Computer Science Journal of Moldova, vol.**18**, no.3(54), 2010

Performance evaluation of clustering techniques for image segmentation

Elmehdi Aitnouri Mohammed Ouali

Abstract

In this paper, we tackle the performance evaluation of two clustering algorithms: EFC and AIC-based. Both algorithms face the cluster validation problem, in which they need to estimate the number of components. While EFC algorithm is a direct method, the AIC-based is a verificative one. For a fair quantitative evaluation, comparisons are conducted on numerical data and image histograms data are used. We also propose to use artificial data satisfying the overlapping rate between adjacent components. The artificial data is modeled as a mixture of univariate normal densities as they are able to approximate a wide class of continuous densities.

Keywords: Performance evaluation, probability density function, clustering algorithm, unsupervised learning, univariate normal mixtures, gray-level histogram.

1 Introduction

Cluster analysis appeared ever since the works in [1, 2]. It began to attract a great deal of attention with the publication of Sokal and Sneath's revolutionary book on numerical taxonomy [3] and the development of high speed computers. More than a hundred different schemes of cluster analysis have been proposed [4-9] which makes it difficult for users to choose the right clustering algorithm for their application. Some authors have partially addressed this problem by evaluating and comparing several clustering techniques [5, 9-13].

©2010 by E. Aitnouri, M. Ouali

For numerical comparison, the common model used to generate test data is the mixture model. Several schemes for generating artificial mixtures of test data have been proposed [9, 14-16]. The task of cluster analysis is cast as the classification of a mixture of populations into its components provided that the number of populations and their parameters are unknown. The contribution of mixture models is not limited to test data generation. They are now used in the design of clustering algorithms. This is due to the ability of mixture models, and particularly mixture of normal densities, to approximate a wide class of continuous probability densities. Parameters estimation in a mixture model is accomplished using the Expectation Maximization algorithm (EM) [17], which maximizes the likelihood function of the observed data.

Mixture models face the difficult problem of determining the number of components in the mixture. This is known as the cluster verification [18]. Because the mixture components are not always well separated, the determination of their number is a difficult task. There are two categories of clustering algorithms: direct and verificative methods. Direct methods are those which exploit geometric properties or other *ad hoc* principles such as inflexion points in histograms [20, 21]; see also [12, 22] for other methods. Their performances depend on the application. Recently, another direct method has been proposed: elimination of false clusters (EFC) [23]. Methods which take explicitly account of the nature of the observed data, perform very well, compared to verificative methods. They are generally simple and fast. Verificative methods on the other hand do not depend on the application. They are general algorithms such as Akaike's information criterion (AIC) [25, 31], the partition coefficient (PC) [26], the ICOMP [27], the minimum description length (MDL) [28] and the minimum message length (MML) [29]. Many applications, such as image segmentation and image retrieval [30], require collecting information about the overall distribution of the pixels in the image. The common approach is to consider the histogram as a mixture of univariate normal densities and to estimate the parameters of each component of the mixture. Cluster validation techniques must be used to estimate the number of components in the

mixture.

In this paper, we propose to benchmark EFC and AIC-based algorithms. The goal of this comparison is twofold: the evaluation of the performances of EFC against AIC-based algorithm on the same benchmarking data since a fair a comparison has never been done before; and the development of a new method for generating test data from a mixture of normal densities. The artificial data shows the property of components separation as they are generated according to components overlapping rate. The overlapping rate allows the control of the degree of overlap between two adjacent components. This results in mixtures with components sufficiently separated so that the clustering algorithm has a chance to distinguish between sub-populations. The paper is organized as follows: standard clustering algorithms and mixture models are reviewed in Section 2. Section 3 deals with comparison purposes and experimental results. Finally, the discussion is presented in Section 4.

2 Unsupervised learning and standard mixtures

Consider that the data can be represented by N random vectors denoted by $X = x_1, x_2, \cdots, x_N$, and assume that it arises from a mixture of M normal densities. The distribution of the data can be approached by a PDF which takes the following form, for any $x \in X$:

$$p(x, \Gamma) = \sum_{j=1}^{M} \kappa_j f_j(x, \Theta_j) \tag{1}$$

where $f_j(.)$ is the j^{th} normal distribution with parameter $\Theta_j = (\mu_j, \sigma_j)$ representing respectively the mean and the standard deviation of the j^{th} component; κ_j are the mixing parameters, with the restrictions that $\kappa_j > 0$ for $j = 1, \cdots, M$ and $\sum_{j=1}^{M} \kappa_j = 1$, and $\Gamma_j = (\Theta_j, \kappa_j)$ totally describe $p(x, \Gamma)$. In the following, we use $\Gamma(x, \Gamma_j)$ to describe the j^{th} component of the mixture. Note that μ_j and σ_j are scalar since we are

dealing with one-dimensional distributions. Such a representation of X is called a mixture representation [20].

Mixture models have shown better data classification capacities than many conventional neural network models such as layered networks trained with the Back-Propagation algorithm. They have been used as basic configurations for radial functions in Radial Basis Networks (RBF) [32]. Various procedures have been developed for determining the parameters of a mixture of normal densities, often based on the maximum likelihood technique, leading to the EM algorithm [17] and stochastic sequential estimation [33]. The technique used to maximize the likelihood function relies on the choice of Γ most likely to give rise to the observed data. For analytical convenience, it is equivalent to minimize the log-likelihood function, which, for the given X yields:

$$E = -\log\{L(X,\Gamma)\} = -\left\{\sum_{n=1}^{N}\log(p(x_n,\Gamma))\right\} \tag{2}$$

$$= -\sum_{n=1}^{N}\log\left\{\sum_{j=1}^{M}\kappa_j f_j(x,\Theta_j)\right\}$$

Here, the log-likelihood function is considered as an error, and its minimization with respect to Γ leads to an estimate, denoted by $\hat{\Gamma} = (\hat{\Theta}, \hat{\kappa})$. A review of the maximum-likelihood technique in the context of mixture models is given in [34]. However, due to the structural complexity of mixture models, most of the maximum likelihood procedures are numerical and iterative, resulting in only locally optimal estimates. The accuracy of the final estimate $\hat{\Gamma}$ depends heavily on the initial value of Γ. This is essentially the reason why clustering algorithms are usually used to produce initial estimates of Γ.

2.1 Cluster analysis and initial estimation of Γ

Clustering algorithms basically perform an unsupervised learning that groups the input data into different categories, from which the initial values of means and widths can be calculated. A common algorithm

widely used in the literature to initialize Γ is the k-means algorithm [35, 36], an effective and popular algorithm developed within the pattern recognition community. Improved schemes for the k-means algorithm using fuzzy concepts (fuzzy c-means) are available in the literature [37]. However, clustering algorithms do not provide an estimation of M, the number of clusters in the data set, but reorganize data into M clusters, with M given by the user. Thus, a key problem, known as cluster validation [18] concerns estimation of M. Most of the previously proposed solutions to the cluster validation problem can be classified into two broad categories [12, 19, 38]: direct approaches and verificative methods. Regarding direct approaches, different methods have been proposed in the literature. However, each method is generally applicable only to a specific type of application data. In our study, we are interested in estimating the number of components in gray-level image histograms. Thus, we developed a direct algorithm, denoted by EFC (Elimination of False Clusters), to solve the cluster validation problem [23, 24]. In the next section, we review the basic principle of the EFC.

2.1.1 EFC algorithm

A gray-level image histogram can be represented by a function, $h(x)$, $x \in Gl_N$, of the gray-level frequencies of the image, where $Gl_N = \{0, 1, \cdots, N-1\}$ corresponds to the set of gray levels of the image. When a given image contains objects/regions having quite different gray-level values, different modes appear in the histogram of the image. This type of histogram is called multi-modal. However, when objects/regions in the image have close gray-level averages, they may overlap to give a single mode. Our hypothesis is that each mode corresponds to a normal distribution. This is acceptable in a large number of practical applications [39]. The EFC algorithm has been developed especially to estimate the number of modes of such histograms. Figure 1 shows a block diagram of the model, which consists of two major steps.

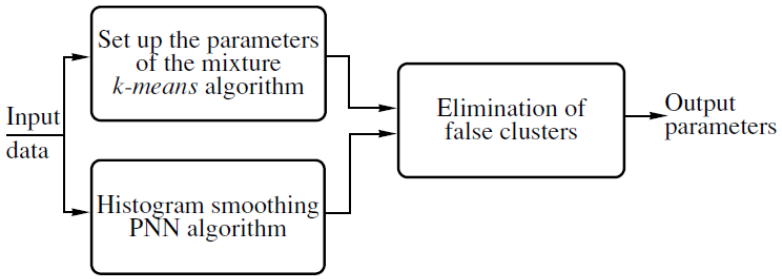

Fig. 1. A block diagram of the EFC algorithm.

In the first step, initial estimation of the mixture parameters is done using the k-means algorithm. In order to approximate each mode by at least one Gaussian, the k-means algorithm is applied with a number of clusters K greater than M, the number of modes in the image histogram. The next step mainly concerns the EFC procedure for suppressing false clusters that may result from the k-means algorithm. Basically, it takes advantage of the Gaussian PDF. Before proceeding with the elimination, a smoothing operation is performed on the histogram using a PNN (Probabilistic Neural Network or, equivalently, Parzen Window) [40]. While this operation is not essential in all cases, it greatly increases the robustness of our model to noise (especially when applied to radar images). Finding the optimal smoothing parameter for the PNN is another interesting problem that we have studied [41].

To choose the best number of clusters to use in the k-means algorithm, we have experimentally studied the accuracy of cluster centers, denoted by y_j, $j = 1, \cdots, K$ estimated by the k-means algorithm, as the function of K, the number of initial clusters. The experiment involves computing the average distance between any true center and the closest center estimated over a set of artificial histograms. In other words, we wanted to get a statistical assessment of the quality of the k-means algorithm in terms of the (average) precision with which the true centers of histogram are estimated. For this purpose, an error function was proposed to measure the quality of the set of clusters computed by the k-means algorithm. The experiment yielded a very interesting relationship between the number of initial clusters K, the true number

of modes M, and the precision of the approximation. From the statistical point of view, the k-means algorithm accurately finds all the true centers if K is chosen as at least $M + 4$. This in itself is an important discovery concerning the k-means algorithm. We further note that the choices $M+2$ and $M+3$ are also good candidates for M. Consequently, for real images it is not necessary to impose a very strict condition on the accuracy with which M is estimated.

Since the k-means algorithm is applied with an initial number of clusters greater than the number of modes, there are false clusters that must be eliminated. The EFC is used to eliminate them. The proposed EFC procedure depends on two parameters, β and γ. β is related to the relative level of the histogram at which the symmetry is measured, because it specifies the percentage of histogram height $h(y_j)$ for any cluster center y_j. In practice, β can be as large as 0.975 and as small as 0.5. The parameter γ is used as a threshold for the acceptable deviation between the true center and the closest center for the clusters computed by the k-means algorithm. If the deviation, written as $||\mu_j - y_j||$, is greater than γ, then y_j is rejected, where μ_j, $j = 1, \cdots, M$, are the real centers of modes. In real applications, μ_j are unknown. Thanks to the fact that a true center divides a mode into two symmetric parts, an equivalent test can be performed without knowing the position of the true center. Reasonable values for γ have been computed experimentally. We have measured the deviation between the true center and the closest center found by the k-means algorithm, for each combination of M and K.

Some performances of the EFC procedure are listed in [23]. The results are encouraging; nevertheless, they should be compared with those of other cluster validation methods. For this purpose, in the next section, we will present the general structure of the second broad category of algorithms, known as verificative methods.

2.2 Verificative Methods

While some practical problems can be solved by using direct approaches, they do not provide a general solution. To find generally

applicable solutions to cluster validation, many researchers have tried to formulate the problem as a multiple-decision problem [25, 26, 27, 28, 29]. The philosophy of these techniques is quite simple. Instead of asking which hypothesis is acting (how many classes are really there), we ask which model, parameterized by K, the number of clusters, best fits the data. However, as the number of clusters increases, the estimated PDF fits the empirical density increasingly tightly, at the expense of its generalization capacity. This problem can be recognized as the bias and variance trade-off in curve-fitting problems [42]. Thus, one should always try to find the minimum number of components describing a PDF, without overfitting the empirical data. In the case of mixture models, we can generate K_{max} models, where K_{max} is given by the user. The choice of the best model thus becomes a model selection problem, since we find ourselves faced with competing K_{max} models. To choose among them, researchers have developed selection criteria. The criteria are composed of two parts. The first part concerns the adequacy, usually evaluated using the maximum likelihood. The second part deals with penalization, which employs an expression that essentially depends on the number of components, K. A general expression for a criterion $C(K)$, where $K = 1, \cdots, K_{max}$ is given by:

$$C(K) = -aL_K + g(K) \tag{3}$$

where a is a scalar, L_K is the logarithm of the likelihood of the model of order K, and $g(.)$ is an increasing function depending on K. The best model is the one which minimizes $C(K)$, namely:

$$\hat{K} = \arg_{min_K} C(K) \tag{4}$$

2.2.1 Implementation

The principle is that for each value of K, $K = 1, \cdots, K_{max}$, Γ is initialized using the k-means algorithm with K initial clusters. Then, a maximum-likelihood solution is obtained using the estimated Γ and the value of K. There are no formal rules for choosing the value of K_{max}. However, several heuristics have been proposed:

- The number of observations N must be greater than the total number of parameters [43].
- Suggested choices for K_{max} include $K_{max} = \sqrt{\frac{N}{2}}$ and $K_{max} = \left(\frac{N}{\log N}\right)^{\frac{1}{3}}$ [27].

The general algorithm can be given as follows:

Algorithm 1.

Input: K_{max}
Output: $\hat{K}$
For $K = 1, \cdots, K_{max}$*:*
  Estimate Γ *using k-means with* K *initial clusters.*
  Compute the maximum likelihood for the estimated Γ*.*
  Compute $C(K)$*.*
  Choose $\hat{K}$*, such that* $\hat{K} = \arg_{min_K} C(K)$*.*

2.2.2 Akaike's information criterion (AIC)

In this paper, we are interested in the AIC. This technique was originally proposed by Akaike [31]. However, different schemes based on the AIC have been developed and used in different applications [19, 44,45,46]. In our work, we use the classical AIC proposed by Akaike and given by:

$$AIC(K) = -2L_K + 2N_p \tag{5}$$

where K is the number of components considered, L_K is the logarithm of the likelihood at the maximum likelihood solution $\hat{\Gamma}$, and N_p is the number of parameters estimated. We select K which leads to the minimum value of $AIC(K)$.

A brief comparison of some criteria for comparing models is presented in [38]. The AIC figures among these algorithms. The AIC performed better than both the ICOMP and the PC, and quite similarly to the MDL, but relatively poorly compared to the MML. Nevertheless, the AIC is the most popular criterion used in the literature. We

have chosen the classical AIC method since it does not depend on the size of the data. This is suitable for image histograms since a normalized histogram represents only the distribution of data, without any information of its size.

3 Comparison between the EFC and the AIC

It is unfortunately not possible to use results obtained respectively from [23] (EFC evaluation) and [38] (AIC evaluation) to compare the EFC and the AIC, for the simple reason that the test data used in the two experiments are different. Furthermore, neither method has been evaluated on an adequately verified data set. In order to perform a fair comparison, we must apply both algorithms to the same test data, which we will describe in the present section.

3.1 Comparison based on one vector

We used an artificial histogram composed of 3 modes as illustrated in Fig.2. To the smoothed histogram we added a Gaussian white noise of width $\sigma_n = 3$. Both algorithms were applied with $K_{max} = 7$. Figure 3 shows the resulting AIC plotted against the number of clusters $K = 1, \cdots, K_{max}$, computed from (3). The maximum likelihood technique has been used to estimate $\hat{\Gamma}_j$, $j = 1, \cdots, K$. We can see clearly from Fig. 3 that the minimum AIC is for $K = 3$, the exact number of components in the mixture.

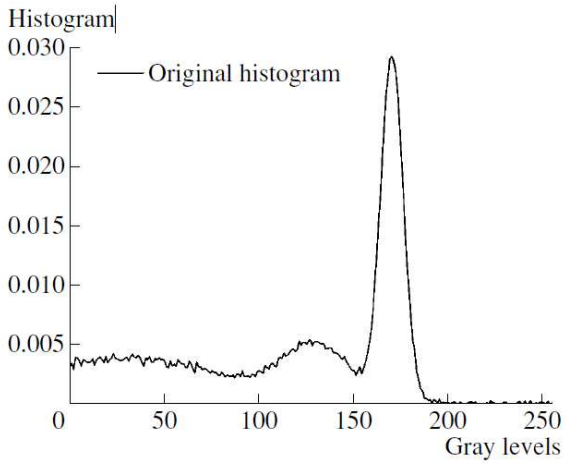

Fig. 2. Original histogram with Γ = ((28.79, 53.24, 0.44), (129.92, 17.6, 0.17), (170.1, 5.92, 0.39)).

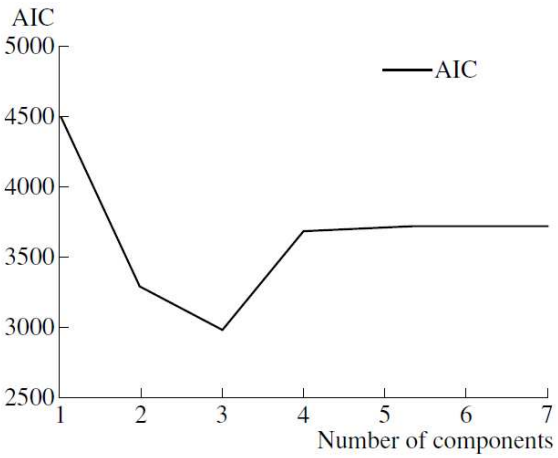

Fig. 3. Values of the AIC versus number of components.

Figure 4 shows the plot of both the original histogram and the

resulting histogram at the $\hat{\Gamma}_3$ solution, given by

$$\hat{\Gamma}_3 = ((36.00, 21.63, 0.26)(141.39, 21.24, 0.25)(168.53, 8.12, 0.49))$$

The estimation using the AIC is good: although there is a relative shift in the means for the first two components of the estimated histogram, the result is still acceptable. For the case of the EFC, Table 1 summarizes the parameters estimated using the k-means algorithm with $K_{max} = 7$. Table 1 is divided into two parts. The first multi-column, denoted by "Before EFC" presents the parameters of each component resulting from applying the k-means algorithm with $K_{max} = 7$. The second part of Table 1, the multi-column "after EFC ML", presents the resulting estimated parameters of each component after applying the EFC procedure and the maximum-likelihood algorithm respectively. In this part, dashes represent components eliminated by the EFC procedure. The values of both β and γ were set as in the experiments performed in [23], namely $\beta = 0.97$ and γ chosen from the γ table.

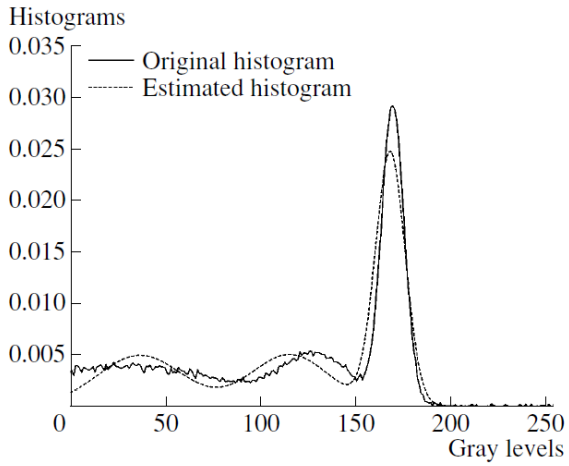

Fig. 4. Reconstructed PDF using the resulting AIC parameters, MSE = 0.0085.

Table 1. Results for the EFC for the artificial histogram, $K_{max} = 7$ and $\beta = 0.97$.

	Before EFC			After EFC & ML		
cluster	means	width	MP	means	width	MP
1	15.33	9.26	0.11	29.5	51.24	0.43
2	44.77	10.08	0.11	–	–	–
3	76.86	11.16	0.09	–	–	–
4	112.32	9.51	0.10	–	–	–
5	136.85	8.72	0.11	131.56	15.84	0.18
6	169.69	6.38	0.45	169.32	6.09	0.39
7	208.14	26.65	0.03	–	–	–

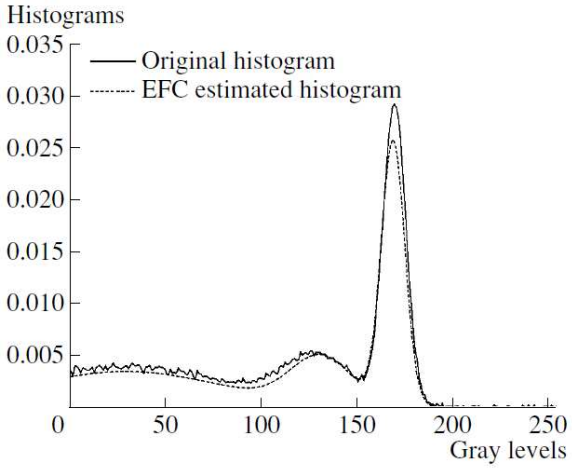

Fig. 5. The reconstructed PDF using the resulting EFC parameters, MSE = 0.0041.

From Table 1, we can see that the EFC procedure has eliminated four spurious clusters. Only clusters corresponding to the true components of the mixture have been kept. Thus, the EFC also finds the exact number of components. Classification using the k-means algo-

rithm, initialized using the values of the remaining cluster means (see Table 1), permits the redistribution of points belonging to eliminated clusters. Finally, we apply the maximum-likelihood technique. Figure 5 shows the plot of both the original and the resulting EFC mixtures for the parameters given in Table 1.

From Figs. 4 and 5, we can see that while both algorithms estimate the exact number of components, the EFC procedure is more accurate than the AIC (see MSE in Figs. 4 and 5). This result is in perfect agreement with the k-means experiment performed in [23]. Indeed, for the case of the AIC, the parameters used to initialize the likelihood were obtained using the k-means algorithm. For the AIC [3], the k-means algorithm was performed with $K = 3$ initial clusters. This is the exact number of components of the test histogram. In this case, the estimated centers of the components are less accurate than those obtained with a large value of K. This does not help the maximum likelihood to result in accurate final estimates. In contrast, for the case of the EFC, the initialization of each component's mean is obtained using the k-means algorithm with $K = K_{max} = 7$ clusters. Therefore, the centers of the real components are very well estimated. This helps the maximum-likelihood procedure to converge, which explains why the parameters estimated using the EFC were clearly more accurate than those estimated by the AIC. We can also evaluate the two algorithms in terms of complexity. For example, we need to perform the k-means algorithm K_{max} times, followed by the maximum likelihood to compute $AIC(K)$, $K = 1, \cdots, K_{max}$. In contrast, the EFC algorithm requires one k-means run, followed by the maximum likelihood. Thus, the EFC technique is roughly K_{max} times faster than the AIC. Although the EFC compares favorably to the AIC in the example described above, it is necessary to examine the effectiveness of the EFC in more general situations. For this purpose, we need to apply both algorithms to a large set of test data. In this way, we can obtain a statistical assessment of the general performance trend. Statistical comparison between algorithms is possible, since mixture models have been used as general test data for clustering algorithms. Indeed, a number of different schemes for generating artificial mixtures have been proposed. Blashfield [9]

and Edelbrock [14] used unconstrained multivariate normal mixtures with fairly complex covariance structures. Milligan and Issac proved in [8] that the generation processes used in [9] and [14] lead to data with overlapping clusters. Other researchers such as Kuiper and Fisher [15], and Bayne et al. [11] have used multivariate normal clusters with simple covariance structure and have directly manipulated variables such as the separation between clusters. In 1985, Milligan and Cooper [12] examined the ability of about 30 different clustering algorithms to determine the number of clusters in a data set. For this purpose, Milligan has developed an algorithm for generating test data [16]. The algorithm is described in nine steps. The main assumption, however, is that the generated data do not overlap in the first dimensional space. The verification of this assumption was mainly done using non-automatic techniques such as visual inspection. All of these generation methods try to define ad hoc criteria in order to handle overlapping components. However, there is no formal definition of the concept of overlap. This raises questions concerning the effectiveness of the generation schemes. In this paper, we introduce a new algorithm for generating mixtures of univariate normal densities. First, we give an analytic definition of the concept of overlap. We have chosen relative component overlap instead of total component overlap in order to preserve the appearance of each component in the mixture. This definition allows us to control the overlapping process, thus offering the possibility of generating a large number of such mixtures with known degrees of overlap. The generation of such examples, denoted by non-overlapped vectors, is described in the next section.

3.2 Generation of non-overlapped vectors

When we want to generate a large set of mixture data, the problem is how to ensure that modes are not totally overlapped. As an example of what we mean by overlapped, we generate a three-component mixture, but due to component overlap, the mixture results in only two components. The example in Fig. 6 illustrates this phenomenon. The mixture in Fig. 6a is actually composed of three components, despite

the fact that only two components are visible. The vector in Fig. 6a is called an overlapped vector; the vector in Fig. 6b on the other hand is not. Note that the difference between the parameters of Fig. 6.a, b is the value of the second width σ_2. We can perform algorithm comparison using overlapped vectors, as in the example of Fig. 6a. Each algorithm will most likely estimate the same number of components. The process is indeed still fair. However, such vectors are not valid for use in evaluating algorithms, due to the degree of component overlap. To avoid such situations, it is necessary to control the overlapping process.

Definition 1. *We define the overlapping rate, denoted by OLR, as,*

$$OLR = \frac{\min(p(x))(\mu_i \leq x \leq \mu_{i+1})}{\min(p(\mu_i), p(\mu_{i+1}))}. \tag{6}$$

In the above formulae, $p(x)$ is the mixture function and $\mu_i < \mu i + 1$. Figure 7 illustrates two different overlappings for a mixture of two Gaussians. $OLR \rightarrow 0$ when the Gaussians are almost totally separated, since $\min(p(x)) \rightarrow 0$, $(\mu_i \leq x \leq \mu_{i+1})$ as in Fig. 7a. $OLR \rightarrow 1$ when $\min(p(x)) \rightarrow \min(p(\mu_i))$ as in Fig. 7b. Thus, OLR specifies the degree of overlap between two adjacent components of a mixture. Our goal here is to develop an automatic algorithm for generating mixture components that are not completely overlapped; in other words $OLR < 1$ should be true for all pairs of adjacent components. Such an algorithm allows us to generate a large set of non-overlapped vectors. Furthermore, we could also control the overlapping rate if we want to perform a refined evaluation of algorithms.

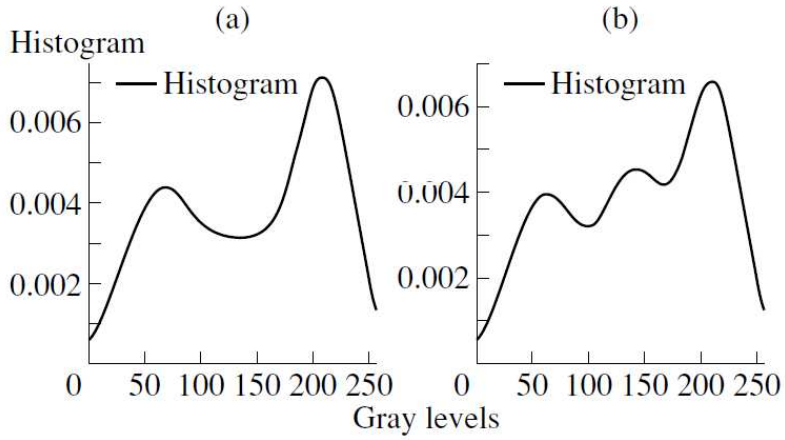

Fig. 6. Artificial histograms generated from a true mixture of two Gaussians. (a) $\Gamma_1 = (60, 28, 0.3)$, $\Gamma_2 = (140, 32, 0.3)$, $\Gamma_3 = (210, 25, 0.4)$. (b) $\Gamma_1 = (60, 31, 0.3)$, $\Gamma_2 = (140, 22, 0.3)$, $\Gamma_3 = (210, 25, 0.4)$.

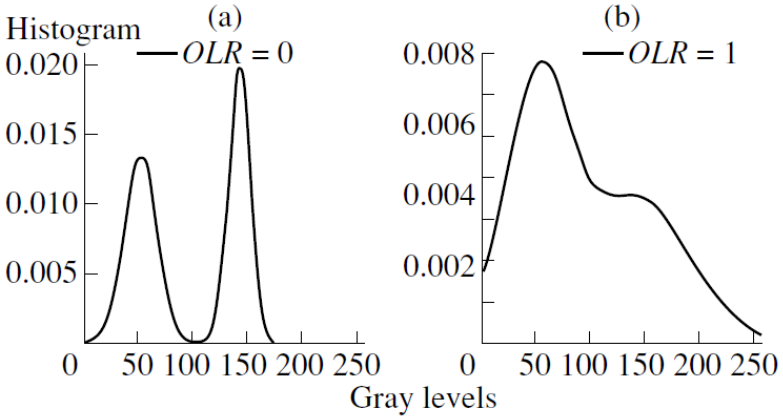

Fig. 7. OLR principle. (a) $OLR \approx 0$, (b) $OLR \approx 1$.

From (6), a general condition for $OLR < 1$ can be obtained using the derivatives of $p(x)$. This idea has been applied to the solution of the edge-detection problem using the Laplacian of Gaussian edge detector [47]. Interestingly, there is an astonishing similarity between the case of overlapping components and the case of a double-step edge. The edge-detection problem involves the appearance of a false edge located between the two true edges when they are smoothed with a Gaussian. In [47], the problem is treated as a mixture of two Gaussians with equal width σ. It is proven that a false edge appears if and only if $2\sigma < \mu_d$, where μ_d is the distance between the means of the two Gaussians. Viewed from the perspective of our case, the false edge corresponds to $\min(p(x))$ defined in (6). Thus, we have:

Corollary 1 - *If we have a mixture of two Gaussians with the same width* σ, $OLR < 1$ *iff* $2\sigma < \mu_d$.

Corollary 1 is a direct consequence of results obtained in [47]. Unfortunately, it is not valid for the case where $\sigma_1 \neq \sigma_2$, and the mathematical approach used in [47] cannot be extended to this general case. As it can be seen in what follows, developing a general condition for $OLR < 1$ is much more difficult. The algorithm will be iterative, dealing with two adjacent components at each iteration. When the width of the first component is fixed, the condition $OLR < 1$ will depend on the width of the next component. Here is a sketch of the algorithm:

3.2.1 Algorithm for generating non-overlapped vectors

Algorithm 2.

Generate M*, the number of components in the mixture.*
For $i = 1, \cdots, M$*:*
    Randomly generate μ_i *and* κ_i *such that* $\mu_i < \mu_{i+1}$, $\kappa_i > 0$, *and* $\sum_{i=1}^{M} \kappa_i = 1$.
    Randomly generate σ_1 *such that* $\sigma_1 < \mu_1 - \mu_2$.
For $i = 2, \cdots, M$*:*
    Compute σ_i *such that* $OLR < 1$.

The number of components M can be set by the user. It is also

convenient, without loss of generality, to generate M means μ_i and sort them so that $\mu_i < \mu_{i+1}$. This will facilitate the execution of the algorithm. However, σ_1 should be smaller than $\mu_2 - \mu_1$; otherwise, there will be no solution for σ_2 to satisfy the non-overlapping condition. Thus, the problem is to find an upper bound for σ_{i+1} given σ_i, in order to ensure that $OLR < 1$.

In order to develop a general algorithm, let us consider the overlap of two adjacent components $\Gamma_1(x)$ and $\Gamma_2(x)$ with $\Gamma_1 \neq \Gamma_2$. Let us denote by $S_\sigma = \kappa\Gamma(\mu - \sigma)$ the height at which the width σ is located for a given component $\Gamma(x)$. For the general case of the overlapping process for two adjacent components $\Gamma_1(x)$ and $\Gamma_2(x)$, we have $S_{\sigma_1} \neq S_{\sigma_2}$ since the two components do not necessarily have the same height.

Definition 2. *We define a notion of apparent width, denoted by $\hat{\sigma}$, as the deviation of the higher component from its center at height S_{σ_l}, for a pair of adjacent components. Here, S_{σ_l} is the height at which the width of the lower component is located.*

We distinguish two cases: $\Gamma_1(\mu_1) > \Gamma_2(\mu_2)$, denoted by case 1, and $\Gamma_1(\mu_1) < \Gamma_2(\mu_2)$, denoted by case 2. Figure 8 illustrates the principle of apparent width for the two cases. Using Definition 2, a generalization of Corollary 1 can be stated as a hypothesis:

Hypothesis 1 - *If we have a mixture of two Gaussians with different heights, $OLR < 1$ iff $\hat{\sigma}_h + \sigma_l < \mu_d$.*

$\hat{\sigma}_h$ is the apparent width of the higher component, σ_l is the width of the lower component, and μ_d is the distance between the two means. It is self-evident that if the components have the same widths and the same heights, Hypothesis 1 collapses into Corollary 1, since $\hat{\sigma}_h = \sigma_l = \sigma$. However, it is very difficult to prove Hypothesis 1. For each of the above cases, we can compute $\hat{\sigma}_h$ as a function of Γ_1 and Γ_2. Then, by introducing the expression of $\hat{\sigma}_h$ in Hypothesis 1, we can solve the resulting relation in order to obtain the bound on σ_2. For the two cases we have:

$$\begin{cases} A\sqrt{\ln(B\sigma_2)} + \sigma_2 < \mu_d & \text{case 1} \\ C\sigma_2\sqrt{\ln\frac{D}{\sigma_2}} + \sigma_1 < \mu_d & \text{case 2} \end{cases} \tag{7}$$

where A, B, C and D are known values. For deduction details, see [48]. By solving (7), we can obtain a condition on the upper bound of σ_2 as a function of the remaining known parameters. However, (7) is non-linear, which will introduce difficulties for obtaining general solutions. The non-linearity of (7) arises from the non-linearity of the Gaussian expression. Thus, it is necessary to simplify (7) by approximating the Gaussian expression.

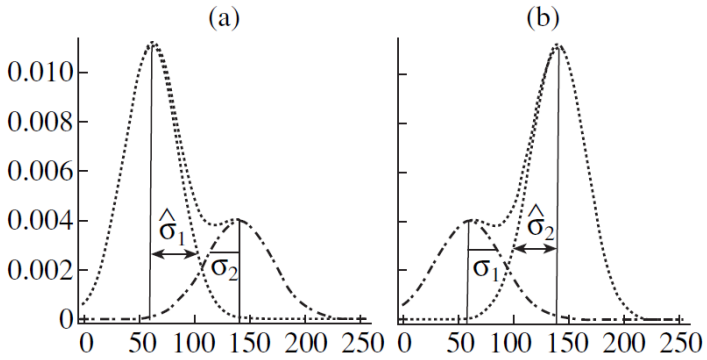

Fig. 8. Apparent width. (a) when $\Gamma_1(\mu_1) > \Gamma_2(\mu_2)$, (b) when $\Gamma_1(\mu_1) < \Gamma_2(\mu_2)$.

3.2.2 Approximation of the Gaussian

Consider a family of lines $\Delta_l = \Delta_1, \Delta_2, \Delta_3, \cdots, \Delta_p$, where Δ_i, $i = 1, \cdots, p$, is a tangent line to the point $(x_{i\sigma} = \mu - i\sigma, p(x_{i\sigma}))$. If we approximate a Gaussian by the series of lines Δ_i, we obtain a piece-wise linear approximation, denoted by $\hat{g}_l(x)$ and given by

$$\begin{cases} \hat{g}_l(x) = i\frac{S}{\sigma}e^{-\frac{1}{2}i^2}\left(x - \mu + \frac{i^2+1}{i}\sigma\right) \\ x \in [\mu - f(i-1)\sigma, \mu - f(i)\sigma] \end{cases} \tag{8}$$

where

$$f(i) = \frac{(i^2+1)e^{-\frac{i^2}{2}} - ((i+1)^2+1)e^{-\frac{(i+1)^2}{2}}}{ie^{-\frac{i^2}{2}} - (i+1)e^{-\frac{(i+1)^2}{2}}}$$

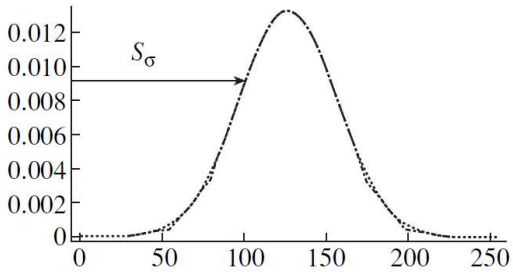

Fig. 9. Approximation of a Gaussian with $p = 3$ and MSE = $7.48742e^{-06}$.

Due to the symmetry of the Gaussian, we have developed only the approximation of the left part. Figure 9 shows the result of this approximation for $p = 3$. The approximation error, denoted by E_{app}, is given by:

$$E_{app}(p) = \frac{1}{\sqrt{2\pi}} \sum_{i=1}^{p} e^{-\frac{i^2}{2}} \times [f(i-1) - f(i)] \left[\frac{f(i) - f(i-1)}{2} - \frac{i^2+1}{i} \right] \tag{9}$$

It is proven in [48] that $E_{app}(p)$ does not depend on the width σ. However, $E_{app}(p)$ decreases as p increases and is almost constant when $p \geq 3$. Note that this approximation has been developed especially for this work. With this new approximation of the Gaussian, the computation of the apparent width $\hat{\sigma}$ can be done on the tangent lines Δ_i , $i = 1, \cdots, p$, and we will have (10) as solutions to the condition of Hypothesis 1:

$$0 < \sigma_2 \leq \frac{-(2\sigma_1 - \mu_d) + \sqrt{\delta}}{2} \text{ (case 1)} \tag{10}$$

where δ is the discriminant of the quadratic form given by

$$\begin{gathered} T_1(\sigma_2) = \sigma_2^2 + (2\sigma_1 - \mu_d)\sigma_2 - \tfrac{\kappa_2}{\kappa_1}\sigma_1^2 \text{ and} \\ 0 < \sigma_2 \leq \frac{\frac{\kappa_2}{\kappa_1} 2\sigma_1 - \sqrt{\delta}}{2} \text{ case 2} \end{gathered} \tag{11}$$

where δ is the discriminant of the quadratic form given by

$$T_2(\sigma_2) = \sigma_2^2 - \frac{\kappa_2}{\kappa_1} 2\sigma_1\sigma_2 + \frac{\kappa_2}{\kappa_1}\mu_d\sigma_1 - \kappa_2\sigma_1^2.$$

All deduction details are available in [48]. The conditions in (10) and (11) express the bound within which σ_2 should be picked in order that the overlap of the two adjacent components satisfies $OLR < 1$. In each step i of the algorithm $i = 1, \cdots, M-1$, (10) and (11) are used to generate σ_{i+1}. By choosing σ_{i+1} as suggested in (10) and (11), we ensure control of all the parameters of the vector. For implementation purposes, especially when we want to generate a large set of non-overlapped vectors, we define other parameter measures that are grouped in a characteristic vector, denoted by CVS (Characteristic Vector of the Set). These measures are:

(a) the number of vectors forming the set;

(b) the maximum number of components in the set;

(c) the minimum number of components in the set;

(d) the minimum distance between the means of components;

(e) the minimum width of Gaussian white noise added to vectors of the set; and finally,

(f) the maximum width of Gaussian white noise added to vectors of the set.

The example used here to compare the EFC with the AIC has a CVS = (1, 3, 3, 12, 3, 3).

3.3 Comparison based on a set of vectors

In this section, we intend to compute a kind of average comparison between the EFC and the AIC. For this purpose, we will use a set of non-overlapped vectors. The evaluation proposed here is divided in two parts. First, we will compute the ability of each algorithm to

estimate the exact number of components. This type of evaluation is used in [12, 23, 38]. The results will be presented in tables showing all statistics. Secondly, we will use the parameters of each component and reconstruct the PDF in order to compute the measure of adequacy of the estimated vectors.

Both experiments use three different sets of 1000 vectors each, generated by the algorithm described in the previous section. The CVSs of the three sets are given respectively by: (1) CVS_1 = (1000, 1, 1, 12, 0, 0), a set containing only vectors of one component, (2) CVS_2 =(1000, 2, 2, 12, 0, 0), a set containing only vectors of two components and (3) CVS_3 = (1000, 3, 3, 12, 0, 0), a set containing only vectors of three components. Moreover, in order to evaluate the robustness of both algorithms against noise, we have used the same generated sets and added a Gaussian white noise of width $\sigma_n = 2$ to form new noisy sets. Both algorithms were applied with $K_{max} = 7$.

The results of the first experiment are presented in Tables 2 and 3 (non-noisy and noisy sets, respectively). Each of these tables is divided into three parts. Each part, identified by its CVS (row 2), presents the application of both algorithms to a given set. The first column gives the different possibilities for the number of components each algorithm can estimate. The cells of the table report the percentage of cases in which the two algorithms estimate the number of components given in column 1. As an example, when the AIC is applied to the non-noisy set with CVS_1, it estimates two components in 42% of cases.

From the results reported in Table 2 (non-noisy sets), the AIC and EFC performances are quite similar. Nevertheless, as the number of components of the set increases, the EFC performs relatively better than the AIC (15% better for the set with CVS_3). For the noisy sets reported in Table 3, we see that the AIC is more robust than the EFC. Indeed, there are no great differences between the AIC results shown in Tables 2 and 3. The EFC is less robust since its performances were degraded by about 12% for all three sets. Note that the smoothing operation using the PNN was not performed in these experiments, in order to evaluate the robustness of the EFC against noise.

The second experiment computes the adequacy degree to which the

Table 2. Comparison between EFC and AIC applied to the noiseless sets. NCE is the estimated number of clusters.

	Noise standard deviation $\sigma_n = 0$					
	CVS_1		CVS_2		CVS_3	
NCE	AIC	EFC	AIC	EFC	AIC	EFC
1	54	56	15	8	0	0
2	42	23	47	52	13	6
3	2	12	21	16	51	66
4	2	5	10	10	28	15
5	0	3	6	12	6	10
6	0	0	1	2	2	2
7	0	0	0	0	0	0

Table 3. Comparison between EFC and AIC applied to noisy samples.

	Noise standard deviation $\sigma_n = 2$					
	CVS_1		CVS_2		CVS_3	
cluster	AIC	EFC	AIC	EFC	AIC	EFC
1	54	46	12	5	0	0
2	38	31	45	43	6	3
3	5	15	25	31	49	54
4	2	4	12	11	25	26
5	1	2	2	6	13	10
6	0	1	1	3	5	5
7	0	0	1	1	2	2

Table 4. Average MSE for each set.

	$\sigma_n = 0$			$\sigma_n = 2$		
	CVS_1	CVS_2	CVS_3	CVS_1	CVS_2	CVS_3
AIC	0.022	0.047	0.248	0.021	0.43	0.0122
EFC	0.008	0.038	0.072	0.019	0.042	0.081

two algorithms fit, measured by the mean square error (MSE). When an algorithm estimates a given number of components K, the maximum likelihood estimates for K components are used to reconstruct the estimated PDF. Thus, a MSE is computed between the original and the estimated vectors. This experiment is applied to both sets, noiseless and noisy. Table 4 shows the average MSEs resulting from the application of the two algorithms to the different sets. We can see from Table 4 that in all cases, the EFC has better overall adequacy than the AIC. The average MSE, however, does not provide specific information regarding the behavior of the adequacy as a function of the estimated number of components. Such information would help to perform an objective comparison.

To this end, we compared the adequacy of the two algorithms using only vectors resulting in the estimation of the same number of components. In other words, instead of using, for example, all the vectors of CVS_1 to compute the MSE, we divide them into groups. Each group contains only vectors resulting in the estimation of the same number of components. Thus, we will have K_{max} groups. We then compute an average MSE for each group. Note that if a group contains only a few vectors, the average MSE can be biased. On the other hand, the average MSE is representative when a group contains a large number of vectors.

Figure 10 shows the plots of adequacy for EFC, for CVS_1 in (a), CVS_2 in (b) and CVS_3 in (c). The adequacy behaves similarly for the two algorithms. Indeed, adequacy for vectors resulting in correct estimation of the number of components M is relatively good compared

to those resulting in a close estimation of M. However, the results totally deteriorate for a significant underestimation of M (see Fig. 10c with CVS_3 for vectors resulting in an estimation of $M = 1$). Finally, the adequacy is better when we overestimate M (see Fig. 10a with CVS_1 for vectors resulting in an estimation of $M = 6$). When no vector results in an estimation of a given number of components, the corresponding error is set to $MSE = 1$, which explains the behavior of the curves for $K = 7$. This overall behavior of the adequacy is also observed for the noisy sets in Fig. 11.

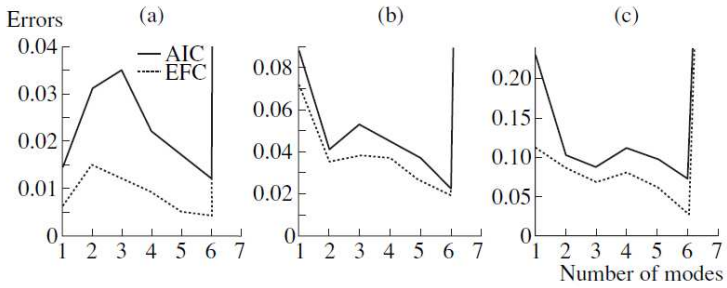

Fig. 10. Adequacy of the AIC and the EFC applied to non-noisy sets: (a) CVS_1, (b) CVS_2, and (c) CVS_3.

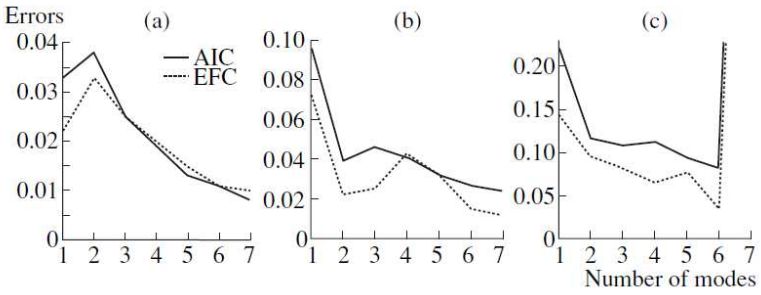

Fig. 11. Adequacy of the AIC and the EFC applied to noisy sets: (a) CVS_1, (b) CVS_2 and (c) CVS_3.

The behavior of clustering algorithms observed above is not unique to our experiments. Indeed, Windham and Cutler [49] observed it,

and developed an algorithm, called MIREV in order to use it. They argued that the best solution may be indicated by an elbow or a knee in a plot of the values versus the number of components. However, the occurrence of the elbow or knee may not necessarily mean that a solution is particularly good, but simply that its neighbors are relatively bad. Note that the experiments performed by Windham and Cutler were done using bivariate normal densities of three components each.

4 Conclusion

The EFC and AIC algorithms are two clustering algorithms pertaining to two different categories: direct and verificative methods. The category of verificative methods provides algorithms than can be used for all types of application data. However, when a prior knowledge about certain characteristics of the application data is available, direct methods can be designed in order to exploit this knowledge. In the case of gray-level image histograms, the EFC clearly outperforms AIC. Indeed, the effectiveness of the EFC has been shown in this paper in terms of its ability to estimate the exact number of modes in a histogram and the adequacy resulting from PDF estimation using the estimated parameters. It is also straightforward to verify that the EFC is roughly K_{max} times ($K_{max} = 7$ in our experiments) faster than the AIC. Note that a more extensive comparison can be conducted, including other clustering algorithms. In this paper, we have conducted a comparison between the EFC algorithm and the AIC algorithm. The comparison was designed to use a novel algorithm for generating mixture test data with non-overlapped components. This algorithm makes it possible to perform statistical tests and evaluations, since it can handle a large number of test data. Moreover, its flexibility allows the design of more detailed and appropriate statistical tests, such as algorithm's robustness in relation to component overlap. This type of test provides information about algorithm limitations. The formal definition of component overlap introduced in this paper can be used to design multivariate mixtures of test data. Indeed, one can keep a Milligan [16] generation framework, while using our algorithm to satisfy the

non-overlap condition necessary in 1D.

References

[1] Tryon, R.C., Cluster Analysis, volume of Ann Arbor, Mich. Edwards Brothers, MA, 1939.

[2] Zubin, J.A., *A Technique for Measuring Like-mindedness*, Abnormal and Social Psychology, 1938, vol. 33.

[3] Sokal, R.R. and Sneath, P.H.A., Principles of Numerical Taxonomy, San Francisco: Freeman, 1963.

[4] Anderberg, M.R., Cluster Analysis for Applications, New York: Academic Press, 1973.

[5] Bailey, K.D., Cluster Analysis, Heise, D., Ed. of Sociological Methodology edition, San Francisco: Jossey- Bass, 1974.

[6] Cormack, R.M., *A Review of Classification*, J. of the Royal Statisticians, Series A, 1971, vol. 134(3).

[7] Everitt, B.S., Cluster Analysis, London: Halstead Press, 1974.

[8] Milligan, G.W., *An Examination of the Effect of Six Types of Errors Perturbation on Fifteen Clustering Algorithms*, Psychometrica, 1980, vol. 45(3), pp. 325–342.

[9] Blashfield, R.K., *Mixture Model Test of Cluster Analysis: Accuracy of Four Agglomerative Hierarchical Methods*, Psychological Bulletin, 1976, vol. 83(3), pp. 377–388.

[10] Wolfe, J.H., *Pattern Clustering by Multivariate Mixture Analysis*, Multivariate Behavioral Analysis, 1970, vol. 5(4), pp. 329–349.

[11] Bayne, C.K., Beauchamp, J.J., Begovich, C.L., and Kane, V.E., *Monte Carlo Comparisons of Selected Clustering Procedures*, Pattern Recognition, 1980, vol. 12(2), pp. 51–62.

[12] Milligan, G.W. and Cooper, M.C., *An Examination of Procedures for Determining the Number of Clusters in a Data Set*, Psychometrica, 1985, vol. 50(2), pp. 159–179.

[13] Dudes, R. and Jain, A.K., *Clustering Techniques: the User's Dilemma*, Pattern Recognition, 1976, vol. 8(4), p. 247.

[14] Edelbrock, C., *Comparing the Accuracy of Hierarchical Grouping Techniques: the Problem of Classifying Everybody*, Multivariate Behavioral Research, 1979, vol. 14, pp. 367–384.

[15] Kuiper, F.K. and Fisher, L., *A Monte Carlo Comparison of Six Clustering Procedures*, Biometrica, 1975, vol. 31(1), pp. 86–101.

[16] Milligan, G.W., *An Algorithm for Generating Artificial Test Clusters*, Psychometrica, 1985, vol. 50(1), pp. 123– 127.

[17] Dempster, A.P., *Maximum Likelihood from Incomplete Data*, Via the EM Algorithm, J. of the Royal Statistical Society, 1977, vol. B 39(1), pp. 1–38.

[18] Jain, A.K. and Dubes, R.C., *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[19] Zhang, J. and Modestino, J.M., *A Model-fitting Approach to Cluster Validation with Application to Stochastic Model-based Image Segmentation*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 1990, vol. 12(10), pp. 1009–1017.

[20] McLachlan, G.J. and Basford, K.E., Mixture Models, New York: Marcel Deker, 1988.

[21] Aitnouri, E. and Ouali, M., *Multithreshold-based SAR image segmentation to targets and shadows detection*, Journal of Applied Remote Sensing (SPIE-JARS), Submitted in October 2010.

[22] Dudes, R. and Jain, A.K., *Validity Studies in Clustering Methodologies*, Pattern Recognition, 1979, vol. 11(4), pp. 235–254.

[23] Aitnouri, E.M., Wang, S., Ziou, D., Vaillancourt, J., and Gagnon, L., *Estimation of a Multi-modal Histogram's PDF using a Mixture Model*, Neural, Parallel & Scientific Computation, 1999, vol. 7(1), pp. 103–118.

[24] Aitnouri, E.M., Wang, S., Ziou, D., Vaillancourt, J., and Gagnon, L., *An Algorithm for Determination of the Number of Modes in Image Histograms*, Vision Interface V'99, Trois-Rivières, 1999, pp. 368–374.

[25] Akaike, H., *Information and an Extension of the Maximum Likelihood Principle*, 2nd Int. Symp. on Information Theory, Budapest, 1973, pp. 267–281.

[26] Bozdek, J.C., Pattern Recognition with Fuzzy Objective Function, New York: Plenum, 1981.

[27] Bozdogan, H., *Mixture-model Cluster Analysis Using Model Selection Criteria and a New Informational Measure of Complexity*. In Bozdogan, H. et al., Eds., The first US/Japan Conf. on the Frontiers of Statistical Modeling: An Informational Approach, Kluwer Academic Publishers, 1994, pp. 69–113.

[28] Rissanen, J., *Modeling by Shortest Data Description*, Automatica, 1978, vol. 14(3), pp. 465–471.

[29] Wallace, C.S. and Boulton, D.M., *An Information Measure for Classification*, Computer Journal, 1968, vol. 11(2), pp. 185–194.

[30] Kelly, P.M., Cannon, T.M., and Hush, D.R., *Query by Image Example: The CANDID Approach*, SPIE, 1995, vol. 242, pp. 238–248.

[31] Akaike, H., On Entropy Maximization Principle, in Applications of Statistics, Krishnaiah, P.R., Ed., North Holland Publishing Company, 1977, pp. 27–41.

[32] Powell, M.J.D., *Radial Basis Functions for Multivariate Interpolation: a Review, Algorithms for Approximation*, 1987, no. 1, pp. 143–167.

[33] Traven, H.G.C., *A Neural Network Approach to Statistical Pattern Classification by Semiparametric Estimation of Probability Density Functions*, IEEE Transactions on Neural Networks, 1991, vol. 2(3), pp. 366–377.

[34] Redner, R.A. and Walker, H.F., *Mixture Densities, Maximum Likelihood and the EM Algorithm*, SIAM Review, 1984, vol. 26(2), pp. 195–239.

[35] Fukunaga, K., Introduction to Statistical Pattern Recognition, New York: Academic Press, 1972.

[36] Tou, J.T. and Gonzalez, R.C., Pattern Recognition Principles, Addison-Wesley, Reading, MA, 1974.

[37] Bezdek, J.C., Pattern Recognition with Fuzzy Objective Function Algorithms, New York: Plenum, 1981.

[38] Oliver, J.J., Baxter, R.A., and Wallace, C.S., *Unsupervised Learning Using MML*. 13th Conf. on Machine Learning, Bari, Italy, 1996, pp. 364–372.

[39] Lawrence, D.B. and Gene Hwang, J.T., *How to Approximate a Histogram by a Normal Density*, The American Statistician, 1993, vol. 47(4).

[40] Parzen, E., *On the Estimation of a Probability Density Function and Mode*, Ann. Math. Stat., 1962, vol. 33, pp. 1065–1076.

[41] Jiang, Q., Aitnouri, E.M., Wang, S., and Ziou, D., *Ship Detection Using PNN Model*. In CD-ROM of ADRO'98, Montreal, 1998.

[42] Bishop, C.M., Neural Networks for Pattern Recognition, Oxford: Clarendon Press, Oxford Univ. Press, 1995.

[43] Sardo, L. and Kittler, J., *Minimum Complexity PDF Estimation for Correlated Data*. Int. Conf. on Pattern Recognition'96, Madisson, 1996, pp. 750–754.

[44] Sclove, S.L., *Application of the Conditional Populationmixture Model to Image Segmentation*, IEEE Tran. Patter. Anal. Machine Intell., 1983, vol. PAMI-5(4), pp. 428–433.

[45] Schwarz, G., *Estimating the Dimension of a Model*, The Annals of Statistics, 1978, vol. 6(3), pp. 661–664.

[46] Oliver, C., Jouzel, F., and Elmatouat, A., *Choice of the Number of Component Clusters in Mixture Models by Information Criteria*, Vision Interface VI'99, Trois- Rivieres, 1999, pp. 74–81.

[47] Tabbone, S., *Détection Multi-échelles de Contours Sous-Pixel et de Jonctions*, PhD thesis, Institut National Polytechnique de Lorraine, France, 1994.

[48] Aitnouri, E.M., Dubeau, F., Wang, S., and Ziou, D., *Generating Test Vectors for Clustering Algorithms*, Research Report 236, Dept. Math. and Comp. Scien., University of Sherbrooke, Sherbrooke, September 1999.

[49] Windham, M.P. and Cutler, A., *Information Ratios for Validating Mixture Analyses*, American Statistical Association, Theory and Methods, 1992, vol. 87(420), pp. 1188–1192.

Mohammed Ouali and Elmehdi Aitnouri, Received September 17, 2010

Mohammed Ouali
BAE Systems Canada, R&D Department
7600 Dr. Frederick Phillips Blvd, Montreal, Qc, Canada K2C 3M5
Phone: (514) 789-3000 (ext 443)
E–mail: mohammed.ouali@usherbrooke.ca

Elmehdi Aitnouri
DZScience-Sherbrooke
#1227-2820, Judge-Morin, Sherbrooke, Qc, Canada, J1E 2R6
Phone: (819) 329-4537 ext. 081
E–mail: elmehdi.aitnouri@usherbrooke.ca

Computer Science Journal of Moldova, vol.**18**, no.3(54), 2010

Perfect Octagon Quadrangle Systems with an upper C_4-system and a large spectrum *

Luigia Berardi, Mario Gionfriddo, Rosaria Rota

To the memory of our dear Lucia

Abstract

An *octagon quadrangle* is the graph consisting of an 8-cycle $(x_1, x_2, ..., x_8)$ with two additional chords: the edges $\{x_1, x_4\}$ and $\{x_5, x_8\}$. An *octagon quadrangle system* of order v and index λ $[OQS]$ is a pair (X, H), where X is a finite set of v vertices and H is a collection of edge disjoint octagon quadrangles (called *blocks*) which partition the edge set of λK_v defined on X. An *octagon quadrangle system* $\Sigma = (X, H)$ of order v and index λ is said to be *upper* $C_4 - perfect$ if the collection of all of the *upper* 4-cycles contained in the octagon quadrangles form a μ-fold 4-cycle system of order v; it is said to be *upper strongly perfect*, if the collection of all of the *upper* 4-cycles contained in the octagon quadrangles form a μ-fold 4-cycle system of order v and also the collection of all of the *outside* 8-cycles contained in the octagon quadrangles form a ϱ-fold 8-cycle system of order v. In this paper, the authors determine the spectrum for these systems, in the case that it is the largest possible.

1 Introduction

A λ-fold *m-cycle system* of order v is a pair $\Sigma = (X, C)$, where X is a finite set of n elements, called *vertices*, and C is a collection of edge disjoint m-cycles which partitions the edge set of λK_v, (the complete graph defined on a set X, where every pair of vertices is joined by λ edges). In this case, $|C| = \lambda v(v-1)/2m$. The integer number λ is also

©2010 by L. Berardi, M. Gionfriddo, R. Rota

* Lavoro eseguito nell'ambito di un progetto PRIN 2008.

called the *index* of the system. When $\lambda = 1$, we will simply say that Σ is an *m-cycle system*. Fairly recently the spectrum (the set of all v such that an *m-cycle system* of order v exists) has been determined to be [1][12]:

$$v \geq m, \text{ if } v > 1, \quad v \text{ is odd}, \quad \frac{v(v-1)}{2m} \text{ is an integer.}$$

The spectrum for λ-fold *m-cycle systems* for $\lambda \geq 2$ is still an open problem.

In these last years, G-decompositions of λK_v have been examined mainly in the case in which G is a polygon with some chords forming an inside polygon whose sides joining vertices at distance two. Many results can be found at first in [4,11,13] and after in [5,10,12]. Recently, octagon triple systems and dexagon triple systems have been studied in [3,14]. Generally, in these papers, the authors determine the spectrum of the corresponding systems and study problems of embedding.
In [6,7,8,9], Lucia Gionfriddo introduced another idea: she studied G-decompositions, in which G is a polygon with chords which determine at least a quadrangle. Further, these polygons have the property of *nesting* C_4-systems, *kite-systems*, etc... . In particular, in [8] she studied *perfect dodecagon quadrangle systems*.

In this paper, where the blocks are dodecagons with chords which join vertices at distance three dividing the dodecagon in five quadrangles, the authors study these systems in the case that the spectrum is the *largest* possible.

2 Some definitions

The graph given in the Fig.1 is called an *octagon quadrangle* and will be also denoted by $[(x_1), x_2, x_3, (x_4), (x_5), x_6, x_7, (x_8)]$. The cycle (x_1, x_2, x_3, x_4) will be the *upper* C_4-cycle, the cycle (x_5, x_6, x_7, x_8) will be the *lower* C_4-cycle, while the cycle $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$

will be the *outside* cycle. Obviously, an *upper* C_4-cycle of an octagon quadrangle OQ can be considered as a *lower* C_4-cycle of OQ and vicev-ersa. It depends only on the representation of the OQ in the plane.

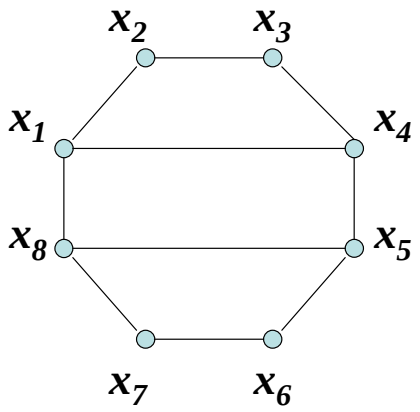

Figure 1. Octagon Quadrangle

An *octagon quadrangle system* of order v and index λ, briefly an OQS, is a pair $\Sigma = (X, B)$, where X is a finite set of v vertices and B is a collection of edge disjoint octagon quadrangles, called *blocks*, which partition the edge set of λK_v, defined on the vertex set X.

An *octagon quadrangle system* $\Sigma = (X, B)$ of order v and index λ is said to be:

i) *upper C_4-perfect*, if all of the *upper* C_4-cycles contained in the octagon quadrangles form a μ-fold 4-cycle system of order v;

ii) *C_8-perfect*, if all of the *outside* C_8-cycles contained in the octagon quadrangles form a ϱ-fold 8-cycle system of order v;

iii) *upper strongly perfect*, if the collection of all of the *upper* C_4-cycles contained in the octagon quadrangles form a μ-fold 4-cycle sys-

tem of order v and the collection of all of the *outside* C_8-cycles contained in the octagon quadrangles form a ϱ-fold 8-cycle system of order v.

In the first two cases, we say that the system has indices (λ, μ) or (λ, ϱ) respectively, in the third case we say that the system has indices (λ, ϱ, μ). It is immediate that any system of order v and index $2k$ can be obtained from a system of the same type of the same order and index k, by a repetition of the blocks.

In the following examples there are OQSs of different types. In them the vertex set is always Z_v and the blocks are given by a given number of base blocks, from which one can obtain their translated blocks and define all the system.

Example 1

The following blocks define a *strongly perfect* $OQS(11)$ of indices (10,8,4): the upper C_4-cycles form an upper C_4-system of index $\mu = 4$ and the outside C_8-cycles form a C_8-system of index $\varrho = 8$.

Base blocks (mod 11):

$$[(0), 6, 4, (10), (8), 3, 7, (1)], [(0), 10, 7, (9), (5), 3, 2, (6)],$$

$$[(0), 10, 7, (3), (9), 5, 2, (6)], [(0), 8, 3, (7), (4), 2, 10, (9)],$$

$$[(1), 0, 4, (6), (7), 5, 8, (9)].$$

Example 2

The following blocks define an upper C_4-*perfect* $OQS(8)$ of indices (10,8,4). It is not C_8-*perfect*. In fact, while the upper C_4-cycles form a C_4-system of index $\mu = 4$, the outside C_8-cycles do not form a C_8-system of index $\varrho = 8$.

Base blocks (mod 7):

$$[(0), 4, 3, (6), \ (2), 1, \infty, (5)],$$

$$[(\infty), 3, 5, (2), (4), 1, 6, (0)],$$

$$[(0, 6, 4, (5), (3), 2, \infty, (1)],$$

$$[(\infty), 2, 5, (0), (6), 1, 4, (3)],$$

where ∞ is a fixed vertex and all the others are obtained cyclically in Z_7.

Example 3

The following blocks define a C_8-*perfect* $OQS(8)$ of indices (10,8,4). It is not upper C_4-*perfect*. In fact, while the outside C_8-cycles form a C_8-system of index $\varrho = 8$, it is not possible to find upper or lower C_4-cycles which form a C_4-system of index $\mu = 4$.

Base blocks (mod 7):

$$[(1), 0, 2, (5), \ (4), 6, \infty, (3)],$$

$$[(\infty), 0, 3, (6), (4), 5, 1, (2)],$$

$$[(6), 0, 5, (2), (3), 1, \infty, (4)],$$

$$[(\infty), 0, 4, (1), (3), 2, 6, (5)].$$

where ∞ is a fixed vertex and all the others are obtained cyclically in Z_7.

Remark: It is immediate that any system of order v and index $2k$ can be obtained from a system of the same type of the same order and index k, by a repetition of the blocks. In this paper we will not use this technique and always we will consider OQSs *without* repeated blocks.

3 Necessary existence conditions

In this section we prove some necessary existence conditions.

Theorem 3.1 *: Let $\Omega = (X, B)$ be an upper strongly perfect OQS of order v and let $\Sigma_1 = (X, B_1)$, $\Sigma_2 = (X, B_2)$ be the corresponding outside $C_8 -$ system and upper $C_4 -$ system, respectively. If the systems $\Omega,\ \Sigma_1, \Sigma_2$ have indices (λ, ϱ, μ), in the order, then:*

i) $\lambda = 5 \cdot k,\ \varrho = 4 \cdot k,\ \mu = 2 \cdot k,$

for some positive integer k;

ii) the largest possible spectrum for upper strongly perfect OQSs is

$$S = \{v \in N : v \geq 8\},$$

and the corresponding minimum values for the indices are:

$$\lambda = 10, \varrho = 8, \mu = 4.$$

Proof. If $\Omega = (X, B)$ is an upper strongly perfect OQS of order v, $\Sigma_1 = (X, B_1)$ and $\Sigma_2 = (X, B_2)$ the outside $C_8 - system$ and the upper $C_4 - system$ respectively and (λ, ϱ, μ) the indices, since $|B| = |B_1| = |B_2|$, then necessarily:

$$\frac{\lambda}{5} = \frac{\varrho}{4} = \frac{\mu}{2}$$

and the statement $i)$ follows.

For $k = 1$ the possible spectrum for strongly perfect OQS is a subset of $S = \{v \in N : v \geq 8\}$. For $k = 2$ the possible spectrum is exactly $S = \{v \in N : v \geq 8\}$. □

Remark: The same conditions are obtained in the case of upper C_4-perfect $OQSs$ but not C_8-perfect, and in the case of C_8-perfect $OQSs$ but not C_4-perfect.

4 Existence of particular octagon systems of indices (10,8,4), without repeated blocks

The systems contained in the following Theorems will be used in what follows.

Theorem 4.1 *: There exist upper strongly perfect OQSs, having order 8,9,10,11,12,13,14,15 and indices (10,8,4).*

Proof. The following OQSs are upper strongly perfect. They have order 8,9,10,11,12,13,14,15 and indices (10,8,4).

i) $\Sigma_9 = (Z_9, B)$, base blocks (mod 9):

$[(0), 4, 8, (1), (5), 2, 7, (3)]$, $[(0), 1, 7, (2), (5), 4, 6, (8)]$,
$[(0), 2, 4, (3), (6), 5, 8, (1)]$, $[(0), 1, 7, (4), (6), 2, 3, (5)]$.

ii) $\Sigma_8 = (W_8, B)$, $W_8 = Z_7 \cup \{\infty\}$, $\infty \notin Z_7$, base blocks (mod 7):

$[(0), 3, 4, (1), (5), 6, \infty, (2)]$,$[(\infty), 4, 2, (5), (3), 6, 1, (0)]$,
$[(0), 1, 3, (2), (4), 5, \infty, (6)]$,$[(\infty), 5, 0, (4), (1), 6, 3, (2)]$.

iii) $\Sigma_{11} = (Z_{11}, B)$, base blocks (mod 11):

$[(0), 5, 7, (1), (3), 8, 4, (10)]$, $[(0), 1, 4, (2), (6), 8, 9, (5)]$,
$[(0), 1, 4, (8), (2), 6, 9, (5)]$, $[(0), 3, 8, (4), (7), 9, 1, (2)]$,
$[(10), 0, 7, (5), (4), 6, 3, (2)]$.

iv) $\Sigma_{10} = (W_8, B)$, $W_{10} = Z_9 \cup \{\infty\}$, $\infty \notin Z_9$, base blocks (mod 9):

$[(0), 4, 7, (1), (3), 8, 6, (5)]$, $[(0), 1, 5, (2), (3), 6, \infty, (4)]$,
$[(\infty), 5, 6, (7), (4), 2, 1, (0)]$, $[(0), 2, 7, (3), (1), 8, \infty, (4)]$,
$[(\infty), 8, 6, (4), (7), 1, 0, (3)]$.

v) $\Sigma_{13} = (Z_{13}, B)$, base blocks (mod 13):

$[(0), 6, 12, (1), (4), 11, 7, (2)]$, $[(0), 1, 12, (2), (6), 7, 4, (3)]$,
$[(0), 2, 11, (3), (8), 6, 9, (4)]$, $[(0), 3, 11, (4), (10), 8, 6, (5)]$,
$[(0), 4, 8, (5), (12), 11, 7, (6)]$, $[(0), 1, 6, (7), (4), 2, 11, (5)]$.

vi) $\Sigma_{12} = (W_{12}, B)$, $W_{12} = Z_{11} \cup \{\infty\}$, $\infty \notin Z_{11}$, base blocks (mod 11):

$[(0), 5, 10, (1), (4), 8, 3, (2)]$, $[(0), 2, 9, (3), (8), 5, \infty, (4)]$,
$[(\infty), 4, 8, (7), (5), 6, 2, (0)]$, $[(0), 1, 10, (2), (6), 7, 4, (3)]$,
$[(0), 3, 10, (4), (7), 6, \infty, (2)]$, $[(\infty), 3, 6, (5), (1), 7, 2, (0)]$.

vii) $\Sigma_{15} = (Z_{15}, B)$, base blocks (mod 15):

$[(0), 7, 13, (1), (4), 6, 5, (2)]$, $[(0), 1, 13, (2), (6), 12, 10, (3)]$,
$[(0), 2, 13, (3), (8), 14, 11, (4)]$, $[(0), 3, 13, (4), (10), 12, 6, (5)]$,
$[(0), 4, 13, (5), (12), 7, 11, (6)]$, $[(0), 5, 4, (6), (8), 12, 11, (1)]$,
$[(14), 1, 8, (7), (4), 11, 10, (3)]$.

viii) $\Sigma_{14} = (W_{14}, B)$, $W_{14} = Z_{13} \cup \{\infty\}$, $\infty \notin Z_{13}$, base blocks (mod 13):

$[(0), 6, 10, (1), (4), 5, 8, (2)]$, $[(0), 1, 12, (2), (6), 4, 9, (3)]$,
$[(0), 2, 11, (3), (8), 5, 10, (4)]$, $[(0), 3, 10, (4), (6), 2, \infty, (1)]$,
$[(0), 1, 11, (5), (10), 6, \infty, (4)]$, $[(\infty), 3, 4, (9), (8), 7, 5, (2)]$,
$[(\infty), 6, 8, (3), (1), 2, 7, (0)]$. □

Theorem 4.2 *: There exist upper C_4- perfect OQSs, having order 8,9,10,11,12,13,14,15 and indices (10,4), which are not C_8- perfect.*

Proof.

The following OQSs are upper C_4- perfect, have order 8,9,10,11, 12, 13,14,15 and indices (10,4), but they are not C_8- perfect.

i) $\Omega_9 = (Z_9, B)$, base blocks (mod 9):

$[(0), 4, 8, (1), (5), 2, 7, (3)]$, $[(0), 1, 7, (2), (5), 4, 6, (8)]$,
$[(0), 2, 4, (3), (6), 5, 8, (1)]$, $[(0), 1, 7, (4), (3), 8, 6, (5)]$.

ii) $\Omega_8 = (W_8, B)$, $W_8 = Z_7 \cup \{\infty\}$, $\infty \notin Z_7$, base blocks (mod 7):

$[(0), 3, 4, (1), (5), 6, \infty, (2)]$,$[(\infty), 4, 2, (5), (3), 6, 1, (0)]$,
$[(0), 1, 3, (2), (4), 5, \infty, (6)]$,$[(\infty), 5, 2, (0), (1), 6, 3, (4)]$.

iii) $\Omega_{11} = (Z_{11}, B)$, base blocks (mod 11):

$[(0), 5, 7, (1), (3), 8, 4, (10)]$, $[(0), 1, 4, (2), (6), 8, 9, (5)]$,
$[(0), 1, 4, (8), (2), 6, 9, (5)]$, $[(0), 3, 8, (4), (7), 9, 6, (5)]$,
$[(10), 0, 7, (5), (4), 6, 3, (2)]$.

iv) $\Omega_{10} = (W_8, B)$, $W_{10} = Z_9 \cup \{\infty\}$, $\infty \notin Z_9$, base blocks (mod 9):

$[(0), 4, 7, (1), (3), 8, 6, (5)]$, $[(0), 1, 5, (2), (3), 6, \infty, (4)]$,
$[(\infty), 5, 6, (7), (4), 2, 1, (0)]$, $[(0), 2, 7, (3), (1), 8, \infty, (4)]$,
$[(\infty), 8, 6, (4), (7), 3, 0, (1)]$.

v) $\Omega_{13} = (Z_{13}, B)$, base blocks (mod 13):

$[(0), 6, 12, (1), (4), 11, 7, (2)]$, $[(0), 1, 12, (2), (6), 7, 4, (3)]$,
$[(0), 2, 11, (3), (8), 6, 9, (4)]$, $[(0), 3, 11, (4), (10), 8, 6, (5)]$,
$[(0), 4, 8, (5), (12), 1, 2, (6)]$, $[(0), 1, 6, (7), (8), 12, 11, (5)]$.

vi) $\Omega_{12} = (W_{12}, B)$, $W_{12} = Z_{11} \cup \{\infty\}$, $\infty \notin Z_{11}$, base blocks (mod 11):

$[(0), 5, 10, (1), (4), 8, 3, (2)]$, $[(0), 2, 9, (3), (8), 5, \infty, (4)]$,
$[(\infty), 4, 8, (7), (5), 6, 2, (0)]$, $[(0), 1, 10, (2), (6), 7, 4, (3)]$,
$[(0), 3, 8, (4), (5), 10, \infty, (2)]$, $[(\infty), 3, 6, (5), (1), 7, 2, (0)]$.

vii) $\Omega_{15} = (Z_{15}, B)$, base blocks (mod 15):

$[(0), 7, 13, (1), (4), 6, 5, (2)]$, $[(0), 1, 13, (2), (6), 12, 11, (3)]$,

$[(0), 2, 13, (3), (8), 14, 11, (4)]$, $[(0), 3, 13, (4), (10), 12, 6, (5)]$,
$[(0), 4, 13, (5), (12), 7, 11, (6)]$, $[(0), 5, 4, (6), (8), 12, 11, (1)]$,
$[(14), 1, 8, (7), (6), 13, 11, (3)]$.

viii) $\Omega_{14} = (W_{14}, B)$, $W_{14} = Z_{13} \cup \{\infty\}$, $\infty \notin Z_{13}$, base blocks (mod 13):

$[(0), 6, 10, (1), (4), 5, 8, (2)]$, $[(0), 1, 12, (2), (6), 4, 9, (3)]$,
$[(0), 2, 11, (3), (8), 5, 10, (4)]$, $[(0), 3, 10, (4), (6), 2, \infty, (1)]$,
$[(0), 1, 11, (5), (10), 6, \infty, (4)]$, $[(\infty), 3, 4, (9), (8), 7, 5, (2)]$,
$[(\infty), 10, 8, (3), (2), 1, 6, (0)]$. □

Theorem 4.3 *: There exist C_8- perfect OQSs, having order 8,9,10,11, 12, 13,14,15 and indices (10,8), which are not upper C_4- perfect.*

Proof.

The following OQSs are C_8- perfect, have order 8,9,10,11,12,13,14,15 and indices (10,8), but they are not upper C_4- perfect.

i) $\Delta_9 = (Z_9, B)$, base blocks (mod 9):

$[(0), 4, 8, (1), (5), 2, 7, (3)]$, $[(0), 1, 7, (2), (5), 4, 6, (8)]$,
$[(0), 2, 4, (3), (6), 5, 8, (1)]$, $[(0), 4, 7, (5), (2), 3, 8, (1)]$.

ii) $\Delta_8 = (W_8, B)$, $W_8 = Z_7 \cup \{\infty\}$, $\infty \notin Z_7$, base blocks (mod 7):

$[(0), 6, 5, (1), (3), 2, \infty, (4)]$,$[(\infty), 6, 3, (5), (4), 1, 2, (0)]$,
$[(0), 3, 5, (2), (6), 4, \infty, (1)]$,$[(\infty), 2, 1, (4), (6), 0, 5, (3)]$.

iii) $\Delta_{11} = (Z_{11}, B)$, base blocks (mod 11):

$[(0), 5, 3, (1), (10), 4, 11, (6)]$, $[(0), 1, 4, (2), (6), 8, 9, (5)]$,
$[(0), 1, 4, (8), (2), 6, 3, (10)]$, $[(2), 5, 3, (9), (6), 10, 7, (1)]$,
$[(10), 0, 7, (5), (4), 6, 3, (2)]$.

iv) $\Delta_{10} = (W_8, B)$, $W_{10} = Z_9 \cup \{\infty\}$, $\infty \notin Z_9$, base blocks (mod 9):

$[(0), 4, 7, (1), (3), 8, 6, (5)]$, $[(0), 1, 5, (2), (3), 6, \infty, (4)]$,
$[(\infty), 5, 6, (7), (4), 2, 1, (0)]$, $[(0), 2, 7, (3), (1), 8, \infty, (4)]$,
$[(\infty), 0, 2, (8), (6), 3, 4, (1)]$.

v) $\Delta_{13} = (Z_{13}, B)$, base blocks (mod 13):

$[(0), 6, 12, (1), (4), 11, 7, (2)]$, $[(0), 1, 12, (2), (6), 7, 4, (3)]$,
$[(0), 2, 11, (3), (8), 6, 9, (4)]$, $[(0), 5, 10, (4), (11), 12, 1, (3)]$,
$[(0), 4, 8, (5), (12), 11, 7, (6)]$, $[(0), 1, 6, (7), (4), 2, 11, (5)]$.

vi) $\Delta_{12} = (W_{12}, B)$, $W_{12} = Z_{11} \cup \{\infty\}$, $\infty \notin Z_{11}$, base blocks (mod 11):

$[(0), 5, 10, (1), (4), 8, 3, (2)]$, $[(0), 2, 9, (3), (8), 5, \infty, (4)]$,
$[(\infty), 9, 10, (8), (5), 6, 2, (0)]$, $[(0), 1, 10, (2), (6), 7, 4, (3)]$,
$[(0), 3, 10, (4), (7), 6, \infty, (2)]$, $[(\infty), 8, 1, (0), (4), 10, 5, (3)]$.

vii) $\Delta_{15} = (Z_{15}, B)$, base blocks (mod 15):

$[(0), 7, 13, (1), (4), 6, 5, (2)]$, $[(2), 5, 11, (0), (4), 7, 9, (1)]$,
$[(0), 2, 13, (3), (8), 14, 11, (4)]$, $[(0), 3, 13, (4), (10), 12, 6, (5)]$,
$[(0), 4, 13, (5), (12), 7, 11, (6)]$, $[(0), 5, 4, (6), (8), 12, 11, (1)]$,
$[(14), 1, 8, (7), (4), 11, 10, (3)]$.

viii) $\Delta_{14} = (W_{14}, B)$, $W_{14} = Z_{13} \cup \{\infty\}$, $\infty \notin Z_{13}$, base blocks (mod 13):

$[(0), 6, 10, (1), (4), 5, 8, (2)]$, $[(0), 1, 12, (2), (6), 4, 9, (3)]$,
$[(0), 2, 11, (3), (8), 5, 10, (4)]$, $[(0), 3, 10, (4), (6), 2, \infty, (1)]$,
$[(0), 1, 11, (5), (10), 6, \infty, (4)]$, $[(\infty), 11, 10, (9), (1), 4, 6, (7)]$,
$[(\infty), 6, 8, (3), (1), 2, 7, (0)]$. □

5 Construction $v \to v+8$

In this section we give a construction for $OQSs$ having indices (10,8,4),(10,4),(10,8), for all possible orders.

Theorem 5.1 *: An upper strongly perfect OQS of order v+8 and indices (10,8,4) can be constructed starting from an upper strongly perfect OQS of order v and indices (10,8,4).*

Proof. Let $A = \{1', 2', 3', 4', 5', 6', 7', 8'\}$, $Z_v = \{0, 1, 2..., v-1\}$, where $A \cap Z_v = \emptyset$. Let $\Sigma = (Z_v, B)$, $\Sigma' = (A, B')$ be two upper strongly perfect OQSs both of indices $(10, 8, 4)$. Define on $Z_v \cup A$ the family H of octagon quadrangles as follows.
Define a partition of A in two sets $L = \{\alpha, \beta, \gamma, \delta\}$, $M = \{a, b, c, d\}$ such that $L \cap M = \emptyset$. Then, H is the family having the blocks:

$[(\alpha), i, \beta, (i+1), (\gamma), i+2, \delta, (i+3)]$, $[(\beta), i+1, \alpha, (i+2), (\delta), i+3, \gamma, (i+4)]$,

$[(\gamma), i, \delta, (i+1), (\alpha), i+2, \beta, (i+3)]$, $[(\delta), i+1, \gamma, (i+2), (\beta), i+3, \alpha, (i+4)]$,

$[(a), i, b, (i+1), (c), i+2, d, (i+3)]$, $[(b), i+1, a, (i+2), (d), i+3, c, (i+4)]$,

$[(c), i, d, (i+1), (a), i+2, b, (i+3)]$, $[(d), i+1, c, (i+2), (b), i+3, a, (i+4)]$,

where i belongs to Z_v.

If $X{=}Z_v \cup A$ and $C = B \cup B' \cup H$, then $\Omega = (X, C)$ is an upper strongly perfect OQS of order $v + 8$ and indices (10,8,4).
If $x, y \in Z_v$ [resp. A], then the edge $\{x, y\}$ is in a block of B [resp. B']: exactly in ten octagon quadrangles, in eight outside C_8-cycles and in four upper C_4-cycles.
If $x \in Z_v$ and $y \in A$, then the edge $\{x, y\}$ is contained in the octagon quadrangles of H. Each vertex $y \in A$ has degree 3 in $2v$ blocks and degree 2 in the other $2v$ blocks, also the edge $\{x, y\}$ is contained exactly in ten octagon quadrangles of H, in eight outside C_8-cycles and in four upper C_4-cycles.

We also observe that the number of blocks of C is:

$$|C| = |B| + |B'| + |H| =$$

$$\frac{v(v-1)}{2} + \frac{8 \cdot 7}{2} + 8 \cdot v = \frac{1}{2} \cdot (v^2 + 15v + 56),$$

which is exactly the number of blocks of an $OQS(v + 8)$ of indices (10,8,4):

$$\frac{(v+8)(v+7)}{2} = \frac{1}{2} \cdot (v^2 + 15v + 56).$$

So, the proof is complete. □

Theorem 5.2 *: An upper C_4-perfect OQS of order $v + 8$ and indices (10,4), which is not C_8-perfect, can be constructed starting from an upper C_4-perfect OQS of order v and indices (10,4).*

Proof. Let $\Sigma' = (A, B')$ be the $OQS(8)$ of indices (10,4), isomorphic to the $OQS(8)$ defined on $Z_7 \cup \{\infty\}$ and defined by the translated one of the following

base blocks (mod 7):
$[(0), 3, 4, (1), (5), 6, \infty, (2)]$,$[(\infty), 4, 2, (5), (3), 6, 1, (0)]$,
$[(0), 1, 3, (2), (4), 5, \infty, (6)]$,$[(\infty), 5, 2, (0), (1), 6, 3, (4)]$.

Following the proof of Theorem 5.1, since Σ' is an upper C_4-perfect $OQS(8)$, but not C_8-perfect (see Theorem 4.2), the statement is proved. □

Theorem 5.3 *: A C_8-perfect OQS of order $v + 8$ and indices (10,8), which is not C_4-perfect, can be constructed starting from a C_8-perfect OQS of order v and indices (10,8).*

Proof. Let $\Sigma' = (A, B')$ be the $OQS(8)$ of indices (10,8), isomorphic to the $OQS(8)$ defined on $Z_7 \cup \{\infty\}$ and defined by the translated one of the following

base blocks (mod 7):
$[(0), 6, 5, (1), (3), 2, \infty, (4)]$,$[(\infty), 6, 3, (5), (4), 1, 2, (0)]$,
$[(0), 3, 5, (2), (6), 4, \infty, (1)]$,$[(\infty), 2, 1, (4), (6), 0, 5, (3)]$.

Following the proof of Theorem 5.1, since Σ' is a C_8-perfect $OQS(8)$, but it is not upper C_4-perfect (see Theorem 4.3), the statement is proved. □

6 Conclusive Existence Theorems

Collecting together the results of the previous sections, we have the following conclusive theorems:

Theorem 6.1 *: There exist upper strongly perfect OQS(v)s of indices (10,8,4) for every positive integer v, $v \geq 8$.*

Proof. The statement follows from Theorems 4.1 and 5.1. □

Theorem 6.2 *: There exist OQS(v)s of indices (10,4), which are upper C_4-perfect but not C_8-perfect, for every positive integer v, $v \geq 8$.*

Proof. The statement follows from Theorems 4.2 and 5.2. □

Theorem 6.3 *: There exist OQS(v)s of indices (10,8), which are C_8-perfect but not upper C_4-perfect, for every positive integer v, $v \geq 8$.*

Proof. The statement follows from Theorems 4.3 and 5.3. □

References

[1] B. Alspach and H. Gavlas, *Cycle decompositions of K_n and $K_n - I$*, J. Combin Theory, Ser. B **81** (2001), 77–99.

[2] L.Berardi, M.Gionfriddo, R.Rota, *Perfect octagon quadrangle systems*, Discrete Mathematics, 310 (2010), 1979–1985.

[3] E.J.Billington, S. Kucukcifci, E.S. Yazici, C.C.Lindner, *Embedding 4-cycle systems into octagon triple systems*, Utilitas Mathematica, 79 (2009), 99–106.

[4] E.J.Billington, C.C.Lindner, *The spectrum for λ-2-perfect 6-cycle systems*, European J. Combinatorics, 13 (1992), 5–14.

[5] L.Gionfriddo, *Two constructions for perfect triple systems*, Bull. of ICA, 48 (2006), 73–81.

[6] L.Gionfriddo, *Hexagon quadrangle systems*, Discrete Maths. 309 (2008), 231–241.

[7] L.Gionfriddo, *Hexagon biquadrangle systems*, Australasian J. of Combinatorics 36 (2007), 167–176.

[8] L.Gionfriddo, *Hexagon kite systems*, Discrete Mathematics, 309 (2009), 505–512.

[9] L.Gionfriddo, *Perfect dodecagon quadrangle systems*, Discrete Mathematics, to appear.

[10] S. Kucukcifci, C.C.Lindner, *Perfect hexagon triple systems*, Discrete Maths., 279 (2004), 325–335.

[11] C.C.Lindner, *2-perfect m-cycle systems and quasigroup varieties: a survey*, Proc. 24th Annual Iranian Math. Conf., 1993.

[12] C.C.Lindner, G.Quattrocchi, C.A.Rodger, *Embedding Steiner triple systems in hexagon triple systems*, Discrete Maths., to appear.

[13] C.C.Lindner, C.A.Rodger, *2-perfect m-cycle systems*, Discrete Maths. 104 (1992), 83–90.

[14] C.C.Lindner, A.Rosa, *Perfect dexagon triple systems*, Discrete Maths. 308 (2008), 214–219.

[15] M.Sayna, *Cycle decomposition III: complete graphs and fixed length cycles*, J. Combinatorial Theory Ser.B, (to appear).

L. Berardi, M. Gionfriddo, R. Rota, Received January 12, 2011

Luigia Berardi
Dipartimento di Ingegneria Elettrica e dell'Informazione,
Universitá di L'Aquila
E–mail: *luigia.berardi@ing.univaq.it*

Mario Gionfriddo
Dipartimento di Matematica e Informatica,
Universitá di Catania
E–mail: *gionfriddo@dmi.unict.it*

Rosaria Rota
Dipartimento di Matematica, Universitá di RomaTre
E–mail: *rota@mat.uniroma3.it*

Computer Science Journal of Moldova, vol.**18**, no.3(54), 2010

Information encryption systems based on Boolean functions

Aureliu Zgureanu

Abstract

An information encryption system based on Boolean functions is proposed. Information processing is done using multidimensional matrices, performing logical operations with these matrices. At the basis of ensuring high level security of the system the complexity of solving the problem of building systems of Boolean functions that depend on many variables (tens and hundreds) is set. Such systems represent the private key. It varies both during the encryption and decryption of information, and during the transition from one message to another.

Keywords: Boolean functions, multidimensional matrices, private keys, security of the system, the complexity of the problem.

1 Introduction

The most popular information encryption systems (IES), based on prime numbers, are shown in [1, 2]. In [3], using ideas from [1], there is proposed a new encryption algorithm, which considerably increases resistance to breakage, keeping the speed encryption and decryption. In [4] there has been proposed another encryption system with a cryptographic power not smaller than those two shown in [1,3], but, at the same time, with an encrypting-decrypting time much smaller. Together with the improving of computing means, the requirements towards IES also increase. Public keys and those private become bigger and bigger, and arithmetic operations with very big numbers become more difficult. As a result, the productivity of the systems decreases considerably. The

©2010 by A. Zgureanu

situation may be changed if we replace these arithmetic operations with logical operations on systems of Boolean functions, represented by multidimensional matrices [4]. Such a solution to the problem is proposed in this paper.

2 Sets of relations and multidimensional matrixes

In accordance with [11], a system A of $N_A = n_1 n_2 n_3 \cdot ... \cdot n_p$ elements $a_{i_1 i_2 i_3 ... i_p}$ $(i_\alpha = 1, 2, 3, ..., n_\alpha;\ \alpha = 1, 2, 3, ..., p)$ that belong to the set Ω and are placed in the points of p-dimensional space of coordinates $i_1, i_2, ..., i_p$ is called a *multidimensional matrix* over the set Ω. The number p is called the *size of the matrix* and shows the number of indexes in the notation of the matrix elements. Size N_A shows the total number of elements in this matrix. Size n_α of the index i_α shows how many values (from 1 to n_α) this index runs. So in this paper, a multidimensional matrix is a direct generalization of the usual two-dimensional matrix.

Consider a family of sets $X = \{X_1, X_2, ..., X_n\}$, where $X_i = \{x_{i1}, x_{i2}, ..., x_{i\lambda_i}\}$, $i = \overline{1, n}$ and the set $\Omega = \{\omega_1, ..., \omega_r\}$ with arbitrary elements (in our case – integer numbers). There are k relations $R_j = R_{X_{j_1} ... X_{j_{d_j}}}$ $(2 \le d_j \le n, j = \overline{1, k},\ j_1, j_2, ..., j_{d_j} \in \{1, 2, ..., n\})$ defined on this family as subsets of Cartesian products $X_{j_1} \times X_{j_2} \times ... \times X_{j_{d_j}}$. The matrixes of these relations are d_j-dimensional with elements from Ω. Let's mark by $\overrightarrow{R}$ the vector with components R_j, that is $\overrightarrow{R} = (R_1, ..., R_j, ..., R_k)$. Let's correlate the following n-dimensional matrix to this vector:

$$A_R = \Phi(\vec{R}). \tag{1}$$

The elements of this matrix are denoted by $a_{s_1 ... s_\tau ... s_n}$. Let's explain how these elements are obtained.

We build the Cartesian product $X_1 \times X_2 \times ... \times X_n = \{x_{11}, ..., x_{1\lambda_1}\} \times ... \times \{x_{n1}, ..., x_{n\lambda_n}\}$, which obviously contains $u = \lambda_1 \cdot ... \cdot \lambda_n$ elements.

With these elements compose a two-dimensional matrix with u rows and n columns (Figure 1, left side).

$$A_R: \quad \begin{array}{c} \\ 1 \\ \vdots \\ i \\ \vdots \\ u \end{array} \begin{array}{c} \begin{array}{cccccc} X_1 & \cdots & X_\tau & \cdots & X_n & \end{array} \\ \begin{pmatrix} x_{11} & \cdots & x_{\tau 1} & \cdots & r_{n1} \\ & & \ddots & & \\ x_{1s_1} & \cdots & x_{\tau s_\tau} & \cdots & x_{ns_n} \\ & & \ddots & & \\ x_{1\lambda_1} & \cdots & x_{\tau\lambda_\tau} & \cdots & x_{n\lambda_n} \end{pmatrix} \end{array} \begin{array}{c} \begin{array}{cccccc} R_1 & \cdots & R_j & \cdots & R_k \end{array} \\ \begin{pmatrix} r_{11} & \cdots & r_{1j} & \cdots & r_{1k} \\ & & \ddots & & \\ r_{i1} & \cdots & r_{ij} & \cdots & r_{ik} \\ & & \ddots & & \\ r_{u1} & \cdots & r_{uj} & \cdots & r_{uk} \end{pmatrix} \end{array}$$

Figure 1.

Compose another two-dimensional matrix $||r_{ij}||$ with u rows and k columns (Figure 1, right side), where $r_{ij} = r_{s_{j_1}\ldots s_{j_d}}$ with elements $s_{j_1}, ..., s_{j_d}$ selected from line i at the places $j_1, ..., j_d$, that indicate the sets $X_{j_1}, ..., X_{j_d}$ where relation R_j is defined.

For simplicity replace the element $x_{\tau s_\tau}$ with its second index as it is shown in Figure 2. The lines of the matrix in the left side of Figure 2 represent indices of the matrix A_R elements. The lines of the matrix in the right side of Figure 2 form the elements of matrix A_R:

$$a_{s_1\ldots s_\tau\ldots s_n} = (r_{i1}, ..., r_{ij}, ..., r_{ik}), \tag{2}$$

To the vector (2) there is associated a number c_i in the base y which satisfies the condition $y > \max \omega_h$, $h = \overline{1, r}$:

$$c_i = r_{i1}y^{k-1} + ... + r_{ij}y^{k-j} + ... + r_{ik} = \sum_{j=1}^{k} r_{ij}y^{k-j}, \ i = \overline{1, u}. \tag{3}$$

So, we obtain the vector

$$\vec{c} = (c_1, ..., c_i, ..., c_u). \tag{4}$$

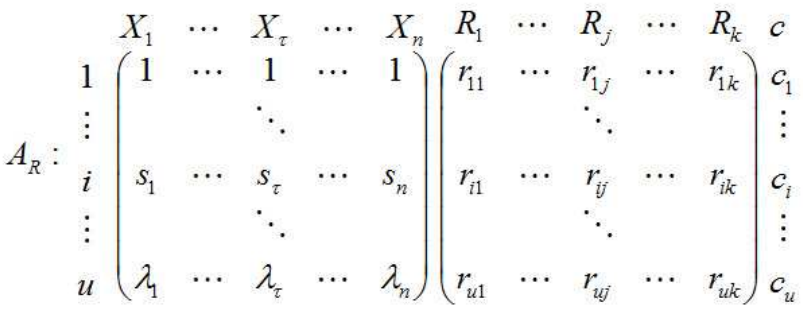

Figure 2.

Thus, using the transformation (1), the vector $\vec{c}$ (3), (4) is put into correspondence to the vector $\vec{R}$. The reverse transformation

$$\vec{R} = \Phi^{-1}(\vec{c}), \tag{5}$$

generally is much more complicated [5], [7], [8].

In some particular cases we can find vector $\vec{R}$ coordinates by vector $\vec{c}$ coordinates. This was achieved when investigating of the distribution of prime numbers in the range of integer numbers. As the result an algorithm for prime numbers generating has been elaborated [9], [10].

If the transformation (5) is difficult we can use this when elaborating the IES.

3 Information encryption systems

We consider a particular case of the exposed above, i.e. $X_1 = X_2 = ... = X_n = \Omega = \{0, 1\}$. We denote the relations defined on these sets by $M_j = M_{X_{j_1}...X_{j_{d_j}}}$ $(2 \leq d_j \leq n,\ j = \overline{1,\ k},\ j_1, j_2, ..., j_{d_j} \in \{1, 2, ..., n\})$, thus obtaining the vector $\vec{M} = (M_1, ..., M_j, ..., M_k)$. Let's correlate an n-dimensional matrix $A_M = \Phi(\vec{M})$, $i = \overline{0, u}$, $j = \overline{1, k}$ [5], presented at Figure 3, to this vector. In this matrix $M_j = M_{X_\tau...X_n}$ and

$m_{ij} = m_{\sigma_\tau \dots \sigma_n} \in \{0, 1\}$. Therefore, this matrix represents a system of k Boolean functions with variables $x_1, ..., x_n$. We correlate the following vector to this matrix:

$$\vec{m} = (m_0, ..., m_i, ..., m_t), t \leq u, \text{ where } m_i = \sum_{j=1}^{k} m_{ij} \cdot 2^{k-j}, i = \overline{0, t}, \quad (6)$$

$$n = \lceil \log_2 t \rceil, k = \lceil \log_2 \max m_i \rceil \quad (7)$$

$$A_R : \begin{array}{c} \\ 0 \\ \vdots \\ i \\ \vdots \\ u \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & \cdots & x_\tau & \cdots & x_n & \end{array} \\ \begin{pmatrix} 0 & \cdots & 0 & \cdots & 0 \\ & & \ddots & & \\ \sigma_1 & \cdots & \sigma_\tau & \cdots & \sigma_n \\ & & \ddots & & \\ 1_1 & \cdots & 1 & \cdots & 1 \end{pmatrix} \end{array} \begin{array}{c} \begin{array}{ccccc} M_1 & \cdots & M_j & \cdots & M_k \end{array} \\ \begin{pmatrix} m_{01} & \cdots & m_{0j} & \cdots & m_{0k} \\ & & \ddots & & \\ m_{i1} & \cdots & m_{ij} & \cdots & m_{ik} \\ & & \ddots & & \\ m_{u1} & \cdots & m_{uj} & \cdots & m_{uk} \end{pmatrix} \end{array} \begin{array}{c} m \\ m_0 \\ \vdots \\ m_i \\ \vdots \\ m_u \end{array}$$

Figure 3.

By analogy (Figure 4) we create another matrix A_D to which we correlate a vector

$$\vec{d} = (d_0, ..., d_i, ..., d_t), t \leq u, \text{ where } d_i = \sum_{j=1}^{k} d_{ij} \cdot 2^{k-j}, i = \overline{0, t}. \quad (8)$$

We may perform logical operations with these matrixes: $A_M \wedge A_D$, $A_M \vee A_D$, $A_M \oplus A_D$ and other, as the result we obtain other matrixes. Let's analyze the operation $\oplus$ (sum modulo 2). Suppose that $A_M \oplus A_D = A_C$. In this case $c_{ij} = m_{ij} \oplus d_{ij}$. Taking into account properties of this operation, we obtain:

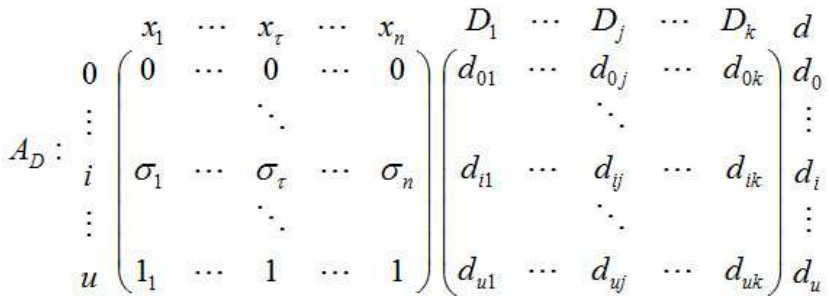

Figure 4.

$$(A_M \oplus A_D) \oplus A_D = A_M \oplus (A_D \oplus A_D) = A_M.$$

Thus

$$A_M \oplus A_D = A_C,\ A_C \oplus A_D = A_M. \tag{9}$$

From (9) it results that the matrix A_D may serve as private key for encryption and decryption of vector $\vec{m}$ which is the ASCII encoding (or any other encoding) of the plaintext M through vector $\overrightarrow{c}$ (ciphertext)

$$\overrightarrow{c} = (c_0, ..., c_i..., c_t),\ t \leq u,$$
$$\text{where } c_i = \sum_{j=1}^{k} c_{ij} \cdot 2^{k-j},\ i = \overline{0,t},\ c_{ij} = m_{ij} \oplus d_{ij}. \tag{10}$$

Let's see how we may create the private key. Suppose that the function is defined by veracity table (see Table 1), where $\varepsilon_0, ..., \varepsilon_u \in \{0, 1\}$.

Let's create the partition $\{\tilde{X}_1, \tilde{X}_2\} = \{\{x_1, ..., x_\tau\}, \{x_{\tau+1}, ..., x_n\}\}$ on set $x = \{x_1, ..., x_n\}$. We create two sets:

- $Y = \{y_0, y_1, ..., y_p, ..., y_{2^\tau - 1}\}$ (formed of binary states that correspond to variables from $\tilde{X}_1$);

Table 1.

	x_1	$\cdots$	x_τ	$\cdots$	x_n	$F(x_1,...,x_n)$
0	0	$\cdots$	0	$\cdots$	0	ε_0
$\vdots$			$\ddots$			$\vdots$
i	σ_1	$\cdots$	σ_τ	$\cdots$	σ_n	ε_i
$\vdots$			$\ddots$			$\vdots$
u	1	$\cdots$	1	$\cdots$	1	ε_u

- and $Z = \{z_0, ..., z_q, ..., z_{2^{n-\tau}-1}\}$ (formed of binary states that correspond to variables from $\tilde{X}_2$).

Then, the Boolean function $F(x_1, ..., x_n)$ may be considered as a binary relation R_{YZ} between the sets Y and Z with the matrix

$$R_{YZ} = \begin{array}{c} \\ y_0 \\ \vdots \\ y_i \\ \vdots \\ y_h \end{array} \begin{array}{c} \begin{array}{ccccc} z_0 & \cdots & z_j & \cdots & z_s \end{array} \\ \left[\begin{array}{ccccc} a_{00} & \cdots & a_{0j} & \cdots & a_{0s} \\ & & \ddots & & \\ a_{i0} & \cdots & a_{ij} & \cdots & a_{is} \\ & & \ddots & & \\ a_{h0} & \cdots & a_{hj} & \cdots & a_{hs} \end{array}\right] \end{array}, h = 2^\tau - 1, s = 2^{n-\tau} - 1,$$

$$\forall i, j a_{ij} = \begin{cases} 1, \text{ if } F(y_i, z_j) = 1, \\ 0, \text{ if } F(y_i, z_j) = 0. \end{cases}$$

According to [6], the subset $S^{z_j}_{F^\varepsilon}$ of the set Y is called ***subset of column*** of the function $F(x_1, ..., x_n)$ for the column z_j and is composed of the elements y_i for which $a_{ij} = \varepsilon$, $\varepsilon \in \{0, 1\}$.

The Boolean function may be defined by the *table of subsets of column* (see Table 2):

It is obvious that $S^{z_j}_{F^0} = Y \backslash S^{z_j}_{F^1}$. Because of this, the subsets $S^{z_j}_{F^0}$ are not indicated in the Table 2. We create partitions $\pi_{x_1}, ..., \pi_{x_\tau}$ on the set Y [6].

Table 2.

	z_0	...	z_j	...	z_s
F^1	$S_{F^1}^{z_0}$	...	$S_{F^1}^{z_j}$	...	$S_{F^1}^{z_s}$

Let's consider a specific case: n=5, $\tau = 3$ (see Table 3). In this case $\tilde{X}_1 = \{x_1, x_2, x_3\}$, $\tilde{X}_2 = \{x_4, x_5\}$ and $Y = \{y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7\} = \{000, 001, 010, 011, 100, 101, 110, 111\}$, $Z = \{z_0, z_1, z_2, z_3\} = \{00, 01, 10, 11\}$. We create the Table 3 ($\varepsilon_0, ..., \varepsilon_{31} \in \{0, 1\}$) and the partitions $\pi_{x_i} = \{\bar{m}_i^0; \bar{m}_i^1\}$, $i = \overline{1,3}$ according to the following conditions:

$$y_j \in \bar{m}_i^{\sigma_i}, \text{ if } x_i = \sigma_i \tag{11}$$

$$\pi_{x_1} = \{\overline{y_0, y_1, y_2, y_3}\,{}_1^0; \overline{y_4, y_5, y_6, y_7}\,{}_1^1\}, \pi_{x_2} = \{\overline{y_0, y_1, y_4, y_5}\,{}_2^0; \overline{y_2, y_3, y_6, y_7}\,{}_2^1\},$$

$$\pi_{x_3} = \{\overline{y_0, y_2, y_4, y_6}\,{}_3^0; \overline{y_1, y_3, y_5, y_7}\,{}_3^1\}.$$

Table 3.

		x_4x_5			
		z_0	z_1	z_2	z_3
	$x_1\,x_2\,x_3$	00	01	10	11
y_0	0 0 0	ε_0	ε_1	ε_2	ε_3
y_1	0 0 1	ε_4	ε_5	ε_6	ε_7
y_2	0 1 0	ε_8	ε_9	ε_{10}	ε_{11}
y_3	0 1 1	ε_{12}	ε_{13}	ε_{14}	ε_{15}
y_4	1 0 0	ε_{16}	ε_{17}	ε_{18}	ε_{19}
y_5	1 0 1	ε_{20}	ε_{21}	ε_{22}	ε_{23}
y_6	1 1 0	ε_{24}	ε_{25}	ε_{26}	ε_{27}
y_7	1 1 1	ε_{28}	ε_{29}	ε_{30}	ε_{31}

To simplify this, we replace elements y_i by their indexes i. Thus, such partitions are obtained:

$$\pi_{x_1} = \{\overline{0,1,2,3}\,{}^{0}_{1}; \overline{4,5,6,7}\,{}^{1}_{1}\}, \pi_{x_2} = \{\overline{0,1,4,5}\,{}^{0}_{2}; \overline{2,3,6,7}\,{}^{1}_{2}\},$$

$$\pi_{x_3} = \{\overline{0,2,4,6}\,{}^{0}_{3}; \overline{1,3,5,7}\,{}^{1}_{3}\}.$$

Let's mark by $\bar{m}^{\sigma_i, ..., \sigma_j, ..., \sigma_p}_{i, ..., j, ..., p}$ the bloc of product of partitions $\pi_{x_i}, ..., \pi_{x_j}, ..., \pi_{x_p}$, where $\sigma_j = 0\,(1)$ if the elements of this bloc belong to the bloc $\bar{m}^0_j$ ($\bar{m}^1_j$) for $j = \overline{i,p}$. We also mark the indicated partitions product by $\pi_{x_i,..., x_j,..., x_p}$. For partitions above we get the following products:

$$\pi_{x_1,x_2} = \{\overline{0,1}^{0,0}_{1,2}; \overline{2,3}^{0,1}_{1,2}; \overline{4,5}^{1,0}_{1,2}; \overline{6,7}^{1,1}_{1,2}\},$$

$$\pi_{x_1,x_3} = \{\overline{0,2}^{0,0}_{1,3}; \overline{1,3}^{0,1}_{1,3}; \overline{4,6}^{1,0}_{1,3}; \overline{5,7}^{1,1}_{1,3}\},$$

$$\pi_{x_2,x_3} = \{\overline{0,4}^{0,0}_{2,3}; \overline{1,5}^{0,1}_{2,3}; \overline{2,6}^{1,0}_{2,3}; \overline{3,7}^{1,1}_{2,3}\},$$

$$\pi_{x_1,x_2,x_3} = \{\overline{0}^{0,0,0}_{1,2,3}; \overline{1}^{0,0,1}_{1,2,3}; \overline{2}^{0,1,0}_{1,2,3}; \overline{3}^{0,1,1}_{1,2,3}; \overline{4}^{1,0,0}_{1,2,3}; \overline{5}^{1,0,1}_{1,2,3}; \overline{6}^{1,1,0}_{1,2,3}; \overline{7}^{1,1,1}_{1,2,3}\}.$$

The Table 2 is obtained when the function is given by veracity table. This table may be also obtained in the case when the function is given in analytical form, for instance in disjunctive normal form:

$$F(x_1, ..., x_n) = u_1 \vee ... \vee u_i \vee ... \vee u_e,$$

where $u_i = x_{i_1}^{\sigma_{i_1}} \wedge ... \wedge x_{i_a}^{\sigma_{i_a}}$, $i_1, i_2, ..., i_a \in \{1, ..., n\}$, $\sigma_{i_1}, ..., \sigma_{i_a} \in \{0, 1\}$, $i = \overline{1,e}$.

There may be distinguished the following 3 cases:

a) $x_{i_1}, ..., x_{i_a} \in \tilde{X}_1$

In this case u_i doesn't depend on variables $x_{\tau+1}, ..., x_n$ and, therefore, the subsets of column are equal and are formed of the elements of the bloc $\bar{m}^{\sigma_{i_1} \dots \sigma_{i_\alpha}}_{i_1 \dots i_\alpha}$ [6]:

$$S^{z_0}_{u_i^1} = ... = S^{z_s}_{u_i^1} = \bar{m}^{\sigma_{i_1} \dots \sigma_{i_\alpha}}_{i_1 \dots i_\alpha}$$

b) $x_{i_1}, ..., x_{i_a} \in \tilde{X}_2$

Taking into account the property

$$u_i = \begin{cases} 1, \text{ if } \forall x_{i_t} \in \{x_{i_1}, ..., x_{i_a}\},\ x_{i_t} = \sigma_{i_t}, \\ 0, \text{ if } \exists\, x_{i_t} \in \{x_{i_1}, ..., x_{i_a}\},\ x_{i_t} \neq \sigma_{i_t} \end{cases}$$

and the definition of the subset of column we get:

$$S^{z_j}_{u_i^1} = \begin{cases} Y, \text{ if for } \forall x_{i_t} \in \{x_{i_1}, ..., x_{i_a}\},\ x_{i_t} = \sigma_{i_t}, \\ \emptyset, \text{ if } \exists\, x_{i_t} \in \{x_{i_1}, ..., x_{i_a}\},\ x_{i_t} \neq \sigma_{i_t}; \end{cases}$$

c) $x_{i_1}, ..., x_{i_b} \in \tilde{X}_1,\ x_{i_{b+1}}, ..., x_{i_a} \in \tilde{X}_2$

In this case

$$S^{z_j}_{u_i^1} = \begin{cases} \bar{m}^{\sigma_{i_1} \dots \sigma_{i_s}}_{i_1 \dots i_s} \text{ if for } \forall x_{i_t} \in \{x_{i_{s+1}}, ..., x_{i_b}\},\ x_{i_t} = \sigma_{i_t}, \\ \emptyset \text{ if } \exists\, x_{i_t} \in \{x_{i_{s+1}}, ..., x_{i_b}\}, \text{ for which } x_{i_t} \neq \sigma_{i_t} \text{ holds.} \end{cases}$$

Considering every conjunction as a Boolean function, we get their subsets of column according to the cases mentioned above. These subsets are given in Table 4. The subsets of column of the given function are obtained in last line. They represent the union of the subsets from every column.

As any analytical form of Boolean function may be reduced to the form (11), then any function given in analytical form may be represented by the table of subsets of column.

The representation of Boolean function by subsets of column gives us the possibility to create the private key in a compact form. Suppose that functions $F_1, ..., F_j, ..., F_k$ with values from the respective columns from the Figure 4 correspond to relations $D_1, ..., D_j, ..., D_k$.

Table 4.

	z_0	...	z_j	...	z_s
u_1^1	$S_{u_1^1}^{z_0}$	...	$S_{u_1^1}^{z_j}$	...	$S_{u_1^1}^{z_s}$
u_2^1	$S_{u_2^1}^{z_0}$	...	$S_{u_2^1}^{z_j}$	...	$S_{u_2^1}^{z_s}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_e^1	$S_{u_e^1}^{z_0}$	...	$S_{u_e^1}^{z_j}$	...	$S_{u_e^1}^{z_s}$
	$S_{F^1}^{z_0}=\bigcup_{i=1}^{e} S_{u_i^1}^{z_0}$	...	$S_{F^1}^{z_j}=\bigcup_{i=1}^{e} S_{u_i^1}^{z_j}$	...	$S_{F^1}^{z_s}=\bigcup_{i=1}^{e} S_{u_i^1}^{z_s}$

Consider functions F_j for which the following conditions are achieved:

$$S_{F_j^1}^{z_0} = S_{F_j^1}^{z_1} = ... = S_{F_j^1}^{z_s} = S_j,\ j = \overline{1,k}.$$

For all the values of j we'll get the vector $\vec{S} = (S_1, ..., S_j, ..., S_k)$ – private key. Suppose that the values of the first τ variables in the index i of d_i form the binary state $\sigma_1 ... \sigma_\tau = y_q$ (Figure 4). Thereby, according to the definition of the subset of column, the values of d_{ij} are obtained from the relation

$$d_{ij} = \begin{cases} 1, \text{if } y_q \in S_j \\ 0, \text{if } y_q \notin S_j \end{cases} \tag{12}$$

Thus, the vector $\vec{S}$ determines univocally the matrix A_D. The subsets S_j are chosen on condition that

$$\bigcup_{j=1}^{k} S_j = Y \tag{13}$$

This condition assures changing the components of the vector $\vec{m}$ through vector $\vec{d}$. Relation (12) assures a rapid calculation of function value on binary state $i = \sigma_1 ... \sigma_\tau ... \sigma_n$.

As $|Y| = 2^\tau$, then for a single function we may create 2^{2^τ} subsets of column, and for k functions we have

$$\lambda = 2^{k \cdot 2^\tau}$$

different keys. As a result, the security of the private key may be chosen by parameter τ and subsets S_j.

According to those mentioned, in the computational software program *Mathematica 6*, there has been elaborated an IES composed of:

- ***key generator,*** which generates vector $\vec{S} = (S_1, ..., S_j, ..., S_k)$. The components S_j are selected randomly as subsets of the set Y and on condition (13). Using (12) and (8) we create vector $\vec{d} = (d_0, ..., d_t)$;

- ***codifier,*** which creates vector $\vec{c} = (c_0, ..., c_t)$ (on the basis of vectors $\vec{m}$ and $\vec{d}$ using (10)), codifies vector $\vec{S} = (S_1, ..., S_j, ..., S_k)$ with the help of the system from [4] or other secure system, and, concatenating it with vector $\vec{c}$, creates vector $\vec{g}$ [4];

- ***decoder,*** which restores the vector $\vec{S} = (S_1, ..., S_j, ..., S_k)$ from the vector $\vec{g}$, creates vector $\vec{d}$ using (8) and (12), creates vector $\vec{m}$ on the basis of the vectors $\vec{c} = (c_0, ..., c_t)$ and $\vec{d} = (d_0, ..., d_t)$ using (9). The initial text is printed on the basis of the vector $\vec{m}$.

Some data concerning functioning of this system (Cripto 3) in comparison with the system RSA are brought in the Table 5. We notice that for RSA both the encrypting and decrypting time grow almost linearly. Beginning with $t = 100000$ the system already meets some difficulties in creating vector $\vec{c}$ because of its too big components. This fact is marked in the Table 5 by symbol ∞. The data correspond to the public key of 2057 bits. The time grows much slowly for the system Cripto 3 and as a result it manages handling messages that contain millions of symbols, and, in the same time, has a very high security.

Table 5.

Encrypting systems	Number of symbols	Encrypting time (sec.)	Decrypting time (sec.)
RSA	100	0.34	5.34
Cripto3		0.31	0.07
RSA	500	0.48	26.90.
Cripto3		0.36	0.08
RSA	1000	0.93.	53.66
Cripto3		0.46	0.10
RSA	10000	9.60	533.40
Cripto3		1.01	0.51
RSA	100000	∞	∞
Cripto3		14.67	5.46
RSA	500000	∞	∞
Cripto3		76.23	34.37
RSA	1000000	∞	∞
Cripto3		82.62	67.20
RSA	2000000	∞	∞
Cripto3		192.06	193.23
RSA	4000000	∞	∞
Cripto3		582.75	431.90

For instance, if $t = 1000000$, then $k = 14$. Consider $\tau = 4$ and, therefore, $\lambda = 2^{224}$. This number is bigger than the number of atoms in the galaxy.

More than that, the key is the variable one. It changes both from one message to another and during the information encrypting. It changed 334 times in the case mentioned above. The data from Table 5 were got using *Athlon (tm) Processor3500*.

This system may be generalized for the case when the functions $F_1, ..., F_j, ..., F_k$ are from q-valent logics. In such a case, both variables $x_1, ..., x_n$ and functions F_j admit values from the set $\Omega = \{0, 1, ..., q-1\}$.

In the Table 1 we have $u = q^n - 1$ for these functions and the last state has the form $q-1...q-1...q-1$. In Figure 3 $m_{ij} \in \Omega$ and in Figure 4 $d_{ij} \in \Omega$. These matrices represent systems of q-valent functions. The formulas (14), (15) and (16) correspond respectively to the formulas (6), (7) and (8):

$$\vec{m} = (m_0, ..., m_i, ..., m_t),\ t \leq u,$$
$$\text{where } \ m_i = \sum_{j=1}^{k} m_{ij} \cdot q^{k-j}, i = \overline{0,t}, \tag{14}$$

$$n = \left\lceil \log_q t \right\rceil,\ k = \left\lceil \log_q \max m_i \right\rceil, \tag{15}$$

$$\vec{d} = (d_0, ..., d_i, ..., d_t),\ t \leq u,\ \text{where } \ d_i = \sum_{j=1}^{k} d_{ij} \cdot q^{k-j}, i = \overline{0,t}. \tag{16}$$

Let's create a new matrix $A_C = A_M + A_D(\text{mod } q)$, where $c_{ij} = m_{ij} + d_{ij}(\text{mod } q)$. Since for q matrices A_D the following relation holds:

$$\overbrace{A_D + A_D + ... + A_D}^{q\ times}(\text{mod } q) = 0\,(\text{zero matrix}),$$

then the equalities (17) and (18) correspond respectively to equalities (9) and (10):

$$A_M + A_D(\text{mod } q) = A_C,\ A_C + (q-1)A_D(\text{mod } q) = A_M, \tag{17}$$

$$\vec{c} = (c_0, ..., c_i..., c_t),\ t \leq u,$$
$$\text{where } c_i = \sum_{j=1}^{k} c_{ij} \cdot q^{k-j},\ \ i = \overline{0,t}\,, c_{ij} = m_{ij} + d_{ij}(\text{mod } q). \tag{18}$$

From (18) it results that if for encrypting the vector $\vec{m}$ we apply the matrix A_D, then for decrypting this vector we apply the matrix $(q-1)A_D$.

From (16) it results that components d_h of the vector $\vec{d}$ belong to the set $\{1, ..., q^k-1\}$ (0 is not included in this set because the state 0....0 doesn't change the components of the vector $\vec{m}$). In order to create this vector we take the last τ variables from the set $\{x_1, ..., x_{n-\tau}, ..., x_n\}$, choose randomly q^τ numbers from the set $\{1, ..., q^k-1\}$ and create the following vector with these numbers:

$$\vec{d} = (d_0, ..., d_h, ..., d_{q^\tau-1}), \quad q^\tau - 1 \leq t,$$

which represents the private key. Components d_h may be repeated an arbitrary number of times. Thereby, the number of different private keys is

$$\lambda = (q^k - 1)^{q^\tau}.$$

Using (14) and (18) we create the vector $\vec{c}$. It results from (17) that

$$m_i = \sum_{j=1}^{k} (c_{ij} + (q-1)d_{ij}) (\text{mod } q) q^{k-j}, i = \overline{0,t}.$$

For the examined case, in the computation software *Mathematica* 6, there was also elaborated an encryption system with a higher speed, depending on q and τ values. For example, the encrypting and decrypting time for $t = 2000000$, $q = 3$, $\tau = 4$ is equal to 130.62 *sec* and 127.45 *sec* respectively in comparison with 192.06 and 193.23 (see Table 5). Generally, a deeper investigation is needed to determine the optimal values for parameters q, τ and t.

For $q > 2$ the private key may be also represented by subsets of column. For this case, in the vector $\vec{S} = (S_1, ..., S_j, ..., S_k)$, every component S_j represents sets of form $\{\{S_j^1\}, \{S_j^2\}, ..., \{S_j^{q-1}\}\}$, where S_j^ε is a subset of the set $\{0, 1, ..., q^\tau - 1\}$, for $\forall \varepsilon \in \{1, ..., q-1\}$, and $S_j^k \bigcap S_j^s = \emptyset$ occurs for $\forall k, s \in \{1, ..., q-1\}$. But, together with the growth of q, there appear difficulties concerning the representation and transmitting of private key. Additional investigations are needed here.

4 Conclusions

1. The elaborated system has information processing speed much higher and also a capacity of solving the problems of much bigger dimensions in comparison with existent encryption systems. The priorities of the system have been highlighted during its testing with vectors that contain hundreds, thousands and millions of components.

2. Due to the fact that the system can operate with small numbers, it may be easily created using different programming languages.

3. The system may be improved using functions with q-valent logics. Deeper investigations are needed in order to achieve this.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* CACM, 21(2), February 1978, pp. 120–126.

[2] El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE TRANS. On inform. Theory, vol. IT-31. pp. 469–472, July 1985.

[3] Bulat M., Zgureanu A., Ciobanu I., Bivol L, *Encryption systems with vector keys.* International scientific conference "Mathematical modeling, optimization and information technologies", Chişinău, 9-21 martie, 2008. ATIC. pp. 281–285.

[4] Bulat M.S., Zgureanu A.F., Chobanu Ya.I., Bivol L.G., *Encryption systems based on n-ary relations.* Systems of management, control and measuring (UKI-08), Russian Conference with international participation, Moscow IPU RAN, 2008. pp. 66–67.

[5] M. Bulat, *Some applications of multidimensional matrices.* Annals of ATIC-2002, v.I (II), pp. 75–82.

[6] M. Bulat, *About one method of Boolean functions differentiation.* Annals of ATIC-2001, v.I (I), pp. 40–47.

[7] M. Bulat M, *Isomorfismo de grandes sistemas.* Acta Academia 2001, Evrica, Chiinu, pp. 161–170.

[8] M. Bulat, A. Zgureanu, I. Ciobanu, L. Bivol, *The inverse transformations of multidimensional matrices.* ASADE Moldova, August 21, 2007, p. 34.

[9] M. Bulat, D. Leon, A. Zgureanu, I. Ciobanu, L. Bivol, *Generadores de numeros primos y factorizadores de numeros compuestos.* Revista de Matematica: Teoria y Aplicaciones, 2006, 13(1) CIMPA-UCR-CCSS: pp.1–15.

[10] M. Bulat, A. Zgureanu, I. Ciobanu, L. Bivol, *Generating of prime numbers based on the multidimensional matrices.* Intern. Algebraic Conf. dedic. to the 100th an-ry of D. K. Fadeev. St P-rg, Russia, 2007, pp. 98–99.

[11] N. P. Sokolov, *Spatial matrices and their application.* Moscow, 1960.

Aureliu Zgureanu, Received November 10, 2010

Aureliu Zgureanu
The Academy of Transport, Computer Science and Communication
Munceşti, 121-a, Chişinău MD-2002 Moldova
E–mail: *aurelzgureanu@gmail.com*
Phone: +373 79 234829, +373 22 473056

Computer Science Journal of Moldova, vol.**18**, no.3(54), 2010

Matrix balancing and robust Monte Carlo algorithm for evaluating dominant eigenpair

Behrouz Fathi Vajargah Farshid Mehrdoust

Abstract

Matrix balancing may effect the stability of algorithms in matrix computations and the accuracy of computed solutions. In this paper, we first introduce an algorithm for matrix balancing. Then, using Monte Carlo method we propose a robust algorithm to evaluate dominant eigenpair of a given matrix. Finally, several randomly generated examples are presented to show the efficiency of the new method.

Keywords: Monte Carlo algorithms; Robust Monte Carlo algorithm; Markov chain; Balancing; Eigenpair; Large scale matrices

1 Introduction

The need to compute dominant eigenpair of matrices arises frequently in scientific and engineering applications with the solution being useful either by itself or as an intermediate step in solving a larger problem. There are many different algorithms presently used to obtain eigenpair of a matrix, among them are the Householder method, the QR method and subspace iteration [6, 7]. Many of these algorithms are inefficient when applied to very large structural systems. Krylov subspace Lanczos method is widely appreciated by the numerical analysis community [6, 7]. The problem of using Monte Carlo and quasi Monte Carlo methods for finding an eigenpair has been extensively studied, for example [2-7]. In this paper, we study the Monte Carlo approach to obtain the dominant eigenpair of matrices with an emphasis on preconditioning

©2010 by B. Fathi Vajargah, F. Mehrdoust

implementation of the corresponding algorithm which is called matrix balancing. We employ a special balancing procedure as a preprocessing step before running the Monte Carlo procedure. Such a balancing procedure ensures robustness of the Monte Carlo algorithm and therefore relatively small values for the stochastic error.

Let A be an $n \times n$ real matrix whose eigenvalues we seek. The pair (λ, x) is called an eigenpair of A if

$$Ax = \lambda x, \quad x \neq 0. \tag{1}$$

In equation (1) the scaler λ and the vector x are called an eigenvalue and eigenvector, respectively. Throughout the paper we suppose that the matrix $A \in \mathbb{R}^{n\times n}$ is diagonalizable with eigenvalues

$$\lambda_{max} = |\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_k|.$$

Note that this implies λ_1 is real, otherwise $\overline{\lambda}_1$(conjugate of λ_1) is another eigenvalue with the same magnitude as λ_1 [7].

2 Monte Carlo approach for computing (λ, x)

Consider the following Markov chain with length i

$$T_i : k_0 \to k_1 \to \cdots \to k_i \tag{2}$$

where $k_j \in \{1, 2, ..., n\}$ for $j = 1, 2, ..., i$.
The following choice of p_α and $p_{\alpha,\beta}$, for constructing $T_i's$, is considered:

$$p(k_0 = \alpha) = p_\alpha, \qquad p(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha\beta}, \tag{3}$$

where p_α and $p_{\alpha\beta}$ show the probability of starting chain at α and transition probability from state α to β, respectively. We further should have that

$$\sum_{\alpha=1}^{n} p_\alpha = 1 \tag{4}$$

and

$$\sum_{\beta=1}^{n} p_{\alpha\beta} = 1 \tag{5}$$

for each $\alpha = 1, 2, ..., n$. Probabilities $p_{\alpha\beta}$ define the transition matrix P.

Let matrix $A \in \mathbb{R}^{n\times n}$ and two vectors $f, h \in \mathbb{R}^n$ are given. Further suppose that the distributions created from the density probabilities p_α and $p_{\alpha\beta}$ are acceptable according to the following definition [2]

$$p_\alpha > 0 \quad \text{when} \quad h_\alpha \neq 0, \qquad p_\alpha \geq 0 \quad \text{when} \quad h_\alpha = 0 \tag{6}$$

and

$$p_{\alpha\beta} > 0 \quad \text{when} \quad a_{\alpha\beta} \neq 0, \qquad p_{\alpha\beta} \geq 0 \quad \text{when} \quad a_{\alpha\beta} = 0. \tag{7}$$

We define the random variable W_j using the following recursive equation

$$W_j = W_{j-1} \frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \quad j = 1, 2, ..., i, \tag{8}$$

where $W_0 = \frac{h_{k_0}}{p_{k_0}}$. Then we may apply the following probability structures

$$\begin{aligned} p_\alpha &= \frac{|h_\alpha|}{\sum_{\beta=1}^{n} |h_\beta|} \\ p_{\alpha\beta} &= \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^{n} |a_{\alpha\beta}|} \qquad \alpha, \beta = 1, 2, ..., n. \end{aligned} \tag{9}$$

Thus, from (8) and (9) we have

$$\begin{aligned} W_i &= \sum_{\beta=1}^{n} |h_\beta| \sum_{\beta=1}^{n} |a_{k_0\beta}| \sum_{\beta=1}^{n} |a_{k_1\beta}| ... \sum_{\beta=1}^{n} |a_{k_{i-1}\beta}| \prod_{j=1}^{i} \frac{a_{k_{j-1}k_j}}{|a_{k_{j-1}k_j}|} \frac{h_{k_0}}{|h_{k_0}|} \\ &= \sum_{\beta=1}^{n} |h_\beta| \prod_{j=0}^{i-1} \sum_{\beta=1}^{n} |a_{k_j\beta}| sign\{h_{k_0} \prod_{j=1}^{i} a_{k_{j-1}k_j}\}. \end{aligned} \tag{10}$$

Now, let us define the random variable $\Gamma^{(i)}$ as

$$\Gamma^{(i)} = W_i f_{k_i} = \sum_{\beta=1}^{n} |h_\beta| \prod_{j=0}^{i-1} \sum_{\beta=1}^{n} |a_{k_j\beta}| sign\{h_{k_0} \prod_{j=1}^{i} a_{k_{j-1}k_j}\} f_{k_i}, \quad (11)$$

where $\Gamma^{(i)}$ will be employed for evaluating $< h, A^i f >$, which is used for estimating λ_{max}. We follow the above discussions in the next theorem.

Theorem 1. *Under assumptions 2 to 9, the random variable $\Gamma^{(i)}$ is an unbiased estimator for $< h, A^i f >$ i.e.,*

$$E[\Gamma^{(i)}] =< h, A^i f > . \quad (12)$$

Proof. *We have*

$$\begin{aligned} E[\Gamma^{(i)}] &= E[\frac{h_{k_0}}{p_{k_0}} \frac{a_{k_0k_1}...a_{k_{i-1}k_i}}{p_{k_0k_1}...p_{k_{i-1}k_i}} f_{k_i}] \\ &= \sum_{k_0,...,k_i=1}^{n} \frac{h_{k_0}}{p_{k_0}} \frac{a_{k_0k_1}...a_{k_{i-1}k_i}}{p_{k_0k_1}...p_{k_{i-1}k_i}} f_{k_i} p_{k_0} p_{k_0k_1}...p_{k_{i-1}k_i} \\ &= \sum_{k_0=1}^{n} h_{k_0} \sum_{k_1=1}^{n} a_{k_0k_1}... \sum_{k_{i-1}=1}^{n} a_{k_{i-2}k_{i-1}} \sum_{k_i=1}^{n} a_{k_{i-1}k_i} f_{k_i} \\ &= \sum_{k_0=1}^{n} h_{k_0} \sum_{k_1=1}^{n} a_{k_0k_1}... \sum_{k_{i-1}=1}^{n} a_{k_{i-2}k_{i-1}} (Af)_{k_{i-1}} \\ &= \sum_{k_0=1}^{n} h_{k_0} (A^i f)_{k_0} =< h, A^i f > . \quad \blacksquare \end{aligned}$$

Now, let us simulate N random paths as T_i defined in (2), (3) and suppose $\Gamma_s^{(i)}$ is the s^{th} realization of random variable $\Gamma_s^{(i)}$ i.e.,

$$\Gamma_s^{(i)} = \sum_{\beta=1}^{n} |h_\beta| \{\prod_{j=0}^{i-1} \sum_{\beta=1}^{n} |a_{k_j\beta}| sign(\prod_{j=1}^{i} a_{k_{j-1}k_j} f_{k_i})\}_s, \qquad s = 1, ..., N.$$

Then the value

$$\bar{\Gamma}^{(i)} = \frac{1}{N} \sum_{s=1}^{N} \Gamma_s^{(i)}, \qquad i \geq 1 \quad (13)$$

is considered as a Monte Carlo approximation of $< h, A^i f >$.

Based on the power method [7], Monte Carlo method for evaluating the dominant eigenvalue as $i \to \infty$, is

$$\lambda_{max} \approx \frac{\overline{\Gamma}_s^{(i)}}{\overline{\Gamma}_s^{(i-1)}}. \tag{14}$$

Let the goal is to find the eigenvector x that corresponds to the eigenvalue λ, then we set $h = e(j) = (0, ..., 0, 1, 0, ..., 0)$, where $e(j)$ is the j^{th} unit vector in $\mathbb{R}^n$, i.e. $(e(j))_\alpha = \delta_{j\alpha}$. It follows that

$$< h, x >= \sum_{\alpha=1}^{n} (e(j))_\alpha x_\alpha$$

and the j^{th} component of eigenvector x using Monte Carlo method is

$$x_j \approx \frac{1}{N} \sum_{s=1}^{N} \{\Gamma^{(i)}[e(j)]\}_s. \tag{15}$$

Theorem 2. *The stochastic error for calculating the dominant eigenvalue of a given matrix $A \in \mathbb{R}^{n\times n}$, based on the Monte Carlo algorithm is minimized if for each i and for some $L > 0$, we have $\sum_{j=1}^{n} |a_{ij}| = L$, i.e. the absolute rowsums of A be a constant number.*

Proof. *Consider the following random variable*

$$\Theta_l(f) = \sum_{i=0}^{l} W_i f_{k_i}, \tag{16}$$

where $f \in \mathbb{R}^n$ is an arbitrary vector and W_i is the variable, defined in (8). It is easy to see that

$$\begin{aligned} Var[\Theta_l(f)] &= \sum_{i=0}^{l} Var[W_i f_{k_i}] + 2 \sum_{i} \sum_{j>i} Cov(W_i f_{k_i} W_j f_{k_j}) \\ &\leq \sum_{i=0}^{l} Var[W_i f_{k_i}] + 2 \sum_{i} \sum_{j>i} \sigma(W_i f_{k_i}) \sigma(W_j f_{k_j}). \end{aligned}$$

Therefore, it is sufficient to minimize $Var[W_i f_{k_i}]$.
We further have

$$\begin{aligned} E[(W_i f_{k_i})^2] &= \sum_{k_0 k_1 \cdots k_i=1}^{n} \frac{h_{k_0}^2}{p_{k_0}} \frac{a_{k_0 k_1}^2 \cdots a_{k_{i-1} k_i}^2}{p_{k_0 k_1} \cdots p_{k_{i-1} k_i}} f_{k_i}^2 \\ &= \prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| \sum_{k_0 k_1 \cdots k_i=1}^{n} |h_{k_0}||a_{k_0 k_1}| \ldots |a_{k_{i-1} k_i}| f_{k_i}^2 \\ &= \prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| \sum_{k_0=1}^{n} |h_{k_0}| (|A|^i, f_{k_i}^2)_{k_0} \\ &= \prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| (|h|, |A|^i f^2). \end{aligned}$$

Now, since f *and* h *are arbitrary vectors, without loss of generality assume that* $f = (1, \cdots, 1)^T$, $h = (\frac{1}{n}, \cdots, \frac{1}{n})$. *Now, we suppose that* $A = |A| = (|a_{i,j}|)_{i,j=1,\ldots,n}$, *thus from the previous equality it follows*

$$\prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| = (h, A^i f),$$

hence we easily conclude that $Var[W_i f_{k_i}]$ *vanishes when* $\sum_{j=1}^{n} |a_{kj}|$ *is equal to a constant value* L, *for* $k = 1, 2, \ldots, n$. ■

The above theorem shows that to reduce the Monte Carlo error it is important to deal with balanced matrices. In the next section, we introduce matrix balancing procedure based on the condition given in Theorem 2.

3 Balancing matrices

Balancing is a preprocessing step, which may produce positive effects on the accuracy and performance of numerical methods for computing eigenvalues. A matrix $A \in \mathbb{R}^{n \times n}$ with a norm that is several orders

of magnitude larger than the modules of its eigenvalues typically has eigenvalues that are sensitive to perturbations in the entries of A. One of the main methods is Sinkhorn-Knopp algorithm [1]. There are other algorithms for balancing that can converge faster than the Sinkhorn-Knopp algorithm, for example, Parlett and Landis [6]. In this paper, we have used a Krylov-based balancing algorithm proposed in [1].

Definition 1. An $n \times n$ matrix A with nonnegative entries is said to be balanced if for each $i = 1, ..., n$, the sum of the entries of its i^{th} row is equal to the sum of the entries of its i^{th} column. In other words,

$$A\mathbf{e} = A^T\mathbf{e}, \tag{17}$$

where $\mathbf{e}$ is the n-vector of all ones.
More generally, an $n \times n$ matrix with arbitrary real entries is said to be balanced in $l^p - norm$ if for each $i = 1, ..., n$ its i^{th} row and column have the same $l^p - norm$. Our employed balancing algorithm is as follows:

Algorithm 1. (Balancing algorithm)

1. **Input** $A \in R^{n\times n}$, t (number of iterations)
2. **for** s=1 : t
2.1 **Set** vector $z_{n\times 1}$ of random numbers $1, -1s$
2.2 **Compute** $p = Az$
2.3 **for** $i = 1 : n$
2.4 **if** $(p(i) = 0)$ then
2.4.1 **Set** $D(i) = 1$
2.4.2 **else** $D(i) = \frac{1}{p(i)}$
2.4.3 **end** for
2.5 **Set** $A = D * A * D^{-1}$
2.6 **end** for
3. **end of** algorithm 1

Now, we present the robust Monte Carlo algorithm based on the balanced matrix. In this algorithm, we use partitioned random number generator for generating Markov chains (random trajectories). In fact, we may divide the interval $(0, 1)$ to r subintervals with equal length $\frac{1}{r}$,

where r is a natural number greater than 1. We use it in rand function of MATLAB software for generating random numbers with more uniformity. We note that in the following algorithm each row of the $N \times i$ matrix ZZ is a Markov chain with the length i. According to Theorem 2, we further consider the vector $h = (\frac{1}{n}, \ldots, \frac{1}{n})^T$.

Algorithm 2. (Robust Monte Carlo algorithm)

1. **Input** the matrix $A_{n\times n}$, the number of Markov chains N and the length of Markov chains i
2. **Call** algorithm 1 for balancing matrix A
3. **Generate** transition probability matrix $P = (p_{ij})_{i,j=1,\ldots,n}$ according to the equation (9)
4. **Generate** an $N \times i$ random matrix ZZ
5. **Set** D1=0; D2=0
6. **for** $s = 1 : N$
7. **Set** $z = ZZ(s, :)$ ▶ s^{th} row of the matrix ZZ
8. **Set** $v = A(z(i-1), :)$
9. **if** $A(z(i-1), z(i)) \neq 0$ **then**
10. **Set** $D1 = D1 + sign(A(z(i-1), z(i))) * norm1(v) * \frac{1}{n}$
11. **Set** $D2 = D2 + h(z(i-1))$
12. **End if**
13. **End for**
14. Approximation of dominant eigenvalue is $D1/D2$
15. **End** algorithm 2

4 Computational results

In the following tables we compare the precision of the computed dominant eigenpair without balancing and after applying balancing algorithm. All test problems were run in MATLAB software on a PC with Intel(R) 1.83 GHz Dual CPU processor. Moreover, the Monte Carlo relative error was computed by the following formula

$$MC\ relative\ error = \frac{|MC\ result - exact\ result|}{|exact\ result|}.$$

First, let us consider the following test matrix

$$A = \begin{pmatrix} 0.6716 & 0.2417 & 0.2461 & 0.4788 & 0.1615 \\ 0.7003 & 0.1290 & 0.3725 & 0.3583 & 0.9478 \\ 0.9097 & 0.3089 & 0.9758 & 0.8556 & 0.5761 \\ 0.2902 & 0.5112 & 0.0766 & 0.6052 & 0.1597 \\ 0.7199 & 0.3323 & 0.2303 & 0.7844 & 0.8600 \end{pmatrix}.$$

Basing on the Theorem 2, the rowsums vector for matrix A before balancing is

$$rowsums = (1.7997, 2.5079, 3.6261, 1.6429, 2.9269)^T,$$

and after balancing is

$$rowsums = (2.3035, 2.0470, 2.4542, 2.1937, 2.5738)^T,$$

where the balanced matrix is

$$A = \begin{pmatrix} 0.6716 & 0.4076 & 0.5187 & 0.4671 & 0.2385 \\ 0.4153 & 0.1290 & 0.4655 & 0.2073 & 0.8299 \\ 0.4316 & 0.2472 & 0.9758 & 0.3960 & 0.4036 \\ 0.2975 & 0.8837 & 0.1655 & 0.6052 & 0.2418 \\ 0.4875 & 0.3795 & 0.3287 & 0.5182 & 0.8600 \end{pmatrix}.$$

As we see, the rowsums after applying balancing algorithm are more centralized. In Table 2, we have compared the precision of the computed eigenvalues without balancing and after applying balancing algorithm for several randomly generated matrices.

Table 1. MC relative error for calculating dominant eigenpair $(\hat{\lambda}, \hat{v})$

Number of trajectories	$\hat{\lambda}, \hat{v}$ (without bal.)	Time (s)	$\hat{\lambda}, \hat{v}$ (with bal.)	Time (s)
100	$0.0797, 0.2504$	0.03	$0.0069, 0.1637$	0.05
1000	$0.0727, 0.1340$	0.14	$0.0041, 0.0571$	0.16

Table 2. MC relative error in computing the dominant eigenvalue for several randomly generated matrices

Dimension	MC method	
	$\widehat{\lambda}_1$ (with bal.)	$\widehat{\lambda}_1$ (without bal.)
100	4.1289×10^{-4}	0.1600×10^{-2}
200	5.4924×10^{-5}	3.0526×10^{-4}
400	1.7083×10^{-4}	0.1100×10^{-2}
800	2.9271×10^{-5}	0.1900×10^{-2}
1600	1.3916×10^{-5}	3.4172×10^{-4}
3200	5.1900×10^{-6}	3.7581×10^{-4}

5 Concluding remarks

In this paper, we have proposed a new algorithm for obtaining dominant eigenpair of desired matrices. By this algorithm, we are able to reduce the stochastic error in Monte Carlo method. The numerical experiments have shown that the Algorithm 1 not only makes the matrix more balanced, but also balances the sum of absolute values in the rows which is desirable according to the Theorem 2. The proposed algorithm has been implemented for large scale matrices in computing dominant eigenvalue and we can conclude that the balancing of the input matrix is very important for improving the accuracy of Monte Carlo algorithm (Figure 1 and Figure 2).

References

[1] Chen T. Y. and Demmel J. W., *Balancing sparse matrices for computing eigenvalues, Linear algebra and its applications,* 309(2000)261-287.

[2] Dimov I.T. and Alexandrov V.N., *A new highly convergent Monte Carlo method for matrix computations.* Mathematics and Computers in Simulation, 47(1998)165-181.

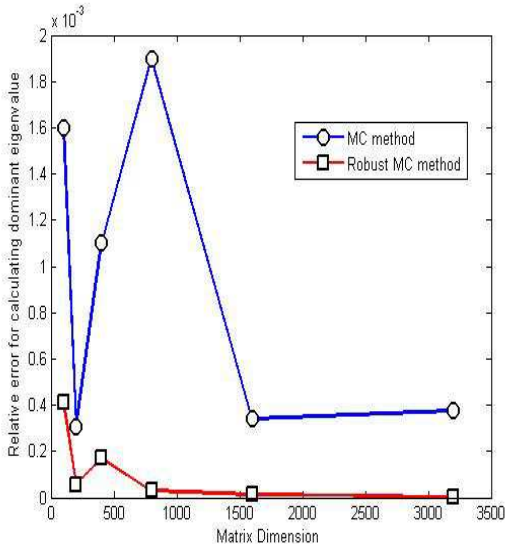

Figure 1. Comparison of relative error between Monte Carlo and robust Monte Carlo methods for various matrices

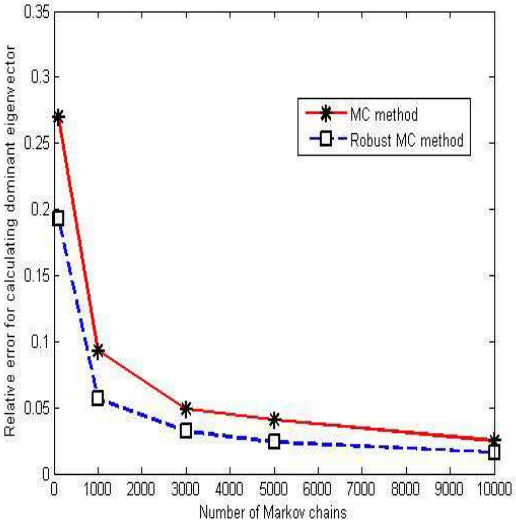

Figure 2. Comparison of relative error between Monte Carlo and robust Monte Carlo methods for various Markov chains

[3] Fathi Vajargah B. and Mehrdoust F., *New Monte Carlo algorithm for obtaining three dominant eigenvalues,* IJAM, 22(2009)553-559.

[4] Fathi Vajargah B. and Mehrdoust F., *New hybrid Monte Carlo methods and computing the dominant generalized eigenvalue*, International journal of computer mathematics, Taylor & Francic, 2009.

[5] Hammersley J. M. and Handscomb D. C., *Monte Carlo Methods.* Methuen, London, (1964).

[6] Kressner D., *Numerical Methods for General and Structured Eigenvalue Problems,* Springer-Verlag Berlin Heidelberg, (2005).

[7] Meyer K. D., *Matrix analysis and applied linear algebra,* SIAM, (2000).

Behrouz Fathi Vajargah, Farshid Mehrdoust

Received June 17, 2010
Revised January 31, 2011

Faculty of Mathematical Sciences,
University of Guilan,
Rasht, P.O. Box: 1914, Iran
E–mail: *fathi@guilan.ac.ir*,
fmehrdoust@guilan.ac.ir

Abstracts of Doctor Habilitatus Thesis

Title: Study of language-theoretic computational paradigms inspired by biology
Author: Serghei Verlan
University: Université Paris Est – Creteil Val de Marne
Date of defence: 22 October 2010

Understanding the surrounding matter, the laws and the functioning of the universe, this is the main challenge of the humankind, appeared at the dawn of times. Significant progress was made during last centuries, especially in the field of physics, which ultimately permitted the engineering of new

devices and the development of new technologies. The mathematics is in tight relation with all these discoveries, as it provided a formal basis for these achievements.

The challenges of the actual science more and more reside in the field of biology. This is quite natural as the fundamental questions about life: what distinguishes living and non-living matter and what is the organization of a living thing, are considered from the very old times. Recent developments in most areas of science permit to approach above questions in a methodological way giving a hope for substantial breakthrough.

Still the biology is an experimental science that accumulates a large collection of knowledge and that needs the help of other disciplines in order to explain and correlate the gathered information. The mathematics is the primary tool sewing this goal, for example, with differential equations in systems biology or statistical analysis for populations evolution.

The computer science also plays an important role in biological investigations. Large-scale databases, data mining, sequence analysis and protein structure prediction – these are several tasks that would never be possible without the arrival of the computers, computational theory and corresponding algorithms. The molecular biology highlighted the key role of DNA and of related mechanisms for the information processing, which is crucial for the understanding of the functioning of living organisms. The computer science traditionally deals with sequences, so it was natural to use the provided tools for the sequence-related research, forming the field of bioinformatics.

We note that computer science is specialized on the discrete representation of the universe and on the representation of information flow processes. Many key concepts from the field where borrowed from the biology. This is why it fits well to the biological modeling providing discrete models in contrast to a modeling with differential equations, which are continuous. This property stimulated the investigation of (computational) models and operations issued from biology. In this case real biological phenomena are abstracted and a discrete system based on the involved operations and functioning is proposed.

There are two motivations for the investigation of such models. The first motivation is very clear – since the model represents an abstraction of a biological phenomena, it is possible to describe the last one in precise terms and provide simulations, estimations and predictions. A close relationship with target biological system permits to express it in a short and clear manner giving the hope to extract additional knowledge about the model. This approach, very nice in theory, encounters important difficulties while faced to the practice. Usually, simple models are too abstract for obtaining non-trivial relations for the initial biological system, while more complete models

encounter rapidly the combinatorial explosion of their size, which makes them very difficult to analyze. The success of the approach relies on a deep knowledge of the target phenomena that permits to find a good balance between abstraction and adequateness.

The second motivation comes from the fact that most successful applications of computer science are based on ideas borrowed from the biology. During the evolution of living organisms, the nature provided solutions for many complicated problems encountered and it is very fruitful to adapt these solutions to different problems. Ant colony optimization, cellular automata, neural networks, evolutionary algorithms are some examples of such an approach.

This thesis is centered around two main topics: insertion-deletion systems and P systems. The first part does a systematical study of the operations of insertion and deletion. While it is possible to consider them as biologically inspired operations, we perform in the thesis a pure theoretical investigation in terms of the theory of formal languages. Thus, our research provides new classes of formal grammars which extend the Chomsky hierarchy by introducing new levels. We introduce a new proof method and show a series of computational-completeness and non-completeness results and discuss possibilities for the extension of the computational power for the latter case.

The research on P systems follows a different motivation. Primary, they represent a general framework for distributed multiset rewriting, which captures important processes in cell biology. In contrast to traditional approaches in systems biology, P systems provide a discrete framework for the representation of molecular interactions[1] that focuses on the structure of the system and on the identification of its components. The field of application for P systems is not limited to systems biology, the topic incorporates concepts from cell biology and aims to propagate them to all fields of the computer science.

The first part of our work targets the core of the P systems framework – its formal definition, which, surprisingly, was not always clear; moreover the introduction of new concepts like the minimal parallelism lead to different interpretations by different authors. We provide a single formal definition, which covers most of the classes of P systems with static structure known up to now.

The second part of Chapter 3 deals with one of most important models in the area of P systems – P systems using only communication, i.e., the objects present in the system cannot evolve, but can only be moved from one

[1] We remark that other discrete models like Petri nets or process algebra are also used, each of them having advantages and limitations.

compartment to another. We generalise the idea of co-transporters from the cell biology and end up with an interesting computational model based on the synchronization of signals. It is worth to note that the obtained model generalises all previous communication- only models of P systems.

The last part of Chapter 3 deals with two concrete problems. The first problem, the problem of the synchronization on a tree, is the generalization of the firing squad synchronization problem for cellular automata, where a linear communication structure is replaced by a hierarchical one. The second problem, the construction of universal P systems of small size, is closely related to the problem of the construction of small size universal Turing machines, which is a fundamental problem of the computer science.

The last part of the thesis presents an exploratory topic, where we performed all the stages for the construction of a new model – we start with a biological phenomena, give its formalization and start theoretical investigations. The closeness of the obtained model to the initial biological subject permit us to affirm that results that are obtained theoretically, can be verified in vitro.

Computer Science Journal of Moldova, vol.**18**, no.3(54), 2010

International Workshop on Intelligent Information Systems

Organized by

the Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova (IMCS)

Location: Chisinau, Republic of Moldova
Dates: September 13-14, 2011

Topics of interest

Topics of interest include, but are not restricted to, the following areas:

- Theoretical Computer Science;
- Formal Models of Computing;
- Information Technologies Applications in Knowledge-Based Society;
- Decision Support Systems;
- Intelligent Agents and Multi-Agent Systems;
- Knowledge Engineering and Management;
- Image Processing;
- Medical & Diagnostic Systems;
- Natural Language Processing;

- Business Intelligence Systems;
- Human-Centered Computing;
- Intelligent User Interfaces.

Call for Papers

Participants are invited to submit high quality papers reporting on original research, scientific results and experiences addressing the areas listed above. Papers should be electronically submitted to the Organizing Committee via email iis2011@math.md.

Selected papers will be published in Computer Science Journal of Moldova (CSJM, ISSN 1561-4042). Requirements for papers publication in CSJM are also available at the following address: http://www.math.md/en/publications/csjm/instructions-authors/.

Important dates

Papers Submission:	May 25, 2011.
Notification of acceptance:	July 15, 2011.
Detailed program will be announced:	September 5, 2011.
Workshop dates:	September 13-14, 2011.

Organizing Committee