchmark	Class		Exact	Approximate	Mixed
	Binary	One-hot			
bbara	LR	LR	+		
bbsse	LR	LR	+		
bbtas	LR	LR	+		
bbtas2a)	LR	LR	+		
beecount	LR	LR	+		
cse	IR	IR	+		
dk14	LR	LR	+		
dk15	LR	LR	+		
dk16	LR	LR	+		
dk17	LR	LR	+		
dk27	~C	C		+	
dk512	LR	~C			+
donfile	LR	LR	+		
ex1	LR	LR	+		
ex2	LR	C			+
ex3	LR	LR	+		
ex4	LF	LF	+		
ex5	LR	LR	+		
ex6	LR	LR	+		
ex7	LR	LR	+		
keyb	LF	IR			+
lion	LR	LR	+		
lion9	LR	LR	+		
mark1	~C	LR			+
mc	LF	LF	+		
modulo12	C	C	+		
opus	LR	LR	+		
planet	LF	C			+
planet11	LF	C			+
pma	LR	LR	+		
s1	LR	LR	+		
s1488	IF	IF	+		
s1494	IF	IF	+		
s1a	LR	LR	+		
s208	LR	LR	+		
s27	LR	LR	+		
s298	~C	LR			+
s386	LR	LR	+		
s420	LR	LR	+		
s510	LR	LR	+		
s8	LR	LR	+		
s820	LR	LR	+		
s832	LR	LR	+		
sand	LR	LR	+		
shiftreg	LR	LR	+		
sse	LR	LR	+		
styr	LR	LR	+		
tav	C	C	+		
tma	LR	LR	+		
train11	LR	LR	+		
train4	LR	LR	+		

ple, encode internal states in such a way as to reduce the power consumption with increase the frequency of the input signal).

6. Conclusions and future work

In presented paper examined the relationship between power consumed by a FSM and the probabilities of ones on the input lines. The results were presented in the form of graphs and their classification done. Partially confirmed the hypothesis that the characteristics of the power consumption as a function of the likelihood corresponds to the Gaussian distribution (56% of the graphs were classified as class 1, 1L and 1R).

Detected a big impact of encoding of internal states of the FSM in the shape of a graph of power dependence on the probabilities of ones on input lines, and thus the assignment to the class. Over 30% of all cases were classified into two different classes on the use of two forms of encoding: binary and one-hot.

Examined the relationship power consumed by a FSM on the probability of changes in the value of the inputs. The results were presented in the form of graphs and their classification done. The encoding had a big impact on the dependence of power on frequency of input signals changes - graphs of 7 examples (less than 14%) were assigned to different classes using different methods of encoding (binary and one-hot). Some graphs are classified into classes LF and IF, in which the frequency increases with decreasing power.

Therefore, it is possible to obtain the desired power consumption characteristics depending on the frequency of the input signal changes by using a FSM equivalent transformations and related forms of encoding the internal states of a FSM.

References

[1] S. Yang, Logic Synthesis and Optimization Benchmarks User Guide:Version 3.0, "Technical Report", Microelectronics Center of North Carolina, 1991, 43 p.

[2] L. Benini, G. De Micheli, State Assignment for Low Power Dissipation, IEEE Journal on Solid-state Circuits, Vol. 30, No. 3 (1995), pp. 259-268.

[3] C.-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A.M. Despain, B. Lin, Power Estimation Methods for Sequential Logic Circuits, IEEE Transactions on VLSI Systems, Vol. 3, No. 3 (1995), pp. 404-416.

[4] M. Pedram, Power simulation and estimation in VLSI circuits, "The VLSI Handbook". Edited by W-K. Chen, The CRC Press and the IEEE Press, 1999.

[5] S. Chattopadhyay, Low Power State Assignment and Flipflop Selection for Finite State Machine Synthesis – a Genetic Algorithmic Approach, IEE Proceedings – Computers and Digital Techniques, Vol-ume: 148, Issue: 45, 2001, pp. 147-151.

[6] A. Tiwari, K.A. Tomko, Saving Power by Mapping Finite-state Ma-chines into Embedded Memory Blocks in FPGAs, Proceedings of Design, Automation and Test in Europe Conference and Exhibition, 16-20 Feb. 2004, Vol. 2, pp. 916-921.

[7] V. Salauyou, T. Grześ, Obliczanie mocy układów sekwencyjnych, "Elektronika" nr 4 (2005), str. 25-27.

[8] V. Salauyou, T. Grześ, Classification of dependence of power dissipated by sequential circuit on input signal parameters, „Image analysis, computer graphics, security systems and artificial intelligence applications”, Wyższa Szkoła Finansów i Zarządzania, Białystok 2005, pp. 139-149.

[9] W.-T. Shiue, Novel State Minimization and State Assignment in Finite State Machine Design for Low-power Portable Devices, Integration, the VLSI Journal, Volume 38, Issue 4 (April 2005), pp. 549-570.

[10] Y. Xia, X. Ye, L. Wang, W. Tao, A. Almaini, A Uniform Framework of Low Power FSM Partition Approach, International Conference on Communications, Circuits and Systems Proceedings, Guilin, 25-28 June 2006, Volume 4, p. 2642-2647.

[11] V. Salauyou, T. Grzes, Zavisimost’ potreblâemoj mošnosti konečnyh avtomatov ot veroâtnosti izmeneniâ vhodnyh signalov, 6. meždunarodnaâ konferenciâ CAD DD’2007, Minsk, 14-15 noâbrâ 2007 g., str. 282-300.

[12] V. Salauyou, T. Grzes, FSM State Assignment Methods for Low-power Design, Proceedings of 6th International Conference on Computer Information Systems and Industrial Management Applications: CISIM’2007, Ełk, June 28-30, IEEE Computer Society, Los Alamitos 2007, pp. 345-348.

[13] T. Grzes, V. Salauyou, I. Bulatava, Power estimation methods in digital circuit design, Optoelectronics, Instrumentation and Data Processing, 2009, V. 45, No. 6, pp. 576-583.

[14] T. Grzes, V. Salauyou, I. Bulatava, Algorithms of coding the internal states of finite-state machine focused on the reduced power consumption, Radioelectronics and Communications Systems, 2010, V. 53, No. 5, pp. 265-273.

Tomasz Grześ

KLASYFIKACJA POBORU MOCY AUTOMATÓW SKOŃCZONYCH

Streszczenie: Zmniejszenie zużycia energii układu cyfrowego można uzyskać na wiele sposobów. Układy scalone wykonane w technologii CMOS zużywają moc, gdy stan na wyjściu elementu logicznego (bramki lub przerzutnika) zmienia się na przeciwny. Dlatego zmniejszenie liczby takich zmian prowadzi do zmniejszenia zużycia energii. W niniejszym artykule zaprezentowano badania zależności mocy pobieranej przez automat skończony od prawdopodobieństw występowania jedynek logicznych na liniach wejściowych i prawdopodobieństwa zmiany wartości na liniach wejściowych. Zaproponowano również klasyfikację wykresów uzyskanych dla wymienionych zależności. Klasyfikacja ta może być zastosowana do oceny wyników procesu redukcji energii oraz sprawdzenia zachowania automatu skończonego przy zmianie właściwości statystycznych sygnałów wejściowych. Zaproponowana klasyfikacja może być również użyta do stworzenia nowych metod i algorytmów zmniejszenia poboru mocy w automatach skończonych.

Słowa kluczowe: moc w automatach skończonych, automaty skończone, klasyfikacja mocy

MODELLING FUZZY BELIEFS OF AGENTS

Magdalena Kacprzak[1], Witold Kosiński[2,3]

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

[2] Department of Computer Science, Polish-Japanese Institute of Information Technology in Warsaw, Poland

[3] Faculty of Mathematics, Physics and Technology, Kazimierz Wielki University of Bydgoszcz, Poland

Abstract: Ordered fuzzy numbers (OFN) were introduced by Kosiński, Prokopowicz and Ślęzak in 2002. The definition of OFN uses the extension of the parametric representation of convex fuzzy numbers. So far, they were applied to deal with optimization problems when data are fuzzy. In 2011 Kacprzak and Kosiński observed that a subspace of OFN called step ordered fuzzy numbers (SOFN) may be equipped with a lattice structure. In consequence, a Boolean operations like conjunction, disjunction and, what is more important, diverse types of implications can be defined on SOFN. In this paper we show how OFN can be applied in multi-agent systems for modelling agents' beliefs about fuzzy expressions. Then we present preliminary version of a logic based on SOFN and study how this logic can be helpful in evaluating features of multi-agent systems concerning agents' fuzzy beliefs.

Keywords: ordered fuzzy numbers, step ordered fuzzy numbers, fuzzy beliefs, multi-agent systems

1. Introduction

In real life we often use notions like bad weather, high temperature, small women, high humidity, obese man, or a firm which does well. Let us focus on these expressions. When we say that somebody is obese, when we talk about obesity? As a criterion we may consider body mass index (BMI) - a measurement which compares weight and height. However, in every day chatting, nobody calculates this index and then such an assessment deeply depends on a performer of the claim. Expressions which are not clear-cut and for which it is difficult to assign one from the values *true* or *false*, occur not only in human communication but also in software engineering, e.g., in rules exploited in fuzzy controllers. In the current work we are going to

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 9, pp. 45-60, 2012.

study and analyze a problem of representation and evaluation of fuzzy expressions in communication performed in multi-agent systems. To capture diversity of approaches concerning expressions like "obesity", in literature are considered multi-valued logics [24,25] or fuzzy logics [31].

The good example of application of fuzzy technique in the context of Multi-Agent Systems (MAS) technology was done by Maione and Naso in [26] in manufacturing control systems. Namely, it is known that agents derive inspiration from communities of intelligent decision makers in uncertain and extremely dynamic environments, and that fuzzy techniques are suited to model human decision-making. Therefore, in [26] the authors discuss the potentialities of the challenging combination of Soft Computing, namely fuzzy logic techniques, and Multi-Agent paradigms in task contracting problems for manufacturing control. In particular, the paper examines if and how much agents' decision schemes benefit from the application of fuzzy methodologies.

The Fuzzy Set Theory gives effective tools to model the degreases of satisfaction of decision objectives and to combine them in a unique criterion of evaluation. In particular, each decision objective can be described with a fuzzy membership function where degree zero (one) expresses the minimum (maximum) satisfaction of the objective, while all the intermediate values represent degrees of partial satisfaction ([3,33]). Then the global objective is the fuzzy aggregation of the weighted goals when t-norm may be used or, parameterized operator providing a more realistic tradeoff between the conflicting objectives (suggested in [33], p. 37) called the "Compensatory AND" operator with some free parameter to be fixed.

In our paper we propose new approach in which **ordered fuzzy numbers** (OFN) are applied. We limit our considerations to multi-agent systems and concentrate on agents' beliefs. Our study is twofold. On the one hand we want to make agents able to use ordered fuzzy numbers in their "thinking" and making decisions. On the other hand we plan to use ordered fuzzy numbers for evaluating agents' beliefs about their beliefs.

The theory of fuzzy numbers [6] is that set up by Dubois and Prade [7], who proposed a restricted class of membership functions, called (L,R)–numbers with shape functions L and R. However, approximations of fuzzy functions and operations are needed if one wants to follow Zadeh's [31] extension principle. It leads to some drawbacks that concern properties of fuzzy algebraic operations, as well as to unexpected and uncontrollable results of repeatedly applied operations. These problems are resolved in ordered fuzzy numbers. OFN were invented by Kosiński, Prokopowicz and Ślęzak in the previous decade [17,18,19,20,21]. The definition of OFN uses the extension of the parametric representation of convex fuzzy numbers. **Step ordered fuzzy**

numbers (SOFN) form a subspace of ordered fuzzy numbers and may be equipped with a lattice structure. Then fuzzy implications can be defined on OFN and SOFN with the help of algebraic operations defined on OFN.

Fuzzy logic, as the originator of this idea Lotfi A. Zadeh noticed (c.f. [32]), has two main directions. Fuzzy logic in the *broad* sense is one of the techniques of soft-computing and serves mainly as apparatus for fuzzy control, analysis of vagueness in natural language and several other application domains. In this field the methods of fuzzification, approximate reasoning and defuzzification are often exploited. Here, we give three examples of works which join the paradigm of multi-agent systems with fuzzy control. One of them is a fuzzy-based approach for partner selection in MAS [30]. In this work, agents, using fuzzy reasoning, can adapt their individual behaviors for partner selection in negotiation. By employing fuzzy logic, the proposed approach can be applied in open and dynamic environments easily and flexibly. The next example is a multi-agent system for knowledge-based access to distributed databases [29]. In this work, the KQML (Knowledge Query and Manipulation Language) is extended with fuzzy linguistic variables to deal with the human style of decision processing and support fuzzy decision making. The last example is a multi-agent systems for environmental control and intelligent buildings [12]. This work refers to the intelligent home project in which home environment is fitted with distributed intelligent home-control agents like WaterHeater, CofeeMaker, DishWasher, etc. All of them exploit in their work fuzzy inferencing.

Fuzzy logic in the *narrow* sense is a branch of many-valued logic based on the paradigm of inference under vagueness. Here, the focus of the research is on a logical system and its metamathematical properties. A basic monograph in this field is written by Hajek [13]. One of the main operations in fuzzy logic are fuzzy implications. Deep study on analytical and algebraic aspects of fuzzy implications are presented by Baczynski and Jayaram [1].

Our current paper refers to both meanings of the term of fuzzy logic. On the one hand, our scientific interests concern multi-agent systems where agents are fuzzy controllers. Our main idea is to enrich such systems with much more sophisticated dialogues than are offered by KQML. On the other hand, we make a first step for introducing a new logic where ordered fuzzy numbers play a crucial role. The main goal of this research is to offer a formal tool adequate for evaluating properties concerning dialogues between fuzzy controllers and other agents.

The paper is organized as follows. Section 1 gives a brief overview of ordered fuzzy numbers. In Section 2, step ordered fuzzy numbers are presented. Section 3 defines a lattice structure on OFN. In section 4 application of OFN in reasoning about agents' beliefs is discussed. Section 5 gives some conclusions.

2. Ordered fuzzy numbers

Proposed recently by the second author and his two coworkers: P.Prokopowicz and D. Ślęzak [17,18,19,20,21] an extended model of convex fuzzy numbers [27] (CFN), called ordered fuzzy numbers (OFN), does not require any existence of membership functions. In this model an ordered fuzzy number is a pair of continuous functions, f and g, say, defined on the interval $[0,1]$ with values in $\mathbf{R}$. To see OFN as an extension of CFN - model, take a look on a parametric representation know since 1986, [9] of convex fuzzy numbers.

The continuity of both parts implies their images are bounded intervals, say UP and $DOWN$, respectively. We may used symbols to mark boundaries for $UP = [l_A, 1_A^-]$ and for $DOWN = [1_A^+, p_A]$. In general, the functions f, g need not to be invertible, only continuity is required. If we add the constant function on the interval $[1_A^-, 1_A^+]$ with its value equal to 1, we might define the membership function

$$\begin{aligned} \mu(x) = \mu_{up}(x), \text{ if } x \in [l_A, 1_A^-] = [f(0), f(1)], \\ \mu(x) = \mu_{down}(x), \text{ if } x \in [1_A^+, p_A] = [g(1), g(0)] \text{ and} \\ \mu(x) = 1 \text{ when } x \in [1_A^-, 1_A^+] \end{aligned} \tag{1}$$

if $f \le g$ are both invertible, i.e. inverse functions $f^{-1} =: \mu_{up}$ and $g^{-1} =: \mu_{down}$ exist, and f is increasing, and g is decreasing. Obtained in this way the membership function $\mu(x), x \in \mathbf{R}$ represents a mathematical object which reminds a convex fuzzy number in the classical sense [5,15].

On OFN four algebraic operations have been proposed between fuzzy numbers and crisp (real) numbers, in which componentwise operations are present. In particular if $A = (f_A, g_A), B = (f_B, g_B)$ and $C = (f_C, g_C)$ are mathematical objects called ordered fuzzy numbers, then the sum $C = A + B$, product $C = A \cdot B$, division $C = A \div B$ and scalar multiplication by real $r \in \mathbf{R}$, are defined in natural way: $r \cdot A = (rf_A, rg_A)$ and $f_C(y) = f_A(y) \star f_B(y)$, $g_C(y) = g_A(y) \star g_B(y)$ where "$\star$" works for "$+$", "$\cdot$", and "$\div$", respectively, and where $A \div B$ is defined, if the functions $|f_B|$ and $|g_B|$ are bigger than zero. Notice that the subtraction of B is the same as the addition of the opposite of B, i.e. the number $(-1) \cdot B$, and consequently $B - B = 0$. From this follows that any fuzzy algebraic equation $A + X = C$ with given A and C as OFN possesses a solution, that is OFN, as well. Moreover, to any convex and continuous fuzzy number correspond two OFNs, they differ by the orientation: one has positive, say (f, g) , another (g, f) has negative.

A relation of partial ordering in the space of all OFN, denoted by $\mathcal{R}$, can be introduced by defining the subset of 'positive' ordered fuzzy numbers: a number $A =$

(f,g) is not less than zero, and by writing

$$A \geq 0 \quad \text{iff } f \geq 0, g \geq 0\,. \tag{2}$$

In this way the set $\mathcal{R}$ becomes a partially ordered ring. Notice, that for each two fuzzy numbers $A = (f_A, g_A), B = (f_B, g_B)$ as above, we may define $A \wedge B =: F$ and $A \vee B =: G$, both from $\mathcal{R}$, by the relations:

$$F = (f_F, g_F), \text{if } f_F = \inf\{f_A, f_B\}, g_F = \inf\{g_A, g_B\}\,. \tag{3}$$

Similarly, we define $G = A \vee B$.

Notice that in the definition of OFN it is not required that two continuous functions f and g are (partial) inverses of some membership function. Moreover, it may happen that the membership function corresponding to A does not exist; such numbers are called improper. In any case for $A = (f,g)$ we call f - the up-part and g - the down-part of the fuzzy number A. To be in agreement with further and classical denotations of fuzzy sets (numbers), the independent variable of the both functions f and g is denoted by y (or some times by s), and their values by x.

3. Step ordered fuzzy numbers

It is worthwhile to point out that a class of ordered fuzzy numbers (OFNs) represents the whole class of convex fuzzy numbers that possess continuous membership functions. To include all CFN (with discoontiuous meembership functions) some generalization of functions f and g is needed. This has been already done by the second author who in [16] assumed they are functions of bounded variation. i.e. they belong to BV. Then all convex fuzzy numbers are contained in this new space $\mathcal{R}_{BV} \supset \mathcal{R}$ of OFN. Then operations are defined $\mathcal{R}_{BV}$ in the similar way, the norm, however, will change into the norm of the cartesian product of the space of functions of bounded variations. Then all convex fuzzy numbers are contained in this new space $\mathcal{R}_{BV}$ of OFN. Notice that functions from BV [23] are continuous except for a countable numbers of points.

Important consequence of this generalization is the possibility of introducing a subspace of OFN composed of pairs of step functions. If we fix a natural number K and split $[0,1)$ into $K-1$ subintervals $[a_i, a_{i+1})$, i.e. $\bigcup_{i=1}^{K-1} [a_i, a_{i+1}) = [0,1)$, where $0 = a_1 < a_2 < \ldots < a_K = 1$, and define a step function f of resolution K by putting u_i on each subinterval $[a_i, a_{i+1})$, then each such function f is identified with a K-dimensional vector $f \sim u = (u_1, u_2 \ldots u_K) \in \mathbf{R}^K$, the K-th value u_K corresponds to

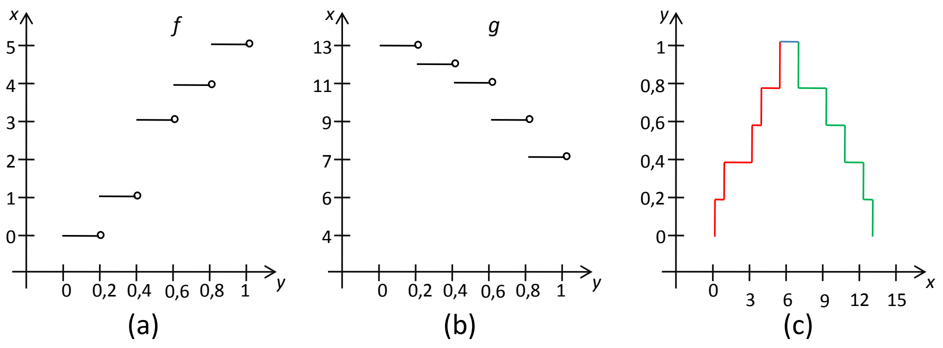


Fig. 1. Example of a step ordered fuzzy number $A = (f, g) \in \mathcal{R}_K$, (a) function f, (b) function g, (c) membership function.

$s = 1$, i.e. $f(1) = u_K$. Taking a pair of such functions we have an ordered fuzzy number from $\mathcal{R}_{BV}$. Now we introduce

Definition 2. By a step ordered fuzzy number A of resolution K we mean an ordered pair (f, g) of functions such that $f, g : [0, 1] \rightarrow \mathbf{R}$ are K-step functions.

We use $\mathcal{R}_K$ for denotation the set of elements satisfying Def. 2. The example of a step ordered fuzzy number and its membership function are shown in Fig. 1. The set $\mathcal{R}_K \subset \mathcal{R}_{BV}$ has been extensively elaborated by our students in [10] and [22]. We can identify $\mathcal{R}_K$ with the Cartesian product of $\mathbf{R}^K \times \mathbf{R}^K$ since each K-step function is represented by its K values. It is obvious that each element of the space $\mathcal{R}_K$ may be regarded as an approximation of elements from $\mathcal{R}_{BV}$, by increasing the number K of steps we are getting the better approximation. The norm of $\mathcal{R}_K$ is assumed to be the Euclidean one of $\mathbf{R}^{2K}$, then we have a inner-product structure for our disposal.

4. Lattice structure on $\mathcal{R}_K$

Let us consider the set $\mathcal{R}_K$ of step ordered fuzzy numbers with operations $\vee$ and $\wedge$ such that for $A = (f_A, g_A)$ and $B = (f_B, g_B)$

$$A \vee B = (sup\{f_A, f_B\}, sup\{g_A, g_B\}), \quad A \wedge B = (inf\{f_A, f_B\}, inf\{g_A, g_B\}).$$

In [14] we have shown that the algebra $(\mathcal{R}_K, \vee, \wedge)$ defines a lattice structure and proved the following theorem.

Theorem 1. The algebra $(\mathcal{R}_K, \vee, \wedge)$ is a lattice.

Let $\mathcal{B}$ be the set of two binary values: $0,1$ and let us introduce the particular subset $\mathcal{N}$ of $\mathcal{R}_K$ $\mathcal{N} = \{A = (\underline{u}, \underline{v}) \in \mathcal{R}_K : \underline{u} \in \mathcal{B}^K, \underline{v} \in \mathcal{B}^K\}$. It means such that each component of the vector $\underline{u}$ as well as of $\underline{v}$ has value 1 or 0. It is easy to observe that all subsets of $\mathcal{N}$ have both a join and a meet in $\mathcal{N}$. In fact, for every pair of numbers from the set $\{0,1\}$ we can determine *max* and *min* and it is always 0 or 1. Therefore $\mathcal{N}$ creates a *complete lattice*. In such a lattice we can distinguish the greatest element $\underline{1}$ represented by $(1,...,1)$ and the least element $\underline{0}$ represented by $(0,...,0)$.

Theorem 2. The algebra $(\mathcal{N}, \vee, \wedge)$ is a complete lattice.

We say that two elements A and B are *complements* of each other if and only if $A \vee B = \underline{1}$ and $A \wedge B = \underline{0}$. The complement of a number A will be marked with $\neg A$ and is defined as follows:

Definition 3. Let $A \in \mathcal{N}$ be a step ordered fuzzy number represented by a binary vector $(a_1, a_2, \ldots, a_{2K})$. By the complement of A we understand

$$\neg A = (1 - a_1, 1 - a_2, \ldots, 1 - a_{2K}).$$

A bounded lattice for which every element has a complement is called a *complemented lattice*. Moreover, the structure of step ordered fuzzy numbers $\{\mathcal{N}, \vee, \wedge\}$ forms a complete and complemented lattice in which complements are unique. In fact it is a *Boolean algebra*. In the example with $K = 2$ a set of universe is created by vectors

$$\mathcal{N} = \{(a_1, a_2, a_3, a_4) \in \mathbf{R}^4 : a_i \in \{0,1\}, \text{for}\, i = 1,2,3,4\}.$$

The complements of elements are $\neg(0,1,0,0) = (1,0,1,1)$, $\neg(1,1,0,0) = (0,0,1,1)$ etc. Now we can rewrite the definition of the complement in terms of a new mapping.

Definition 4. For any $A \in \mathcal{N}$ we define its negation as

$$N(A) := (1 - a_1, 1 - a_2, \ldots, 1 - a_{2K}), \text{if}\, A = (a_1, a_2, \ldots, a_{2K}).$$

It is obvious, from Definitions 3 and 4, that the negation of given number A is its complement. Moreover, the operator N is a strong negation, because is involutive, i.e.

$$N(N(A)) = A \text{ for any } A \in \mathcal{N}.$$

One can refer here to known facts from the theory of fuzzy implications (cf. [1,2,8]) and to write the strong negation N in terms of the standard strong negation N_I on the unit interval $I = [0,1]$ defined by $N_I(x) = 1 - x, x \in I$, namely $N((a_1,a_2,\ldots,a_{2K})) = ((N_I(a_1),N_I(a_2),\ldots,N_I(a_{2K}))$.

In the classical Zadeh's fuzzy logic the definition of a fuzzy implication on an abstract lattice $\mathcal{L} = (L, \leq_L)$ is based on the notation from the fuzzy set theory introduced in [8].

Definition 5. Let $\mathcal{L} = (L, \leq_L, 0_L, 1_L)$ be a complete lattice. A mapping $I : L^2 \to L$ is called a fuzzy implication on $\mathcal{L}$ if it is decreasing with respect to the first variable, increasing with respect to the second variable and fulfills the border conditions

$$I(0_L,0_L) = I(1_L,1_L) = 1_L\,, I(1_L,0_L) = 0_L\,. \tag{4}$$

Now, possessing the lattice structure of $\mathcal{R}_{\mathcal{K}}$ (SOFN) and the Boolean structure of our lattice $\mathcal{N}$, we can repeat most of the definitions know in the Zadeh's fuzzy set theory. The first one is the Kleene–Dienes operation, called a binary implication, already introduced in our previous paper [14] as the new implication (cf. Definition 4 in [14])

$$I_b(A,B) = N(A) \vee B\,, \text{ for any } A,B \in \mathcal{N}\,. \tag{5}$$

In other words, the result of the binary implication $I_b(A,B)$, denoted in [14] by $A \to B$, is equal to the result of operation *sup* for the number B and the complement of A:

$$A \to B = sup\{\neg A, B\}.$$

Next we may introduce the Zadeh implication by

$$I_Z(A,B) = (A \wedge B) \vee N(A)\,, \text{ for any } A,B \in \mathcal{N}\,. \tag{6}$$

Since in our lattice $\mathcal{R}_K$ the arithmetic operations are well defined we may introduce the counterpart of the Lukasiewicz implication by

$$I_L(A,B) = C\,, \text{where } C = 1 \wedge (1 + B - A)\,. \tag{7}$$

In the calculating the RHS of (7) we have to regard all numbers as elements of $\mathcal{R}_K$, since by adding the ordered fuzzy number A from $\mathcal{N}$ to the crisp number 1 we may leave the subset $\mathcal{N} \subset \mathcal{R}_{\mathcal{K}}$. However, the operation $\wedge$ will take us back to the lattice $\mathcal{N}$. It is obvious that in our notation $1_N = 1$. The explicit calculation will be: if $C = (c_1,c_2,\ldots,c_{2K)}), A = (a_1,a_2,\ldots,a_{2K}), B = (b_1,b_2,\ldots,b_{2K})$, then $c_i = min\{1, 1 - a_i + b_i\}$, where $1 \leq i \leq 2K$.

5. Application of OFN in reasoning about fuzzy beliefs

In this section we show how agents' attitudes can be modelled by means of ordered fuzzy numbers. Assume a model of a multi-agent system which is consistent with the formalism used in the software tool Perseus [4]. In the future this verifier may be extended to analyze also distributed systems where agents have fuzzy beliefs. In this formalism, a model of a multi-agent system is assumed to be an enriched Kripke structure $\mathcal{M} = (Agt, S, RB, I, v)$ where

- $Agt = \{1, 2, 3, \ldots, n\}$ is a set of names of agents,
- S is a non-empty set of states (the universe of a structure),
- RB is a (doxastic) function which assigns to every agent a binary relation, $RB : Agt \longrightarrow 2^{S \times S}$ - this function gives an interpretation for agents' beliefs,
- I is an interpretation of actions, $I : \Pi_0 \longrightarrow (Agt \longrightarrow 2^{S \times S})$ where Π_0 is a set of atomic actions,
- v is a valuation function, $v : S \longrightarrow \{\mathbf{0}, \mathbf{1}\}^{V_0}$ where V_0 is a set of propositions.

This model should be extended with a set L of **linguistic variables**. By linguistic variables we mean variables which values are from the set of words or sentences of a natural (or artificial) language. Formally it is a foursome $l = (Z, T, U, m)$ where

- Z is a name of variable l,
- T is a set of fuzzy terms which can be assigned to l,
- U is a digital interval of values of l,
- m is a rule which assigns ordered fuzzy numbers to terms from the set T.

For example, let l describes *speed* of a car, then fuzzy terms which can be assigned to l are *back*, *very slowly*, *slowly*, *fast*, digital values for *speed* are assumed to be from the interval $[-50, 100]$. Ordered fuzzy numbers assigned for *speed* are figured in Fig. 4a.

Since a current state of a multi-agent system changes dynamically it is reasonable to expect that a rule m of a linguistic variable l will be different in different states. Therefore we propose to extend definition of l and assume that $l = (Z, T, U, m_{AS})$ where $m_{AS} : T \times Agt \times S \rightarrow OFN$ is a function which for every agent and every state of a system assigns an ordered fuzzy number being an interpretation for elements from T. In other words, in various situations for various agents different rule m may be accepted.

Now, consider a system with three agents A, B, C. The aim of the system is to simulate a movement of a point laying on a segment (cf. [22]). Agent A observes speed of the point. Agent B observes a distance from the point to the given stop point.

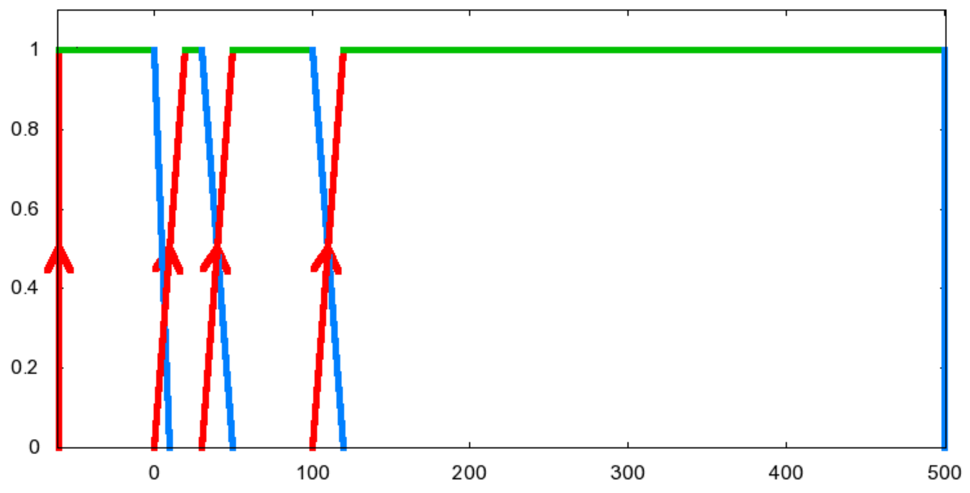


Fig. 2. Linguistic variable: *distance*

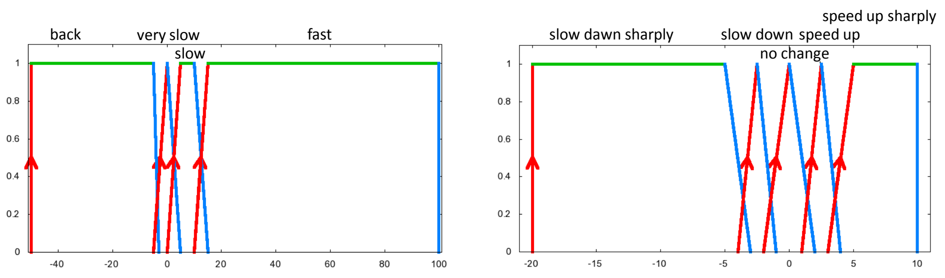


Fig. 3. Linguistic variables: *speed* and *acceleration*

Agent C is a fuzzy controller. Agents A and B provide to the controller data about the speed and the distance of the point. We assume that *speed* is a linguistic variable with values from the interval [-50,100] and terms “back, very slowly, slowly, fast”. Similarly, *distance* is a linguistic variable with values from the interval [-50,300] and terms “too close, close, moderately close, far”. Rules assigned ordered fuzzy numbers to the above terms at the initial state are presented in Fig. 3 and 4a. The tasks of agents A and B are to measure speed and the distance of the point (respectively), exchange digital values for fuzzy expressions and then provide these data to the controller. The task of the controller is to stop the point before the given stop point. It operates

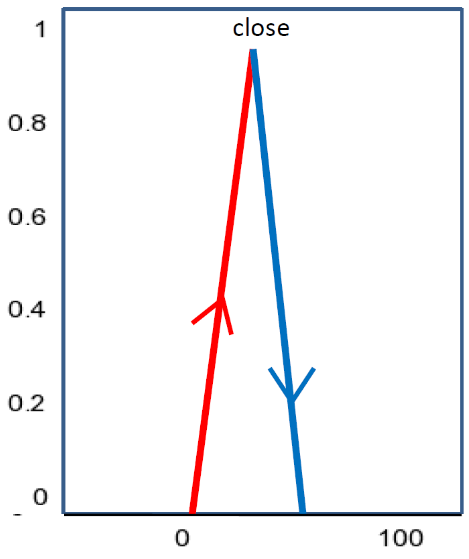


Fig. 4. Linguistic variable: *distance*

on the third linguistic variable, i.e., *acceleration*. Values of this variable are form the interval [-20,10] and are described by terms “slow dawn sharply, slow down, no change, speed up, speed up sharply”. The rule which assigns ordered fuzzy numbers to these terms are pictured in Fig. 4b. The controller given fuzzy terms concerning *speed* and *distance* uses special rules to control the movement of the point. A base of rules for this example is given in Table 1.

Table 1. Rules of the fuzzy controller.

	too close	close	moderately close	far
back	speed up	speed up sharply	speed up sharply	speed up sharply
very slow	slow down sharply	no change	speed up	speed up sharply
slow	slow down sharply	slow down	no change	speed up
fast	slow down sharply	slow down sharply	slow down	no change

Finally the controller determines the output fuzzy set describing acceleration and transforms it into real value.

Application of OFN in modelling agents' beliefs has great advantage since allows for manipulating fuzzy expressions rather then strict digital values. It makes agents' communication easier and faster and definitely simplifies knowledge bases and rules which agents use in their decision process. In the example, agent *C* affects the speed of the point but it does not care when it is to close of too far. It is a role of agent *B*. In other words, agent *B* is a specialist which has abilities to evaluate the distance and current scenario and then judge, e.g., whether the distance is to small. For instance, if a point simulates a truck 1 meter to a wall means "close", but if this point simulates an ant 1 meter can be treated as very log distance. It is a specialist *B* job to assessment this.

An extension of a model of a multi-agent system to linguistic variables and exchanging a rule m with a set of rules m_{AS} causes that agents can manipulate fuzzy expressions and what is more can have different point of view on criteria which determine their subjective interpretations of this expressions.

The most important problem when we consider fuzzy beliefs of agents is how to check properties of such defined systems. The question is about a language in which we can evaluate whether some property is true or not. Leu us discus it now. Assume that in the above example is also agent *D*. It tries to guess the behavior of agent *C*, i.e., agent *D* needs to learn what action *C* decided to perform. *D* beliefs that if *A* says that the point is *close* and *B* says that the point moves *slowly* then *C* decides to *slow up*. To create a formula describing this property use here a commonly accepted language of epistemic logic based on Kripke structure [11]. In this formalism we can write

$$B_D(A\_close) \wedge B_D(B\_slow) \rightarrow B_D(C\_slow\_up)$$

where $B_D(T)$ informally means that agent *D* beliefs that *T* holds. Our aim is to verify whether this formula is true in a model of the system from the example.

Furthermore, we know that agent *D* is not sure about beliefs of agents *A* and *B* and assumes that terms *close* and *slow* are interpreted by ordered fuzzy numbers depicted in Fig. 5. Notice that *D* departs from the truth (cf. Fig. 3) but not so much. However, if we take into account classical two-valued logic then at the initial state of the system formula $B_D(A\_close) \wedge B_D(B\_slow)$ is not true. It stems from the fact that beliefs of *D* about beliefs of *A* and *B* are not true. For some digital values they agree but for another not. Although the OFN representations are not the same they are very similar. In two-valued logic we lose this important information. Therefore, we propose to use new, innovative approach in which step ordered fuzzy numbers are applied. In Section 3 we showed that SOFN creates a lattice with Boolean operations of conjunction, disjunction and implication. Therefore it is possible to employ these numbers as a logical values for OFN. Let v_l be a valuation function which for every formula assigns an ordered fuzzy number and assume that (111111) means absolutely true and (000000) means absolutely false. Values between (111111) and (000000), like e.g. (10100) express different kinds of half-truth. Below is given a hypothetical assignment:

(a) $v_l(B_D(A\_close)) = (101101)$
(b) $v_l(B_D(B\_slow)) = (100111)$
(c) $v_l(B_D(C\_slow\_up)) = (000000)$

Analyze intuitions concerning these values. In (a) it is assumed that agent D does not know exactly for which digital values from [-50,500] term "*close*" is ascribed since the assigned value does not equal to (11111). However, if the interval [-50,500] is divided into 3 parts then in parts one and three agent D agrees with agent B. Similar interpretation is for value (100111) assigned to formula $B_D(B\_slow)$. In (c) it is assume that agent D has no idea what and when agent C says about acceleration of the point. Based on these values we can determine value of the whole formula:

$$v_l(B_D(A\_close) \wedge B_D(B\_slow) \rightarrow (B_D(C\_slow\_up)) = (011010)$$

It means that agent D guesses faultlessly the kind of activity of the controller when considered digital values of speed and distance are very small, middle or very high. Such information surely can not be expressed by classical logical values *true* and *false*. Although multi-valued and fuzzy logics can deal with more than two values such a precise knowledge can be captured only by ordered fuzzy numbers.

6. Conclusion

The paper lays the foundations of new logic based on step ordered fuzzy numbers which will be very helpful in capturing how agents can reason about fuzzy expressions. This is innovative approach to modelling agents' beliefs and their uncertainty about beliefs of other agents. We show motivation for introducing such a new logic. The application of it we mainly find in analyzing agents' communication when knowledge base of agents is represented by a set of ordered fuzzy numbers expressing diverse agents' attitudes. Furthermore, step ordered fuzzy numbers, when are applied as logical values for propositions and other formulas of the applied language, give much more information than that something is *true* or *false*. We hope that this innovative approach is very promising in specification and verification of multi-agent systems where some software engineering ideas are applied, e.g. where fuzzy control is suitable. It could be also very useful in reasoning about software agents which are decision support systems. For example, we can analyze activity of agents which assist clients with their decisions in e-shops, i.e., agents which support users of a system in choosing a right product.

References

[1] Baczyński M., Jayaram B. (2008), *Fuzzy Implications*, Series: Studies in Fuzziness and Soft Computing , vol. 231, Springer, Berlin 2008, ISBN: 978-3-540-69080-1,

[2] Baczyński M. (2009), S-implications in Atanassov's intuitionistic and interval-valued fuzzy set theory revisited, in *Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets,Generalized Nets and Related Topicss, Volum 1: Foundation*, IBS PAN - SRI PAS, Warsaw, 2009, pp.33–42,ISBN: 13 9788389475299

[3] Bellmann R.E., Zadeh L.A. (1970), Decision Making in a Fuzzy Environment, *Management Science*, **17** (4), 141–164.

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

[5] Chen Guanrong, Pham Trung Tat (2001), *Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*, CRS Press, Boca Raton, London, New York, Washington, D.C., 2001.

[6] Czogała E., Pedrycz W. (1985), *Elements and Methods of Fuzzy Set Theory* (in Polish), PWN, Warszawa, Poland, 1985.

[7] Dubois D., Prade H. (1978), *Operations on fuzzy numbers*. Int. J. Syst. Sci., Vol. 9, No. 6, pp. 613–626, 1978.

[8] Fodor J. C., Roubens M. (1994), *Fuzzy Preference Modelling and Multicriteria Decision Support*, Kluwer Academic Publishers, Dordrecht, 1994.

[9] Goetschel R. Jr., Voxman W. (1986), Elementary fuzzy calculus, *Fuzzy Sets and Systems*, **18** (1), 31-43.

[10] Gruszczyńska A., Krajewska I. (2008), *Fuzzy calculator on step ordered fuzzy numbers*, in Polish, UKW, Bydgoszcz, 2008.

[11] Fagin R., Halpern J. Y., Moses Y., Vardi M. Y. (1995), Reasoning about Knowledge, MIT Press, Cambridge, 1995.

[12] Fuhrmann T., Neuhofer B. (2006), *Multi-Agent Systems for Environmental Control and Intelligent Buildings*, University of Salzburg, Austria, 2006.

[13] Hajek, P. (1998), Metamathematics of fuzzy logic, Dordrecht: Kluwer, 1998.

[14] Kacprzak M., Kosiński W. (2011), On lattice structure and implications on ordered fuzzy numbers, Proc. of the 7th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-2011), 18-22 July 2011, Aix-les-Bains, France, pp. 267 - 274.

[15] Klir G.J. (1997), Fuzzy arithmetic with requisite constraints, *Fuzzy Sets and Systems*, **91**, pp. 165–175, 1997.

[16] Kosiński W. (2006), On fuzzy number calculus, *Int. J. Appl. Math. Comput. Sci.*, **16** (1), pp. 51–57, 2006.

[17] Kosiński W., Prokopowicz P. , Ślęzak D. (2002), Fuzzy numbers with algebraic operations: algorithmic approach, in: *Intelligent Information Systems 2002*, M. Klopotek, S.T. Wierzchoń, M. Michalewicz(Eds.) Proc.IIS'2002, Sopot, June 3-6, 2002, Poland, pp. 311-320, Physica Verlag, Heidelberg, 2002.

[18] Kosiński W., Prokopowicz P., Ślęzak D. (2002), Drawback of fuzzy arithmetics - new intutions and propositions, in: *Proc. Methods of Aritificial Intelligence*, T. Burczyński, W. Cholewa, W. Moczulski(Eds.), pp. 231-237, PACM, Gliwice, Poland, 2002.

[19] Kosiński W., P. Prokopowicz P., Ślęzak D. (2003), On algebraic operations on fuzzy numbers, in *Intelligent Information Processing and Web Mining*, Proc. of the International IIS: IIPWM,03 Conference held in Zakopane, Poland, June 2-5,2003, M. Klopotek, S.T. Wierzchoń, K. Trojanowski(Eds.), pp. 353-362, Physica Verlag, Heidelberg, 2003.

[20] Kosiński W., Prokopowicz P., Ślęzak D. (2003), Ordered fuzzy numbers, *Bulletin of the Polish Academy of Sciences, Sér. Sci. Math.*, **51** (3), 327-338, 2003.

[21] Kosiński W., Prokopowicz P. (2004), Algebra of fuzzy numbers (In Polish: Algebra liczb rozmytych), *Matematyka Stosowana. Matematyka dla Społeczeństwa*, **5** (46), 37–63, 2004.

[22] Kościeński K. (2010), *Moduł schodkowych liczb rozmytych w sterowniu ruchem punktu materialnego*, in Polish, PJIIT, Warszawa, 2010.

[23] Lojasiewicz S. (1973), Introduction to the Theory of Real Functions (in Polish), Biblioteka Matematyczna, Tom 46, PWN, Warszawa, 1973.

[24] Lukasiewicz J.(1958), Elements of the Mathematical Logic (in Polish), PWN, Warszawa, 1958.

[25] Malinowski G. (2001), Many-Valued Logics, in Goble, Lou, ed., The Blackwell Guide to Philosophical Logic. Blackwell.

[26] Maione G., Naso D. (2003), A soft computing approach for task contracting in multi-agent manufacturing control, *Journal Computers in Industry*, **52** (3), 2003.

[27] Nguyen H.T. (1978), A note on the extension principle for fuzzy sets, *J. Math. Anal. Appl.* **64**, pp. 369-380, 2003 .

[28] Prokopowicz P. (2005), *Algorithmization of Operations on Fuzzy Numbers and its Applications* (In Polish: Algorytmizacja działań na liczbach rozmytych i jej zastosowania), Ph. D. Thesis, IPPT PAN, kwiecień 2005.

[29] Sajja P. S. (2008), Multi-Agent System for Knowledge-Based Access to Distributed Databases, *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 3, 2008.

[30] Ren F., Zhang M., Bai Q. (2007), A Fuzzy-Based Approach for Partner Selection in Multi-Agent Systems, *6th IEEE/ACIS International Conference on Computer and Information Science ICIS*, pp. 457-462, 2007.

[31] Zadeh L. A.(1983), The role of fuzzy logic in the management of uncertainty in expert systems, *Fuzzy Sets and Systems*, **11**(3), pp. 199–227, 1983.

[32] Zadeh L. A. (1994), Preface, in R. J. Marks II (ed.), *Fuzzy logic technology and applications*, IEEE Publications, 1994.

[33] Zimmermann, H.-J. (1993), *Fuzzy Sets Theory and Its Applications*, Kluwer Academic Press, 1993.

MODELOWANIE ROZMYTYCH PRZEKONAŃ AGENTÓW

Streszczenie: Skierowane liczby rozmyte (SLR) zostały wprowadzone przez W. Kosińskiego, P. Prokopowicza i D. Ślęzaka w 2002 roku. Definicja skierowanych liczb rozmytych wykorzystuje rozszerzenie parametrycznej reprezentacji wypukłych liczb rozmytych. SLR do tej pory były wykorzystywane do rozwiązywania problemów optymalizacyjnych dla rozmytych danych. W 2011 roku M. Kacprzak i W. Kosiński zaobserwowali, że schodkowe skierowane liczby rozmyte (SSLR) stanowiące podprzestrzeń SLR tworzą kratę. W konsekwencji, Boolowskie operacje takie jak koniunkcja, alternatywa oraz różne rodzaje (rozmytych) implikacji mogą być zdefiniowane w zbiorze schodkowych skierowanych liczb rozmytych. Celem niniejszej pracy jest pokazanie nowego zastosowania SLR jakim jest modelowanie przekonań agentów w środowisku systemów wieloagentowych, gdy przekonania te dotyczą rozmytych pojęć i danych. Jest to pierwszy krok w kierunku stworzenia pełnej logiki opartej na wartościach ze zbioru SSLR. Logika ta umożliwi analizę własności systemów, w których agenci mają rozmyte przekonania.

Słowa kluczowe: skierowane liczby rozmyte, schodkowe skierowane liczby rozmyte, rozmyte przekonania, systemy wieloagentowe

FINDING SIMILAR DOCUMENTS IN WEB SEARCH RESULTS

Urszula Kużelewska

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: Searching the Web is a challenging task. According to the Zamir and Etzioni's definition, Internet is "unorganized, unstructured and decentralized place". Although there are powerful search engines available, the number of indexed web pages exceeds 1 trillion [20] and still grows. Most of the search engines return list of documents from their bases sorted according to their relevance to a search query. Such approach is not the best, because the returned list is very long and may contain documents not related to the query. To increase efficiency of a searching process one may identify groups of similar documents from result list. One of the tools to do it are traditional clustering algorithms. The article presents clustering Web search results directly from a search engine as well as sets created from results for different queries. Documents were grouped using the following methods: EM and XMeans.

Keywords: Web search results clustering, documents similarity, snippets clustering

1. Introduction

Internet is a popular place to share our knowledge or opinion as well as a source of interesting and valuable information. Although the number of web sites is huge, they are useless if they are difficult to find. "The revolution that the Web has brought to information access is not so much due to the availability of information (huge amounts of information has long been available in libraries and elsewhere), but rather the increased efficiency of accessing information, which can make previously impractical tasks practical" [4]. Increased number of information is not related to simultaneous increase in its quality. It requires from the search engines continuous developing and improving standard of generated results.

One of the approaches to this problem is Search Results Clustering (SRC) - techniques of identification groups of similar web sites. According to Weiss [12] the result of SRC are groups of documents, which are organized according to the

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 9, pp. 61-76, 2012.

common theme and described comprehensibly to the human. Such approach does not affect quality or length of a result list, however improves the time of access to relevant information.

The purpose of this paper is to present clustering of Web search results, which have been generated using selected clustering methods with different techniques used in document processing. The main idea of application of traditional clustering algorithms in Web search results domain is not novel, however many new grouping methods have been proposed as well as new procedures in whole process of snippet clustering, which is the reason for testing them in the domains like SRC.

The article is organized as follows: the second section presents short review of methods used in finding similarities in documents from Web search results, the third section describes application of clustering algorithms in SRC domain. The fourth section presents some approaches to the problem of Web search clustering evaluation and the following part contains results of experiments and the last section concludes the paper.

2. Search Results Clustering

Lists of results returned by search engines consist of elements relevant to a query. Each element relates to the particular web page and contains a title, a domain address and a small portion of text from the page (snippet). Search Results Clustering methods work on data preprocessed from titles and/or text of snippets.

The preprocessing of text is as follows: first, all capital letter are replaced by small ones, next, the words without any meaning, such as "is" or "the", are removed, and then the remaining text is stemmed. Stemming is a procedure of extracting constant part of words having different form of inflectional. To give an example, "computer", "computers" or "computerization" could have one common stem - "compute". Words after stemming are determined as terms. The benefits from preprocessing are reduction of the number of words and improvement similarity among elements in final clusters.

One of the steps of Search Results Clustering process is definition of labels describing the generated groups. Depending on the method this part can be performed before or after clustering phase. Typical example solution of this problem is identification of the most frequent words in every group.

Before documents are clustered, they are transformed from their letter representation to numbers in Vector Space Model (VSM) [8]. The numbers relate to the relevance selected words in particular documents. The selected word are called terms.

This process enables application an arbitrary clustering algorithm. The methods of the relevance calculation is described below.

The equation describing a document in Vector Space Model is as follows:

$$D_i = (d_{i1}, d_{i2}, ..., d_{in}) \tag{1}$$

where components d_{ij} refer to the level of description as well as diversification of the individual terms and n is a number of terms selected for representation of documents.

It has been proposed many methods of document description in VSM. One of them is binary representation: if a term from VSM vector is present in the examining document, the relevant component is equal 1. In the other case it is equal 0. More useful are methods based on term or document frequency (TF, DF, TFIDF). One of them, TFIDF, is described as follows:

$$TFIDF(D_i) = TF(t_j, D_i) \cdot IDF(t_j) \tag{2}$$

where component $TF(t_j, D_i)$ refers to the number of occurrences of term t_j in document D_i (see Equation 3) and $IDF(t_j)$ refers to the number of occurrences of this term in all documents (see Equation 4).

$$TF(t_j, D_i) = \begin{cases} 0, & for\; n_{ji} = 0; \\ \frac{n_{ji}}{n_{max}}, & for\; n_{ji} > 0. \end{cases} \tag{3}$$

where n_{ji} denotes a number occurrences of term t_j in document D_i and n_{max} denotes maximal number from occurrences of every term from VSM vector.

$$IDF(t_j) = \log\left(\frac{N}{N_j}\right) \tag{4}$$

where N denotes a number of all documents and N_j - a number of documents containing term t_j.

Documents can be described by all terms present in them, however, to increase time efficiency as well as quality of results terms to VSM vector are selected. It can be used a simple method of selecting the most often terms or one of more complex procedures. The procedures are also based on indices described above, such as TF or IDF.

3. SRC algorithms

Over recent 10 years many important SRC algorithms and systems have been proposed. Despite homogenous contents of clusters, the methods should create compact cluster labels as well as be very effective regarding time. Classical approach to SRC problem applies traditional clustering methods, such as k-means or EM, however there are also systems using different solutions. The most popular are Grouper [14], Carrot [11] and Vivisimo [22]. The algorithms based on traditional clustering are: HKA [5], WISE [1] and ICSE [9].

One of the approaches to document clustering is based on Suffix Tree Clustering (STC) technique, where grouped are phrases instead of individual words. The idea of STC construction has been adapted in Carrot system. The authors created a very extended system with many stemming techniques. They also defined relationship between two main STC parameters: merge threshold and minimum base cluster score, and quality of generated results. A distinguishing feature is dealing with snippets in Polish language.

LINGO was proposed by Osiński [6] in his master thesis and finally became a part of Carrot system. In LINGO was used latent semantic indexing originally adapted in SHOC [15]. In this algorithm the author solve the problem of inadequate labels generation, which occurs in the previous methods, starting the procedure from identification of descriptions of clusters (description comes first). Then documents are assigned to the cluster with the label most matched to their content.

One of the fastest method in traditional clustering domain is k-means. For this reason it is a very popular technique in partitioning document partitioning. Mahdavi and Abolhassani have applied modified k-means method in document clustering domain. They have combined k-means with Harmony Search optimization method to avoid convergence to local optimum and tested the methods on Euclidean as well as cosine similarity/dissimilarity measures.

ICSE (Intelligent Cluster Search Engine) is a system, which also uses k-means algorithm. Documents from a result list are grouped into most relevant, relevant and irrelevant clusters. The clustering method identifies the web pages, which are relevant to the search query in order to increase the relevance rate of search results.

WISE is a meta-search engine, which builds a hierarchical structure of clusters. The clusters contain related web pages expressing one meaning of the query. It uses PoBOC soft clustering algorithm, which is based on graph theory.

4. Evaluation of web search results clustering

Objective assessment of a result of partitioning is a challenging task. Clustering scheme of an internet search list, as well as a clustering result in general, is difficult to evaluate, because it depends on a purpose of the solution and subjective expectations of results' recipients [11]. Considering the reason above, the most useful evaluation is satisfaction of a user searching the information.

However, there are also objective approach to this problem, such as IR (Information Retrieval) indices and criterion merge-then-cluster.

IR indices are composed of precision and recall components. The precision compares a number of documents related to a search query with a number of all documents received from a search engine in answer to the query.

The criterion merge-then-cluster assumes evaluation appropriately prepared set of documents. Original partition is known and compared to generated clustering. The comparison may be performed using traditional measures from clustering evaluation domain [3]. Example of a such index is Rand expressed in the following equation:

$$Rand = \frac{a+d}{a+b+c+d} \tag{5}$$

where a denotes a number of pairs of elements belonging to the same group in both the original as well as the generated solution, b is a number of pairs of elements belonging to different groups in the original, but to the same group in the generated solution, c is a number of pairs of elements belonging to the same group in original, but to different groups in generated solution, d is a number of pairs of elements belonging to different groups in the original as well as in the generated solution.

In experiments presented in this article subjective as well as objective evaluation have been performed.

5. Results of experiments

The system Search Engine used in the following experiments was created at Bialystok University of Technology as a part of a master thesis "An intelligent search engine using clustering methods to optimize search results" [10]. The results were generated by Bing [16] search engine and the clustering algorithms were taken from Weka system [13]. The system allows data entry as XML file, as well, which allows objective evaluation of quality of results. A browser window of the system is presented in Figure 1.

In the system the step of clustering may be realised by one of the algorithms: EM [2] and XMeans [7]. EM clusters elements basing on probability of their membership to each group. In the system, it is possible to run EM without giving the information about a number of clusters. XMeans is an extension of k-means method. One of improvements is automatic calculation of a number of groups. However a user is required to give this value, the final result may contain a different, more optimal, number of clusters.

The experiments consist of 3 parts: on-line clustering data from Bing Web Service, clustering data from file containing different as well as similar description of elements between clusters (created from results of various queries) and clustering benchmark data from Credo repository [19].

5.1 Experiments with data clustering from a search engine

In this experiment, the query "*Java*" was entered in Bing search engine and returned 100 results. The results were clustered using 4 algorithms: Lingo, STC [18], XMeans and EM [10]. In case of methods requiring input parameters (XMeans and in some cases EM), their values were adjusted to obtain the smallest number of clusters as well as equal clusters' size. EM was started with the following values of parameters: unknown number of clusters, length of description vector: 25, documents description method: TFIDF. XMeans was started with the numbers of clusters equal 20, length of description vector: 45 and documents description method: TFIDF.

Tables 1 and 2 show a brief summary of the results generated by respectively Lingo and STC algorithms and XMeans and EM methods, which allows subjective evaluation and comparison of generated results. The tables contain only larger clusters (of size greater than 3 elements).

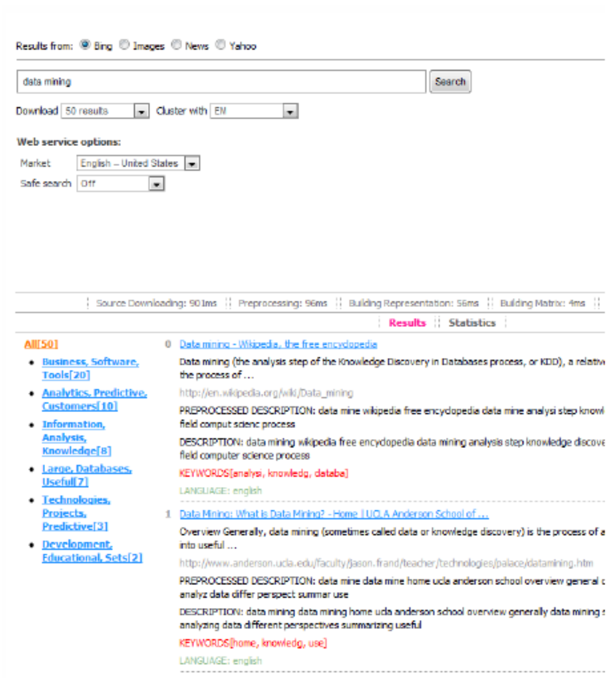
Lingo algorithm split the result list in many groups containing small number (10 and less) of elements, whereas STC, unfortunately, has identified one large group composed of 88 items and several smaller clusters. XMeans and EM methods generated several groups ranging in sizes from several to 40 items. Labels of the groups concerned domains: computer technology and programming and were phrases (in case of Carrot system) or consisted of one word (in case of SearchEngine program). The greatest group of STC method had label *Java*, which is undesirable, because it equals the input query. However, interesting labels were created in case of XMeans algorithm: *Sun* and *Oracle*, concerning companies connected with Java programming language.

Table 1. Clustering results of *Java* query generated by Lingo and STC methods

Method	Group label	Number of results	Method	Group label	Number of results
Lingo	Java Tutorials	10	STC	Java	88
	Java Coffee	8		Programming	16
	Java.net	8		Download	12
	Java Technology	7		Source	12
	Downloads	6		Software	11
	Java Community	6		Tutorials	11
	Java Language	6		Coffee	9
	Learn Java	6		Java Programming	8
	Open Source	5		Java.net	8
	Source Code	5		Developers	8
	Standard Edition	5		Standard Edition	7
	Tutorial	5		Java Software	6
	Browser	4		Open Source	5
	Guide	4		Virtual Machine	5
	Java Programming Resources	4	-	-	-
			-	-	-
	Virtual Machine	4	-	-	-

Table 2. Clustering results of *Java* query generated by XMeans and EM methods

Method	Group label	Number of results	Method	Group label	Number of results
XMeans	Development	23	EM	Programming	40
	Programming	18		Developers	21
	Information	9		Tutorials	19
	Code	8		Download	14
	Software	7		Software	5
	Sun	7		-	-
	Oracle	6	-	-	-
	Learn	5	-	-	-


Results from: [✓ Bing] Images News Yahoo

data mining | Search

Download 50 results Cluster with EM

Web service options:

Market English – United States

Safe search Off

Source Downloading: 901ms ¦ Preprocessing: 96ms ¦ Building Representation: 56ms ¦ Building Matrix: 4ms

Results ¦ **Statistics**

All[50]

- Business, Software, Tools[20]
- Analytics, Predictive, Customers[10]
- Information, Analysis, Knowledge[8]
- Large, Databases, Usefull[7]
- Technologies, Projects, Predictive[3]
- Development, Educational, Sets[2]

0 Data mining - Wikipedia, the free encyclopedia

Data mining (the analysis step of the Knowledge Discovery in Databases process, or KDD), a relativ
the process of ...

http://en.wikipedia.org/wiki/Data_mining

PREPROCESSED DESCRIPTION: data mine wikipedia free encyclopedia data mine analysi step knowl
field comput scienc process

DESCRIPTION: data mining wikipedia free encyclopedia data mining analysis step knowledge discove
field computer science process

KEYWORDS[analysi, knowledg, databa]

LANGUAGE: english

1 Data Mining: What is Data Mining? - Home | UCLA Anderson School of ...

Overview Generally, data mining (sometimes called data or knowledge discovery) is the process of a
into useful ...

http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm

PREPROCESSED DESCRIPTION: data mine data mine home ucla anderson school overview general c
analyz data differ perspect summar use

DESCRIPTION: data mining data mining home ucla anderson school overview generally data mining s
analyzing data different perspectives summarizing useful

KEYWORDS[home, knowledg, use]

LANGUAGE: english

Fig. 1. Example window of Search Engine system

Tables 3 and 4 contains selected pages from the result list returned by Bing search engine and group labels to which they have been assigned by 4 examined clustering methods.

The results are slightly different depending on the algorithm, however each of clustering scheme is correct regarding content of clusters (excluding STC). The pages: "*The Java^{TM} Tutorials*" and "*Learn Java – Tutorials, Tips, Help...*" were placed depending on the algorithm: in the group *Java Tutorials* (Lingo), *Tutorials* (EM) and in two groups *Programming* and *Learn* (XMeans). One of the mentioned pages was assigned by Lingo to two different groups (*Java Tutorials* and *Java Language*). The page from Wikipedia ("*Java (programming language) – Wikipedia,...*") was clustered as *Java Language* (Lingo), *Sun* (XMeans) or *Programming* (EM) and the page "*Java Programming Resources – Java, Java, and ...*" was assigned to the cluster *Java Tutorials* (Lingo) and *Programming* (XMeans and EM). The remaining two pages concerning technology was grouped by Lingo to the cluster *Java Technology* and to the cluster *Developers* (XMeans and EM),

Table 3. Part of clustering results of *Java* query generated by Lingo and STC methods (some URLs have been shortened)

Page title & Page URL	Group label	
	Lingo	STC
The Java^{TM} Tutorials http://download.oracle.com/javase/tutorial/	Java Tutorials Java Language	Java
Learn Java – Tutorials, Tips, Help ... http://java.about.com/	Java Tutorials	Java
Java (programming language) - Wikipedia, ... http://en.wikipedia.org/.../Java_(programming...	Java Language	Java
Java Programming Resources – Java, Java, and ... http://www.apl.jhu.edu/ hall/java/	Java Tutorials	Java
Oracle Technology Network for Java Developers http://www.oracle.com/.../java/index.html	Java Technology	Java
Nokia Developer - Java http://www.developer.nokia.com/Develop/Java/	Java Technology	Java

Table 4. Part of clustering results of *Java* query generated by XMeans and EM methods (some URLs have been shortened)

Page title & Page URL	Group label	
	XMeans	EM
The Java^{TM} Tutorials http://download.oracle.com/javase/tutorial/	Programming	Tutorials
Learn Java – Tutorials, Tips, Help ... http://java.about.com/	Learn	Tutorials
Java (programming language) - Wikipedia, ... http://en.wikipedia.org/.../Java_(programming...	Sun	Programming
Java Programming Resources – Java, Java, and ... http://www.apl.jhu.edu/ hall/java/	Programming	Programming
Oracle Technology Network for Java Developers http://www.oracle.com/.../java/index.html	Oracle	Developers
Nokia Developer - Java http://www.developer.nokia.com/Develop/Java/	Developers	Developers

however, only XMeans assigned one page connected with Oracle company to *Oracle* group. Unfortunately, STC algorithm placed all of the pages in one group *Java*.

5.2 Experiments with small datasets

In this experiment datasets were created from Google [17] search results of queries concerning cities names. It has been selected snippets from two initial pages of results. The queries were composed to examine abilities of the methods (XMeans and EM) to separate completely different as well as slightly similar clusters.

The first case of data, 2*cities*, contains results of *Warsaw* nad *New York* queries. Numbers of snippets in each list were 18, 20 and 20 respectively (see Table 5). In the second case - 3*cities* - results of query *London* have been added to the previous file. Finally, in the last data set - 3*cities&airport* - snippets from results of *Warsaw airport* query have been joined.

Despite of subjective evaluation of clusterings (see summary of selected results in Tables 6 and 7), it was performed objective assessment - calculation Rand index (see Table 8).

Table 5. Components (queries) of 2*cities*, 3*cities* and 2*cities&airport* data

Data set	Query	Number of results
2cities	Warsaw	18
2cities	New York	20
3cities	Warsaw	18
3cities	New York	20
3cities	London	20
3cities&airport	Warsaw	18
3cities&airport	New York	20
3cities&airport	London	20
3cities&airport	Warsaw airport	10

Table 6. Results of clustering into 3 groups of 3*cities&airport* data generated by EM method

Group label	Number of elements	Original components
New	16	16(New York)
London	24	2(Warsaw), 3(New York), 19(London)
Warsaw	29	17(Warsaw), 2(London), 10(Airport)

Table 7. Results of clustering into 4 groups of 3*cities&airport* data generated by EM method

Group label	Number of elements	Original components
New	20	20(New York)
London	17	1(Warsaw), 2(Airport), 14(London)
Warsaw	26	18(Warsaw), 6(London), 2(Airport)
Airport	6	6(Airport)

Table 8. Results of clustering (Rand coefficient) of 2*cities*, 3*cities* and 2*cities&airport* data

Data set	Method name	Number of groups	Rand
2cities	EM	-	0.55
2cities	EM	2	0.95
3cities	EM	-	0.75
3cities	EM	3	0.69
3cities	XMeans	3	0.85
3cities&airport	EM	3	0.81
3cities&airport	EM	4	0.86

An interesting experiment was one, when data contained 3 main groups (*New York*, *Warsaw*, *London*) and one another - *Warsaw airport*, which might be a subgroup of *Warsaw* cluster. Tables 6 and 7 show results of clustering the data, when a number of groups was set to 3 and 4 respectively.

In the first case all the results from query *Warsaw airport* is clustered to *Warsaw* group (forming a subgroup), whereas in the following experiment, in which the number of groups is equal to the number of result lists - most of them is separated into additional cluster.

5.3 Experiments with large dataset

The dataset - *ambiguous* - is taken from Credo repository. The topics of the queries were selected from the ambiguous Wikipedia list. Elements of the list contain the word "disambiguation" in the titles, e. g. "Aida" is a title of opera by Giuseppe Verdi, as well as "a set of defined interfaces and formats for representing common data analysis objects, primarily used by researchers in high-energy particle physics" [21]. In the experiments it has been used 100 results concerning 10 main topis, which were clustered using EM and XMeans methods.

In both cases of algorithms, a number of clusters was given: 10 in EM and ranging from 10 to 20 in XMeans. The other parameters were as follows: documents description method: TFIDF and length of description vector was ranging from 10 to 20. Selected results are shown in Tables 9, 10 and 11.

Table 9. Clustering results of *ambiguous* data generated by EM method

Group label	Number of elements	Original components
B-52	98	98 (B-52)
Bronx	97	97 (Bronx)
Cube	91	91 (Cube)
Aida	92	92 (Aida)
Eos	92	92 (Eos)
Camel	98	98 (Camel)
Beagle	97	97 (Beagle)
Sea	1	1 (Aida)
Coral	238	7 (Aida), 3 (Bronx) 2 (B-52), 4 (Cain) 3 (Beagle), 2 (Camel) 100 (Coral Sea), 9 (Cube) 8 (Eos), 100 (Excalibur)
Cain	96	96 (Cain)

Tables show, that the identified groups are homogenous in most of cases. The groups generated by EM method (see Table 9) contain more than 90 of 100 elements from the original partition. The only exception are groups: *Sea*, which is composed of 1 element and *Coral* containing mostly 2 original groups: *Coral Sea* and *Excalibur*.

Table 10 presents a result created by XMeans method, when a number of groups was set on 10. The algorithm generated 8 groups: 5 large homogenous (70-90 elements) clusters, 2 small, but composed of only one original partition. Unfortunately, there is also the greatest cluster, which contains 5 original groups.

In the following part of experiment, the number of groups given to XMeans has been increased to 20. The method has generated 16 clusters, which sizes were ranging from 2 to 94. Contrary to the previous part, in this the formed groups (except for 2) were homogenous.

In this experiment the results were also evaluated by Rand index. Its value for EM result (see Table 9) was 0.956, whereas for XMeans partitioning were 0.77 (see Table 10) and 0.966 (see Table 11). The high values (around 1) indicate great compatibility the generated results with original groups.

Table 10. Clustering results (8 groups) of *ambiguous* data generated by XMeans method

Group label	Number of elements	Original components
Bronx	98	98 (Bronx)
Cube	88	88 (Cube)
Aida	76	76 (Aida)
Camel	89	89 (Camel)
Music	11	11 (Camel)
Sea	17	17 (Aida)
Coral	521	7 (Aida), 2 (Bronx) 100 (B-52), 1 (Cain) 100 (Beagle), 100 (Eos) 100 (Coral Sea), 12 (Cube) 100 (Excalibur)
Cain	99	99 (Cain)

6. Conclusions

The purpose of this paper was to present possibilities of application of clustering methods in grouping Web search results. Two algorithms were selected and used in the experiments - EM and XMeans. The experiments were divided into 3 parts: clustering on-line snippets from a search engine, verification of ability to discover different as well as similar clusters and clustering large data containing ambiguous search results. The results were evaluated by Rand index or compared to partitionings from Lingo and STC systems.

There are many improvements to make, however the presented results show great usefulness of traditional clustering methods in the domain of SRC. In the first, on-line experiment, the created clusters consisted of similar snippets, which were described by adequate labels. Moreover, the groups were not fragmented and the labels were diversified, as well.

In the remaining experiments, in most cases, original clusters were properly identified by the examined methods: EM and XMeans. Generated clusters in many results were homogenous and Rand coefficient was about 0.8. It is particularly evident in case of ambiguous data. Labels of clusters were relevant to its content, however in future it is desirable to describe them by phrases. It is worth recalling the fact of identification of a subgroup in the second experiment.

It may be concluded, that as long as clustering algorithms are proposed they should be checked in SRC domain. Particularly interesting are methods generating

Table 11. Clustering results (16 groups) of *ambiguous* data generated by XMeans method

Group label	Number of elements	Original components
Aida	87	87 (Aida)
Cube	81	81 (Cube)
Camel	61	61 (Camel)
Eos	75	75 (Eos)
Bronx	87	87 (Bronx)
Information	33	33 (Camel)
Amp	2	2 (Cube)
B-52	85	85 (B-52)
Coral	84	84 (Coral Sea)
Sea	17	15 (Coral Sea), 1 (Aida) 1 (Eos)
Musicals	72	8 (Excalibur), 10 (Aida) 9 (B-52), 8 (Bronx) 4 (Beagle), 3 (Camel) 11 (Cain), 1 (Coral Sea) 7 (Eos), 11 (Cube)
Cain	86	86 (Cain)
Beagle	94	94 (Beagle)
Excalibur	89	89 (Excalibur)
Reviews	30	3 (Excalibur), 2 (Aida) 1 (B-52), 3 (Bronx) 2 (Cain), 5 (Cube) 14 (Eos)
Photo	17	5 (B-52), 2 (Bronx) 1 (Cain), 1 (Cube) 2 (Beagle), 3 (Eos) 3 (Camel)

compact and separable groups as well as able to identify hierarchical relationships in clustering results.

References

[1] R. Campos, G. Dias, C. Nunes, WISE: Hierarchical Soft Clustering of Web Page Search Results based on Web Content Mining Techniques, Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence, 2006, pp. 301-304

[2] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, 39, 1977, pp. 1–38

[3] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, Journal of Intelligent Information Systems, 17(2/3), 2001, pp. 107–145

[4] S. Lawrence, C. L. Giles, Accessibility and Distribution of Information on the Web, Nature (400), 1999, pp. 107-109

[5] M. Mahdavi, H. Abolhassani, Harmony K-means algorithm for document clustering, Data Mining and Knowledge Discovery(18), 2009, pp. 370–391

[6] S. Osiński, An algorithm for clustering of web search results, Master Thesis, Poznan University of Technology, 2003

[7] D. Pelleg, A. Moore, X-means: Extending K-means with Efficient Estimation of the Number of Clusters, Proceedings of International Conference on Machine Learning, 2000, pp. 727-734

[8] G. Salton, A Vector Space Model for Automatic Indexing, Communications of the ACM, 18(11), 1975, pp. 613-620

[9] M. Sathya, J. Jayanthi, N. Basker, Link Based K-Means Clustering Algorithm for Information Retrieval, Proceedings of IEEE-International Conference on Recent Trends in Information Technology, 2011, pp. 1111-1115

[10] W. Rakowski, An intelligent search engine using clustering methods to optimize search results, Master Thesis (in Polish), Bialystok University of Technology, 2011

[11] D. Weiss, A Clustering Interface for Web Search Results in Polish and English, Master Thesis, Poznan University of Technology, 2001

[12] D. Weiss, The search for meaning in a haystack, Seminar of Institute of Linguistics, Polish Academy of Science (in Polish), 2003

[13] I.H. Witten, E. Frank, M.A. Hall, Weka: data mining software in Java, `[http://www.cs.waikato.ac.nz/ml/weka/]` (26.06.2012)

[14] O. Zamir, O. Etzioni, Grouper: A Dynamic Clustering Interface to Web Search Results, WWW Computer Networks 31(11-16), 1999, pp. 1361-1374

[15] D. Zhang, Y.Dong, Semantic, Hierarchical, Online Clustering of Web Search Results, APWeb'2004, 2004, pp. 69-78

[16] Bing Search Engine, [http://www.bing.com] (10.09.2011)

[17] Google Search Engine, [http://www.google.pl] (15.10.2012)

[18] Carrot2 Clustering Engine, [http://search.carrot2.org/stable/search]

[19] Web search results datasets, [http://credo.fub.it/] (26.06.2012)

[20] Google Blog, [http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html], (21.06.2012)

[21] Wikipedia description of Aida expression, [http://en.wikipedia.org/wiki/Aida_%28disambiguation%29]

[22] Vivisimo company, [http://vivisimo.com] (10.09.2011)

IDENTYFIKOWANIE DOKUMENTÓW PODOBNYCH W WYNIKACH WYSZUKIWANIA W SIECI WWW

Streszczenie: Przeszukiwanie sieci WWW jest niezmiernie trudnym zadaniem. Według Zamira i Etzioniego Internet to "miejsce bez struktury, niezorganizowane i zdecentralizowane". Chociaż istnieją potężne narzędzia w postaci wyszukiwarek internetowych, ich użycie staje się z czasem trudniejsze, gdyż ilość zaindeksowanych stron internetowych przekracza 1 bln [20] i nadal rośnie. Większość wyszukiwarek generuje wyniki posortowane według ich zgodności z treścią zapytania w postaci bardzo długich list. Takie podejście nie jest najlepszym rozwiązaniem z powodu rozmiaru list oraz zawierania w nich dokumentów nie związanych z zapytaniem. W celu zwiększenia efektywności przeszukiwania Internetu można zastosować grupowanie podobnych dokumentów z generowanej przez wyszukiwarki listy wyników. Jednym z takich narzędzi są tradycyjne algorytmy grupujące. W artykule przedstawiono wyniki grupowania dokumentów bezpośrednio z listy zwróconej przez wyszukiwarkę oraz zbiorów dokumentów utworzonych z wyników wyszukiwania dla kilku zapytań. Wykorzystano następujące metody grupujące: EM i XMeans.

Słowa kluczowe: grupowanie wyników wyszukiwania, podobieństwo dokumentów, grupowanie snippetów

Artykuł zrealizowano w ramach pracy badawczej nr S/WI/5/08.

MODELING DYNAMICAL SYSTEMS BY MEANS OF DYNAMIC BAYESIAN NETWORKS

Anna Łupińska–Dubicka

Faculty of Computer Science, Białystok University of Technology

Abstract: Bayesian networks (BNs) are powerful tools for modeling complex problems involving uncertain knowledge. They have been employed in practice in a variety of fields. Their extension to time-dependent domains, dynamic Bayesian networks (DBNs) allow to monitor and update the system as time procedes and predict further behavior of the system. Most practical uses of DBNs involve temporal influences of the first order, i.e., influences between neighboring time steps. This choice is a convenient approximation influenced by the existence of efficient algorithms for first order models and limitations of available tools. This paper presents how to create higher order dynamic Bayesian networks and shows that introducing higher order influences can improve the accuracy of the model. To introduce the formalism to the readers, it describes a hypothetical simplified model based on a DBN.

Keywords: Bayesian networks, temporal dependencies, dynamical models, dynamic Bayesian networks

1. Introduction

The world around us is dynamic. Most of the physiological processes occurring in the human body, like many natural phenomena or processes are of dynamical character. Consequently, the modeled phenomena very often are generate time series data. Every variable is observed in successive moments of time. In addition, the analyzed phenomenon is affected not only by the current observations, but also the findings in previous periods. In this case, using static models can be inadequate and can lead to obtaining in correct results. To present and analyze the phenomena that change over time we need to use a dynamic model that takes into account the relationship between the values of the model parts in different moments of time.

This paper concentrates on models that belong to the class of probabilistic graphical models, with their two prominent members, Bayesian networks (BNs) [20] and

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 9, pp. 77-92, 2012.

dynamic Bayesian networks (DBNs) [7]. BNs are widely used as practical tools for knowledge representation and reasoning under uncertainty in equilibrium systems. DBNs extend them to time-dependent domains by introducing an explicit notion of time and influences that span over time. Most practical uses of DBNs involve temporal influences of the first order, i.e., influences between neighboring time steps. This choice is a convenient approximation influenced by existence of efficient algorithms for first order models and limitations of available tools. After all, introducing higher order temporal influences may be costly in terms of the resulting computational complexity of inference, which is NP-hard even for static models. Limiting temporal influences to influences between neighboring states is equivalent to assuming that the only thing that matters in the future trajectory of the system is its current state. Many real world systems, however, have memory that spans beyond their current state.

The idea of increasing modeling accuracy by means of increasing the time order of the model was illustrated by Shannon [21]. In his seminal paper, he shows sentences in the English language, generated by a series of Markov chain models of increasing time order, trained by means of the same corpus of text. The following example sentences come from [3].

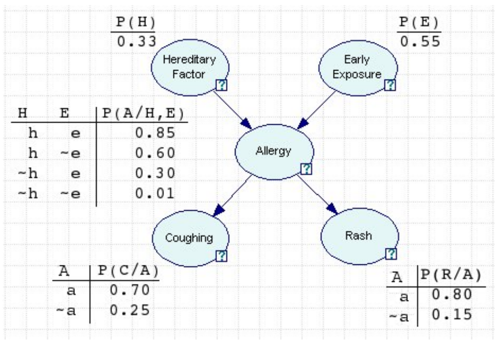
"saade ve mw hc n entt da k eethetocusosselalwo gx fgrsnoh,tvettaf aetnlbilo fc lhd okleutsndyeosthtbogo eet ib nheaoopefni ngent" In the above text, letters were generated by a zero order model. The only assumption was that the various letters appear with the different probablity. However, if we look at the various expressions of any language, we can see that the letters are very common in a certain context. In the case of English, the letter Q, in all probability will be the next U. Sample text generated using the chain the first order, in which the choice of each letter is a random function of its predecessor: *"t I amy, vin. id wht omanly heay atuss n macon aresethe hired boutwhe t, tl, ad torurest t plur I wit hengamind tarer-plarody thishand"* Going one step further, we can see that the digram TH in the English language often vowels A, E, I, O or U, slightly less consonant R or W and rarely other letters. In the following piece of text used the second-order Markov chain: *"Ther I the heingoind of-pleat, blur it dwere wing waske hat throos. Yout lar on wassing, an sit". "Yould," "I that vide was nots ther."* And so can look sample text, if we use the model of the fourth order. As we can see, most of the words are the words of English: *"His heard.» «Exactly he very glad trouble, and by Hopkins! That it on of the who difficentralia. He rushed likely?» «Blood night that."*

The resemblance of the latter sentence to ordinary English text, an informal measure of the model's accuracy, has increased dramatically between the first and the fourth orders. A first order model was essentially impotent in its ability to model the problem.

This paper presents how to create higher order dynamic Bayesian networks and shows that introducing higher order influences can improve the accuracy of the model. To introduce the formalism to the readers, it describes a hypotethical simplified model based on a DBN. The remainder of the paper is structured as follows. Section 2. introduces Bayesian networks. Section 3. presents a short description of methods dealing with dynamical processes. Section 4. presents in details dynamic Bayesian networks, one of the extension of BNs. Section 5. concludes the paper.

2. Bayesian Networks

Bayesian networks (BNs), also called belief networks or causal networks, belong to the family of probabilistic graphical models (GMs). These graphical structures are used to represent knowledge about an uncertain domain. In particular, each node in the graph represents a random variable, while the edges between the nodes represent probabilistic dependencies among the corresponding random variables Formally, a BN $\mathcal{B}$ is a pair $<\mathcal{G},\Theta>$, where $\mathcal{G}$ is an acyclic directed graph in which nodes represent random variables $X_1,\ldots,X_n$ and edges represent direct dependencies between pairs of variables. Θ represents the set of parameters that describes the probability distribution for each node X_i in $\mathcal{G}$, conditional on its parents in $\mathcal{G}$, i.e., $P(X_i|Pa(X_i))$. Often, the structure of the graph is given as a causal interpretation, convenient from the point of view of knowledge engineering and user interfaces. BNs allow for computing probability distributions over subsets of their variables conditional on other subsets of observed variables.



P(H)
0.33

P(E)
0.55

H	E	P(A/H,E)
h	e	0.85
h	~e	0.60
~h	e	0.30
~h	~e	0.01

A	P(C/A)
a	0.70
~a	0.25

A	P(R/A)
a	0.80
~a	0.15

Fig. 1. A simple BN illustrating selected causes and effects of allergy in children

Consider the simple BN shown in Figure 1, illustrating various causes and effects of allergy in children. All variables in this example are Boolean. The tendency to develop allergies has a hereditary component: Allergic parents are more likely to have allergic children, whose allergies are likely to be more severe than those from non-allergic parents. Exposure to allergens, especially in early life, is also an important risk factor for allergy. When an allergen enters the body of an allergic child, the child can cough or develop a rash. Figure 1 shows the dependency structure among the variables and the conditional probability distributions for each of the variables.

3. Modeling dynamical relationships

While Bayesian networks are powerful tool for representing uncertainty, they do not provide direct mechanism for representing temporal dependencies. Most of the events that we meet in our everyday life are not detected based on a particular point in time. They can be described through the multiple states of observations that all together lead up to final event. The ability to model effectively the temporal aspects of the domain plays cruicial role in modeling the World. For example, in medicine representing and reasoning about time is cruitial for prevention, diagnosis, terapeutic management, or prognosis. In industry, capturing the temporal aspects is essential for diagnosis and prediction of events and disturbances. The efforts to introduce temporality into Bayesian networks have resulted in a variety of networks for applications such as planning, diagnosis, forecasting, and scheduling.

Dean and Kanazawa [7] proposed a temporal belief network, a directed graphical model where nodes represent the truth of a state variable at the single point in time. The network is arranged into time slices representing the system's complete state at the single point in time, and time slices are duplicated over a predetermined and fixed–length time grid representing the time interval of interest. Links between state variables within the time slice are disallowed.

Network of dates proposed by Berzuini [4] represents a departure from the multiple instantiations approach because each temporal duration is represented by a node. Berzuini associates a probability density with each temporal random variable to represent a continous time.

An extention proposed by Santos and Young [14] is based on the definition of temporal aggretage (TA). Each aggregate represents a process changing over time. A temporal aggregate consists of the set of states, that the process can take on, and a set of temporal intervals each having an associated random variable. Their Probablistic Temporal Network (PTN) is a directed graph in which the nodes are TAs and the edges are the causal/temporal causal relationships between aggregates.

In Modifiable Temporal Bayesian Networks with Single–granularity (MTBN–SG) Aliferis and Cooper [1] distinguished three types of the variables: ordinary variables (corresponding to potentially observed phenomena), mechanism variables (corresponding to causal mechanism between variables), time–lag quantifier variables (corresponding to the time lag between the cause variable and the effect variable). All these variables create directed, possibly cyclic, graph over a range of time points.

Basing on the fact that in many cases there are only few state changes in the temporal range of interest Arroyo–Figueroa and Sucar [2] proposed an extension of Bayesian networks, called Temporal Nodes Bayesia Networks (TNBN). Their approach is based on the definition of *temporal node*, that is defined by a set of ordered pairs: the value of the variable and a time interval during the variable can change its state. Temporal interval of each temporal node are relative to the parent nodes. Each arc in TNBN corresponds to a causal–temporal relation.

In Networks of Probabilistic Events in Discrete Time (NPEDT) [12], like in TNBNs, each variable represent an event that can occur only once. However, they differ from TNBNs in that time is discretized using the same unit for all variables. The value taken on by a variable indicates the absolute, not relative, time at which the event occurs.

4. Dynamic Bayesian Networks

Dynamic Bayesian networks [7] are a temporal extension of Bayesian networks for modeling dynamic systems. While the Bayesian network shows the cumulative probability distribution over a set of random variables independent of time, the dynamic Bayesian networks can be seen as a multi–dimensional representation of a random process. DBNs allow the interpretation of the present, the reconstruction of the past and the forecasting of the future. Phenomena are located in time, and location at the time is ordered by the "earlier and later" relationship. Mostly due to the computational complexity of the inference algorithms, time is treated as a discrete variable. It should be noted that term *dynamic* means that we model the system's development over time and not that the model structure and its parameters change over time, even though the latter is theoretically possible.

4.1 Network structure

A DBN is a directed, acyclic graphical model of a stochastic process. It consists of time–steps and each of time–steps containts its own variables. The most common approach is usually assumed that the network meets the Markov property, i.e., the value of future states of the process are determined only by its current state, regardless of the past. In other words, the future states of the process are conditionally independent of the past states. Such a network is called a first order network. Usually, the DBN is often defined as a pair of BNs $(\mathcal{B}^1, \mathcal{B}^{\rightarrow})$, where $\mathcal{B}^1$ represents a priori probability distribution $P(\mathbf{Z}^1)$ of the model. Typically, $Z^t = (\mathbf{U}^t, \mathbf{X}^t, \mathbf{Y}^t)$, where $\mathbf{U}$ represents the input, $\mathbf{X}$ represents the hidden and $\mathbf{Y}$ represents the output variables of the model. $\mathcal{B}^{\rightarrow}$ is a two time slice BN (2TBN), that defines the transition distribution $P(\mathbf{Z}^t|\mathbf{Z}^{t-1})$ as follows [18]:

$$P(\mathbf{Z}^t|\mathbf{Z}^{t-1}) = \prod_{i=1}^{n} P(Z_i^t|Pa(Z_i^t)) \ ,$$

where Z_i^t is the i–th node at time t. $Pa(Z_i^t)$ are the parents of Z_i^t from the same or from the previous time–slice. The joint probablity distribution for a sequence of length T can be obtained by *unrolling* the $\mathcal{B}^{\rightarrow}$ network:

$$P(\mathbf{Z}^{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{n} P(Z_i^t|Pa(Z_i^t)) \ .$$

Consider a two years old child whose parents suffer from allergy and who has been exposed to allergens. We know that this child has not developed any symptoms of allergy in the previous year. Suppose that we want to know the probability that allergy appears in the third year. If we use the BN pictured in Figure 1, we omit all historical information except that for the current year. Figure 2a) shows a DBN of first temporal order, which allows us to predict the probability of the child developing allergy in this and in the future years. Number of slices is the number of steps for which we perform the inference. The time step that is chosen for a dynamic Bayesian network varies on considered phenomenon. In this example, one step means one year. Temporal plate is the part of a DBN that contains nodes changing over time. *Hereditary Factor* is time invariant and, hence, is outside of the temporal plate. As we can see, now the value of the *Allegry* variable depends not only on *Hereditary factor* and *Early exposure* but also on its value from previous time step.

As mentioned before, the state at time t generally depends on the states at previous $k = 1$ time steps. There is nothing in the theory that prevents k from being any number between 1 and $t - k$. However, modeling dependencies of higher orders is very rare in the literature, probably because of the availability of both theoretical tools (algorithms) and practical (availability of software). In many cases, taking into consideration only the first–order dependences is probably sufficient. However, there is a possibility that some phenomena could be modeled with higher accuracy if they also take account of the influence of states earlier than immediately preceding the

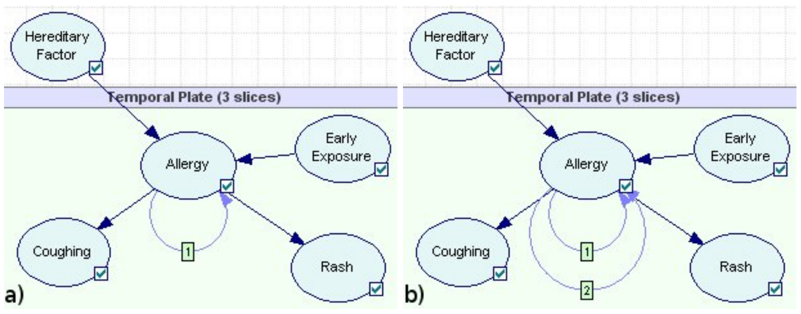


Fig. 2. A DBN modeling causes and effects an allergy in children: a) first order b) second order

current state of the model. It is likely that such simplification of dynamic models can lead to incomplete and even erroneous results. According to Murphy [16], it is possible to simulate k^{th}–ordered processes by means of first order Markov processes by adding new variables (called lag variables). This approach, unfortunately, obscures the model by introducing many unnecessary extra variables.

The SMILE$^{☺}$ library, the part of the software developed by Decision Systems Laboratory of the University of Pittsburgh, provides an implementation of the extended DBNs formalism. This implementation is, to the best of my knowledge, the first implementation of temporal reasoning that provides support for k^{th}-order temporal arcs.

In case of k^{th}–ordered processes, $\mathcal{B}^{\rightarrow}$ can be defined not as 2TBN, but as a $(k+1)$TBN and describes the transition model $P(\mathbf{Z}^t|\mathbf{Z}^{t-1},\mathbf{Z}^{t-2},\ldots,\mathbf{Z}^{t-k})$ as follows:

$$P(\mathbf{Z}^t|\mathbf{Z}^{t-1},\mathbf{Z}^{t-2},\ldots,\mathbf{Z}^{t-k}) = \prod_{i=1}^{n} P(Z_i^t|Pa(Z_i^t))\ ,$$

In this case, the set of parents $Pa(Z_i^t)$ can contain nodes not only from the previous time–slice, but also from time–slices further in the past. The joint probablity distribution for a sequence of length T can be obtained by *unrolling* the (k+1)TBN network:

$$P(\mathbf{Z}^{1:T}) = \prod_{t=1}^{T}\prod_{i=1}^{n} P(Z_i^t|Pa(Z_i^t))\ .$$

Typically, the older the child, the lower the probability of allergy appearing. And, generally, a child who has not developed allergy two years in a row has a lower chance of developing allergy in the third year. A reasonable expectation is that modeling higher order dependencies should increase the accuracy of the model. Figure 2b) shows a second time-order DBN, i.e., a model in which there are two temporal arcs from node *Allergy*, the first order takes the information from one step before, the second from two steps before.

4.2 Inference

The main goal of inference for DBNs is to calculate $P(X_{t1}|y_{t2:t3})$, where X_{t1} is the value at time $t1$ and $Y_{t2:t3}$ represents all observation between times $t2$ and $t3$. There exist several interesting ways of performing inference on DBNs. The three most common types are illustrated below.

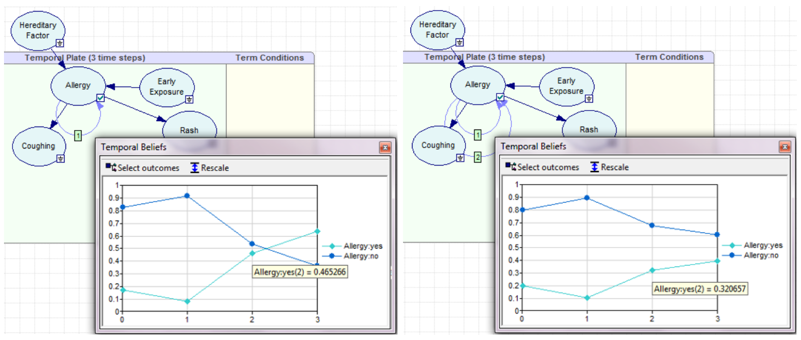


Fig. 3. First (a) and second (b) order DBN's temporal beliefs and probability of allergy in the third year (filtering)

Filtering. The current belief state is computed given all evidence from the past. It is used to keep track of the current state for making rational decisions. This task is traditionally called *filtering*, because we are filtering out the noise from the observations. However, in some circumstances the term *monitoring* might be more appropriate.

For example, we want to know the probability that allergy appears in the third year. Figure 3a) presents temporal beliefs for the first order network. As we can see, the probablity of allergy in the third year equals 46.5%. On the other hand, in Figure 3b) we can see that the probablity of allegry in the third year for the second order model is lower than from the first order model and equals 32.1%.

Smoothing. Sometimes we want to estimate the state of the past, given all the evidence up to the current time, i.e., $P(X_{t-l}|y_{1:t})$, where $l > 0$ is how far we want to look back. This type of the inference can be useful to get a better estimate of the past state, because more evidence is avaiable at time t than at time $t-l$.

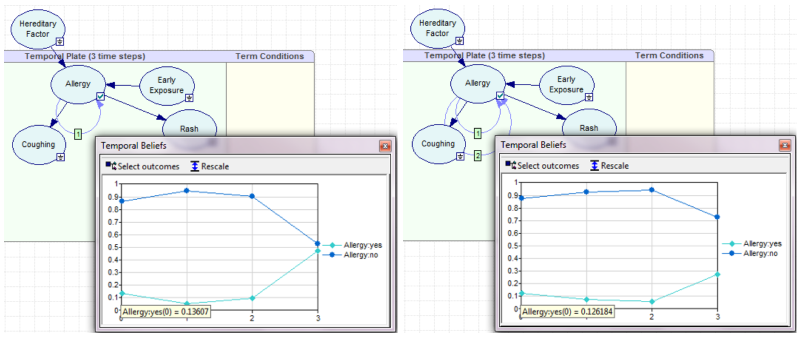


Fig. 4. First (a) and second (b) order DBN's temporal beliefs and probability of allergy in the first year (smoothing)

For example, we want to know how probable was that our three years old child suffered from allergy in its first year of life, given information of lack of any symptoms of allergy in the subsequent years. Figure 4 shows that for the first order dynamic network this probability equals 13.6%, and for the second order network it equals 12.6%.

Prediction. In addition to estimating the current or past state, we might want to predict the future, i.e., compute $P(X_{t+h}|y_{1:t})$, where $h>0$ is how far we want to look ahead. This kind of inference can be used to evaluate the effect of possible actions on the future state.

For example, we want to know the probability that our child will suffer from allegry in its the fourth year even though it has not developed any symptoms of allergy up to thrird year. In Figure 5, we can see again that the second order model is more precize and estimates this probablity at 39.7% while for the first order model this probability equals 63.6%.

While dynamic Bayesian networks are an extension of Bayesian networks there already exist algorithms for inference, divided into two major categories: exact inference and approximate infenerce.

Exact inference. The first approach for exact inference in DBNs is based on the notion that an unrolled DBN is in fact the same as a static BN. In this case we can

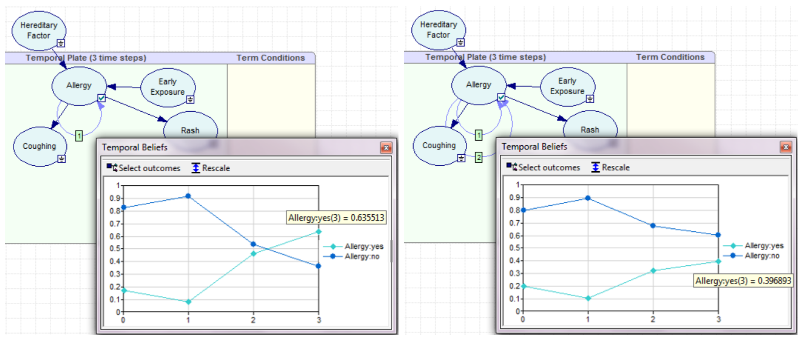


Fig. 5. First (a) and second (b) order DBN's temporal beliefs and probability of allergy in the fourth year (prediction)

use any of the inference algorithm for static BNs, such as the variable elimination or the junction tree algorithm. The basic idea of variable elimination is to take a probabilistic model over n variables and reduce it to a model over $n-1$ variables, while maintaining the ability of the model to answer queries of interest. This process is repeated until we have a trivial model from which we can look up answers immediately. The complexity of the algorithm depends on the amount of work it takes to eliminate a variable, which is sensitive to the order in which variables are eliminated.

The juntion tree algorithm converts the original Bayesian network into a tree structure where each node corresponds to a certain set of nodes (a clique) in the original network. This tree is obtained by following steps: (1) constructing an undirected graph called the moral graph from the belief network, (2) selectively adding arcs to the moral graph to form a triangulated graph, (3) identifying the maximal cliques from the triangulated graph, (4) building the junction tree, starting with cliques as the nodes, where each link between two cliques is labeled by a separator set. Unfortunately, when we create a junction tree from an unrolled DBN, the cliques tend to be very large, often making exact inference intractable.

The second approach for exact inference is to convert the DBN into a hidden Markov model (HMM) and apply the forward–backward algorithm. Converting a DBN to a HMM is only possible with discrete state DBNs. If there are N hidden variables per slice, and each such variable can have up to M values, the resulting HMM will have $S = M^N$ states. As long as S is not to large, this is good method, since the forwards–backwards algorithm is exact, and is very simple to implement.

The frontier algorithm [22] is based on the notion that in the forward–backward algorithm for HMM, variable X_t d–separates the past from the future. We can generalize this idea to DBNs noting that all nodes in a time slice d–separate the past from the future. This set of nodes is called the *frontier*. The modified $2TBN$ is here called a 1.5DBN H_t (H for half), because it is created by taking all nodes from slice 2 and only outgoing nodes from slice 1. For each H_t we construct the junction tree. Inference is performed on each separate tree and messages are passed between them via the interface nodes.

The interface algorithm [18] optimalizes the frontier algorithm by using to d–separate the past from the future only these hidden nodes in a time slice which have outgoing arcs. This subset of the frontier is called the *forward interface*.

Approximate inference Although the exact inference is always possible in discrete–state models, very often it can be computationally prohibitive. When faster results are needed, we can use approximate methods to perform inference. Generally, we can distinguish two types of approximations: *deterministic* and *stochastic*. Deterministic algorithms for the discrete–state DBNs include: the loopy belief propagation, the Boyen–Koller algorithm, and the factored frontier algorithm. Stochastic algorithms can be divided into two groups: offline and online. Offline methods are often based on importance sampling (likelihood weighting) or Monte Carlo Markov Chain (MCMC). Online methods usually rely on particle filtering (PF), also known as sequentional Monte Carlo, the bootstrap filter, the condensation algorithm, survival of the fittest, or interacting particle approximations.

In loopy belief propagation (LBP) we apply the Pearl's algorithm [20] to the original graph even if this graph contains loops (undirected cycles). In theory, this runs the risk of double counting information. However, it was shown that in practise this method works suprisingly well [19].

The basic idea behind the Boyen–Koller (BK) algorithm [6] is to approximate the joint distribution over the interface as a product of marginals. The marginals are exactly updated using the junction tree algorithm. BK constructs the junction tree for the 1.5 DBN H_t, but does not require that all the interface nodes are in the same clique. Since BK does exact inference in a two–slice DBN, sometimes it can be intractable.

The factored frontier (FF) algorithm [17] also represents the belief state as a product of marginals and thus is very similar to the Boyen–Koller algorithm. However, FF is simpler than BK, because instead of relying on the juction tree algorithm it computes the marginals directly.

Stochastic algorithms have the advantage over deterministic algorithms that they are easier to implement and that they are able to handle arbitrary models. Unfortunately, their main disadvantage is speed, they are often significantly lower than the deterministic methods, what makes them unsuitable for large model and/or large datasets. In [9] Doucet *et al.* present the Rao–Blackwellised Particle Filtering algorithm, which main idea is to integrate out some of the variables using exact inference, and apply stochastic to the remaining ones.

4.3 Learning

Creating dynamic Bayesian networks can be a complicated task. One way to construct Bayesian networks is from domain knowledge. However, in many domains, domain knowledge is not sufficient to construct a complete Bayesian network. In this case, Bayesian networks can be learned from data when data are available. The learning problem can be categorized into two groups depending on the knowledge about the structure: 1) parameter learning problem when the structure is known, and 2) structure learning problem when the structure is unknown, where the parameter learning is a part of the structure learning problem and is used as an inner loop of structure learning. Generally, the techniques for learning DBNs are the same as the techniques for learning static Bayesian networks.

Parameter learning. When the structure of the model is known, the learning task becomes one of parameter estimation. In case when we have a full sampling data, we determine the probability distibutions by computing statistics from the data samples. We want to find the values of the parameters of each CPD that maximize the likelihood of the training data, containing S independent sequences. Each of the sequences has the observed values of all n nodes per slice for each of T slices. When N is small compared to the number of parameters that we are estimating, we use the Maximum A Posterori (MAP) estimates rather than the Maximum Likelihood (ML) estimates.

In case when there are hidden variables and/or missing data, exact methods for computing the probablity distributions become intractable. In such case, we must use iterative methods, such expectation–maximization (EM) [15] or gradient ascent [5] algorithm to find a local maximum of the ML/MAP function. The key similarity between them is that the information they require is computed with the inference routines. Another similarity is that in general, both are guaranteed to find only a local optimum in the parameter space. The gradient ascent techniques have the advantage of greater generality, while the EM algorithm has the advantages of simplicity and robustness. Gradient ascent can be thought of as moving a point corresponding to the parameter values through parameter space so as to maximize the likelihood function. The EM algorithm consists of two major steps: an expectation step followed by a maximization step. The expectation is with respect to the unknown underlying variables, using the current estimate of the parameters and conditioned upon the observation. The maximization step then provides a new estimate of the parameters. These two steps are iterated until convergence.

The parameter learning algorithm implemented in the SMILE☺ library is based on fact that a DBN has a limited number of CPDs that need to be learned. By decomposing the DBN into several BNs we can use existing BN learning algorithms. The DBN is unrolled for $k+1$ time–slices, where k denotes the temporal order of the Markov process. Then, the unrolled DBN is decomposed into separate BNs accountable for the initial paramaters and the temporal parameters.

Structure learning. If we know a number and type of some states in the network, but we do not know their relations and mutual independence, according to [13] we need a structure learning algorithm searching over the space of possible, alternative structures to identify the one (or those) having the highest score. by the data. This requires a scoring function for candidate structures and an efficient search procedure, since the list of potential structures grows exponentially with the number of nodes. In addition to the computational cost, another important consideration is the amount of data that is required to reliably learn structure. But, in practise we can reduce the data requiremetns considerably, because we often have strong prior knowledge about the structure or at least parts of it. Dojer [8] proposed an algorithm for finding an optimal structure of the BN from data, relying on the relaxation of the acyclity constraint. While the unrolled DBN is always acyclic, we can use this algorithm also for learning the DBN structure.

Friedman [10,11] propose one method for learning both the network structure and the parameters from partially observed data the structural expectation–maximization algorithm (SEM). It can be described as an iteration of following steps: 1) adding a new node to the network, representing a hidden variable, and 2) finding as good as possible network connections for given set of nodes. These steps are repeated as long as the network keeps improving.

5. Summary

Because majority of events encountered in every day life are described by sets of observations taken in successive moments of time, we need models capable of dealing with temporal dependencies. This paper concentrats on dynamic Bayesian networks, an temporal extension of Bayesian networks that allow to model dynamical processes. It illustrates, by mean of an example, different cases of the inference in DBNs and shortly described learning concepts depending on given knowledge. Most practical uses of DBNs involve temporal influences of the first order, i.e., influences between neighboring time steps. It is likely that this can lead to incomplete or even erroneous results. This paper presents usage of higher orders dynamic Baseyian networks and shows that introducing influences of states earlier than the state immediately preceding the current state can improve accuracy of inference, not only in case of prediction but also in case of smoothing and filtering. In addition, this article shortly presents a description of different approaches dealing with dynamical processes.

Acknowledgments

The empirical part of the paper was performed using SMILE☺, an inference engine, and GeNIe, a development environment for reasoning in graphical probabilistic models, both developed at the Decision Systems Laboratory, University of Pittsburgh, and available at `http://genie.sis.pitt.edu/`.

References

[1] Constantin Aliferis and Gregory Cooper. A Structurally and Temporally Extended Bayesian Belief Network Model: Definitions, Properties, and Modeling Techniques. In *Proceedings of the Twelfth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 28–39, San Francisco, CA, 1996. Morgan Kaufmann.

[2] Gustavo Arroyo-Figueroa and Luis Enrique Sucar. Temporal Bayesian Network of Events for Diagnosis and Prediction in Dynamic Domains. *Applied Intelligence*, 23:77–86, October 2005.

[3] Jon .L. Bentley. *Programming pearls*. ACM Press Series. Addison-Wesley, 2000.

[4] Carlo Berzuini. Representing time in causal probabilistic networks. In *Uncertainty in Artificial Intelligence 5 Annual Conference on Uncertainty in Artificial Intelligence (UAI-89)*, pages 15–28, Amsterdam, NL, 1989. Elsevier Science.

[5] John Binder, Daphne Koller, Stuart Russell, Keiji Kanazawa, and Padhraic Smyth. Adaptive Probabilistic Networks with Hidden Variables. In *Machine Learning*, pages 213–244, 1997.

[6] Xavier Boyen and Daphne Koller. Approximate Learning of Dynamic Models. In *IN NIPS-11*, pages 396–402. MIT Press, 1998.

[7] Thomas Dean and Keiji Kanazawa. A Model for Reasoning About Persistence and Causation. *Computational Intelligence*, 5(2):142–150, 1989.

[8] Norbert Dojer. An Efficient Algorithm for Learning Bayesian Networks from Data. *Fundam. Inf.*, 103(1-4):53–67, January 2010.

[9] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao–Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Proceedings of the Sixteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 176–183, San Francisco, CA, 2000. Morgan Kaufmann.

[10] Nir Friedman. Learning Belief Networks in the Presence of Missing Values and Hidden Variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, 1997.

[11] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the Structure of Dynamic Probabilistic Networks. pages 139–147. Morgan Kaufmann, 1998.

[12] Severino F. Galán, Francisco Aguado, Francisco Javier Díez, and José Mira. Nasonet, Modeling the Spread of Nasopharyngeal Cancer with Networks of Probabilistic Events in Discrete Time. *Artificial Intelligence in Medicine*, 25(3):247–264, 2002.

[13] David Heckerman. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1996.

[14] Eugene Santos Jr. and Joel D. Young. Probabilistic temporal networks. Technical report, Air Force Institute of Technology, 1996.

[15] Steffen L. Lauritzen. The EM Algorithm for Graphical Association Models with Missing Data. *Computational Statistics and Data Analysis*, 19:191–201, February 1995.

[16] Kevin Murphy and Saira Mian. Modelling gene expression data using dynamic bayesian networks. Technical report, 1999.

[17] Kevin Murphy and Yair Weiss. The Factored Frontier Algorithm for Approximate Inference in DBNs. In *Proceedings of the Seventeenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 378–385, San Francisco, CA, 2001. Morgan Kaufmann.

[18] Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.d. dissertation, University of California Berkeley, Computer Science Division, 2002.

[19] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *In Proceedings of Uncertainty in AI*, pages 467–475, 1999.

[20] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann PUBLISHERs, Inc., San Mateo, CA, 1988.

[21] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[22] Geoffrey Zweig and Stuart Russell. Speech recognition with dynamic bayesian networks. Technical report, 1998.

MODELOWANIE SYSTEMÓW DYNAMICZNYCH PRZY UŻYCIU DYNAMICZNYCH SIECI BAYESOWSKICH

Streszczenie: Sieci Bayesowskie (Bayesian networks, BNs) są popularnym narzędziem do reprezentacji wiedzy w warunkach niepewności. Znalazły praktyczne zastosowanie w wielu dziedzinach. Ich rozszerzenie o domenę czasową, dynamiczne sieci bayesowskie (dynamic Bayesian networks, DBNs) umożliwiają monitorowanie oraz aktualizację systemów zmieniających się wraz z upływem czasu, a także predykcję przyszłego stanu takiego systemu. Większość praktycznych zastosowań dynamicznych sieci Bayesowskich bierze pod uwagę tylko zależności pierwszego rzędu, to znaczy, że bieżący stan systemu zależy tylko od jego stanu w bezpośrednio poprzedzającym go kroku czasowym. Takie założenie jest uproszczeniem, wynikającym najprawdopodobniej z braku efektywnych narzędzi zdolnych obsłużyć modele wyższych rzędów. Niniejszy artykuł przedstawia na przykładzie sposób w jakim tworzy się modele wyższych rzędów oraz pokazuje, wpływy wyższych rzędów mogą zwiększyć jakość modelu.

Słowa kluczowe: sieci bayesowskie, zależności temporalne, modele dynamiczne, dynamiczne sieci bayesowskie

SERVICE STRATEGIES IN TANDEM SERVER NETWORKS WITH FEEDBACK AND BLOCKING

Walenty Oniszczuk

Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

Abstract: In this paper, we consider specialized tandem server networks with finite buffer capacities, feedback and intelligent service strategy, which are one of the key elements in ensuring quality of service in computer systems. Here, the two strategies of tasks service are presented and compared. Generally, in this paper two models of linked computer servers with blocking and with feedback service according to the HOL priority scheme are investigated. These kinds of models, describe behaviour of computer tandem networks, exposed to open Markovian queuing models with blocking. These models which are illustrated below are very accurate, derived directly from two-dimensional state graphs. In our examples, the performance is calculated and numerically illustrated by regulating intensity of the input flow and varying buffer capacities.

Keywords: feedback and blocking, service strategy, network performance analysis

1. Introduction

Queuing network models have been widely used to evaluate the performance of computer systems and communication networks. Queuing theory was developed to understand and to predict the behaviour of real life systems. Queuing networks models with finite capacity queues and blocking and feedback have been introduced and applied as more realistic models of systems with finite capacity resources and with population constrains [1, 2, 5, 12, 13]. Over the years high quality research has appeared in diverse journals and conference proceeding in the field of computer science, traffic engineering and communication engineering. However, there are still many important and interesting finite capacity queues under various blocking mechanisms and synchronization constraints to be analyzed [3, 4, 6, 9-11].

Most of specialized computer systems are connection oriented, which are also known as linked in series. There are many blocking models of linked in series networks that can be used to provide insight into the performance of those networks.

Blocking models, if they can be solved efficiently, are often used in network planning and dimensioning. Due to obvious resource constrains, realistic models have finite capacity buffers, where the queue length cannot exceed its arbitrary maximum threshold. When the queue length reaches its capacity, the buffer and the server are said to be full (blocking factors). Queuing network models (QNMs) with finite capacity queues and blocking provide powerful and practical tools for performance evaluation and predication of discrete flow systems in computer systems and networks.

Despite all the research done so far, there are still many important and interesting models to be studied. For example, finite capacity queues under various blocking mechanisms and synchronization constraints, such as those involving feedback service with priority scheduling, where in a feedback queue, a task with a fixed probability can return to the previous node immediately after its service at the current node.

The paper is organized as follows. Section 2, describes the analytical models of a tandem network with two proposed feedback service strategy in the first server. Models implementation and numerical examples are described in Section 3. And finally, conclusions are drawn in Section 4.

2. Models description and notations

The most common queuing models assume that the interarrival and service times are exponentially distributed. Equivalently, the arrival and service processes follow a Poisson distribution. That is, if the interarrival times are exponentially distributed then the number of arrivals at the system follows a Poisson distribution. Similarly, for the service process.

We consider an open queuing model of tandem networks with a single task class and three stations: a source, station *A* and *B* (see Figure 1). Tasks arrive from the source at station *A* according to the Poisson process with rate λ. The service rates at each station are μ_1^A, μ_2^A (for feed backed tasks) and μ^B, respectively. After service completion at station *A*, the task proceeds to station *B*. Once it finishes at station *B*, it gets sent back to station *A* for re-processing with probability σ. We are also assuming that tasks are leaving the network with $1-\sigma$ probability. Service at each station is provided by a single exponential server.

In the first model, a feed backed task is served at station *A* according to a non-preemptive priority scheme (Head-of-Line (HoL) priority discipline) independently of all other events, where a task cannot get into service at station *A* (it waits at station *B* - blocking factor) until the task currently in service is completed. Once finished, each re-processed task departs from the network. The successive service times at both stations are assumed to be mutually independent and they are independent of the state of the network.

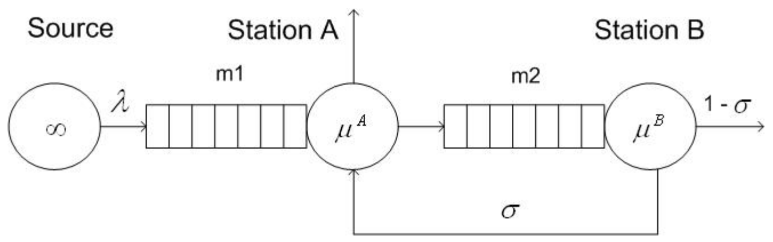


Fig. 1. Illustration of the three-station network model with feedback

Between station *A* and station *B* is a common waiting buffer with finite capacity *m*2. When this buffer is full, the accumulation of new tasks from the first station is temporarily suspended. Similarly, if the first buffer (with capacity *m*1) ahead of the first station is full, then the source station is blocked.

In theory, any Markov model can be solved numerically. In particular, solution algorithm for Markov queuing networks with blocking and priority feedback service is a five-step procedure:

a) Definition of the tandem station state space and choosing its state space representation.
b) Enumerating all the transitions that can possible occur among the states.
c) Definition of the transition rate matrix that describes the network evaluation that means generating the transition rate.
d) Solution of linear system of the global balance equations to derive the stationary state distribution vector (computing appropriate probability vector).
e) Computation, from the probability vector, of the average performance indices.

2.1 First service strategy for feed backed task: Head of Line

According to general assumptions, a continuous-time homogeneous Markov chain can represent this tandem network. The queuing network model reaches a steady-state condition and the underlying Markov chain has a stationary state distribution. The underlying Markov process of a queuing network with finite capacity queues has finite state space. If the numbers of tasks located simultaneously in the network in the first and second servicing stations are denoted by *i* and *j* plus an index *k* depicted

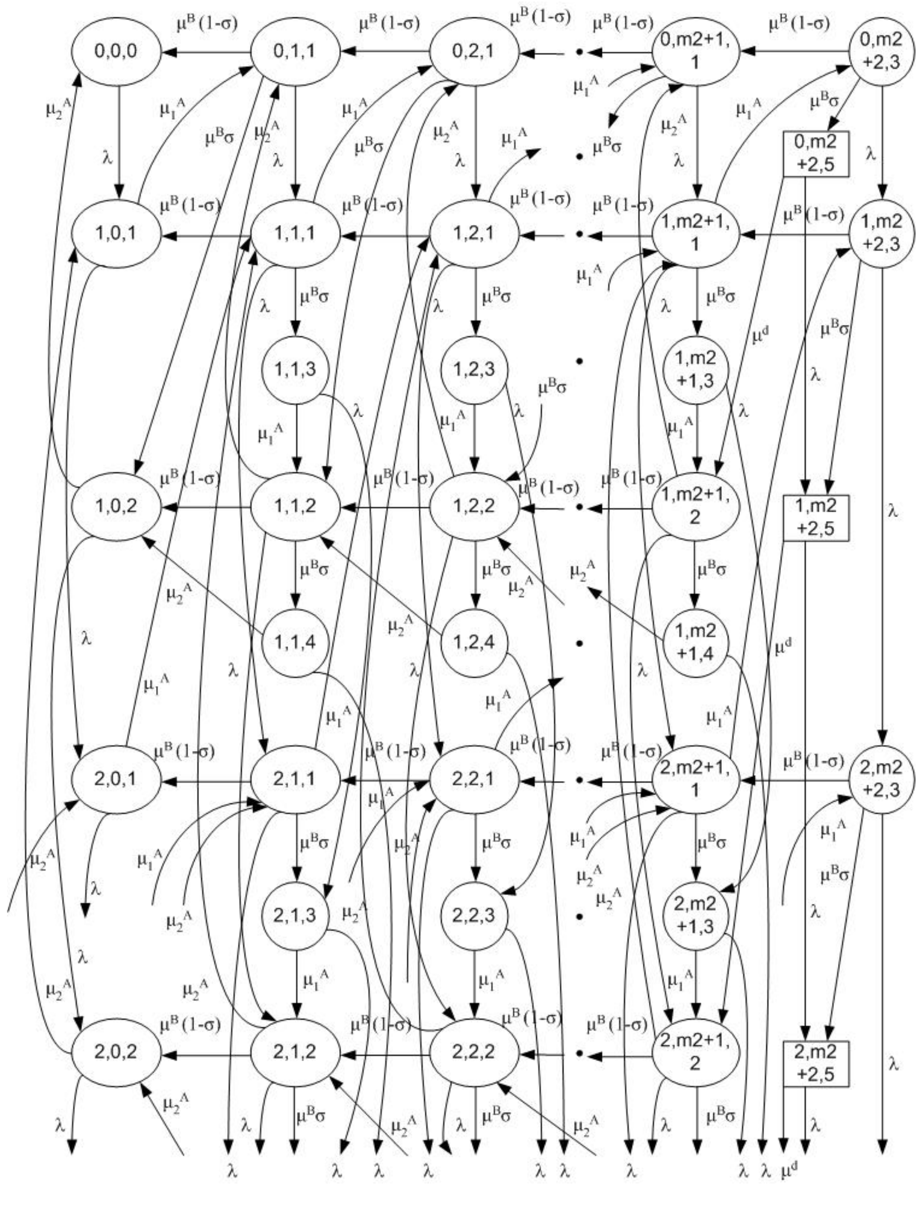


Fig. 2. Two-dimensional tandem network state diagram (first part)

the state of the each server, then a Markov model with two-dimensional state space is defined in this paper (see Figure 2 and Figure 3).

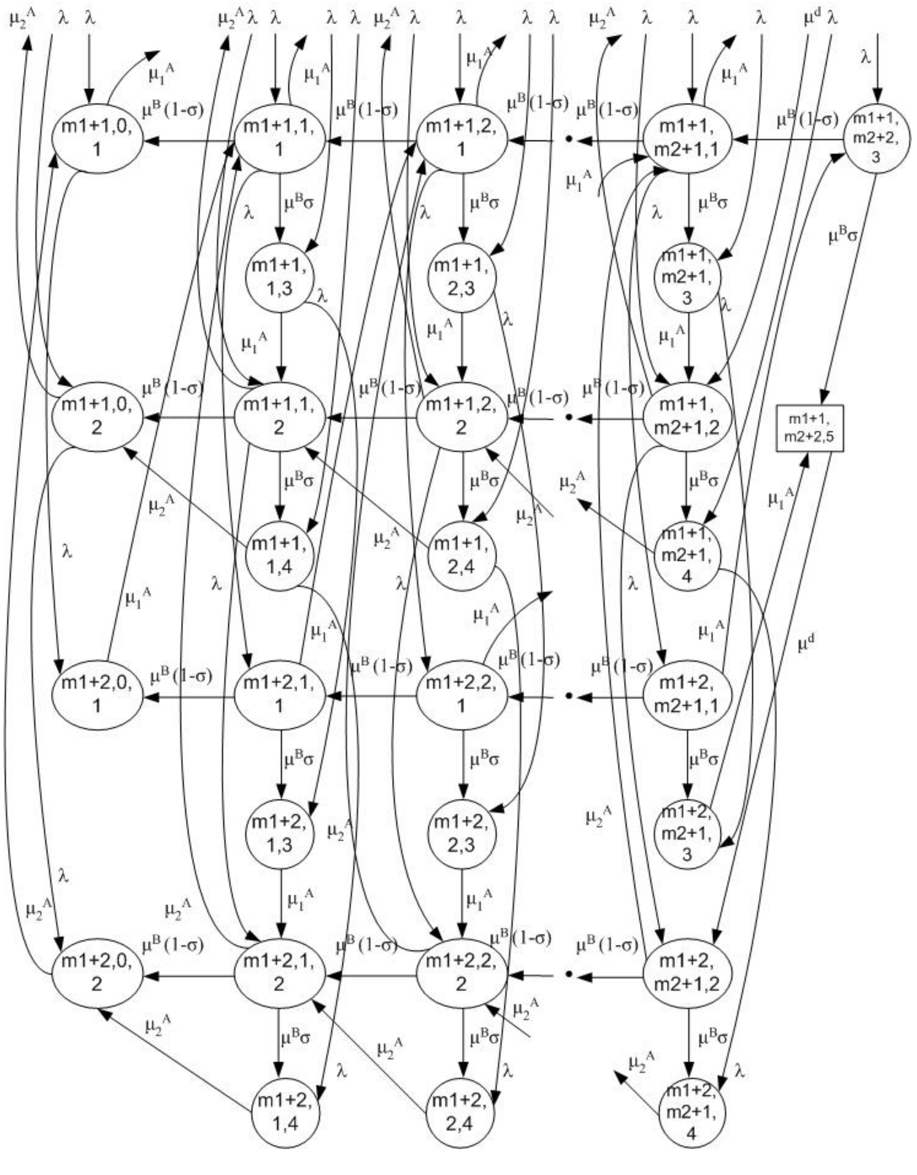


Fig. 3. Two-dimensional tandem network state diagram (second part)

In these diagrams, the index k has the following values: $0,1,2,3,4,5$. The value $k=0$ describes the idle network, $k=1$ servicing the ordinary tasks, $k=2$ servicing priority tasks, $k = 3$ blocking tasks, $k = 4$ blocking tasks and priority service, $k = 5$ describes a deadlock.

Of course, in this special type of tandem network a deadlock may occur. For example, let us suppose that station A is blocked by station B, because buffer in B station is full. A deadlock will occur if the task in service at station B must be sent to station A upon completion of its service. We assume that a deadlock is detected instantaneously and resolved immediately, with some negligibly delay time by exchanging both the blocked tasks simultaneously with the mean rate equal to μ^d.

This kind of service strategy with HoL priority service for feed backed tasks is described more detail in [7]. In mentioned paper, the procedure of constructing the steady-state equations in the Markov model is shown. Of cause, the stationary probability vector for this model can be obtained using numerical methods for linear systems of equations. Based on this stationary probability vector we can calculate the quality of service (QoS) parameters and the measures of effectiveness for this model.

2.2 Second service strategy: no two priority services in succession

This service strategy principle include:

1. Procedure for feedback tasks **"no two priority services in succession"** (preventing a possible congestion in the first buffer),
2. Mechanisms for checking the current buffer occupancy (resource allocation policy by blocking operations),
3. Procedures for detecting and resolving a possible deadlock.

In this strategy, we assume that deadlocks are detected and resolved instantaneously without any delay, simply by exchanging the blocked tasks.

Notation: the state space of this tandem network model can be described by random variables (i, j, k), where i indicates the number of tasks at the first station, j indicates the number of tasks at second server and k represents the state of each server (see Figure 4 and Figure 5). Here, the index k may have the following values: $0, 1, 2, 3, 4$. If $k = 0$ - idle network, $k = 1$ - regular task service, $k = 2$ - priority task service, $k = 3$ - blocking one station and regular task service at the other one, $k = 4$ - blocking one station and priority task service at the other one. Based on our analysis of the state space diagrams, the steady-state equations in this Markov model ware constructed. More detail this procedure is shown in [8]. In this paper the procedures for calculation the measures of effectiveness and quality of service (QoS) parameters are presented.

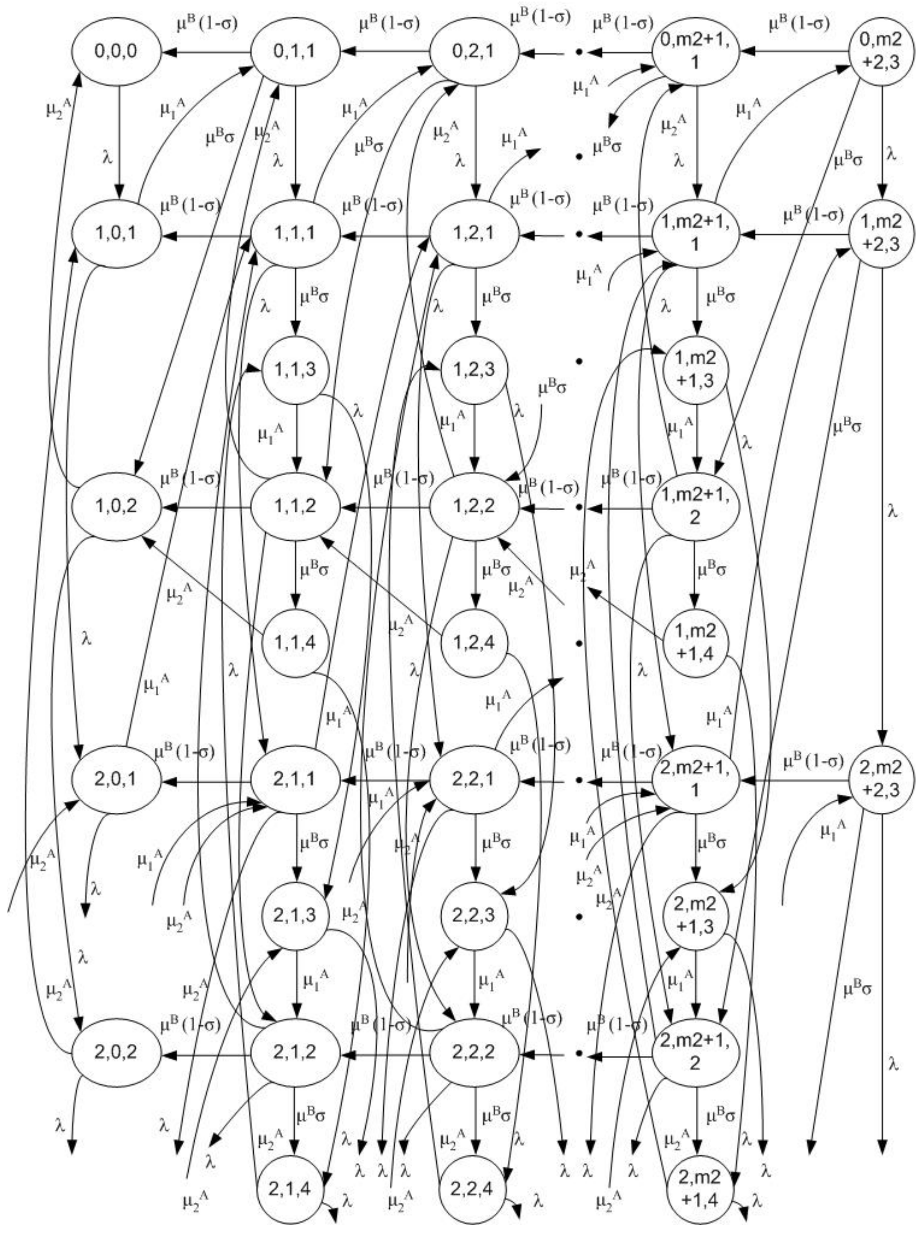


Fig. 4. Second strategy two-dimensional state diagram (first part)

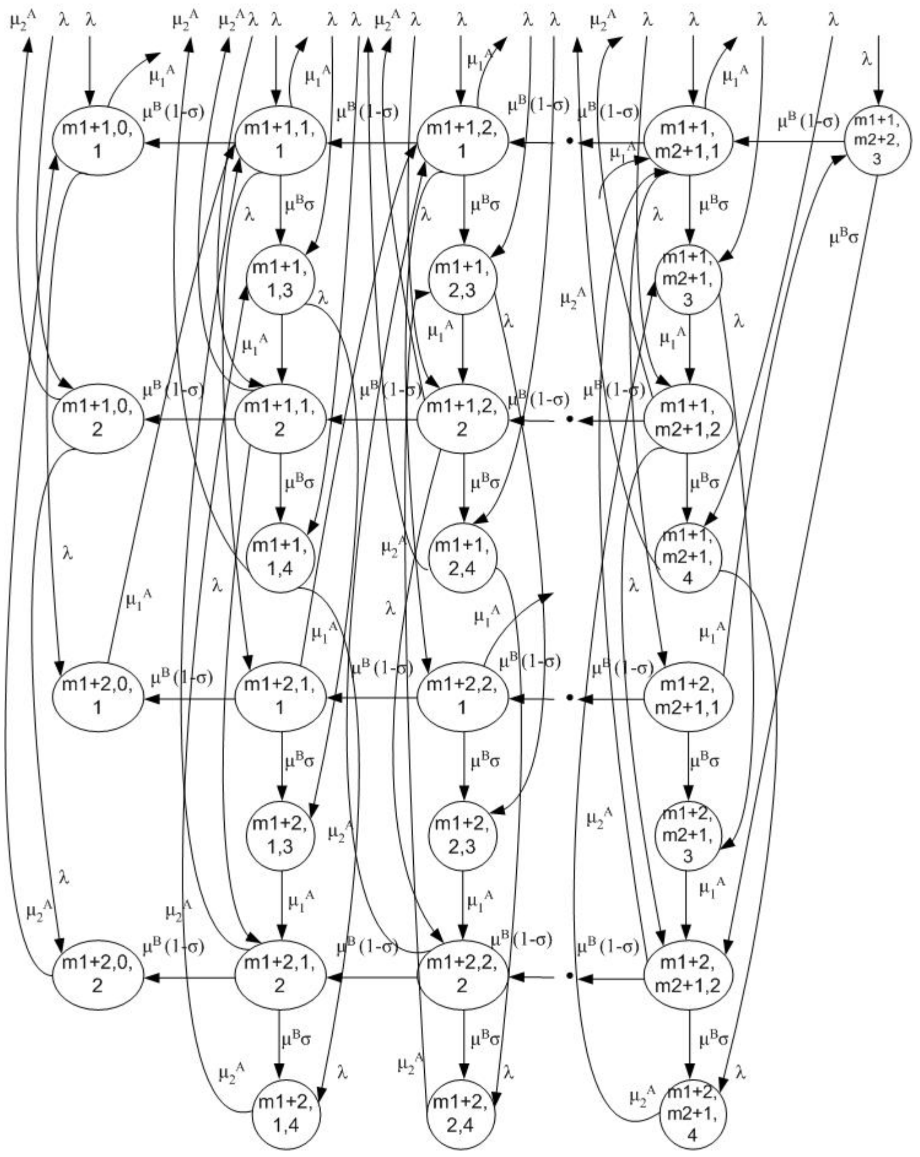


Fig. 5. Second strategy two-dimensional state diagram (second part)

3. Numerical examples

To demonstrate our analysis of two service strategies in a tandem server network with feedback and blocking presented in Section 2, we have performed numerous calculations. These calculations were realized for many parameters combinations by varying the arrival rate (λ) from source station and by varying the buffer capacities $m1$ and $m2$.

For the first group of calculations the following parameters were chosen: the service rates in station A and station B are equal to $\mu_1^A = 3.0$, $\mu_2^A = 2.5$, $\mu^B = 2.0$. The inter-arrival rate λ from the source station to station A is changed from 0.5 to 4.0 with step of 0.5 and the feedback probability is chosen as $\sigma = 0.7$. The buffers capacities are set to $m1 = 10$ and $m2 = 8$.

Based on such parameters the following results were obtained and the majority of them are presented in Table 1, where λ is the inter-arrival rate from the source station to station A, $p-bB$ is the station B blocking probability, $n-B$ is the average number of tasks in the second station, $w-B$ is the mean waiting time in buffer B, $t-bB$ is the mean blocking time in B station, $utl-B$ is buffer $m2$ utilization coefficient and $\lambda1$ is the effective input stream intensity from source station to first server (blocking factor).

Note: the columns with *italic* contains the results for second service strategy (no two priority services in succession), the standard columns - results for first Head of Line (HoL) service strategy.

Table 1. The measures of effectiveness and Quality of Service parameters.

λ	**p − bB**	*p − bB*	**n − B**	*n − B*	**w − B**	*w − B*	**t − bB**	*t − bB*	**utl − B**	**λ1**
0.5	0.036	*0.043*	0.393	*0.403*	0.052	*0.053*	0.013	*0.015*	0.286	0.500
1.0	0.143	*0.183*	1.679	*1.999*	0.487	*0.607*	0.052	*0.066*	0.642	1.000
1.5	0.272	*0.339*	6.028	*7.939*	2.349	*3.144*	0.100	*0.122*	0.958	1.428
2.0	0.287	*0.345*	7.051	*8.490*	2.806	*3.386*	0.105	*0.124*	0.989	1.619
2.5	0.287	*0.345*	7.088	*8.497*	2.822	*3.389*	0.105	*0.124*	0.990	1.746
3.0	0.287	*0.345*	7.091	*8.498*	2.823	*3.389*	0.105	*0.124*	0.990	1.842
3.5	0.287	*0.345*	7.091	*8.498*	2.823	*3.389*	0.105	*0.124*	0.990	1.919
4.0	0.287	*0.345*	7.091	*8.498*	2.823	*3.389*	0.105	*0.124*	0.990	1.981

For the second group of experiments the following parameters were chosen: the service rates in station A and station B are equal to $\mu_1^A = 4.0$, $\mu_2^A = 3.5$, $\mu^B = 4.0$. The inter-arrival rate λ from the source station to station A is 2.0. The feedback probability σ is 0.2. Buffer capacities $m1$ and $m2$ are changed within the range from 1 to 10. For this series of experiments, the following results were obtained and the selected set of them are presented in Table 2.

In this table m is buffers capacities in A and B stations, $v1-A$ and $v2-B$ are mean number of tasks in first and second buffers, $q1-A$ and $q2-B$ are mean response times at station A and B, $\rho - A$ and $\rho - B$ are utilization coefficients at both stations respectively.

Note: similarly as in the first experiment, the columns with *italic* contains the results for second service strategy (no two priority services in succession), the standard columns - results for first HoL service strategy.

Table 2. Selected measures of effectiveness.

m	*v1-A*	**v1-A**	*v2-B*	**v2-B**	*q1-A*	**q1-A**	*q2-B*	**q2-B**	ρ - *A*	ρ - *B*
1	*0.304*	0.304	*0.202*	0.199	*0.400*	0.400	*0.301*	0.300	*0.600*	*0.492*
2	*0.498*	0.449	*0.358*	0.350	*0.443*	0.443	*0.340*	0.338	*0.607*	*0.527*
3	*0.627*	0.628	*0.473*	0.460	*0.473*	0.473	*0.369*	0.365	*0.610*	*0.546*
4	*0.713*	0.715	*0.554*	0.537	*0.494*	0.494	*0.389*	0.385	*0.612*	*0.556*
5	*0.771*	0.775	*0.610*	0.590	*0.507*	0.509	*0.403*	0.398	*0.613*	*0.562*
6	*0.810*	0.815	*0.648*	0.626	*0.517*	0.518	*0.412*	0.407	*0.614*	*0.565*
7	*0.836*	0.843	*0.673*	0.649	*0.523*	0.525	*0.419*	0.413	*0.614*	*0.567*
8	*0.853*	0.862	*0.689*	0.664	*0.528*	0.530	*0.423*	0.417	*0.614*	*0.568*
9	*0.865*	0.874	*0.700*	0.674	*0.530*	0.533	*0.426*	0.419	*0.614*	*0.569*
10	*0.873*	0.883	*0.707*	0.680	*0.532*	0.535	*0.427*	0.421	*0.614*	*0.569*

The results of the experiments clearly show that the effect of the properly chosen service strategy in tandem network with feedback must be taken into account when analyzing performance such computer network. As noted above, feedback probability σ and blocking factor considerably change the performance measures in such networks.

4. Conclusions

An approach to compare the effectiveness of two service strategies in linked in series servers with blocking and feedback has been presented. Tasks blocking probabilities and some other fundamental performance characteristics of such network are derived, followed by numerical examples. The results confirm importance of a special treatment for the models with blocking and with HoL feedback service, which justifies this research. Moreover, our proposal is useful in designing buffer sizes or channel capacities for a given blocking probability requirement constraint. The results can be used for capacity planning and performance evaluation of real-time computer networks where blocking and feedback are present.

References

[1] S. Balsamo, V. De Nito Persone, R. Onvural, Analysis of Queueing Networks with Blocking, Kluwer Academic Publishers, Boston, 2001.

[2] M.C. Clo, MVA for product-form cyclic queueing networks with blocking, Annals of Operations Research, Vol. 79, pp. 83-96, 1998.

[3] A. Economou, D. Fakinos, Product form stationary distributions for queueing networks with blocking and rerouting, Queueing Systems, Vol. 30 (3/4), pp. 251-260, 1998.

[4] C.S. Kim, V. Klimenok, G. Tsarenkov, L. Breuer, A. Dudin, The BMAP/G/1-> ·/PH/1/M tandem queue with feedback and losses, Performance Evaluation, Vol. 64, pp. 802-818, 2007.

[5] J. B. Martin, Large Tandem Queueing Networks with Blocking, Queueing Systems, Vol. 41 (1/2), pp. 45-72, 2002.

[6] W. Oniszczuk, Modeling of dynamical flow control procedures in closed type queuing models of a computer network with blocking, Automatic Control and Computer Sciences, Vol. 39, Issue 4, pp. 60-69, 2005.

[7] W. Oniszczuk, Blocking and Deadlock Factors in Series Linked Servers with HOL Priority Feedback Service, Polish Journal of Environmental Studies, Vol. 16, No. 5B, pp. 145-151, 2007.

[8] W. Oniszczuk, An Intelligent Service Strategy in Linked Networks with Blocking and Feedback, Studies in Computational Intelligence N. 134 "New Challenges in Applied Intelligence Technologies", Springer-Verlag, Berlin, Heidelberg, pp. 351-361, 2008.

[9] W. Oniszczuk, Semi-Markov-based approach for analysis of open tandem networks with blocking and truncation, International Journal of Applied Mathematics and Computer Science, Vol. 19, No. 1, pp. 151-163, 2009.

[10] W. Oniszczuk, Analysis of linked in series servers with blocking, priority feedback service and threshold policy, International Journal of Computer Systems Science and Engineering, Vol. 5, No.1, pp.1-8, 2009.

[11] W. Oniszczuk, Loss Tandem Networks with Blocking Analysis - A Semi-Markov Approach, Bulletin of the Polish Academy of Sciences: Technical Sciences, Vol. 58, No. 4, pp. 673-681, 2010.

[12] R. Onvural, Survey of closed queuing networks with blocking, Computer Survey, Vol. 22 (2), pp. 83-121, 1990.

[13] H.G. Perros, Queuing Networks with Blocking. Exact and Approximate Solution, Oxford University Press, New York, 1994.

STRATEGIE OBSŁUGI W TANDEMACH SERWERÓW Z POWTÓRNĄ OBSŁUGĄ I BLOKADAMI

Streszczenie: W artykule poruszono zagadnienia związane z modelowaniem sieci serwerów z buforami o ograniczonej pojemności, powtórną priorytetową obsługą, które są ważnym elementem, w badaniu parametrów jakości obsługi w systemach komputerowych. Do badań i analizy wybrano dwie strategie powtórnej obsługi zadań w pierwszym z serwerów. Modele analityczne takich sieci stanowisk obsługi przedstawione są tutaj, jako otwarte markowowskie systemy kolejkowe z blokadami,. Tego typu modele w sposób najbardziej pełny odwzorowują ewolucję takich systemów w czasie. Zbudowano dwuwymiarowe grafy takich modeli tandemów oraz na przykładach pokazano jak zmieniają się ich miary wydajności i jakości obsługi, gdy zmienia się intensywność wejściowego strumienia i pojemność buforów.

Słowa kluczowe: powtórna obsługa, strategie obsługi, analityczne modele sieci

RAPID PROTOTYPING OF DEDICATED SYSTEMS BASED ON SHARED MEMORY ARCHITECTURE: METHOD AND EXAMPLE

Grzegorz Rubin[1], Mirosław Omieljanowicz[2], Alexander Petrovsky[2]

[1] The State College of Computer Science and Business Administration Lomza, Poland

[2] Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

Abstract: The aim of this paper is to present the method of rapid prototyping for reducing development cost of dedicated systems. In this paper the designing method for real-time embedded systems is proposed. At first the principle of specific universal balanced architecture is shown. Approach is based on modeling using modification of Petri nets called Hardware Petri Nets implemented in form of CAD software. Dedicated system is made using special computation architecture on FPGA. An example of designing processor for TVDFT is also given.

Keywords: rapid prototyping, shared memory architecture, FPGA

1. Introduction

The design and implementation of dedicated DSP (Digital Signal Processing) system is quite complex and time-consuming. Nowadays objective in times of high market pressure and ever decreasing time-to-market, automatization of the design and implementation are crucial. Finding an universal solution suited to wide range of DSP algorithms is permanently actual task. Rapid prototyping aims to reducing development cost[1]. A prototype is constructed prior to the system production version to gain information that guides analysis and design. This paper presents the method of designing real-time dedicated systems prepared for implementation in FPGA. In this approach a modification of Petri nets called Hardware Petri Nets (HPN)[2] is used. Implementation in FPGA chip using special computation architecture called dynamic reconfigurable computation architecture (DRCA) is also presented. Reconfiguration is made by altering the connections and control scheme of the universal computation module (UCM). The main advantage of proposed method is that there is no need to change hardware even if all used algorithms must be changed. There is a possibility to make all design processes to be almost fully automated from modeling to working prototype. Proposed method could be described in the form of the following algorithm:

1. Specification of tasks including: functional requirements, real-time requirements, bandwidth, maximum power consumption, maximum cost, etc.
2. Functional modeling and testing .
   - 2.1. The choice of algorithms which conform functional requirements from the point 1.
   - 2.2. Functional modeling using available computer programs (i.e. Matlab, Simulink).
   - 2.3. Functional verification of algorithms and if it doesn't comply with requirements: change algorithms (back to the point 2.1) or (as a last resort) change the specification (back to the point 1).
3. Synthesis of a processor structure.
   - 3.1. Design of a processor architecture based on DRCA concept.
   - 3.2. Mapping (selected at the point 2) algorithm to the chosen (at the point 3.1) architecture.
   - 3.3. Modeling and testing using application based on proposed HPN and if it doesn't comply with functional requirements: change the architecture (back to the point 3.1).
4. Synthesis of the hardware.
   - 4.1. Modeling calculation timing and generation of the control microcode using application based on proposed HPN.
   - 4.2. Synthesis of the processor and testing according to the all requirements specified at the point 1. The non-compliance forced to modify or change the architecture (back to the point 3.1).
5. End.

The next part of the article shows cosiderations which lead to choice of the architecture for universal computational module and presents an example of using the proposed method to design a processor for TVDFT transformation .

2. Principle of universal computation module and dynamic reconfigurable internal architecture

At the architectural level the main interest is to plan the overall organization of the compound system using processing elements (PE), memories, communication channels and control elements. Finding a universal solution suited to wide range of DSP algorithms is crucial task in terms of rapid prototyping . One of possible approaches is called shared-memory architecture (fig. 1 A).

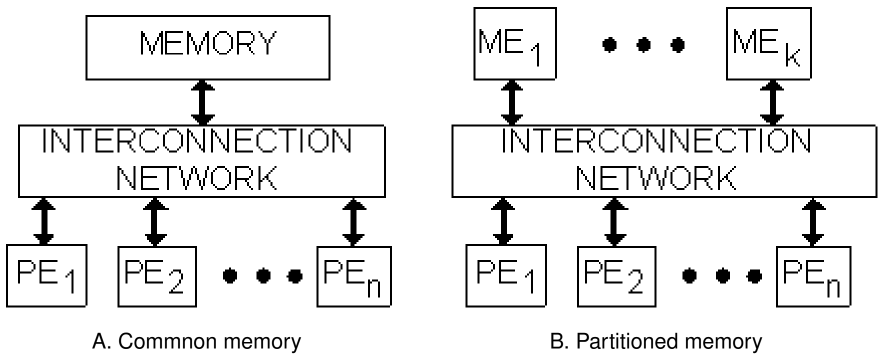


Fig. 1. Shared memory architecture

The idea is very simple. In order to provide simultaneously PE's with input data, the shared memory is partitioned into blocks. Using a rotating access scheme each processor gets access to the memories once per N (number of PE's) cycles. During this time processor either writes or reads data from memory. All PE's have the same duration time slot to access to the memories and access conflict is completely avoided. Shared-memory approach for DSP application is detailed in [3]. The well-known disadvantage of shared-memory architecture is memory bandwidth bottleneck. In order to avoid bandwidth bottleneck and simultaneously provide the processors with several (K) input data, the shared-memory is partitioned into K memories (fig.1 B).

In this paper a special instance of that architecture called universal computation module (UCM) is presented. The main target is to find balance between complexity of interconnection network, type of computation model of PE's (serial vs. parallel), number of PE's and memory size. Chosen compromise should fulfill following factors: required performance, minimal power consumption and cost in terms of chip area. Another important requirement is to create flexible, easy reconfigurable architecture suited to wide range of DSP algorithms.

Processing elements (PE's) usually perform simple memory less mapping of the input values to a single output values. The PE's can be in parallel or serial form. Memory elements (ME) comparing to PE's are slow. It is obvious that there is a need for additional register to fulfill performance requirements. By bit parallel PE high speed register will play a trivial (one word) cache memory role. By bit serial PE there must be a shift register. The RAM addressing requires only cyclic work. Number of RAM words should be enough to store all variables accordingly to realized algorithm. Interconnection network (ICN)should provide the communication channel needed to supply PE's with proper data and to store results in the proper memories. The data movement should be kept simple, regular and uniform. Major design issues involve the topology of the communication network and its bandwidth.

There is a variant of shared-memory architecture that seems be well-matched to wide range of DSP algorithms and can be very well-balanced in term of processing capacity and communication bandwidth. This modification splits interconnections

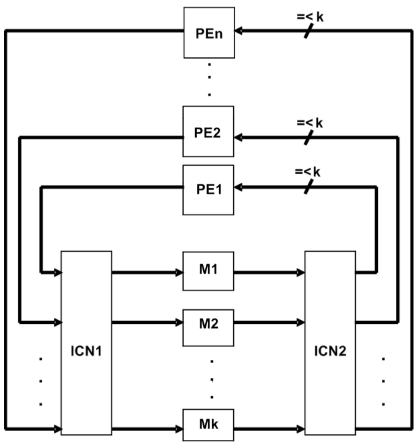


Fig. 2. Shared memory architecture with split intercommunication network [3]

into two parts (fig.2). In this architecture to obtain balance [3] there should be:

$$T_{PE} >= 2 * N * \frac{T_M}{W_M} \tag{1}$$

where: $T_(PE)$ - operation time of PEs, N - number of PEs, T_M - memory access time, W_M - word width in memory. Then number of PEs should be:

$$N <= (T_{PE}/T_M) * W_M/2 \tag{2}$$

Typically $T_{PE} = \frac{T_M}{4}$ and data word length is 16 to 24 bits. Thus, balanced shared-memory module will have 2 or 3 PE's. After that a trade-off between parallel and serial form of PE's and ICN was made.

It is obviously that bit-parallel version of PE's have several times higher performance than bit-serial one at the same clock. However, taking into account whole module with PE's, input and output registers, memory and interconnection network, advantage of parallel form is not so clear. Including power consumption and chip area, serial form could be more convenient. Generally smaller chip area and smaller clock lead to smaller power consumption. The requirements of the PE is that it completes its operation within the specified time limit. Self explanatory chip area of single serial PE is much smaller then parallel PE, but to get the same performance needs faster clock. Parallel PE's leads to more connections lines, consistently more chip area and power. Parallel PE looks to have several times bigger computational throughput (than serial by the same clock), however when consider more than one PE in shared memory architecture it could be impossible to use high speed clock because of noise in signal propagation on parallel buses. Otherwise control part of whole system in serial PE version may be in micro program fashion, where implemented algorithm will be changed by the way of changing control memory contest. Using parallel PE control part must be significantly changed due to change type of computational task.

In universal architecture on fig. 2 it is convenient to use serial bit PE's. Only single wire is required for each signal, hence interconnections are simple and consume small chip area. The memory has been split into *K* logical memories that *K* values can be accessed simultaneously. Hence every PE can have *K* inputs and memory conflicts are avoided. However to have nice working architecture there is a need to change the data format from serial to parallel and parallel to serial without influence of interconnections. It is easy to accomplish using shift register as a part of memory block as is shown on fig. 3.

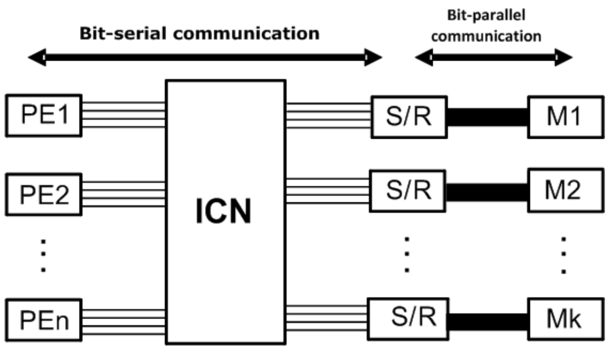


Fig. 3. Shared memory with bit-serial and bit- parallel communication [3]

Above brief considerations shows that shared-memory architecture with 2-3 serial PE's can be easier to implement and well suited to wide range of DSP algorithms (dynamic reconfiguration made by microcode change). Processing elements used in that type of shared memory architecture should have three functions: bit-serial full addition (inc. carry), bit-serial multiplication, negation and two serial inputs and one output. Other arithmetic operations will be done as a sequence of additions. Block diagram of proposed UCM with 3 PE's is shown on fig.4. Because of three PE's, UCM is equipped with six shift registers and two independent memories. Those registers are used as single word cache memory and as serial to parallel and parallel to serial translators on communication path to RAM memory. Hence interconnection network is very simple. This leads to small chip area and possibility of using high speed clock. To achieve flexibility in organization of computation process, connection network should enable access to any RAM for every PE. Thus more than 3 PE's give more connections lines and more chip area (only for wires) so the balance between chip area for logic and connections will be loosed.

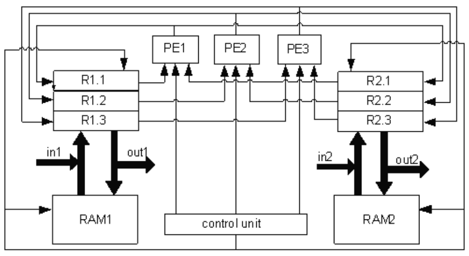


Fig. 4. Proposed universal computation module - UCM

Proposed kind of shared-memory architecture in form of UCM offers good balance in terms of chip area, power consumption, computational throughput and flexibility but offers relatively small performance to run advanced DSP algorithms. To fulfill requirements of complex DSP task more computation power is needed. It can be achieved by connecting some number of UCM's. Changing number of UCM's and way in which they are connected and controlled we get dynamic reconfigurable computation architecture - DRCA. In such a manner almost every required performance could be achieved.

To be able to use proposed UCM in rapid protoyping process it is necessary to have specific modeling method (taking into account parallel processing). That can be done using Petri Nets theory. There are few known examples of that approach [2, 5, 6]. For proposed approach a special modification of Petri Nets called Hardware Petri Nets (HPN)was developed Proposed HPN allows for deadlock prevention [7] in parallel processes. Every transition corresponds to enable signal in physical device. Moreover there is possibility to design hierarchical structure of the net. Every transition and place can be designed as the subnet. Simulation of hierarchical structure is also possible.

Developed HPN concept was implemented as CAD-type application. This allows automatation of the designing process (important feature in rapid prototyping method). Produced software environment has a graphical model creation tool and allows simulation of algorithms step by step. A sample graph corresponding to UCM is presented on fig.5. Formal analysis, hierarchical designing and simulation are possible too. Each part of designed algorithms can be independently simulated. That is useful for error correction. As the result of software simulation we have control

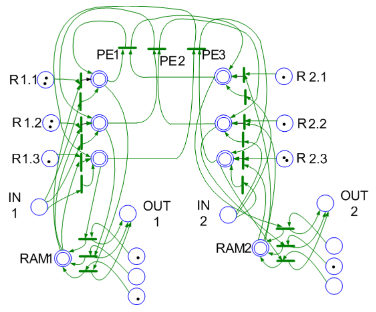


Fig. 5. Hardware Petri Net model for universal computation block

vectors for FPGA device. These vectors can be written as a text file, then loaded to Xilinx FPGA device simulator. Such approach allows for better scheduling, because of running every elements of architecture as fast as it is possible, when proper data are ready for processing. These operations ending protopyting process.

In the next part of this article the example of realization of TVDFT transformation based on proposed approach is presented.

3. Rapid prototyping of TVDFT

A wide variety of signal processing functions can be hosted on the UCM, including complete subsystems that encompass multiple algorithms. The TVDFT transformation [8] will be used as the example. TVDFT is given by equation:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\varphi(n,k)}w(n), k = 0..K \tag{3}$$

where: $X(k)$ - spectral component corresponding to k- th harmonic, N - length of analysis frame, $x(n)$ - input signal, $w(n)$ - time window, K - number of orders in input signal. Function $\varphi(n,k)$ is given as

$$\varphi(0,k) = 0; \quad \varphi(n,k) = \sum_{i=0}^{n} \frac{2\pi k(f_0(i) - f_0(i-1))}{2F_s} \tag{4}$$

where $f_0(i)$ is fundamental frequency at time specified by i, F_s is sampling frequency. In case of linear change of fundamental frequency formula (4) can be written as follows:

$$\varphi(n,k) = \frac{2\pi nk}{F_s}(f_0 + \frac{2\Delta f}{2N}) \tag{5}$$

where: f_0 -fundamental frequency at the beginning of analysis frame, Δf is fundamental frequency change within analysis frame. Hence, TVDFT formula (3) can be written as follows:

$$Re\,X(k) = \sum_{0}^{N-1} x_w(n)\cos(\frac{2\pi nk}{F_s}(f_0 + \frac{2\Delta f}{2N})) \tag{6}$$

$$Im\,X(k) = \sum_{0}^{N-1} x_w(n)\sin(\frac{2\pi nk}{F_s}(f_0 + \frac{2\Delta f}{2N})) \tag{7}$$

where: $x_w(n) = x(n)w(n)$.

Formulas (6),(7) show, that for practical realization of TVDFT, two sine wave generators with linear change of frequency [4] and one multiplication block must be used. Block diagram of computations is show on fig.6. According to this at the first step we must calculate value of sine and cosine function (generators blocks) then at the second step we must multiply that value by input signal (transform block). The sine and cosine function algorithm [4, 9] was based on formula $(-1)^k(\bar{A_n} - kM)[M - (\bar{A_n} - kM)]$, where n - number of current sample, M - amount of samples per half period, $\bar{A_n}$ - value of n'th sample, k - integral part from division n/M. This equation leads to computation block diagram presented on fig.7.

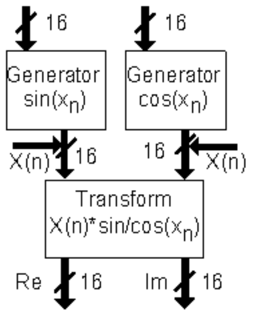


Fig. 6. Block computation diagram for TVDFT

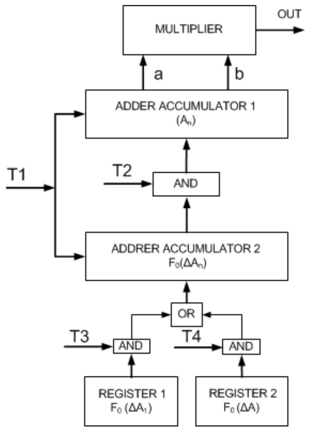


Fig. 7. Generator with linear time-varying frequency [9]

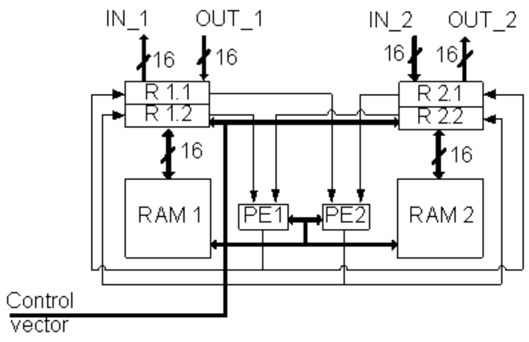


Fig. 8. UCM for sine/cosine generators

The balanced computation module for such algorithm needs two serial PE's, so there are only four shift registers for communication between RAM's and PE's. Thus it gives UCM in form shown on fig.8. Size of memories is 3 of 16- bit words (RAM1[0..2] and RAM2[0..2]). According to block diagram of the TVDFT to accomplish whole task we need 3 computation modules. For improving performance it is possible to use parallel or cascade connection of computation units. On fig.9 two UCM's calculate sine and cosine values in parallel form and then the results get into the inputs of additional UCM unit, where, input signal $x(n)$ is multiplied by sine value. The result (according formula (6),(7)) accumulates in memory .

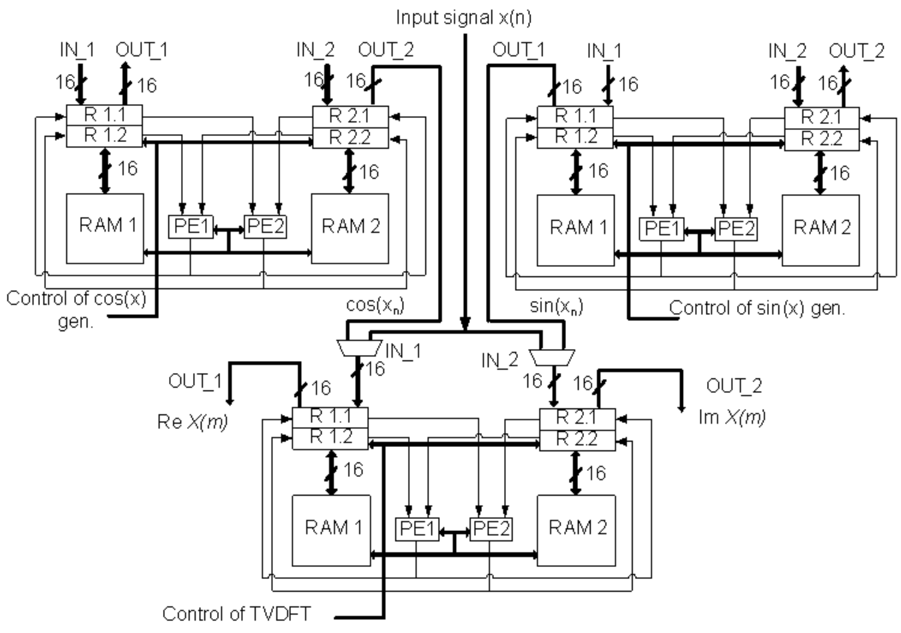


Fig. 9. Three UCM's for TVDFT realization

The next prototyping phase is to make computation model and to create all necessary control signals. First stage was for generator block and second one for transform block. Set of operations for generator block mapped on HPN model is given bellow (operations from 1 to 8 repeats for every N - value of sine/cosine function):
1) Load data into input registers.
R1.1<- IN1, R2.1<- RAM2[0]

R1.2<- RAM1[2], R2.2<- RAM2[1]
2) Calculations.
PE1<- R1.1+R1.2, PE2<- R1.2+not(R2.2)
3) Writing the results into output registers.
R2.1<- PE1, R1.1<- PE1, R2.2<- PE2
4) Writing the result from output registers to memories.
RAM2[2]<- R2.2, RAM1[0]<- R1.1, RAM2[1]<- R2.2,
5) Load data into input registers.
R1.1<- RAM1[0], R2.1<- RAM2[1]
R1.2<- RAM1[2], R2.2<- RAM2[2]
6) Calculations.
PE1<- R1.1+R1.2, PE2<- R1.2(R2.2)*
7) Writing the results into output registers.
R1.1<- PE1, R2.1<- PE1, R2.2<- PE2
8) Writing the result from output register to output.
Out1<- R2.2
where: *IN1* - at first step starting frequency value and then relative value of frequency changing per step (in case of the generator with time-varying frequency[4]), *Out* - value of sine/cosine function.

According to presented set of operation and prototyping method we built HPN model (fig.10) and made appropriate simulation. It allows for mapping and scheduling (see results on fig.11) for first stage - given sinus generators algorithm. It the same way the second stage was made - modeling of transform block (fig.6) using HPN.

As the end of prototyping process an FPGA implementation of TVDFT computational unit was made on XILINX VIRTEX-II family. Obtained result was compared to prototype created in traditional way, i.e using HDL language and libraries in Xilinx ISE Design Suite. Summary of results are shown in Table 1. Using proposed method shorter time to make prototype and better hardware in terms of cost, power, bandwith were achieved as well.

4. Summary

In this paper the method of rapid prototyping to reduce development cost of dedicated real-time systems was presented . Proposed method is based on developed DRCA architecture (with UCM as a main building block) and Hardware Petri Nets as automated modeling tool. The balance between processing elements (PE's), count of memories and interconnection network was achieved. Processing element based on bit- serial arithmetic (multiplication and addition) was also given. As an example

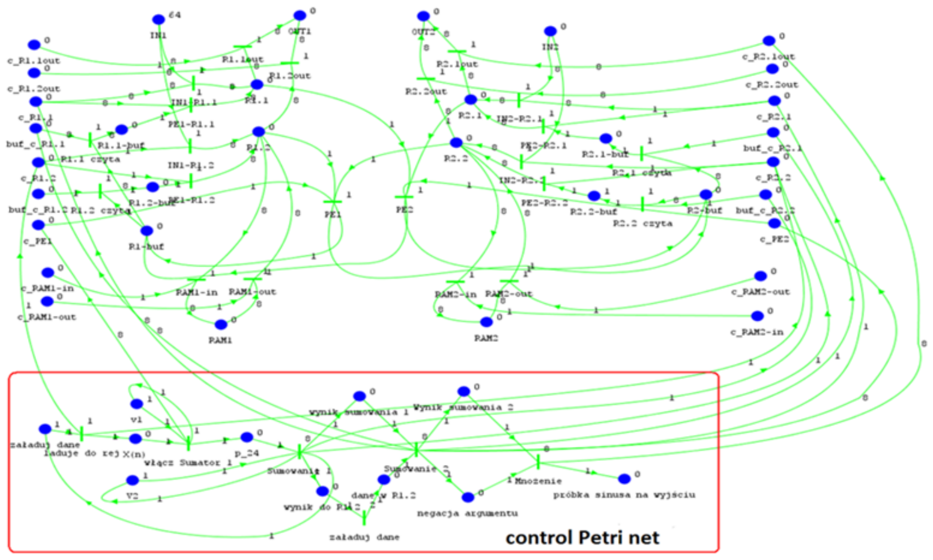
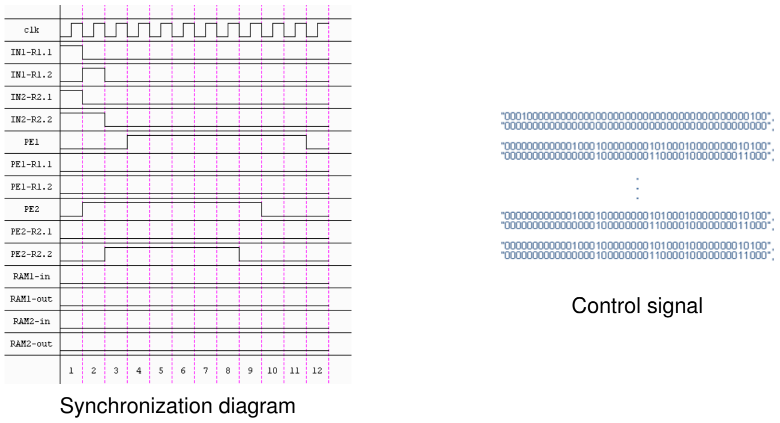


Fig. 10. Mapped algorithm of sin/cos generator



```
"0001000000000000000000000000000000000000000100",
"0000000000000000000000000000000000000000000000";

"0000000000001000100000000010100010000000010100",
"0000000000000000100000000011000010000000011000";

                       .
                       .
                       .

"0000000000001000100000000010100010000000010100",
"0000000000000000100000000011000010000000011000";

"0000000000001000100000000010100010000000010100",
"0000000000000000100000000011000010000000011000";
```

Fig. 11. Result of simulation using application based on HPN

Table 1. Summary of implementation TVDFT made on the DRCA and generally available solutions

	DRCA build on UCM		Standard HDL project	
Number of Slices:	34 out of 14336	0%	279 out of 14336	5%
Number of Slice Flip Flops:	33 out of 28672	0%	392 out of 28672	1%
Number of 4 input LUTs:	68 out of 28672	0%	1270 out of 28672	4%
Number of bonded IOBs:	197 out of 720	27%	197 out of 720	27%
Total estimated power consumption	410 mW		622mW	
Min. input arrival time before clock:	2.383 ns		2.383 ns	
Minimum period:	2.690 ns		20,302ns	
Max. Freq.:	372MHz		256MHz	
Max. output required time after clock:	3.847 ns		8.821 ns	
Maximum combinational path delay:	3.847 ns		3.847 ns	

of DRCA appliance the TVDFT algorithm was implemented. Presented in this paper rapid prototyping method is an solution suited to wide range of DSP algorithms.

References

[1] S. Zoran, Prototyping embedded DSP systems - from specification to implementation, In the Proc. EUSIPCO 2004, Vienna 2004, pp. 1625-1632

[2] A. A. Petrovsky, Mietody i mikroprocesnyje sredstwa obratotki sziroko polosnych i bistroprotiekajuszczych procesow w realnow wremieni (in russian) Nauka i Tiechnika, Minsk, 1988

[3] L. Wanhammar , DSP integrated circuits, Academic Press, USA, 1999.

[4] G.Rubin, M. Omieljanowicz, A. Petrovsky, Reconfigurable FPGA- based hardware accelerator for embedded DSP, MIXDES 2007, Ciechocinek, 2007, pp.147-151

[5] G. Rubin, A. Petrovsky, M. Omieljanowicz, Multilevel hardware Petri nets for rapid prototyping design platform. The 12th Intern. Conference Mixed design on integrated circuits and systems, MIXDES 2005, Kraków, Poland, 20-25 June 2005, pp.147-152.

[6] M. Adamski, M. Węgrzyn, Hierarchically Structured Colored Petri Net Specification and Validation of Concurent Controllers, Proc. In 39th Interenational Scientific Colloquium, IWK'94, Ilmenau, Germany, 1994, Band 1, pp. 517-522.

[7] G. Rubin, A. Petrovsky, M. Omieljanowicz, Rapid prototyping of Real time DSP-systems based on accurate simulation computation processes using Petri nets, WSFiZ Press, vol. I, Bialystok, 2005, pp. 409-417.

[8] A. Petrovsky, P. Zubrycki, A. Sawicki, Tonal and noise separation based on a pitch synchronous DFT analyzer as a speech coding method, The proc. of the ECCTD 03, vol.III, Cracow 2003, pp. 169-172.

[9] M. Omielianowicz, P. Zubrycki, A. Petrovsky, G. Rubin FPGA-based algorithms and hardware for generating digital sine wavesd, Mixed design of integrated circuits and systems : MIXDES'2002 : 9th International Conference, Wrocław June 20-22, 2002, pp. 279-284.

SZYBKIE PROTOTYPOWANIE DEDYKOWANYCH SYSTEMÓW W OPARCIU O ARCHITEKTURĘ WSPÓŁDZIELONEJ PAMIĘCI: METODA I PRZYKŁAD

Streszczenie: Celem tego artykułu jest zaprezentowanie metody szybkiego prototypowania redukującego koszty opracowania dedykowanych systemów obliczeniowych. Zaprezentowane rozwiązanie jest nakierowane na projektowanie systemów wbudowanych czasu rzeczywistego. W pierwszej części publikacji opisano zasady wyboru uniwersalnego modułu obliczeniowego i zaprezentowano uzyskaną architekturę przygotowaną do implementacji sprzętowej w układach FPGA. Określono też metodę modelowania za pomocą dedykowanych sieci Petri, nazwanych sprzętowymi sieciami Petri, wykorzystywanych w postaci oprogramowania typu CAD. To oprogramowanie pozwala na szybkie utworzenie modelu bloku obliczeniowego, następnie na przeprowadzenie automatycznej weryfikacji jego poprawności i ostatecznie wygenerowania wektorów sterujących pracą. Wykorzystując opracowane: uniwersalną architekturę modułu obliczeniowego i narzędzie typu CAD stworzono metodę pozwalającą na szybkie uzyskanie sprzętowego prototypu (na bazie podzespołów FPGA) układów obliczeniowych z zakresu cyfrowego przetwarzania sygnałów w czasie rzeczywistym. W drugiej części artykułu przedstawiono przykład wykorzystania zaproponowanej metody do zaprojektowania układu obliczeniowego realizującego przekształcenie TVDFT.

Słowa kluczowe: szybkie prototypowanie, architektura ze współdzieloną pamięcią, FPGA

Artykuł zrealizowano w ramach pracy badawczej S/WI/4/08.

UNSUPERVISED CLASSIFICATION AND PARTICLE SWARM OPTIMIZATION

Adam Truszkowski, Magdalena Topczewska

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: This article considers three algorithms of unsupervised classification - *K-means*, *Gbest* and the *Hybrid* method, the last two have been proposed in [14]. All three algorithms belong to the class of non-hierarchical methods. At first, the initial split of objects into known in advance number of classes is performed. If it is necessary, some objects are then moved into other clusters to achieve better split - between cluster variation should be much larger than within cluster variation. The first algorithm described in this paper (*K-means*) is well-known classical method. The second one (*Gbest*) is based on the particle swarm intelligence idea. While the third is a hybrid of two mentioned algorithms. Several indices assessing the quality of obtained clusters are calculated.

Keywords: unsupervised classification, clustering, particle swarm optimization

1. Introduction

The aim of unsupervised classification or cluster analysis is to search the data for a structure of natural groupings to understand the complex nature of multivariate relationships. This group of methods are exploratory techniques and can be widely used in many fields, for example in medicine to search for sets of symptoms or treatments, in marketing to segment of target groups, etc. Thus it can provide an informal means for suggesting hypotheses concerning relationships. It can be also helpful to assess dimensionality of data and to identify outliers [5].
The process of grouping is performed on the basis of similarities or dissimilarities (distances) of objects and this is the only assumption of this group of methods. Of course, to obtain good results some kind of basic exploration should be made to decide how to measure the association between the objects. Due to the fact that we usually have to deal with large data sets, we can rarely examine all group combinations. So we can use some of the wide variety of algorithms which emerged to find

reasonable clusters in data.
Each algorithm solving clustering problem belongs to one of two types – hierarchical or non-hierarchical. Hierarchical methods create a tree structure called dendrogram by splitting or merging recursively existing groups of objects. In the non-hierarchical methods objects are divided into known in advance number of clusters, and this division is then corrected by moving some objects between clusters to obtain better split. Among others *K-means* is the most popular of this kind. The applications of this algorithm cover many areas and range from market segmentation [7], portfolio analysis [12], to image retrieval [8] or academic performance [11]. The algorithms inspired by nature can be alternative proposal. They have also many applications, for instance image classification [10], robotic mapping [3], document clustering [1], etc. Comparing with other artificial intelligence algorithms, like genetic or evolutionary algorithms, *swarm intelligence* (*SI*) is relatively new group of methods, but regard to its increasing popularity it seems promising. *Particle swarm optimization* (*PSO*) belongs to the *SI* group of methods. It is inspired by social behavior of bird flocking and fish schooling [6]. Each element (bird, fish) called a *particle* respects few primary rules – moves toward the best position of the swarm leader and shares information with its neighbours.
This article considers thee non-herarchical algorithms of clustering: *K-means*, *Gbest* (based on particle swarm intelligence idea) and the *Hybrid* method. Several indices of obtained quality of splits, like adjusted Rand index, mistakes matrix, inter cluster, intra cluster and validity quantities, are calculated.

2. Methods

The first algorithm described in this paper (*K-means*) is a well-known classical method. The second one – *Gbest* – is based on the particle swarm intelligence idea. While the third one is a hybrid of two mentioned algorithms. In general, at first, the initial split of objects into known in advance number of classes is performed. If it is necessary, some objects are then moved into other clusters to achieve better split and improve results - between cluster variation should be much more larger than within cluster variation.
The crucial issue is how to measure the distance between objects. In the pattern recognition systems, for example during examination of profiles or patterns, it is much better to use correlation measures than distance ones to avoid connection of objects with different profiles. To calculate the similarity between elements the most commonly used is Euclidean distance. We can also apply Manhattan, Jaccard or Gower distances depending on the considered problem.

In this paper only the Euclidean distance is applied due to the fact that it is a measure calculated as a distance between two points joined with a straight line. If there is no additional information about the data and the objects are points in $\mathbb{R}^{N_d}$, where $\mathbb{R}^{N_d}$ is a real N_d-dimensional vector space, this distance is a natural way to calculate the similarity between objects.

Given a set of N_o objects $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{N_o}\}$, each object is described by N_d values of attributes. The notation used in the article to all three methods is as follows:
N_o – number of vectors (objects in the data set),
N_d – number of dimensions (attributes describing each object),
N_c – number of clusters,
n_j – number of objects in j-th cluster,
$\mathbf{m}_j$ – the vector describing j-th cluster centroid,
$\mathbf{C}_j$ – the subset of objects belonging to the j-th cluster.

2.1 *K-Means* algorithm

The *K-means* algorithm was described in [9] and assigns each object to the cluster having the nearest centroid. Instead of K in this article the N_C denotes the number of clusters. The steps of this method are presented in the listing 1. Three steps are performed in this algorithm. In the first step the initial partition of objects into predefined N_C clusters is made. In the second one, the objects are allocated into clusters - the object is located to the cluster according to the rule of the nearest centroid. The distance between $\mathbf{z}_p$ object and $\mathbf{m}_j$ centroid is calculated as Euclidean distance

$$d(\mathbf{z}_p, \mathbf{m}_j) = \sqrt{\sum_{k=1}^{N_d} (z_{pk} - m_{jk})^2} \tag{1}$$

Then the mean values of clusters (centroids) are recalculated

$$\mathbf{m}_j = \frac{1}{n_j} \sum_{\forall \mathbf{z}_p \in \mathbf{C}_j} \mathbf{z}_p \tag{2}$$

and updated. The step two is repeated, because it may turned out that after recalculation of centroids some objects should be reallocated. If there are no more reassignments, the algorithm is stopped. The stop criterion can also consider the number of iterations or timeout has been exceeded, etc.

Rather than starting with partition of data into initial clusters, the N_C initial centroids can be specified and then all objects are divided into groups.

Algorithm 1 K-Means

```
1: Partition the vectors into $N_C$ initial clusters
2: repeat
3:     For each data vector, assign it to the cluster whose centroid is nearest (usually computed distance
       is Euclidean distance with either standardized or unstandardized vectors (1)
4:     Recalculate the cluster centroids using (2)
5: until no more reassignments take place
```

The final result depends largely on the initial means. Poor selection of centroids might cause poor quality of obtained clusters.

2.2 *Gbest* algorithm

In this and the next subsection two methods based on particle swarm optimization (PSO) are introduced. Particle swarm optimization is inspired by social behavior of bird flocking and fish schooling [6]. Each element is called a *particle* and represents a potential solution of optimization problem. Therefore, the whole swarm is a set of potential solutions. During the optimization process each particle changes its position, remembering its best position and best positions of its neighbors. It moves toward the best position of the swarm leader and shares information with its neighbors.

The algorithm based on PSO idea is the *Gbest*, in which i-th particle $\mathbf{x}_i = (\mathbf{m}_{i1}, \mathbf{m}_{i2}, \dots, \mathbf{m}_{iN_c})$ is initialized as a vector of randomly proposed centroids for N_c clusters. Therefore the swarm represents candidates for clusters obtained from data. The number of particles in a swarm is chosen arbitrarily by the user. Every particle stores three types of information: the current position ($\mathbf{x}_i$), the current velocity ($\mathbf{v}_i$) and its best position ($\mathbf{P}_i$) In the next step, for each particle and each object the distance between the object and centroids is calculated and then the element is allocated into the cluster according to the rule of the nearest centroid. Besides, the value of the fitness function is evaluated. The fitness function calculated as the quantization error has the form

$$J_e = \frac{\sum_{j=1}^{N_c} \left(\sum_{\forall \mathbf{z}_p \in C_{ij}} d(\mathbf{z}_p, \mathbf{m}_j) / |C_{ij}| \right)}{N_C} , \tag{3}$$

where $|C_{ij}|$ denotes the number of objects belonging to ij-th cluster.
Then the best local ($\mathbf{P}_i$) and global particle ($\mathbf{G}_i$) is found and actualized and finally, the centroids in every particle are recalculated using equation for actualization of the velocity of the particle

$$v_i(t+1) = \omega v_i(t) + c_1 \phi_1(t)(\mathbf{P}_i(t) - x_i(t)) + c_2 \phi_2(t)(\mathbf{G}(t) - x_i(t)) , \tag{4}$$

and new position of the particle

$$x_i(t+1) = x_i(t) + v_i(t+1) \ , \tag{5}$$

where ω – inertia weight, c_1 and c_2 are acceleration constances, $\phi_1, \phi_2 \sim U(0,1)$, P_i is the best position of the i-th particle, G is the best global position.
The standard *Gbest* algorithm is presented in the listing 2.

Algorithm 2 Gbest

```
 1: Initialize each particle to contain $N_C$ randomly selected cluster centroids
 2: for $t = 1$ to $t_{max}$ do
 3:     for each particle $i$ do
 4:         for each data vector $z_p$ do
 5:             calculate the Euclidean distance (1) $d(\mathbf{z}_p, \mathbf{m}_{ij})$ to all $C_{ij}$ cluster centroids
 6:             assign $\mathbf{z}_p$ to $C_{ij}$ cluster such that $d(\mathbf{z}_p, \mathbf{m}_{ij}) = \min_{\forall c=1,\ldots,N_C}(\mathbf{z}_p, \mathbf{m}_{ij})$
 7:             calculate the fitness using (3)
 8:         end for
 9:         Update the best global and best local positions
10:         Update the cluster centroids in a particle using (4) and (5)
11:     end for
12: end for
    where:
    $t_{max}$ – the maximum number of iterations.
```

2.3 *Hybrid* algorithm

The *Hybrid* method presented in this subsection is similar to the *Gbest* algorithm. The only difference is a placement of centroids, found by *K-means* method, into a set of particles. The aim is to facilitate the task of searching the solution and the swarm leader with the best position. The hybrid algorithm has four steps:

1. Determine the N_c number of clusters and input the data set.
2. Execute *K-means* method and find the centroids of clusters.
3. Put the result of the previous step as initial values of one particle and for the rest particles of the swarm use values initialized randomly.
4. Execute *Gbest* algorithm with initialized swarm.

2.4 Indication of achieved clusters

To measure the correspondence between partitions of the objects several indices might be applied. In this article we concentrate on the adjusted Rand index, mistakes matrix, inter cluster, intra cluster and validity quantities.
The *Adjusted Rand Index* (ARI) [4] is the modification of the *Rand Index* [13]. The aim of ARI is to prove the disagreement or agreement between two divided groups of objects and classes which they belong to. The maximum value of ARI and RI equals 1 and it means that there is complete agreement between obtained clusters and original classes. Since the RI lies between 0 and 1, its expected value must be greater than or equal 0. The expected value of ARI is 0 and this index has wider range of values. Low values indicate poor agreement between clusters and classes.

3. Results

In this article seven data sets were chosen to experiments. First five sets are artificial problems and can be downloaded from [15].

- *Cl-3* – data set consisted of three classes of objects described by 2 attributes. In each class there are 200 objects. Elements of every class can be described by normal distribution with mean vectors $\mu_1 = (0,0)$, $\mu_2 = (2.5,1)$, $\mu_3 = (-0.5,3)$ and the same covariance matrices $\Sigma_1 = \Sigma_2 = \Sigma_3$.
- *Mag* – data set consisted of two classes of normally distributed objects from two dimensional space with mean vectors: $\mu_1 = (0.1342, 0.0229)$, $\mu_2 = (3.1186, 0.0978)$ and covariance matrices:

$$\Sigma_1 = \begin{bmatrix} 2.2617 & 2.1477 \\ 2.1477 & 2.4903 \end{bmatrix} \qquad \Sigma_2 = \begin{bmatrix} 2.3649 & 2.0096 \\ 2.0096 & 2.0425 \end{bmatrix}$$

  100 elements for every class were randomly generated from normal distribution with mentioned parameters.
- *Mag2-out* – *Mag* data increased by one outlier $(-1,4)$.
- *Squares4* – data set consisted of four classes with 100 generated objects in each class. The classification rule for this problem was as follows:

$$class = \begin{cases} 1 & \text{if } (z_1 >= 4 \text{ and } z_1 <= 6 \text{ and } z_2 >= 4 \text{ and } z_2 <= 6) \\ 2 & \text{if } (z_1 >= 7 \text{ and } z_1 <= 9 \text{ and } z_2 >= 4 \text{ and } z_2 <= 6) \\ 3 & \text{if } (z_1 >= 4 \text{ and } z_1 <= 6 \text{ and } z_2 >= 8 \text{ and } z_2 <= 10) \\ 4 & \text{if } (z_1 >= 7 \text{ and } z_1 <= 9 \text{ and } z_2 >= 8 \text{ and } z_2 <= 10) \end{cases}$$

- *Squares2* – objects from *Squares4* data set where original classes 1 and 4 were connected to one class with label 1, remaining two classes created new class with label 2. Thus the obtained location of objects in classes is like in XOR truth table problem.
- *Iris* – The well-known and well-understood problem of three species of irises, where each of 150 flowers is described by four attributes and belongs to one of three classes.
- *Wine* – The well known classification problem with three classes, 178 objects and 13 attributes.

The results presented in the tables below are the mean values for twenty starts of the algorithms.

3.1 Experiment 1

The first experiment concerned selection of particle swarm optimization algorithm parameters to achieve the best numerical values and to use them in second experiment.
At first, the relationship between ϕ_1 and ϕ_2 in (4) was examined. If $\phi_1 < \phi_2$ occurs it means that the centroids of the particle consider more their best local position. In the case when $\phi_1 > \phi_2$ occurs, the centroids consider more the best global position. If there is equality between ϕ_1 and ϕ_2 the centroids are influenced by the best local and global positions to the same extent.
As the example we present selected results for *Cl-3* data set with three separate clusters in the Table 1. The best results were obtained for $\phi_1 = 0.8$ and $\phi_2 = 0.2$. The similar results, when the relationship $\phi_1 > \phi_2$ was observed for all presented data sets.

Table 1. Fitness function results for parameters ϕ_1 and ϕ_2 for *Cl-3* data set

ϕ_1	ϕ_2	*Gbest* method	*Hybrid* method
0.1	0.9	1.0160	0.6867
0.2	0.8	0.9112	0.6866
0.3	0.7	0.9638	0.6866
0.4	0.6	0.7752	0.6866
0.5	0.5	0.7866	0.6866
0.6	0.4	0.7423	0.6866
0.7	0.3	0.7338	0.6866
0.8	0.2	0.7012	0.6865
0.9	0.1	0.7225	0.6866

Next, the relationship between parameters c_1 and c_2 for the same data set was examined. Results are presented in the Table 2. Both parameters are the acceleration constants with which the centroids in a particle move. Too high value may cause the falling out of the swarm territory, while too small may cause very slow relocation of the particles and finally not getting the optimal solution. To assess the best set of c_1 and c_2 values the fitness function of the best particle was used. The best results as the quantization error equaled to the fitness function value was achieved for $c_1 = c_2 = 0.5$ for examined data set.

Table 2. Fitness function results for parameters c_1 and c_2 for *Cl-3* data set

c_1	c_2	*Gbest* method	*Hybrid* method
0.1	0.9	0.8909	0.6866
0.2	0.8	0.7658	0.6861
0.3	0.7	0.9503	0.6867
0.4	0.6	0.7216	0.6849
0.5	0.5	0.7125	0.6834
0.6	0.4	0.7989	0.6855
0.7	0.3	0.8231	0.6866
0.8	0.2	0.7597	0.6865
0.9	0.1	0.9548	0.6867

For the selected data set we examined the influence on the results by different number of particles in the swarm. The *Gbest* and *Hybrid* methods belong to meta-heuristic optimization algorithm and wrong number of particles may be crucial to obtain the optimal solution. In the Table 3 the average time with standard deviation of three executions of two algorithms are shown for 3, 10, 50 and 100 particles. Moreover the values of classification error and the Rand index values were tested. The classification error for 3 particles equals 3.6% indicates that in the *Gbest* method 22 objects were located to the wrong cluster comparing with their original class. In other cases all objects were classified correctly.

Despite the fact that times are smaller for the *Hybrid* method, it should be remembered that as the first step the *K-means* algorithm is performed. Other parameters, like inertia coefficient or number of particles in the swarm, were also tested, but due to limited space results are not presented in this article.

3.2 Experiment 2

In the second experiment we tested three clustering methods on the data sets described at the beginning of this section and the quality of obtained clusters. Analyz-

Table 3. Average execution time, classification error (B) and Adjusted Rand index (ARI) for *Cl-3* data set

No of particles	*Gbest*			*Hybrid*		
	Time	B	ARI	Time	B	ARI
3	91.38 ± 0.778	0.036	0.890	89.29 ± 1.396	0.0	1.000
10	193.53 ± 1.388	0.0	1.000	131.99 ± 0.522	0.0	1.000
50	369.33 ± 6.698	0.0	1.000	276.37 ± 10.494	0.0	1.000
100	982.85 ± 23.160	0.0	1.000	761.64 ± 7.883	0.0	1.000

ing the values of indices in the Table 4 it can be concluded that we have to deal with different sets of data. Some of them, like *Cl-3* and *Squares4*, have distinct clusters of data, which during clustering process are completely correctly separated. In this case Adjusted Rand Index equals 1 and it means that there is a complete agreement between obtained clusters and original classes. The classification error equaled 0 indicates that there are no objects not correctly classified (located in a wrong cluster).

Table 4. Indices for quality of clusters obtained by 3 grouping methods – classification error (B) and Adjusted Rand Index (ARI)

Data set	Method					
	K-means		*Gbest*		*Hybrid*	
	ARI	B	ARI	B	ARI	B
Cl-3	1.000	0.000	1.000	0.000	1.00	0.000
Mag2	0.129	0.245	0.211	0.215	0.170	0.230
Mag2-out	-0.108	0.457	0.481	0.209	-0.094	0.452
Squares4	1.000	0.000	1.000	0.000	1.000	0.000
Squares2-2c	-0.499	0.500	-0.499	0.500	-0.499	0.500
Squares2-4c	0.001	0.500	0.001	0.500	0.001	0.500
Iris	0.693	0.120	0.806	0.067	0.693	0.120
Wine	0.427	0.197	0.654	0.118	0.613	0.129

The *Mag* and *Mag-out* sets consist of objects belonging to two classes described by normal distribution. The scatter plot of the classes is presented in the Figure 1 in the top-left corner. Each class has the long narrow ellipse shape. In this case all the methods yield rather poor values of indices. The best values of ARI have been achieved for *Gbest* algorithm – *Mag*: 0.211 and *Mag-out*: 0.481. The classification error has the smallest value also for the same method. In the case of *Mag* data 21.5% objects have been classified incorrectly (Fig. 1), while for the *Mag-out* set this value was 20.9% (Fig 2). The problem of the second data set is the outlier, for which the separate cluster is suggested by the *Gbest* method. For these data sets the process of variables standarization was made, but it has almost no influence on the results. The *K-means* method prefers round-shape clusters. Instead of using Euclidean distance, the Mahalanobis distance could be applied to this kind of data. The alternative is to apply *whitening process* described in [2].

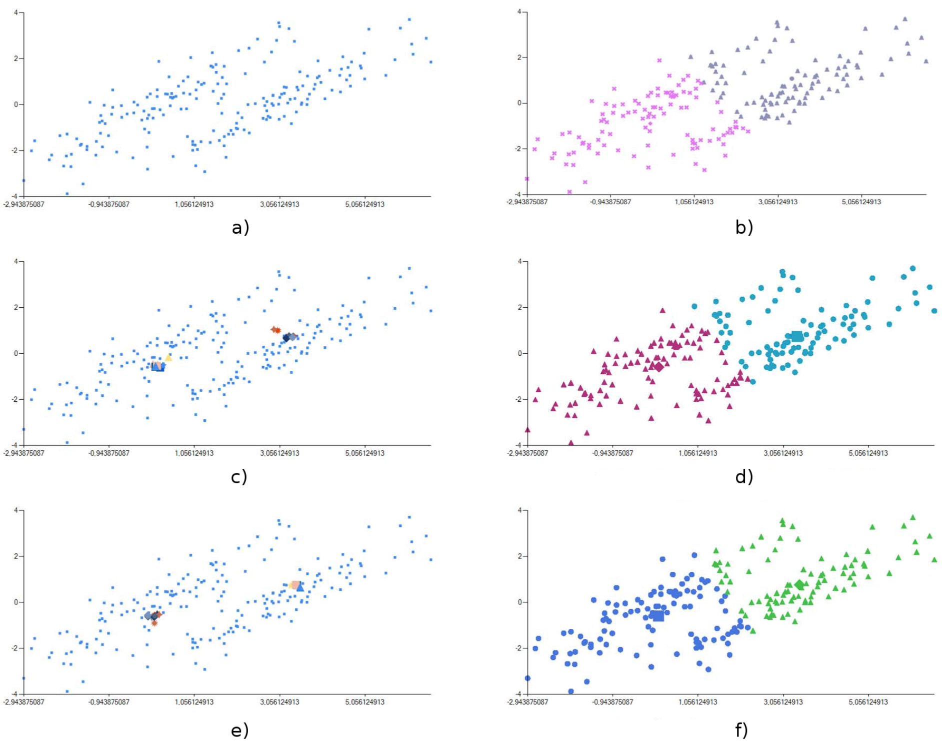


Fig. 1. *Mag* data set: a) scatter plot; particles and centroids for two algorithms: c) *Hybrid* , e) *Gbest*; final clusters obtained for three algorithms: b) *K-means*, d) *Hybrid*, f) *Gbest*

Analyzing *Squares2-2c* and *Squres2-4c* data in all cases there was 50% incorrectly classified objects. The agreement of achieved clusters and original classes was poor. For the first data set there were only two centroids and original classes placed like in the XOR problem. The obtained clusters merged two groups of objects placed

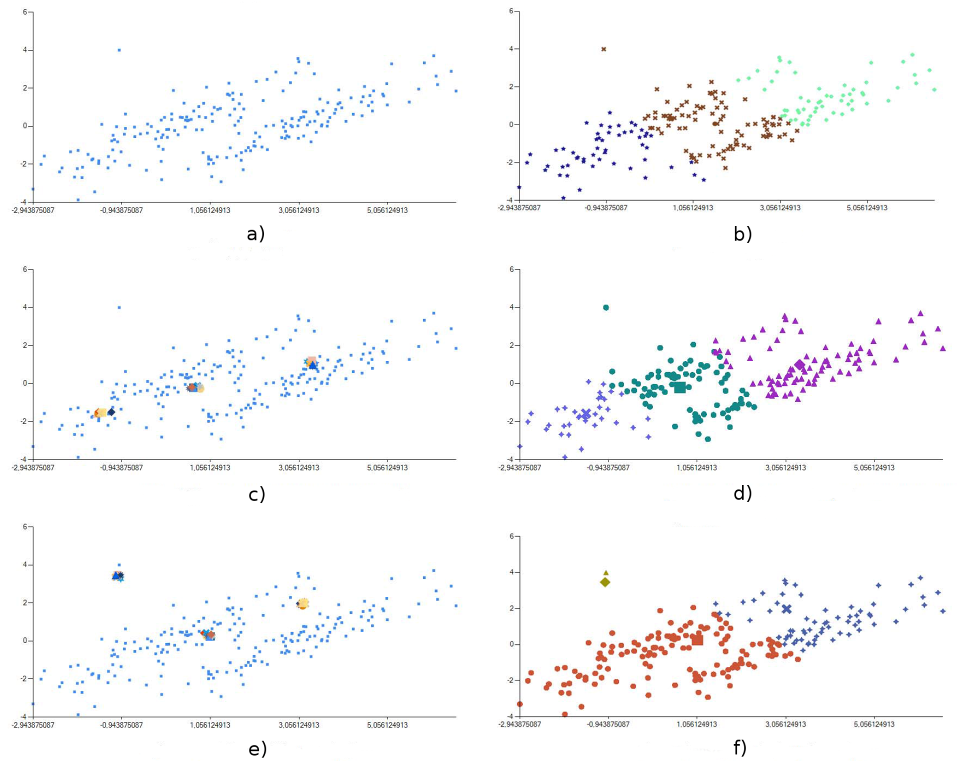


Fig. 2. *Mag-out* data set: a) scatter plot; particles and centroids for two algorithms: c) *Hybrid* , e) *Gbest*; final clusters obtained for three algorithms: b) *K-means*, d) *Hybrid*, f) *Gbest*

at the top into one cluster, and two groups below to the other. Therefore the classification error in each case equaled 50%. In the case of *Squres2-4c* set, there were four clusters – each distinct group of objects constituted distinct cluster. Instead of two original groups, we achieved for clusters. The ARI equaled 0.001 for all methods.
The best value of the ARI in the case of *Iris* set was obtained for the *Gbest* method – ARI=0.806, that means that there is good agreement between clusters and original classes. There were only 6.7% of incorrectly classified objects. For the *Wine* set the *Gbest* was the best algorithm – ARI equaled 0.654 and 11.8% of objects were located incorrectly to the clusters.
To compare algorithms based on the PSO in more detail, the indices like inter-cluster, intra-cluster distances and validity are presented in the Table 5.

Table 5. Indices assessing the quality of clusters obtained by two clustering methods based on PSO – Inter-cluster distance, Intra-cluster distance and Validity

Method	Inter-cluster	Intra-cluster	Validity
Mag			
Gbest	3.5600 ± 1.7601	0.0184 ± 0.0017	0.0148 ± 0.0015
Hybrid	4.2829 ± 1.3883	0.0152 ± 0.0004	0.0141 ± 0.0280
Mag-out			
Gbest	1.6997 ± 1.1968	0.0184 ± 0.0017	0.0245 ± 0.0294
Hybrid	3.0887 ± 1.0358	0.0187 ± 0.0013	0.0099 ± 0.0146
Squares4			
Gbest	0.7333 ± 0.4669	0.0091 ± 0.0006	0.0170 ± 0.0098
Hybrid	0.9879 ± 0.0183	$0.0074 \pm 1.4E-6$	0.0075 ± 0.0001
Squares2-2c			
Gbest	2.9201 ± 1.5894	0.0083 ± 0.0002	0.0119 ± 0.0263
Hybrid	4.0120 ± 0.0139	$0.0081 \pm 3.8E-8$	$0.0020 \pm 6.9E-6$
Squares2-4c			
Gbest	1.2531 ± 1.0588	0.0079 ± 0.0006	0.0489 ± 0.1211
Hybrid	1.0168 ± 0.0196	$0.0074 \pm 9.5E-7$	0.0073 ± 0.0001
Iris			
Gbest	1.8501 ± 1.3326	0.0141 ± 0.0024	0.0435 ± 0.1331
Hybrid	2.2151 ± 1.0719	0.0120 ± 0.0012	0.0083 ± 0.0067
Wine			
Gbest	5.1038 ± 4.3797	0.0390 ± 0.0096	0.0258 ± 0.0314
Hybrid	1.2613 ± 0.8971	0.0476 ± 0.0029	0.0484 ± 0.0190

Considering inter-cluster and intra-cluster distances, the smaller values for the former ensures larger separation between groups of objects, while the latter ensures more compact clusters with smaller variance. In general, for five out of seven examined data sets the inter-cluster distances were larger for the *Hybrid* method than for the *Gbest*.

4. Conclusions

The paper concerns three algorithms of clustering data. Two are based on the swarm intelligence idea [14] and one is classical, well-known method. Two experiments were performed to present the comparison between considered methods. The aim of the first experiment was to check what is the influence of the different values of parameters on the results obtained with *Gbest* and *Hybrid* algorithms. The second experiment described the clusters found by three methods and their quality calculated by using various indices.

It was found that for two data sets *Gbest* method gives larger inter-cluster distances and smaller intra-cluster distances than the *Hybrid* algorithm.
In the future the larger data sets will be tested and selection of the best parameters in these cases will be performed.

References

[1] X. Cui, T.E. Potok and P. Palathingal, Document Clustering using Particle Swarm Optimization *IEEE Swarm Intelligence Symposium*, The Westin , 2005.

[2] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Morgan Kaufmann, San Francisco, 1990.

[3] T. Hardin, X. Cui, R.K. Ragade, J.H. Graham and A.S. Elmaghraby, A Modified Particle Swarm Algorithm for Robotic Mapping of Hazardous Environments, *The 2004 World Automation Congress*, Seville, Spain, 2004.

[4] L. Hubert and P. Arabie, Comparing partitions., *Journal of Classification*, 193-218, 1985.

[5] R.A. Johnson, D.W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice-Hall International, Inc., 1992.

[6] J. Kennedy and R. Eberhart, Particle swarm optimization., *Proceedings of IEEE International Conference on Neural Networks*, IEEE Press, Piscataway, NJ, USA, 1942-1948, 1995.

[7] R.J. Kuo, L.M. Ho and C.M. Hu, Integration of self-organizing feature map and K-means algorithm for market segmentation, *Computers & Operations Research*, Vol. 29, Issue 11, 1475–1493, 2002.

[8] H. Liu and X. Yu, Application Research of k-means Clustering Algorithm in Image Retrieval System, Proceedings of the Second Symposium International Computer Science and Computational Technology(ISCSCT ’09), Huangshan, P.R. China, 274-277, 2009

[9] J.B. McQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1, Berkeley, Calif.: University of California Press, 281-297, 1967.

[10] M. Omran, A. Salman and A.P. Engelbrecht, Image classification using particle swarm optimization, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning* (SEAL 2002), Singapore, 370-374, 2002.

[11] O.J. Oyelade, O.O. Oladipupo, I.C. Obagbuwa, Application of k-Means Clustering algorithm for prediction of Students’ Academic Performance, *International Journal of Computer Science and Information Security*, Vol. 7, No. 1, pp. 292-295, 2010.

[12] R. Pietrzykowski and W. Zieliński and D. Kozioł, Application of k-means method for a portfolio of shares taxonomy (in Polish), Wyd. Wyższej Szkoły Ekonomiczno–Informatycznej, Warszawa, s.3, 74-76, 2005.

[13] W.M. Rand, Objective criteria for the evaluation of clustering methods, *Journal of the American Statistical Association*, 66, 846-850, 1971.

[14] D.W. van der Merwe and A.P. Engelbrecht, Data clustering using particle swarm optimization, The 2003 Congress on Evolutionary Computation (CEC'03), vol. 1, 215- 220, Canbella, Australia, 2003.

[15] http:/ /aragorn.pb.bialystok.pl/~magda/data/dane.zip

KLASYFIKACJA NIENADZOROWANA I OPTYMALIZACJA ROJEM CZĄSTEK

Streszczenie: W niniejszym artykule porównywane są trzy algorytmy analizy skupień - metoda k-średnich, algorytm gbest oraz metoda hybrydowa. Algorytmy gbest oraz hybrydowy zostały zaproponowane w publikacji [14]. Wszystkie trzy metody należą do rodziny metod niehierarchicznych, w których na początku tworzony jest podział obiektów na znaną z góry liczbę klastrów. Następnie, niektóre obiekty przenoszone są pomiędzy klastrami, by uzyskać jak najlepszy podział - wariancja pomiędzy skupieniami powinna być znacznie większa niż wariancja wewnątrz skupień. Pierwszy algorytm (k-means) jest znaną, klasyczną metodą. Drugi oparty jest na idei inteligencji roju cząstek. Natomiast trzeci jest metodą hybrydową łączącą dwa wymienione wcześniej algorytmy. Do porównania uzyskanych skupień wykorzystano kilka różnych indeksów szacujących jakość otrzymanych skupień.

Słowa kluczowe: klasyfikacja nienadzorowana, analiza skupień, optymalizacja rojem cząstek

LISTA RECENZENTÓW / THE LIST OF REVIEWERS (2012)

1. Ryszard Antkiewicz (Warszawa)
2. Adam Borowicz (Białystok)
3. Krzysztof Ciesielski (Warszawa)
4. Norbert Dojer (Warszawa)
5. Mieczysław Jessa (Poznań)
6. Krzysztof Kajstura (Bielsko-Biała)
7. Piotr Kisielewski (Kraków)
8. Miron Kłosowski (Gdańsk)
9. Joanna Kołodziej (Kraków)
10. Zbigniew Koza (Wrocław)
11. Wojciech Kwedlo (Białystok)
12. Szymon Łukasik (Kraków)
13. Grażyna Mirkowska-Salwicka (Warszawa)
14. Krzysztof Smółka (Łódź)
15. Agnieszka Nowak – Brzezińska (Sosnowiec)
16. Robert Nowicki (Częstochowa)
17. Marek Parfieniuk (Białystok)
18. Rafał Skinderowicz (Sosnowiec)
19. Władysław Szcześniak (Gdańsk)
20. Krzysztof Trojanowski (Warszawa)
21. Sławomir Wierzchoń (Warszawa)
22. Aleksander Zgrzywa (Wrocław)