Computer Science Journal of Moldova, vol.**14**, no.3(42), 2006

Generating Languages by P Systems with Minimal Symport/Antiport *

Artiom Alhazov Yurii Rogozhin

Abstract

It is known that P systems with two membranes and minimal symport/antiport rules are "almost" computationally complete as generators of number or vector sets.

Interpreting the result of the computation as the sequence of terminal symbols sent to the environment, we show that P systems with two membranes and symport rules of weight two or symport/antiport rules of weight one generate all recursively enumerable languages.

1 Introduction

Membrane System (also called P system), which is a model of living cell, was introduced by Gh. Păun recently ([21, 22]). Membrane systems are distributed parallel computing devices, processing multisets of objects, synchronously, in compartments delimited by a membrane structure. The objects, which correspond to chemicals evolving in the compartments of a cell, can also pass through membranes. The membranes form a hierarchical structure (they can be dissolved, divided, created, and their permeability can be modified). A sequence of transitions between configurations of a P system forms a computation. The result of a halting computation is the number of objects present at the end of the computation in a specified membrane, called the output membrane. The objects can also have a structure of their own that can

*The authors acknowledge the project 06.411.03.04P from the Supreme Council for Science and Technological Development of the Academy of Sciences of Moldova.

be described by strings over a given alphabet of basic molecules - in this case the result of a computation is a set of strings. An important version of membrane systems deals with membranes arranged not in a hierarchical structure (which mathematically corresponds to a tree), but in a tissue-like structure (which mathematically corresponds to a graph). Many open problems related to the computational power of P systems remain. The challenge is especially interesting, and apparently difficult, for restricted versions of P systems, in our case for P systems with symport/antiport rules.

P systems with symport/antiport rules are parallel distributed systems, processing multisets according the rules that move the objects between the regions. They were first introduced in [20]; symport rules move objects across a membrane (which is a separator of regions) together in one direction, whereas antiport rules move objects across a membrane in opposite directions.

A comprehensive overview of the most important results obtained in the area of P systems and tissue P systems with *symport/antiport* rules, with respect to the development of computational completeness results improving descriptional complexity parameters as the number of membranes and cells, respectively, the weight of the rules and the number of objects can be found in [1] and the last results in [5]. We consider P systems with symport/antiport rules and minimal cooperation, i.e., P systems with symport/antiport rules of weight one and P systems with symport rules of weight two, such systems are called P systems with minimal symport/antiport.

In this paper we consider generating languages by such P systems with two membranes, in the way it has been done, e.g., for transitional P systems and for P systems with active membranes, see [22]. This is a natural generalization of studies of the generative power of P systems with symport/antiport. Since the environment in this model is considered not empty at the beginning of the computation, we distinguish the result by considering the sequence of objects from a terminal sub-alphabet, sent into the environment.

2 Preliminaries

Let O be a finite set, O^* is a free monoid generated by O. Consider $x \in O^*$; we will denote all permutations of x by $Perm(x)$.

By RE we denote the family of recursively enumerable languages.

2.1 Counter Automata with an Output Tape

A non-deterministic counter automaton (see [9], [1]) with an output tape is a tuple

$$M = (d, T, Q, q_0, q_f, P), \text{ where}$$

- d is the number of counters
  (we will use the notation $D = \{1, \cdots, d\}$);
- T is an output alphabet;
- Q is a finite set of states;
- $q_0 \in Q$ is the initial state;
- $q_f \in Q$ is the final state;
- P is a finite set of instructions.

The instructions operating on counters are of the forms

$$(q_i \to q_l, k\gamma), \text{ with } q_i, q_j \in Q,\ q_i \neq q_f,\ k \in D,\ \gamma \in \{+, -, =0\}$$

(changing the state from q_i to q_l and applying operation γ to counter k). The operations are *increment* (add one to the value of the counter), *decrement* (subtract one from the value of the counter) and *zero-test* (test whether the value of the counter is zero or not), respectively. If an empty counter is decremented or a non-empty counter is tested for zero, then the computation is blocked.

The instructions operating on the tape are of the form

$$(q_i \to q_l, write(c)), \text{ where } q_i, q_j \in Q,\ q_i \neq q_f \text{ and } c \in T$$

(changing the state and writing the symbol c on the tape). One can also speak about the *halt* instruction of the counter automaton, assigned to the final state q_f.

A transition of the counter automaton consists in updating or checking the value of a counter, or writing a symbol on the tape, according to an instruction of one of the types described above and by changing the current state to another one. The computation starts in state q_0 with all counters equal to zero. The result of the computation is a word consisting of the symbols written on the tape when the automaton halts in state q_f.

The result we will use is that counter automata are computationally complete, and only two counters are needed.

2.2 P Systems with Symport/Antiport

The reader is supposed to be familiar with basic elements of membrane computing, e.g., from [22]; comprehensive information can be found in the P systems web page, [29].

A *P system with symport/antiport rules* is a construct

$$\Pi = (O, T, E, \mu, w_1, \ldots, w_k, R_1, \ldots, R_k), \text{ where}$$

1. O is a finite alphabet of symbols called *objects*;
2. $T \subseteq O$ is the terminal alphabet;
3. $E \subseteq O$ is the set of objects that appear in the environment in an infinite number of copies;
4. μ is a *membrane structure* consisting of k membranes that are labelled in a one-to-one manner by $1, 2, \ldots, k$;
5. $w_i \in O^*$, for each $1 \leq i \leq k$, is a finite multiset of objects associated with the region i (delimited by membrane i);
6. R_i, for each $1 \leq i \leq k$, is a finite set of symport/antiport rules associated with membrane i; these rules are of the forms (x, in)

and (y, out) *(symport rules)* and $(y, out; x, in)$ *(antiport rules)*, respectively, where $x, y \in O^+$.

A P system with symport/antiport rules is defined as a computational device consisting of a set of k hierarchically nested membranes that identify k distinct regions (the membrane structure μ), where to each membrane i there are assigned a multiset of objects w_i and a finite set of symport/antiport rules R_i, $1 \leq i \leq k$. A rule $(x, in) \in R_i$ permits the objects specified by x to be moved into region i from the immediately outer region. Notice that for P systems with symport rules the rules in the skin membrane of the form (x, in), where $x \in E^*$, are forbidden. A rule $(x, out) \in R_i$ permits the multiset x to be moved from region i into the outer region. A rule $(y, out; x, in)$ permits the multisets y and x, which are situated in region i and the outer region of i, respectively, to be exchanged. It is clear that a rule can be applied if and only if the multisets involved by this rule are present in the corresponding regions. The weight of a symport rule (x, in) or (x, out) is given by $|x|$, while the weight of an antiport rule $(y, out; x, in)$ is given by $max\{|x|, |y|\}$.

As usual, a computation in a P system with symport/antiport rules is obtained by applying the rules in a non-deterministic maximally parallel manner. Specifically, in this variant, a computation is restricted to moving objects through membranes, since symport/antiport rules do not allow the system to modify the objects placed inside the regions. Initially, each region i contains the corresponding finite multiset w_i, whereas the environment contains only objects from E that appear in infinitely many copies.

2.3 Generating Languages

A computation is halting if starting from the initial configuration, the system reaches a configuration where no rule can be applied anymore.

Consider the output alphabet T. Throughout the computation we register when objects from T are sent to the environment. The result of a halting computation is the sequence of objects from T sent to the environment in the order in which it happened (if multiple objects from

T are ejected in the environment simultaneously, all permutations are considered). [1]

Example 1 *Consider the following P system*

$$\begin{aligned} \Pi &= (O = \{p, a, b\}, T = \{a, b\}, E = \{p, a, b\}, \mu = [_1 \ ]_1, w_1, R_1), \\ w_1 &= paab, \\ R_1 &= \{r_1 : (pa, out; pb, in), r_2 : (ab, out; a, in)\}. \end{aligned}$$

The environment contains an unbounded supply of objects p, a, b, so the state of the system is determined by objects in the skin membrane. The system starts with one copy of p, two copies of a and one copy of b in the skin membrane, simultaneously applying rules r_1 and r_2. Having one copy of each of the objects p, a, b, two computations are possible:

$$\boxed{pab}_1 \Rightarrow_1^{r_1} \boxed{pbb} \qquad or \qquad \boxed{pab}_1 \Rightarrow^{r_2} \boxed{pa}_1 \Rightarrow^{r_1} \boxed{pb}_1.$$

Consider the terminal objects sent out in either case. These computations correspond to generating a language

$$Perm(aab) \cdot a \cup Perm(aab) \cdot Perm(ab) \cdot a,$$

consisting of 9 words.

We denote the family of languages generated by a P system with symport/antiport rules with at most $m > 0$ membranes, symport rules of weight at most $s \geq 0$, and antiport rules of weight at most $t \geq 0$ by

$$LOP_m(sym_s, anti_t)$$

If $t = 0$, we may omit $anti_t$.

[1] Intuitively, sending the terminal objects out of the skin membrane is like writing them on the output tape.

3 Main Results

Theorem 1 $LOP_2(sym_1, anti_1) = RE$.

Proof. Without loss of generality we simulate a counter automaton with an output tape $M = (d, T, Q, q_0, q_f, P)$ which starts with empty counters. We also suppose that all instructions from P are labeled in a one-to-one manner with elements of $\{1, \ldots, n\} = I$, n is a label of the *halt* instruction and $I' = I \setminus \{n\}$. We denote by I_+, I_-, and $I_{=0}$ the set of labels for the "increment" -, "decrement" -, and "test for zero" -instructions, respectively ("increment"-instructions include also instructions operating on the tape). We use the next notation: $C = \{c_k\}, k \in D$.

We construct the P system Π_1 as follows:

$$
\begin{aligned}
\Pi_1 &= (O, T, E, [_1\, [_2\ \ ]_2\,]_1, w_1, w_2, R_1, R_2),\\
O &= E \cup \{X, Y_1, Y_2, J_1, J_2, J_3\} \cup \{b_j, d_j \mid j \in I\},\\
E &= T \cup Q \cup C \cup \{a_j, e_j \mid j \in I\} \cup \{a'_j \mid j \in I' \setminus I_-\} \cup \{J_0, F, Y_3\},\\
w_1 &= q_0 J_1 J_2 J_3,\\
w_2 &= X J_3 Y_1 Y_2 \prod_{j \in I} b_j \prod_{j \in I} d_j,\\
R_i &= R_{i,s} \cup R_{i,r} \cup R_{i,f}, \quad i = 1, 2.
\end{aligned}
$$

We code the counter automaton as follows:

Region 1 will hold the current state of the automaton, represented by a symbol $q_i \in Q$ and the value of all counters, represented by the number of occurrences of symbols $c_k \in C$, $k \in D$, where $D = \{1, ..., d\}$.

We split our proof into several parts that depend on the logical separation of the behavior of the system. We will present the rules and the initial symbols for each part, but we remark that the system we present is the union of all these parts. The rules R_i are given by three phases:

1. START: preparation of the system for the computation.
2. RUN: simulation of instructions of the counter automaton.

3. END: terminating the computation.

The parts of the computations illustrated in the following describe different phases of the evolution of the P system. For simplicity, we focus on explaining a particular phase and omit the objects that do not participate in the evolution at that time. Each rectangle represents a membrane, each variable represents a copy of an object in a corresponding membrane (symbols outside of the outermost rectangle are found in the environment). In each step, the symbols that will evolve (will be moved) are written in **boldface**. The labels of the applied rules are written above the symbol $\Rightarrow$.

1. START.

We use here the following idea: in our system we have a symbol X which moves from region 2 to region 1 and back in an infinite loop. This loop may be stopped only if all stages completed correctly.

$$\begin{aligned} R_{1,s} &= \emptyset, \\ R_{2,s} &= \{\mathtt{2s1} : (X, out), \mathtt{2s2} : (X, in)\}. \end{aligned}$$

Notice that some rules are never executed during a correct simulation: applying them would lead to an infinite computation. To help the reader, we will underline the labels of such rules in the description below.

2. RUN.

$$\begin{aligned} R_{1,r} &= \{\mathtt{1r1} : (q_i, out; a_j, in) \mid (j : q_i \to q_l, k\gamma) \text{ or} \\ &\quad (j : q_i \to q_l, write(c)) \in P,\ \gamma \in \{+, -, = 0\},\ k \in D,\ c \in T\} \\ &\cup \{\mathtt{1r2} : (q_f, out; a_n, in)\} \\ &\cup \{\mathtt{1r3} : (b_j, out; a'_j, in) \mid j \in I' \setminus I_-\} \\ &\cup \{\mathtt{1r4} : (b_j, out)) \mid j \in I_-\} \\ &\cup \{\mathtt{1r5} : (a_j, out; J_0, in),\ \mathtt{1r6} : (J_1, out; b_j, in) \mid j \in I\} \\ &\cup \{\mathtt{1r7} : (J_0, out; J_1, in)\} \end{aligned}$$

$$
\begin{aligned}
&\cup \quad \{\texttt{1r8} : (a'_j, out; c_k, in) \mid (j : q_i \to q_l, k+) \in P\} \\
&\cup \quad \{\texttt{1r9} : (a'_j, out; c, in) \mid (j : q_i \to q_l, write(c)) \in P\} \\
&\cup \quad \{\texttt{1r10} : (c, out) \mid c \in T\} \\
&\cup \quad \{\texttt{1r11} : (a'_j, out) \mid j \in I_{=0}\} \\
&\cup \quad \{\texttt{1r12} : (d_j, in) \mid j \in I_{=0} \cup I_+\} \\
&\cup \quad \{\texttt{1r13} : (c_k, out; d_j, in),\ \underline{\texttt{1r14}} : (J_3, out; d_j, in) \\
&\qquad \mid (j : q_i \to q_l, k-) \in P\} \\
&\cup \quad \{\texttt{1r15} : (d_j, out; e_j, in) \mid j \in I\} \\
&\cup \quad \{\texttt{1r16} : (e_j, out, q_l, in) \mid (j : q_i \to q_l, k\gamma) \text{ or} \\
&\qquad (j : q_i \to q_l, write(c)) \in P,\ \gamma \in \{+, -, = 0\},\ k \in D,\ c \in T\} \\
&\cup \quad \{\texttt{1r17} : (e_n, out; F, in), \texttt{1r18} : (b_n, out)\} \\
&\cup \quad \{\underline{\texttt{1r19}} : (J_3, out; J_1, in)\} \\
&\cup \quad \{\underline{\texttt{1r20}} : (\#, out),\ \underline{\texttt{1r21}} : (\#, in)\}. \\
R_{2,r} \ = \ &\{\texttt{2r1} : (b_j, out; a_j, in), \texttt{2r2} : (a_j, out; J_2, in), \\
&\underline{\texttt{2r3}} : (a_j, out; J_1, in) \mid j \in I\} \\
&\cup \quad \{\texttt{2r4} : (d_j, out; b_j, in) \mid j \in I\} \\
&\cup \quad \{\texttt{2r5} : (J_2, out; d_j, in) \mid j \in I_- \cup I_+\} \\
&\cup \quad \{\texttt{2r6} : (J_2, out; a'_j, in),\ \texttt{2r7} : (a'_j, out; d_j, in) \mid j \in I_{=0}\} \\
&\cup \quad \{\underline{\texttt{2r7}} : (a'_j, out; c_k, in) \mid j \in I_{=0}\} \\
&\cup \quad \{\underline{\texttt{2r8}} : (\#, out; J_0, in)\ \}.
\end{aligned}
$$

First of all, we mention that if during the phase RUN object J_3 comes to the environment (rules 1r14, 1r19), it remains there forever (**Scenario 0**). Then during the phase END the second symbol J_3 from region 2 will be moved to region 1 that leads to an infinite computation (by rule 1f4, see phase END).

Let us explain the synchronization of a_j coming to the environment and b_j leaving the environment: the first one brings J_0 into region 1 while the latter brings J_1 into the environment; then rule 1r7 returns J_0 and J_1 to their original locations.

If a_j comes to the environment without b_j leaving it, J_1 remains

in region 1 (or 2) and J_0 comes to region 1 (**Scenario 1**), so 2r8 is applied, causing an endless computation since 1r20 or 1r21 is always applicable.

If b_j leaves the environment without a_j coming there, J_0 remains in the environment and J_1 comes there (**Scenario 2**), so 1r19 is applied and symbol J_3 appears in the environment. Thus, the computation never halts, see scenario 0.

We also mention that applying rule 2r3 causes scenario 1. Therefore, in order for a computation to halt, no underlined rules should be applied.

We will now consider the "main" line of computation.

" Increment" -instruction:

$$q_l\mathbf{a_j}a_tc_ke_jJ_0\boxed{\mathbf{q_i}J_1J_2J_3\boxed{b_jd_j\#}} \Rightarrow^{1r1} q_iq_la_ta'_jc_ke_jJ_0\boxed{\mathbf{a_j}J_1J_2J_3\boxed{\mathbf{b_j}d_j\#}}$$
$$\Rightarrow^{2r1} q_iq_l\mathbf{a'_j}a_tc_ke_jJ_0\boxed{\mathbf{b_j}J_1\mathbf{J_2}J_3\boxed{\mathbf{a_j}d_j\#}} \Rightarrow^{1r3,2r2}$$
$$q_iq_la_t\mathbf{b_jc_k}e_j\mathbf{J_0}\boxed{\mathbf{a_j}\mathbf{a'_j}\mathbf{J_1}J_3\boxed{J_2d_j\#}} \quad \text{(A)}$$
$$\Rightarrow^{1r5,1r6,1r8} q_iq_la_ja'_ja_te_j\mathbf{J_1}\boxed{\mathbf{b_j}c_k\mathbf{J_0}J_3\boxed{J_2\mathbf{d_j}\#}} \Rightarrow^{1r7,2r4}$$
$$q_iq_la_ja'_ja_t\mathbf{e_j}J_0\boxed{c_k\mathbf{d_j}J_1J_3\boxed{b_jJ_2\#}} \Rightarrow^{1r15}$$
$$q_i\mathbf{q_l}a_ja'_ja_t\mathbf{d_j}J_0\boxed{c_k\mathbf{e_j}J_1J_3\boxed{b_jJ_2\#}} \Rightarrow^{1r12,1r16}$$
$$q_ia_ja'_j\mathbf{a_t}e_jJ_0\boxed{\mathbf{q_l}c_k\mathbf{d_j}J_1J_3\boxed{b_j\mathbf{J_2}\#}} \Rightarrow^{1r1,2r5}$$
$$q_iq_la_ja'_je_jJ_0\boxed{\mathbf{a_t}c_kJ_1J_2J_3\boxed{b_jd_j\#}}$$

In that way, q_i is replaced by q_l and c_k is moved from the environment into region 1. In configuration (A) rule 1r9 may be applied instead of rule 1r8 and after that rule 1r10, so terminal symbol c will be sent to the environment. Thus we model instruction $j : (q_i \to q_l, write(c))$ of counter automaton M. Symbol d_j returns to region 2 in the first step of the simulation of the next instruction (the last step of the illustration).

Notice that symbols a_j, b_j, a'_j, d_j, e_j, J_2, J_1, J_0 have returned to their original positions.

" Decrement" -instruction:

$$q_l\mathbf{a_j}a_te_jJ_0\boxed{\mathbf{q_i}c_kJ_1J_2J_3\boxed{b_jd_j\#}} \Rightarrow^{\texttt{1r1}} q_iq_la_te_jJ_0\boxed{\mathbf{a_j}c_kJ_1J_2J_3\boxed{\mathbf{b_j}d_j\#}}$$
$$\Rightarrow^{\texttt{2r1}} q_iq_la_te_jJ_0\boxed{\mathbf{b_j}c_kJ_1\mathbf{J_2}J_3\boxed{\mathbf{a_j}d_j\#}} \Rightarrow^{\texttt{1r4,2r2}}$$
$$q_iq_la_t\mathbf{b_j}e_j\mathbf{J_0}\boxed{\mathbf{a_j}c_k\mathbf{J_1}J_3\boxed{d_jJ_2\#}} \Rightarrow^{\texttt{1r5,1r6}} q_iq_la_ja_te_j\mathbf{J_1}\boxed{\mathbf{b_j}c_k\mathbf{J_0}J_3\boxed{\mathbf{d_j}J_2\#}}$$
$$\Rightarrow^{\texttt{1r7,2r4}} q_iq_la_ja_t\mathbf{e_j}J_0\boxed{\mathbf{d_j}c_kJ_1J_3\boxed{b_jJ_2\#}} \Rightarrow^{\texttt{1r15}}$$
$$q_i\mathbf{q_l}a_ja_t\mathbf{d_j}J_0\boxed{\mathbf{c_k}\mathbf{e_j}J_1J_3\boxed{b_jJ_2\#}} \quad \text{(B)}$$
$$\Rightarrow^{\texttt{1r13,1r16}} q_ia_j\mathbf{a_t}c_ke_jJ_0\boxed{\mathbf{q_l}\mathbf{d_j}J_1J_3\boxed{b_j\mathbf{J_2}\#}} \Rightarrow^{\texttt{1r1,2r5}}$$
$$q_iq_la_jc_ke_jJ_0\boxed{\mathbf{a_t}J_1J_2J_3\boxed{b_jd_j\#}}$$

In the way described above, q_i is replaced by q_l and c_k is removed from region 1 to the environment. If symbol c_k is absent in region 1 in configuration (B) it enforces the applying of rule 1r14 that leads to an infinite computation. Symbol d_j returns to region 2 in the first step of the simulation of the next instruction (the last step of the illustration). Notice that symbols a_j, b_j, d_j, e_j, J_2, J_1, J_0 have returned to their original positions.

" Test for zero" -instruction:

q_i is replaced by q_l if there is no c_k in region 1, otherwise a'_j in region 2 exchanges with c_k in region 1 and the computation will never stop.

(i) *There is no c_k in region 1:*

$$q_la_t\mathbf{a_j}a'_je_jJ_0\boxed{\mathbf{q_i}J_1J_2J_3\boxed{b_jd_j\#}} \Rightarrow^{\texttt{1r1}} q_iq_la_ta'_je_jJ_0\boxed{\mathbf{a_j}J_1J_2J_3\boxed{\mathbf{b_j}d_j\#}}$$
$$\Rightarrow^{\texttt{2r1}} q_iq_la_t\mathbf{a'_j}e_jJ_0\boxed{\mathbf{b_j}J_1\mathbf{J_2}J_3\boxed{\mathbf{a_j}d_j\#}} \Rightarrow^{\texttt{1r3,2r2}}$$

$$q_i q_l a_t \mathbf{b_j} e_j \mathbf{J_0} \boxed{\mathbf{a_j} \mathbf{a'_j} \mathbf{J_1} J_3 \boxed{\mathbf{J_2} d_j \#}} \Rightarrow^{\texttt{1r5,1r6,2r6}}$$

$$q_i q_l a_j a_t e_j \mathbf{J_1} \boxed{\mathbf{b_j} \mathbf{J_0} J_2 J_3 \boxed{a'_j \mathbf{d_j} \#}} \qquad \text{(C)}$$

$$\Rightarrow^{\texttt{1r7,2r4}} q_i q_l a_j a_t \mathbf{e_j} J_0 \boxed{\mathbf{d_j} J_1 J_2 J_3 \boxed{a'_j b_j \#}} \Rightarrow^{\texttt{1r15}}$$

$$q_i \mathbf{q_l} a_j a_t \mathbf{d_j} J_0 \boxed{\mathbf{e_j} J_1 J_2 J_3 \boxed{a'_j b_j \#}} \Rightarrow^{\texttt{1r12,1r16}}$$

$$q_i a_j \mathbf{a_t} e_j J_0 \boxed{\mathbf{q_l} \mathbf{d_j} J_1 J_2 J_3 \boxed{\mathbf{a'_j} b_j \#}} \Rightarrow^{\texttt{1r1,2r7}} q_i q_l a_j e_j J_0 \boxed{\mathbf{a_t} \mathbf{a'_j} J_1 J_2 J_3 \boxed{b_j d_j \#}}$$

In this case, q_i is replaced by q_l. Notice that symbols a_j, a'_j, b_j, d_j, e_j, J_2, J_1, J_0 have returned to their original positions. Symbol d_j returns to region 2 in the first step of the simulation of the next instruction (the last step of the illustration) and symbol a'_j returns to the environment in the second step of the simulation of the next instruction.

(ii) *There is some c_k in region 1:*
Consider configuration (C) with object c_k in region 1:

$$q_i q_l a_j a_t e_j \mathbf{J_1} \boxed{\mathbf{b_j} \mathbf{c_k} \mathbf{J_0} J_2 J_3 \boxed{\mathbf{a'_j} \mathbf{d_j} \#}} \Rightarrow^{\texttt{1r7,2r4,2r7}}$$

$$q_i q_l a_j a_t \mathbf{e_j} J_0 \boxed{\mathbf{a'_j} \mathbf{d_j} J_1 J_2 J_3 \boxed{b_j c_k \#}} \Rightarrow^{\texttt{1r11,1r15}}$$

$$q_i \mathbf{q_l} a_j a'_j a_t \mathbf{d_j} J_0 \boxed{\mathbf{e_j} J_1 J_2 J_3 \boxed{b_j c_k \#}} \Rightarrow^{\texttt{1r12,1r16}}$$

$$q_i a_j a'_j \mathbf{a_t} \mathbf{e_j} J_0 \boxed{\mathbf{q_l} \mathbf{d_j} J_1 J_2 J_3 \boxed{b_j c_k \#}}$$

Now rule `1r15` again will be applied and two symbols a_t, a_s appear in several steps in region 1 that leads to an infinite computation (see scenario 1).

Let us consider the symbols from region 2 visiting the environment and going back: $2 \to 1 \to 0 \to 1 \to 2$ (b_j, d_j for all instructions) and the symbols from the environment visiting region 2 and going back: $0 \to 1 \to 2 \to 1 \to 0$. The latter ones are: a_j for all instructions, and also $\{a'_j \mid j \in I_{=0}\}$.

Then we have to argue that if they **Return** to their "home region" ($2 \to 1 \to 2$ or $0 \to 1 \to 0$) or **Repeat** their visit to the "opposite region" before returning "home" ($2 \to 1 \to 0 \to 1 \to 0$ or $0 \to 1 \to 2 \to 1 \to 2$), an infinite computation is unavoidable, or such case is not possible.

$a_j, j \in I$. Return: see scenario 1; repeat: impossible without b_j.

$a'_j, j \in I_{=0}$. Return: as d_j cannot come back to region 2 (is not possible to apply rule `2r6`), it again goes to the environment (see d_j, repeat); repeat: impossible without J_2.

$b_j, j \in I$. Return: if a_j comes to the environment, scenario 1 takes place. If a_j returns to region 2, rule `2r3` is applied; repeat: symbol J_1 will be moved to the environment, where it changes with J_3 (rule `1r19`), so it leads to an infinite computation (scenario 0).

$d_j, j \in I$. Return: e_j stays in the environment, the simulation stops and the computation never ends due to `2s1`, `2s2`; repeat: two symbols a_t, a_s appear in several steps in region 1 that leads to an infinite computation (see scenario 1).

3. END.

$$\begin{aligned} R_{1,f} &= \{\texttt{1f1} : (Y_1, out; Y_3, in),\ \texttt{1f2} : (Y_2, out)\} \\ &\cup \{\texttt{1f3} : (X, out; Y_2, in),\ \texttt{1f4} : (J_3, out, J_3, in)\}. \\ R_{2,f} &= \{\texttt{2f1} : (Y_1, out; F, in),\ \texttt{2f2} : (Y_2, out; F, in),\ \texttt{2f3} : (F, out)\} \\ &\cup \{\texttt{2f4} : (J_3, out; Y_3, in)\}. \end{aligned}$$

Once the counter automaton reaches the final state, q_f is in region 1 and it exchanges with object a_n (rule `1r2`), object F will be moved to region 1 in several steps. It takes Y_1, Y_2 and J_3 to region 1, in either order. The duty of Y_2 is to bring X to the environment (the object X can oscillate for indefinite time, but we are interested in halting computations). The duty of Y_1 is to bring J_3 from region 2 to region 1. If during the previous steps of simulation of counter automaton M object J_3 from region 1 was moved to the environment (by rules `1r14` or `1r19`), rule `1f4` will be applied, leading to an infinite computation.

If on the previous steps of simulation of counter automaton M object J_1 was moved to region 2 (by rules 2r3), rule 2r8 will be applied in several steps, and the computation never halts.

Thus, at the end of a terminating computation terminal word $w \in T$ will be sent to the environment. □

Theorem 2 $LOP_2(sym_2) = RE$.

Proof. Our proof is based on the construction introduced in Theorem 2 from [2]. As in the proof of Theorem 1 we simulate a counter automaton $M = (d, T, Q, q_0, q_f, P)$ which starts with empty counters. Again we suppose that all instructions from P are labelled in a one-to-one manner with elements of $\{1, \ldots, n\} = I$ and that I is the disjoint union of $\{n\}$ as well as I_+, I_-, and $I_{=0}$, where by I_+, I_-, and $I_{=0}$ we denote the set of labels for the "increment", "decrement", and "test for zero" instructions, respectively (recall that "increment"-instructions include also instructions operating on the tape). Moreover, we define $I' = I \setminus \{n\}$ and $Q' = Q \setminus \{q_0\}$. We also suppose that there is only one instruction with initial state q_0 (labeled with number 1) and only one instruction with the final state q_f (labeled with number n).

We construct the P system Π_2 as follows:

$$
\begin{aligned}
\Pi_2 &= (O, T, E, [_1\ [_2\ \ ]_2\ ]_1, w_1, w_2, R_1, R_2),\\
O &= E \cup \{\#_1, \#_2, \$, f\} \cup Q \cup \{b_j, g_j \mid j \in I\} \cup \{g'_j \mid j \in I'\},\\
E &= T \cup \{a_j, a'_j, d_j, d'_j \mid j \in I\} \cup C,\\
C &= \{c_i \mid 1 \le i \le d\},\\
w_1 &= \#_2 \$ f q_0 a_1 \prod_{j \in I} b_j,\\
w_2 &= \#_1 \prod_{q_i \in Q'} q_i \prod_{j \in I} g_j \prod_{j \in I'} g'_j,\\
R_i &= R_{i,s} \cup R_{i,r} \cup R_{i,f}, i \in \{1, 2\}.
\end{aligned}
$$

We code the counter automaton as follows: The environment will hold the current state of the automaton, represented by a symbol $q_i \in$

Q, membrane 1 will hold the value of all counters, represented by the number of occurrences of symbols c_k, $k \in D$, where $D = \{1, ..., d\}$. We also use the following idea realized by phase START below: in our system we have a symbol $\#_2$ moving from the environment to membrane 1 and back in an infinite loop. This loop can only be stopped if all stages have completed correctly. Otherwise, the computation will never stop.

Again as in Theorem 1 we split our proof into several parts that depend on the logical separation of the behavior of the system. We will present the rules and the initial symbols for each part, but we remark that the system that we present is the union of all these parts.

The rules R_i are given by three phases:

1. START: preparation of the system for the computation.
2. RUN: simulation of instructions of the counter automaton.
3. END: terminating the computation.

1. START.

$$\begin{aligned} R_{1,s} &= \{\texttt{1s1} : (\#_2, out), \texttt{1s2} : (\#_2, in)\}, \\ R_{2,s} &= \emptyset. \end{aligned}$$

Notice that system Π_2 begins its functioning by applying rule `1s1` and moving objects $q_0 a_1$ to region 2 (see phase RUN below). Thus system Π_2 starts to simulate the counter automaton M.

2. RUN.

$$\begin{aligned} R_{1,r} &= \{\texttt{1r1} : (q_i a_j, in) \mid (j : q_i \to q_l, k\gamma) \text{ or} \\ &\quad (j : q_i \to q_l, write(c)) \in P,\ \gamma \in \{+, -, = 0\},\ k \in D,\ c \in T\} \\ &\cup \{\texttt{1r2} : (b_j g_j, out) \mid j \in I_+ \cup I_{=0}\} \\ &\cup \{\texttt{1r3} : (c_k b_j, in) \mid (j : q_i \to q_l, k+) \in P, k \in D\} \\ &\cup \{\texttt{1r4} : (g_j c_k, out) \mid (j : q_i \to q_l, k-) \in P, k \in D\} \end{aligned}$$

$$
\begin{aligned}
&\cup \ \{\texttt{1r5} : (a'_j g_j, in) \mid j \in I'\} \\
&\cup \ \{\texttt{1r6} : (\#_1, out), \texttt{1r7} : (\#_1, in)\} \\
&\cup \ \{\texttt{1r8} : (d_j b_j, in) \mid j \in I_{=0}\} \\
&\cup \ \{\texttt{1r9} : (d_j c_k, out) \mid (j : q_i \to q_l, k = 0) \in P, k \in D\} \\
&\cup \ \{\texttt{1r10} : (a'_j q_l, out) \mid (j : q_i \to q_l, k\gamma) \text{ or} \\
&\quad (j : q_i \to q_l, write(c)) \in P,\ \gamma \in \{+, -\},\ k \in D,\ c \in T\} \\
&\cup \ \{\texttt{1r11} : (a'_j g'_j, out), \texttt{1r12} : (d'_j g'_j, in), \\
&\quad \texttt{1r13} : (d'_j, out) \mid j \in I_{=0}\} \\
&\cup \ \{\texttt{1r14} : (d_j q_l, out) \mid (j : q_i \to q_l, k = 0) \in P, k \in D\}, \\
&\cup \ \{\texttt{1r15} : (c b_j, in) \mid (j : q_i \to q_l, write(c)) \in P\} \\
&\cup \ \{\texttt{1r16} : (c, out) \mid c \in T\}, \\
R_{2,r} \ = \ &\{\texttt{2r1} : (a_j b_j, in) \mid j \in I'\} \\
&\cup \ \{\texttt{2r2} : (q_i, in) \mid q_i \in Q\} \\
&\cup \ \{\texttt{2r3} : (b_j g_j, out) \mid j \in I'\} \\
&\cup \ \{\texttt{2r4} : (a'_j \$, in) \mid j \in I\} \\
&\cup \ \{\texttt{2r5} : (\#_1 \$, out)\} \\
&\cup \ \{\texttt{2r6} : (a'_j g_j, in) \mid j \in I'\} \\
&\cup \ \{\texttt{2r7} : (a'_j q_l, out) \mid (j : q_i \to q_l, k\gamma) \text{ or} \\
&\quad (j : q_i \to q_l, write(c)) \in P,\ \gamma \in \{+, -\},\ k \in D,\ c \in T\} \\
&\cup \ \{\texttt{2r8} : (a'_j g'_j, out) \mid j \in I_{=0}\} \\
&\cup \ \{\texttt{2r9} : (d'_j g'_j, in) \mid j \in I_{=0}\} \\
&\cup \ \{\texttt{2r10} : (d'_j q_l, out) \mid (j : q_i \to q_l, k = 0) \in P, k \in D\}.
\end{aligned}
$$

Notice that the starting configuration of Π_2 corresponds to the result of the first step of the simulation of the starting instruction (`1r1` is "already made").

"Increment" instruction:

$$a'_j c_k \mathbf{q_i a_j} \boxed{b_j \$ \boxed{g_j q_l \#_1}} \Rightarrow^{\texttt{1r1}} a'_j c_k \boxed{\mathbf{q_i a_j b_j} \$ \boxed{g_j q_l \#_1}} \Rightarrow^{\texttt{2r1,2r2}}$$

$$a'_j c_k \boxed{\$\boxed{q_i a_j \mathbf{b_j g_j} q_l \#_1}} \Rightarrow^{\texttt{2r3}} a'_j c_k \boxed{\mathbf{b_j g_j}\$\boxed{q_i a_j q_l \#_1}} \Rightarrow^{\texttt{1r2}}$$
$$\mathbf{c_k b_j g_j a'_j}\boxed{\$\boxed{q_i a_j q_l \#_1}} \quad \text{(A)}$$
$$\Rightarrow^{\texttt{1r3,1r5}} \boxed{c_k b_j g_j a'_j \$\boxed{q_i a_j q_l \#_1}}$$

Now there are two variants of computations (depending on the application of rule `1r2` or rule `2r6`):

a) Applying rule `1r2`:

$$c_k a'_j \boxed{c_k \mathbf{b_j g_j a'_j}\$\boxed{q_i a_j q_l \#_1}} \Rightarrow^{\texttt{1r2,2r4}}$$
$$\mathbf{c_k b_j g_j a'_j}\boxed{c_k \boxed{q_i a_j \mathbf{a'_j q_l}\$\#_\mathbf{1}}} \Rightarrow^{\texttt{1r5,1r3,2r5,2r7}}$$
$$\boxed{b_j g_j a'_j a'_j q_l c_k c_k \$\#_1 \boxed{q_i a_j}} \quad \cdots$$

After that application of rules `1r6` and `1r7` leads to infinite computation.

b) Applying rule `2r6`:

$$\boxed{c_k b_j \mathbf{g_j a'_j}\$\boxed{q_i a_j q_l \#_1}} \Rightarrow^{\texttt{2r6}} \boxed{c_k b_j \$\boxed{q_i a_j g_j \mathbf{a'_j q_l} \#_1}} \Rightarrow^{\texttt{2r7}}$$
$$\boxed{\mathbf{a'_j q_l} c_k b_j \$\boxed{q_i a_j g_j \#_1}} \Rightarrow^{\texttt{1r10}} a'_j q_l \boxed{c_k b_j \$\boxed{q_i a_j g_j \#_1}}$$

q_i is replaced by q_l and c_k is moved into region 1. In configuration (A) rule `1r15` may be applied instead of rule `1r3` and after that rule `1r16`, so terminal symbol c will be sent to the environment. Thus we model instruction $j : (q_i \to q_l, write(c))$ of counter automaton M.

"Decrement" instruction:

$$a'_j \mathbf{q_i a_j} \boxed{c_k b_j \$\boxed{g_j q_l \#_1}} \Rightarrow^{\texttt{1r1}} a'_j \boxed{c_k \mathbf{q_i a_j b_j}\$\boxed{g_j q_l \#_1}} \Rightarrow^{\texttt{2r1,2r2}}$$
$$a'_j \boxed{c_k \$\boxed{q_i a_j \mathbf{b_j g_j} q_l \#_1}} \Rightarrow^{\texttt{2r3}} a'_j \boxed{\mathbf{c_k g_j} b_j \$\boxed{q_i a_j q_l \#_1}} \Rightarrow^{\texttt{1r4}}$$

$$c_k \mathbf{g_j a'_j} \boxed{b_j \$ \boxed{q_i a_j q_l \#_1}} \Rightarrow^{\texttt{1r5}} c_k \boxed{b_j g_j a'_j \$ \boxed{q_i a_j q_l \#_1}}$$

Now there are two variants of computations (depending on the application of rule `1r4` or rule `2r6`).

c) Applying rule `1r4`:

$$a'_j c_k \boxed{b_j \mathbf{c_k g_j a'_j} \$ \boxed{q_i a_j q_l \#_1}} \Rightarrow^{\texttt{1r4,2r4}} \mathbf{a'_j g_j} c_k c_k \boxed{b_j \boxed{q_i a_j \mathbf{a'_j q_l} \$ \#_1}}$$
$$\Rightarrow^{\texttt{1r5,2r5,2r7}} c_k c_k \boxed{a'_j a'_j g_j q_l b_j \$ \#_1 \boxed{q_i a_j}} \cdots$$

After that the application of rules `1r6` and `1r7` leads to an infinite computation.

d) Applying rule `2r6`:

$$c_k \boxed{b_j \mathbf{g_j a'_j} \$ \boxed{q_i a_j q_l \#_1}} \Rightarrow^{\texttt{2r6}} c_k \boxed{b_j \$ \boxed{q_i a_j g_j \mathbf{a'_j q_l} \#_1}} \Rightarrow^{\texttt{2r7}}$$
$$c_k \boxed{\mathbf{a'_j q_l} b_j \$ \boxed{q_i a_j g_j \#_1}} \Rightarrow^{\texttt{1r10}} c_k a'_j q_l \boxed{b_j \$ \boxed{q_i a_j g_j \#_1}}$$

In that way, q_i is replaced by q_l and c_k is removed from region 1.

"Test for zero" instruction:

q_i is replaced by q_l if there is no c_k in region 1 (case e)), otherwise the computation will never stop (case f)).

Case e):

$$a'_j d_j d'_j \mathbf{q_i a_j} \boxed{b_j \$ \boxed{g_j g'_j q_l \#_1}} \Rightarrow^{\texttt{1r1}} a'_j d_j d'_j \boxed{\mathbf{q_i a_j b_j} \$ \boxed{g_j g'_j q_l \#_1}} \Rightarrow^{\texttt{2r1,2r2}}$$
$$a'_j d_j d'_j \boxed{\$ \boxed{q_i a_j \mathbf{b_j g_j} g'_j q_l \#_1}} \Rightarrow^{\texttt{2r3}} a'_j d_j d'_j \boxed{\mathbf{b_j g_j} \$ \boxed{q_i a_j g'_j q_l \#_1}} \Rightarrow^{\texttt{1r2}}$$
$$d'_j \mathbf{b_j d_j g_j a'_j} \boxed{\$ \boxed{q_i a_j g'_j q_l \#_1}} \Rightarrow^{\texttt{1r8,1r5}} d'_j \boxed{d_j b_j g_j a'_j \$ \boxed{q_i a_j g'_j q_l \#_1}}$$

Again there are two variants of computations, depending on the application of rule `1r2` or rule `2r6`, where applying rule `1r2` leads to an infinite computation (see case a)). Hence, we only consider the case of applying rule `2r6`:

$$d'_j\boxed{d_jb_j\mathbf{g_j}\mathbf{a'_j}\$\boxed{q_ia_jg'_jq_l\#_1}} \Rightarrow^{\texttt{2r6}} d'_j\boxed{d_jb_j\$\boxed{q_ia_jg_j\mathbf{a'_j}\mathbf{g'_j}q_l\#_1}} \Rightarrow^{\texttt{2r8}}$$
$$d'_j\boxed{\mathbf{a'_j}\mathbf{g'_j}d_jb_j\$\boxed{q_ia_jg_jq_l\#_1}} \Rightarrow^{\texttt{1r11}} a'_j\mathbf{g'_j}\mathbf{d'_j}\boxed{d_jb_j\$\boxed{q_ia_jg_jq_l\#_1}} \Rightarrow^{\texttt{1r12}}$$
$$a'_j\boxed{d_jb_j\$\mathbf{g'_j}\mathbf{d'_j}\boxed{q_ia_jg_jq_l\#_1}} \Rightarrow^{\texttt{2r9}} a'_j\boxed{d_jb_j\$\boxed{q_ia_jg_jg'_j\mathbf{d'_j}\mathbf{q_l}\#_1}} \Rightarrow^{\texttt{2r10}}$$
$$a'_j\boxed{\mathbf{d'_j}\mathbf{q_l}\mathbf{d_j}b_j\$\boxed{q_ia_jg_jg'_j\#_1}} \Rightarrow^{\texttt{1r13,1r14}} a'_jd_jd'_jq_l\boxed{b_j\$\boxed{q_ia_jg_jg'_j\#_1}}$$

Thus, q_i is replaced by q_l.

Case f):

$$a'_jd_jd'_j\mathbf{q_i}\mathbf{a_j}\boxed{c_kb_j\$\boxed{g_jg'_jq_l\#_1}} \Rightarrow^{\texttt{1r1}} a'_jd_jd'_j\boxed{c_k\mathbf{q_i}\mathbf{a_j}\mathbf{b_j}\$\boxed{g_jg'_jq_l\#_1}} \Rightarrow^{\texttt{2r1,2r2}}$$
$$a'_jd_jd'_j\boxed{c_k\$\boxed{q_ia_j\mathbf{b_j}\mathbf{g_j}g'_jq_l\#_1}} \Rightarrow^{\texttt{2r3}} a'_jd_jd'_j\boxed{c_k\mathbf{b_j}\mathbf{g_j}\$\boxed{q_ia_jg'_jq_l\#_1}} \Rightarrow^{\texttt{1r2}}$$
$$d'_j\mathbf{b_j}\mathbf{d_j}\mathbf{g_j}\mathbf{a'_j}\boxed{c_k\$\boxed{q_ia_jg'_jq_l\#_1}} \Rightarrow^{\texttt{1r8,1r5}} d'_j\boxed{\mathbf{c_k}\mathbf{d_j}\mathbf{g_j}\mathbf{a'_j}b_j\$\boxed{q_ia_jg'_jq_l\#_1}}$$
$$\Rightarrow^{\texttt{1r9,2r6}}$$
$$c_kd_jd'_j\boxed{b_j\$\boxed{q_ia_jg_j\mathbf{a'_j}\mathbf{g'_j}q_l\#_1}} \Rightarrow^{\texttt{2r8}} c_kd_jd'_j\boxed{\mathbf{a'_j}\mathbf{g'_j}b_j\$\boxed{q_ia_jg_jq_l\#_1}} \Rightarrow^{\texttt{1r11}}$$
$$a'_jc_kd_j\mathbf{g'_j}\mathbf{d'_j}\boxed{b_j\$\boxed{q_ia_jg_jq_l\#_1}} \Rightarrow^{\texttt{1r12}} a'_jc_kd_j\boxed{b_j\$\mathbf{g'_j}\mathbf{d'_j}\boxed{q_ia_jg_jq_l\#_1}} \Rightarrow^{\texttt{2r9}}$$
$$a'_jc_kd_j\boxed{b_j\$\boxed{q_ia_jg_jg'_j\mathbf{d'_j}\mathbf{q_l}\#_1}} \Rightarrow^{\texttt{2r10}} a'_jc_kd_j\boxed{\mathbf{d'_j}q_lb_j\$\boxed{q_ia_jg_jg'_j\#_1}} \Rightarrow^{\texttt{1r13}}$$
$$a'_jc_kd_jd'_j\boxed{q_lb_j\$\boxed{q_ia_jg_jg'_j\#_1}}$$

Further we continue our work only by applying the rules `1s1` and `1s2`, thus, the computation will never stop.

3. END.

$$\begin{aligned} R_{1,f} &= \{\texttt{1f1} : (q_f a_n, in)\} \\ R_{2,f} &= \{\texttt{2f1} : (\#_2 g_n, in), \texttt{2f2} : (f a_n, in), \texttt{2f3} : (f g_n, out)\}. \end{aligned}$$

Object $\#_2$ is moved to region 2, so we stop without continuing the loop.

Thus, at the end of a terminating computation terminal word $w \in T$ will be sent to the environment. □

4 Discussion

Consider P systems with two membranes and symport of weight at most two or symport/antiport rules of weight one. It has been shown in [5] for both classes that they generate all recursively enumerable sets of positive integers and some finite sets of non-negative integers containing zero. In the present work we look at generating languages by the same systems, taking sequences of terminal objects sent into the environment as the output of the system. We have shown that both classes are computationally complete.

The case of one membrane remains to be investigated. Unlike generating numbers, it is possible to generate infinite sets:

Example 2

$$\begin{aligned} \Pi_3 &= (O = \{a, b\}, T = \{a\}, E = O, \mu = [_1 \ ]_1, w_1 = b, R_1), \\ R_1 &= \{(b, out; a, in), (a, out; b, in), (a, out)\}; \end{aligned}$$

$$\begin{aligned} \Pi_4 &= (O = \{a, b\}, T = \{a\}, E = O, \mu = [_1 \ ]_1, w_1 = ba, R_1), \\ R_1 &= \{(ba, out), (ba, in), (b, in)\}; \end{aligned}$$

$$L(\Pi_1) = L(\Pi_2) = a^+$$

Let us return to the way the behaviour of the system is defined. Notice that $T \subseteq E$ holds for both proofs (the output symbols are available in the environment in unbounded quantity). Therefore, when a terminal symbol is ejected into the environment, it is not important whether it is allowed to re-enter the system (it has been registered and "released") or not (it remains on the "tape"). In the next paragraph, however, we assume the first case (the other one is more restricted).

We come back to systems with multiple membranes. Clearly, such computational power is only possible when we register just the terminal symbols. What happens if we require

$$T = \{a \in O \mid |u|_a > 0,\ (u, out) \in R_1 \text{ or } (u, out; v, in) \in R_1\},$$

i.e., all symbols that may be sent into the environment constitute the terminal alphabet? It turns out that the total number of objects present inside the system cannot exceed the initial value plus the size of the output word (multiplied by a constant if we drop the restriction on the weight of symport/antiport rules). Following the argument that the total number of configurations with at most N objects is a polynomial with respect to N (depending on the size of the alphabet and the number of membranes), it is not difficult to see that the condition above bounds the power of such systems by $LOGSPACE$ [15] (i.e., that of Turing machines with a working tape of size $O(\log n)$, where n is the size of the output word).

Finally, it is possible to define P systems accepting words by registering the terminal symbols that enter the skin membrane from the environment. In principal, similar computational completeness results may be obtained. We did not include such results in this article because it would be non-deterministic acceptance and because there are multiple ways one could define the behaviour of such P systems.

The authors thank Dr. Rudolf Freund for the useful discussions of the definitions.

References

[1] A. Alhazov, R. Freund, Yu. Rogozhin. *Computational Power of Symport/Antiport: History, Advances, and Open Problems.* International Workshop in Membrane Computing (WMC6) (R. Freund, G. Lojka, M. Oswald, Gh. Păun, Eds.) Vienna Institute of Technology, Vienna (2005) 44–78 and also Membrane Computing, International Workshop, WMC 2005, Vienna, 2005, Revised Selected and Invited Papers (R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, Eds.), *Lecture Notes in Computer Science*, Springer **3850** (2006) 1–30.

[2] A. Alhazov, R. Freund, Yu. Rogozhin. *Some Optimal Results on Communicative P Systems with Minimal Cooperation.* Cellular Computing (Complexity Aspects), ESF PESC Exploratory Workshop (M.A. Gutiérrez-Naranjo, Gh. Păun, M.J. Pérez-Jiménez, Eds.), Fénix Editora, Sevilla, (2005) 23–36.

[3] A. Alhazov, M. Margenstern, V. Rogozhin, Yu. Rogozhin, S. Verlan. *Communicative P Systems with Minimal Cooperation.* Membrane Computing, International Workshop, WMC 2004, Milan, 2004, Revised Selected and Invited Papers (G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa, Eds.) *Lecture Notes in Computer Science*, Springer **3365** (2005) 161–177.

[4] A. Alhazov, Yu. Rogozhin. *Minimal Cooperation in Symport/ Antiport P Systems with One Membrane.* Third Brainstorming Week on Membrane Computing (M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-Campero, D. Sburlan, Eds.) RGNC TR *01/2005*, University of Seville, Fénix Editora, Sevilla (2005) 29–34.

[5] A. Alhazov, Yu. Rogozhin. *Towards a Characterization of P Systems with Minimal Symport/Antiport and Two Membranes.* In: Pre-proc. of the 7th Workshop on Membrane Computing, WMC7, 17 – 21 July, 2006, Lorentz Center, Leiden (2006) 102–117. Revised Selected and Invited Papers in: *Lecture Notes in Computer Science*, Springer **4361** (2006) 135–153.

[6] A. Alhazov, Yu. Rogozhin, S. Verlan. *Symport/Antiport Tissue P Systems with Minimal Cooperation.* Cellular Computing

(Complexity Aspects), ESF PESC Exploratory Workshop (M.A. Gutiérrez-Naranjo, Gh. Păun, M.J. Pérez-Jiménez, Eds.), Fénix Editora, Sevilla (2005) 37–52.

[7] F. Bernardini, M. Gheorghe. *On the Power of Minimal Symport/Antiport.* Workshop on Membrane Computing, WMC 2003 (A. Alhazov, C. Martín-Vide, Gh. Păun, Eds.), Tarragona, 2003, TR *28/03*, Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona (2003) 72–83.

[8] F. Bernardini, A. Păun. *Universality of Minimal Symport/ Antiport: Five Membranes Suffice.* Membrane Computing, International Workshop, WMC 2003, Tarragona, Revised Papers (C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa, Eds.), *Lecture Notes in Computer Science*, Springer **2933** (2004) 43–45.

[9] R. Freund, M. Oswald. *GP Systems with Forbidding Context.* Fundamenta Informaticae, IOS Press **49**, 1–3 (2002) 81–102.

[10] R. Freund, M. Oswald. *P Systems with Activated/Prohibited Membrane Channels.* Membrane Computing International Workshop, WMC-CdeA 02, Curtea de Argeş, 2002. Revised Papers (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds.), *Lecture Notes in Computer Science*, Springer **2597** (2003) 261–268.

[11] R. Freund, A. Păun. *Membrane Systems with Symport/Antiport: Universality Results.* Membrane Computing International Workshop, WMC-CdeA 02, Curtea de Argeş, 2002. Revised Papers (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds.), *Lecture Notes in Computer Science*, Springer **2597** (2003) 270–287.

[12] P. Frisco. *About P Systems with Symport/Antiport.* Second Brainstorming Week on Membrane Computing (Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, Eds), TR *01/2004*, Research Group on Natural Computing, University of Seville (2004) 224–236.

[13] P. Frisco, H.J. Hoogeboom. *P Systems with Symport/Antiport Simulating Counter Automata.* Acta Informatica, Springer **41**, 2–3 (2004) 145–170.

[14] P. Frisco, H.J. Hoogeboom. *Simulating Counter Automata by P Systems with Symport/Antiport.* Membrane Computing International Workshop, WMC-CdeA 02, Curtea de Argeş, 2002. Revised Papers (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds.), *Lecture Notes in Computer Science*, Springer **2597** (2003) 288–301.

[15] J.Gruska. *Foundations of Computing.* International Thomson Computer Press (1997).

[16] L. Kari, C. Martín-Vide, A. Păun. *On the Universality of P Systems with Minimal Symport/Antiport Rules.* Aspects of Molecular Computing - Essays dedicated to Tom Head on the occasion of his 70th birthday, *Lecture Notes in Computer Science*, Springer **2950** (2004) 254–265.

[17] M. Margenstern, V. Rogozhin, Yu. Rogozhin, S. Verlan. *About P Systems with Minimal Symport/Antiport Rules and Four Membranes.* Fifth Workshop on Membrane Computing (WMC5), (G. Mauri, Gh. Păun, C. Zandron, Eds.), Universitá di Milano-Bicocca, Milan (2004) 283–294.

[18] C. Martín-Vide, A. Păun, Gh. Păun. *On the Power of P Systems with Symport Rules,* Journal of Universal Computer Science, **8**, 2 (2002) 317–331.

[19] M.L. Minsky. *Finite and Infinite Machines.* Prentice Hall, Englewood Cliffs, New Jersey (1967).

[20] A. Păun, Gh. Păun. *The Power of Communication: P Systems with Symport/Antiport.* New Generation Computing, Japan, **20** (2002) 295–305.

[21] Gh. Păun. *Computing with Membranes.* Journal of Computer and Systems Science, **61** (2000) 108–143.

[22] Gh. Păun. *Membrane Computing. An Introduction.* Springer-Verlag (2002).

[23] Gh. Păun. *Further Twenty Six Open Problems in Membrane Computing (2005).* Third Brainstorming Week on Membrane Computing (M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-

Campero, D. Sburlan, Eds.) RGNC TR *01/2005*, University of Seville, Fénix Editora, Sevilla (2005) 249–262.

[24] Gh.Păun. *2006 Research Topics in Membrane Computing.* Fourth Brainstorming Week on Membrane Computing, vol. *1* (M.A. Gutiérrez-Naranjo, Gh. Păun, A. Riscos-Núñez, F.J. Romero-Campero, Eds.), Fénix Edit., Sevilla (2006), 235–251.

[25] G. Rozenberg, A. Salomaa (Eds.). *Handbook of Formal Languages* (3 volumes). Springer-Verlag, Berlin (1997).

[26] Gy. Vaszil. *On the Size of P Systems with Minimal Symport/ Antiport.* Fifth Workshop on Membrane Computing (WMC5) (G. Mauri, Gh. Păun, C. Zandron, Eds.), Universitá di Milano-Bicocca, Milan (2004) 422–431.

[27] S. Verlan. *Optimal Results on Tissue P Systems with Minimal Symport/ Antiport.* Presented at EMCC meeting, Lorentz Center, Leiden (2004).

[28] S. Verlan. *Tissue P Systems with Minimal Symport/Antiport.* Developments in Language Theory, DLT 2004 (C.S. Calude, E. Calude, M.J. Dinneen, Eds), *Lecture Notes in Computer Science*, Springer **3340**, (2004) 418–430.

[29] P Systems Webpage, `http://psystems.disco.unimib.it`

Artiom Alhazov, Yurii Rogozhin, Received December 1, 2006

Dr. Artiom Alhazov
Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova
email: *artiom@math.md*
Research Group on Mathematical Linguistics
Rovira i Virgili University, Tarragona, Spain
email: *artiome.alhazov@estudiants.urv.cat*

Dr.hab. Yurii Rogozhin
Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova
email: *rogozhin@math.md*

Computer Science Journal of Moldova, vol.**14**, no.3(42), 2006

Applying Evolution Strategies for Chest Radiographs Segmentation

Oliviu Matei

Abstract

Segmentation is one of the key areas in computer vision. Advanced techniques, such as active contour models and active shape models suffer from many drawbacks. An improvement of the two approaches, based in evolution strategies, is presented. Some of its advantages lie in less computational resources and less a priori knowledge required. At the end of the paper, the results of the experiments carried out with the new approach are presented.

1 Introduction

The ultimate goal of machine vision is image understanding - the ability not only to recover image structure, but also know what it represents. An attractive field of application for image segmentation is the medicine. Almost every object of interest in the human body can vary in size, shape and appearance. Often this makes the task of automatically identifying and segmenting interesting structures, such as organs and bones, very difficult.

This article presents a segmentation technique based on evolution strategies. We compare this method with the classical ones - active contour models and active shape models. At the end we show the results of image segmentation applying the proposed technique.

1.1 Related work

Many researchers approached the segmentation problem in a bottom-up fashion: emphasis was on the analysis and design of filters for the detection of local structures such as edges, ridges, corners and 'T'-junctions. The structure of an image can be described as a collection of such syntactical elements and their (spatial) relations, and such descriptions can be used as input for generic segmentation schemes. Unfortunately, these segmentations are often not very meaningful. On the other hand, top-down strategies (also referred to as model-based or active approaches) for segmentation were used successfully in highly constrained environments, e.g., in industrial inspection tasks. Often these methods are based on template matching. But template matching, or related techniques, are likely to fail if the object and/or background exhibit a large variability in shape or gray-level appearance, as is often the case in real-life images and medical data. Active contours or snakes [3], [9] and wave propagation methods such as level sets [16], have been heralded as a new paradigms for segmentation.

Other researchers experimented with hand-crafted parametric models. An illustrative example is the work of Yuille et al. [17] where a deformable model of an eye is constructed from circles and parabolic patches and a heuristic cost function is proposed for the gray-level appearance of the image inside and on the border of these patches.

There is a need for generic segmentation schemes that can be trained with examples as to acquire a model of the shape of the object to be segmented (with its variability) and the gray-level appearance of the object in the image (with its variability). Such methods use statistical techniques to extract the major variations from the prototypes in a principled manner. Several of such schemes have been proposed. One of the methods is active shape models (ASMs) put forward by Cootes and Taylor [7]. The shape model in ASMs is given by the principal components of vectors of landmark points.

2 Statistical Shape Models

Here we describe a statistical model of shape that will be used to represent objects in images. The shape of an object is represented by a set of points. Our aim is to derive models which allow us to both analyze new shapes and to synthesize shapes similar to those in the training set.

Good choices for landmarks are points that can be consistently located from one image to another. In two dimensions, points could be placed at clear corners of object boundaries, 'T'-junctions between boundaries or easily located biological landmarks. However, there are rarely enough of such points to give more than a sparse description of the shape of the target object. This list would be augmented with points along boundaries that are arranged to be equally spaced between well defined landmark points.

3 Active Contour Models

An active contour (also known as a snake by virtue of the nature of its evolution) is an energy minimizing spline ([4]) that is constrained by its own internal forces of continuity and curvature, while external forces drive it towards desired image features. The external forces are provided by the image and where appropriate, information from a higher-level process may be included. The problem is formulated as an energy minimization problem. Typically there are many local minima in the energy function. In this situation the contour may become trapped in strong neighboring edges resulting in a false boundary. Prior knowledge may be included in the process in order to achieve this. A model-free interpretation is limited because the problem is underconstrained. Various methods have been proposed to avoid insignificant false edges in [3], [8] and [14]. However, even with the application of these techniques the contour may still become trapped by false edges.

There are two key difficulties with active contour algorithms. First, the initial contour must, in general, be close to the true boundary or else it will likely converge to the wrong result. The second problem is

that active contours have difficulties progressing into concave boundary regions [17].

4 Active Shape Models

This section briefly reviews the ASM segmentation scheme. We follow the description and notation of [6]. In principle, the scheme can be used in nD, but in this paper we give a two-dimensional (2-D) formulation.

A shape model relies on representing the objects by a set of labelled points; each point is placed on a particular part of the object. The model gives the average positions of the points and a description of the main modes of variations found in the training set. Locating an example of such a model in an image involves choosing values for each of the parameters so as to best fit the model to the image. An initial guess for the best shape, orientation, scale and position can be refined by comparing the hypothesized model example with the image data and using differences between model and image to deform the shape. The method has similarities with the ACM's, but differs in that global shape constraints are applied; to make this distinction clear, the term "Active Shape Models" was adopted. The key point is that an instance of a model can only deform in ways found in its training set.

5 Evolution Strategies

Evolution strategies (ESs) are search algorithms which imitate the principles of natural evolution as a method to solve parameter optimization problems. They are based on the idea that among a population of examples, only members which are the fittest can survive and breed so that the examples in following generations are improved.

A general evolutionary algorithm is defined below:

Generate random population
REPEAT

evaluate fitness of current population
select chromosomes, based on fitness
for reproduction perform crossover and mutation
to give new improved population

UNTIL finished

An evolution strategy must have the following five components:

1. a definition of the objective function
2. a definition and implementation of the genetic representation
3. a way to create an initial population of potential solutions
4. a definition and implementation of the genetic operators
5. values for various parameters that the evolution strategy uses (population size, probabilities of applying operators etc.)

Once these three have been defined, the evolution strategy should work fairly well. Beyond that, one can try many different variations to improve performance, find multiple optima (species - if they exist), or parallelize the algorithms.

5.1 Objective Function

In our experiments, we applied three types of objective functions:

- The first function used was the magnitude of the edges. We assumed that the object (specifically the lung) is delimited by clear edges.

  The magnitude (or edge strength) is computed using the Sobel operator. It uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The magnitude is then:

$$|G| = |G_x| + |G_y|$$

Since the edges are defined as the pixels with the highest magnitude, the evolutionary strategy has to find those pixels which maximize the magnitude. Mathematically, this is:

$$f(x) = \sum_{i=1}^{n} G_{p(i)}$$

where $f(x)$ is the objective function and $G_{p(i)}$ is the magnitude of the pixel $p(i)$.

This was performed in two ways.

The first approach was the detection of edges in a raw image, respectively in a smoothed image. As expected, the results were much better in the smoothed image. On the raw image, the population got stuck often in local optima. Whereas a smoothing of the image assured a "smoother" intensity of the pixels. Another approach was based on edge detection on a sharped image. The sharpening was performed using Laplace operator. A disadvantage of this operator is the doubling of some edges in the image and the small neighborhood it takes into account. To prevent these, we applied a gaussian operator.

Compared with the previous approach, the results became better. Sharpening the image, the search was not performed on the image space (formed by the intensities of the pixels), but rather on the gradient space, which is the absolute difference between neighboring pixels. This makes the approach less sensitive to intensity variations.

- The second objective function was an extension of the previous one. For sharpening the image, we took into account not only the magnitude, but also the direction of the edge. The sharpening was performed using Prewitt operator.

In this case, for each landmark there were two criteria: the magnitude and the direction of the edge in that point. The direction of an edge is determined using the Susan edge detector čitesmith99susan, which, because of the lack of space, will not be detailed here.

In this case, the objective function is:

$$f(x) = \alpha \sum_{i=1}^{n} G_{p(i)} + \beta \sum_{i=1}^{n} R_{p(i)}$$

where $G_{p(i)}$ is the magnitude of the pixel $p(i)$, $R_{p(i)}$ is the direction of the edge in the pixel $p(i)$ and α and β are two parameters tuning the importance of one factor or the other. They must fulfill the condition: $\alpha + \beta = 1$. If $\alpha = 1$, then this objective function is identical with the previous one. If $\alpha = 0$, then the objective function takes into account only the direction of the edges.

This approach runs well only on some images. That is because corresponding landmarks do not have corresponding edges directions. However, on shapes that can be easily defined mathematically, the results were very satisfactory.

- The third function is a more general case, in which we consider not only the magnitude of the edge, but also the gray-level appearance of the edge.

  In some cases it is efficient to assume that the points lie on strong edges and to search for such in an image. However this is not always satisfactory and it is necessary to have more general model of the gray-level appearance. The gray-level appearance model that describes the typical image structure around each landmark is obtained from pixel profiles, sampled (using linear interpolation) around each landmark, perpendicular to the contour.

  This requires a notion of connectivity between the landmark points from which the perpendicular direction can be computed. The direction perpendicular to a landmark (x_n, y_n) is computed

by rotating the vector that runs from (x_{n-1}, y_{n-1}) to (x_{n+1}, y_{n+1}) over 90°. We assume that all objects we use are closed contours, so for the first landmark, the last landmark and the second landmark are the points from which a perpendicular direction is computed; for the last landmark, the second to last landmark and the first landmark are used.

On either side k pixels are sampled using a fixed step size, which gives profiles of length $2k + 1$. Cootes and Taylor [6] propose to use the normalized first derivatives of these profiles to build the gray-level appearance model. The derivatives are computed using finite differences between the $(j-1)$th and the $(j+1)$th point:

$$g_{ijk} = I_j(y_{i(k+1)}) - I_j(y_{i(k-1)}) \tag{1}$$

where y_{ik} is the kth point along the ith profile and $I_j(y_{ik})$ is the gray level in image j at that point. The normalization is such that the sum of absolute values of the elements in the derivative profile is 1:

$$g'_{ij} = \frac{g_i j}{\sum_{k=1}^{n_p} |g_{ijk}|} \tag{2}$$

Denoting these normalized derivative profiles as $g_1, g_2, ..., g_s$, the mean profile $\overline{g}$ and the covariance matrix S_g are computed for each landmark.

$$\overline{g_i} = \frac{1}{N_s} \sum_{j=1}^{Ns} g'_{ij} \tag{3}$$

This allows for the computation of the Mahalanobis distance [11] between a new profile and the profile model

$$f(g_i) = (g_i - \overline{g}) S_g^{-1} (g_i - \overline{g}) \tag{4}$$

Minimizing $f(g_i)$ is equivalent to maximizing the probability that g_i originates from a multidimensional Gaussian distribution. In other words, minimizing $f(g_i)$ is equivalent to maximizing the probability that the points are the desired ones.

5.2 Genetic Representation

An individual is represented by all the landmark points for a training image. Noting each landmark point by (x_i, y_i), the set of all landmarks is then:

$$x = (x_1, y_1, x_2, y_2, ..., x_n, y_n) \tag{5}$$

where n is the number of landmarks defined for one training image.

However, an interesting idea introduced in ES is that an individual is not a simple float or binary value, like in the case of GAs, but is a pair of float values, i.e., $v = (x, \sigma)$, where the first vector x represents a point in the search space, and the second vector σ is a vector of standard deviations. For our case, an individual is represented by:

$$x = ((x_1, y_1, x_2, y_2, ..., x_n, y_n), (\sigma_1, \sigma_2, ..., \sigma_s, ..., \sigma_{2s})) \tag{6}$$

5.3 The Initial Population

Once a suitable representation has been decided upon for the chromosomes, it is necessary to create an initial population to serve as the starting point for the genetic algorithm. The initial population is usually created randomly. When there is some information about the solution, some heuristics can be used. Each subsequent population that is built (each subsequent generation) uses the previous population as a base - taking the more fit individuals to breed better solutions.

In this case, we can use the training landmarks as forming the initial population, rather than random points. However, a random population should lead to the same good results in segmentation, although after a larger number of generation. On the other hand, an heuristic initial population would yield good results faster because the initial shapes resemble the one to be segmented.

5.4 Genetic Operators

To produce an offspring, the system acts in several stages:

1. Select two individuals:

$$(x^1, \sigma^1) = [(x_1^1, ..., x_n^1, y_1^1, ..., y_n^1), (\sigma_1^1, ..., \sigma_{2n}^1)] \tag{7}$$

and

$$(x^2, \sigma^2) = [(x_1^2, ..., x_n^2, y_1^2, ..., y_n^2), (\sigma_1^2, ..., \sigma_{2n}^2)] \tag{8}$$

and apply a recombination operator. The types of crossover that we used are:

(a) discrete, in which case the offspring is

$$(x, \sigma) = [(x_1^{q_1}, ..., x_n^{q_s}, y_1^{q_{s+1}}, ..., x_{2n}^{q_{2n}}), (\sigma_1^{q_1}, ..., \sigma_{2n}^{q_{2n}})] \tag{9}$$

where $q_i = 1$ or $q_i = 2$ with equal probabilities for all $i = 1, 2, ..., s$. That means that each component i comes either from the first or from the second parent.

(b) intermediate, when the offspring is

$$(x, \sigma) = [(x_1^{q_1}, ..., x_n^{q_s}, y_1^{q_{s+1}}, ..., y_n^{q_{2n}}), (\sigma_1^{q_1}, ..., \sigma_{2n}^{q_{2n}})] \tag{10}$$

$$(x, \sigma) = [(\frac{x_1^1 + x_1^2}{2}, ..., \frac{x_n^1 + x_n^2}{2}, \frac{y_1^1 + y_1^2}{2}, ..., \frac{y_n^1 + y_n^2}{2}, (\frac{\sigma_1^1 + \sigma_1^2}{2}, ..., \frac{\sigma_{2n}^1 + \sigma_{2n}^2}{2})] \tag{11}$$

The intermediate crossover yielded better results. The explanation is somehow obvious. Whereas the discreet crossover maintains at least one of the coordinates of either parents, the intermediate crossover creates a completely new offspring, with completely new coordinates.

2. Apply mutation to the offspring obtained. Here is the novel feature of $(\mu + \lambda) - ES$ and $(\mu, \lambda) - ES$. By this operator, the

control parameters adapt themselves. After mutation, the individual $v = (x, \sigma)$ becomes $v' = (x', \sigma')$, where

$$\sigma' = \sigma \dot{e}^{N(0,\delta\sigma)} \tag{12}$$

and

$$x' = x + N(0, \sigma') \tag{13}$$

In the above relations, $\delta\sigma$ is an extra parameter and $N(0, \delta\sigma)$ represents a random number within a normal (Gaussian) distribution with mean zero and standard deviation $\delta\sigma$.

In ESs, mutation realizes a kind of hill-climbing search when it is considered in combination with selection. The σ_i parameters related to the x_i can be seen as preferred search directions along the axes of coordinates. Generally, the best search direction (with the largest climbing, in mathematics called *gradient*) is not along the axis. Thus the trajectory of the population through the search space is zigzagging along the gradient, hence it is suboptimal. In order to avoid this shortcoming, an additional control parameter (θ) has been introduced. An individual is represented by a triple (x, σ, θ). The recombination operators become now:

$$\sigma' = \sigma \dot{e}^{N(0,\delta\sigma)} \tag{14}$$

$$\theta' = \theta + N(0, \sigma') \tag{15}$$

$$x' = x + C(0, \sigma', \theta') \tag{16}$$

The parameter $\delta\theta$ is an extra parameter, like $\delta\sigma$. The parameter $C(0, \sigma', \theta')$ is the correlation matrix.

5.5 ES Parameters

An important step in designing an evolution strategy is the choice of the parameters.

1. The individual length is twice as much as the number of landmarks. If there are n landmarks (x_i, i_i) defined for each image, an individual is:

$$x = (x, \sigma, \theta) = [(x_1, ..., x_n, y_1, ..., y_n)(\sigma_1, ..., \sigma_{2n})(\theta_1, ..., \theta_{2n})] \tag{17}$$

2. The size of the population, μ represents the number of individuals that undergo evolution (crossover and mutation).

3. The number of offspring, λ is the number of individuals created after a generation evolves. It is an intermediary population, from which, the best individuals will survive and make up a new generation.

4. The selection procedure can be done in two ways:

   - selecting the fittest μ individuals from the intermediary population consisting of λ offspring. This type of selection is referred to as (μ, λ) selection
   - selecting the fittest μ individuals from the population consisting of the μ parents along with the λ offspring. This is referred to as $(\mu + \lambda)$ selection

5. Mutation parameters $\delta\sigma$ and $\delta\theta$ are needed for defining the step of the mutation. The larger the two parameters, the more significant the mutation.

6. The total number of generations represents the maximum number of iterations until the algorithm stops. Normally, the algorithm should end when the objective function is reached or when it is close enough. If this does not happens, the algorithm runs until the maximum number of generations.

6 Why Evolution Strategies?

6.1 Evolution Strategies vs. Genetic Algorithms

Both ESs and Genetic Algorithms (GAs) belong to a wider class of search algorithms, known as Evolutionary Algorithms or Evolutionary Computation. Both approaches to computational problem solving rely on the biological paradigms of organic evolution, postulated by Darwin. They explore the search space by means of population of search points which undergo evolution by crossover, mutation and selection.

Cootes et al. [5] and Huang et. al. [12] proposed the use of GAs for segmentation. Unlike them, we propose ESs for the same task. The reasons for that are described further.

1. A reason for choosing ESs consists in the way the individuals are represented. The ESs individuals are represented by floats. On the other hand, the GAs operate on binary strings. Only recently, they have been used also float representation.

2. The selection process represents another reason for using ESs. The new generation is selected deterministic as consisting of the fittest μ individuals out of λ (in the case of $(\mu, \lambda) - ES$) or $\mu + \lambda$ individuals (for $(\mu + \lambda) - ES$).

   In a generation of the GAs, all the individuals have a chance to be selected according to their fitness. Thus even the weakest ones can be parts of the new population. Thus the selection is weakly controlled.

3. Another reason for choosing the ESs lies in the way the reproduction parameters are stored and handled. In GAs, the probability of crossover and the probability of mutation remain constant throughout the evolution process and are external to the population. On the contrary, these parameters evolve every new generation in ESs. This self-adaptation is quite important for fine local tuning and has its counterpart in GAs research.

6.2 ESs vs. ACMs

As can be seen in section 3 the main disadvantage of the ACMs is their additional computational complexity. There are very complex calculations to be done, including integrals and derivatives.

Second, most approaches are based on explicit updating schemes which demand very small time steps. The steps used by ESs have two advantages:

- can be tuned with the evolution parameters, namely $\delta\sigma$ and $\delta\theta$

- vary over time, from small steps to larger ones, depending on the fitness of the individuals

Third, an ACM requires the definitions of both an external energy as well as an internal one. The external energy is dependent on the application and is similar to the target shape for ASMs and to the objective function for ESs. Straightforward functions have been defined by Gunn et. al. [10] and by Xu at al. [15].

However, the internal function may have significant drawbacks:

- It may be easily defined by regular shapes, but difficult or even impossible to define for more complex, real life shapes.
- Quite often only one function does not suffice. Most of the times, the internal forces of a shape can be defined by more functions.

ESs do not necessarily make use of the internal energies. However, such heuristic knowledge can be embedded successfully into the evolution, which leads to faster and more reliable convergence. In this case, the size of the population is important, because the individuals "pull" each other towards the contour of the object.

According to Yuille et al. [13], the active contours have difficulties progressing into concave boundary regions. On the other hand, the ESs do not have such difficulties as they do not make use of any function defining an internal energy.

6.3 ESs vs. ASM

Cootes et. al. [7] propose, as improvement of the ASMs, the alignment of images. This leads to better results. The drawbacks of such an approach are:

- It requires extra computational resources.
- The alignment works well on the training images, but the image to be segmented cannot be aligned because there are no landmarks defined yet. This makes that approach almost useless in real life.

The ASMs require an approximate position of the object to be segmented, whereas ESs are global optimization techniques. In other words, they "search" for the objects independently of the position and shape of the object on the training image. However, training shapes spread all over the images would yield better results in terms of time and segmentation.

The ASMs can get stuck in secondary edges and cannot be avoided due to the small steps in which the shape is updated. This situation can occur also in ESs. However, due to mutation, some individuals may get out of them and eventually reach the global optimum.

7 Results

We carried out three types of experiments. The objects in the images were annotated by a number of fixed landmarks and a closed contour between those fixed points from which a number of equidistant landmark points were sampled. No shape alignment was performed.

As shown in section 5.5, an individual consists of a set of landmarks describing a contour. A training set of images were marked manually by two physicians. An average of the two segmentations yielded the initial population, which, further on, underwent evolution.

To evaluate the results, the following measure was used: the area correctly classified as object was divided by the area classified as object plus the area classified incorrectly as object or as background. In other words: true positive area divided by true positive plus false positive plus false negative area. This performance measure yields 1 for a perfect result and 0 if there is no overlap at all between the detected and true object. This measure is probably more directly related to what is expected of a segmentation in practice than the average distance between the true and detected location of each landmark in the shape model because the latter is not sensitive to shifts of the landmarks along the contour.

The first set of experiments consisted of segmentation of rectangles of different sizes. This step was required for rough assessment of the

methods. It gave the first impressions of various setups. The number of fixed points was 4 (in the corners of the rectangles) and in total 24 points were used.

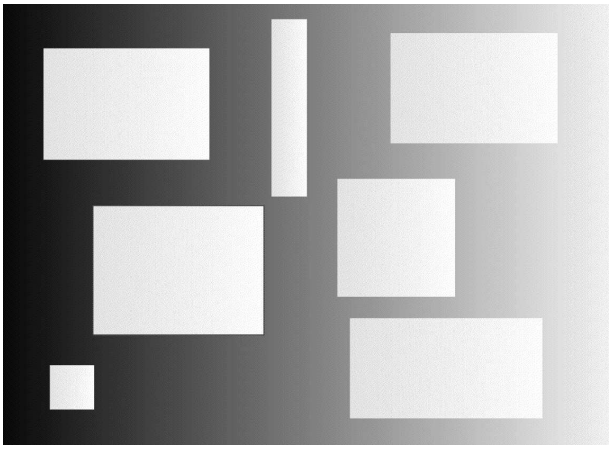

Figure 1. Rectangles used for segmentation

The other experiments were performed on 78 standard postero-anterior (PA) chest radiographs. In these images, both the left and the right lung fields were segmented as separate objects. The images were randomly selected from a tuberculosis screening program and contained both normal and abnormal cases of patients of 16 years and older. The images were printed on 10 x 10 cm film, digitized with a scanner to 1000 x 1000 pixels with 12 bit intensity.

The lung fields in the test set of the chest radiographs were manually segmented by a human observer independently. Therefore the results of the algorithms could be compared with the output of the observer. For these experiments, 15 fixed landmarks have been annotated and in total 60 points were used.

The second set of experiments consisted in segmentation of the left lung field.

The results we obtained are summarized in the Table 1.

Figure 2. Rx images used for left lung segmentation

Table 1. Segmentation results for the left lung fields

Segmentation	**Accuracy** $\pm\sigma$
Segmentation using the edge magnitude	$88.4\% \pm 7.4\%$
Segmentation using the edge magnitude and direction	$94.5\% \pm 9.87\%$
Segmentation using the gray-level appearance	$95.2\% \pm 6.4\%$

The third set of experiments consisted in segmentation of the right lung field.

Figure 3. Rx images used for right lung segmentation

The results we obtained for the right lung field segmentation are summarized in the Table 2.

As can be noticed from the experiments, the results for the left and for the right lung are pretty similar. The three segmentation criteria used yielded quite some different results. The edge magnitude gave the worst results, but with the smallest standard deviation. The second

Table 2. Segmentation results for the right lung fields

Segmentation	**Accuracy** $\pm\sigma$
Segmentation using the edge magnitude	$87.3\% \pm 5.21\%$
Segmentation using the edge magnitude and direction	$93.5\% \pm 8.6\%$
Segmentation using the gray-level appearance	$96.1\% \pm 6.58\%$

criterion, edge magnitude and direction resulted pretty good results, but the standard deviation was large, which means that it works well only on some images. The last criterion, gray-level appearance, gave the best results, with small standard deviation, which means good and constant results.

Further we will compare the results of our approach with results obtained by other authors. Bram van Ginneken et. al. [1] carried out similar research on lung fields segmentation. They used 230 standard chest radiographs and applied ASMs and ASMs with optimal features. Their results are concluded in the tables below. The results of the right lung segmentations are presented in Table 3.

Table 3. Segmentation results of the experiments carried out by Ginneken for the right lung fields

Segmentation	**Accuracy** $\pm\sigma$
ASMs	$88.2\% \pm 7.4\%$
ASMs with optimal features	$92.9\% \pm 2.6\%$

The results of the left lung segmentations are presented in Table 4.

It is difficult to compare results of two researches carried out on different sets of images and in different conditions. However, it is noticeable that the results presented in [1] and obtained by us are very similar. Moreover, unlike Ginneken, our results are more compact (σ is slightly lower in our case), which means more consistent segmentations.

We consider interesting to see the similarities and differences

Table 4. Segmentation results of the experiments carried out by Ginneken for the left lung fields

Segmentation	**Accuracy** $\pm\sigma$
ASMs	$86.1\% \pm 10.9\%$
ASMs with optimal features	$88.7\% \pm 11.4\%$

between segmentations based on evolutionary methods for various anatomical shapes. Huang et. al. [12] used genetic algorithms for eye location. They used 10 faces for training and 20 face images for tests, out of which their algorithm missed two eye locations and no false positives were identified. In other words, the accuracy of the algorithm was about 90%. Eye detection is a quite simple task because it involves a rather coarse segmentation.

Cootes et al [5] applied ASMs and genetic algorithms for locating structures in medical images. They defined a 96 point heart model on a set of echocardiograms. They obtained good results after 50 iterations, but did not assessed the accuracy of the location.

8 Conclusions

The method we describe allow flexible models of image objects such as lungs to be built easily from sets of example images. The technique can be applied to a wide variety of objects in different imaging modalities.

Cootes [5] proposed the use of genetic algorithms for the searches performed by the ASMs. Our approach, on the contrary, substitutes entirely the ASMs. The method we presented could be applied for segmentation of any anatomical structure. However, our experiments were carried out only for lung fields.

An enhancement of this approach is the use of the ESs in a multiresolution framework. This means the use of ESs within various resolutions, starting with a coarse image and ending with the finest resolution.

Beichel et al. [2] researched active appearance model matching.

A combination between active appearance with evolution strategies would be a very good research subject as well as an improvement of the approach we have presented in this paper.

References

[1] J. Staal B. ter Haar Romeny B. van Ginneken, A.F. Frangi and M.A. Viergever. Active shape model segmentation with optimal features. *IEEE Transactions on medical imaging*, 21(8), 2002.

[2] R. Beichel, H. Bischof, and M. Sonka. Robust active appearance model matching. In *Information Processing in Medical Imaging, 19th International Conference*, pages 114–125, 2005.

[3] D. L. Pham C. Xu and J. L. Prince. *Handbook of Medical Imaging, Vol. 2:Medical Image Processing and Analysis*. SPIE PRESS, 2000.

[4] Jr. C. Xu, A. Yezzi and J. L. Prince. On the relationship between parametric and geometric active contours. In *Proc. of 34th Asilomar Conference on Signals, Systems, and Computers*, pages 483 –489, 2000.

[5] T. F. Cootes, A. Hill, and J. Haslam. The use of active shape models for locating structures in medical images. *Image and Vision Computing*, 12(6):355–366, 2001.

[6] T. F. Cootes and C. J. Taylor. Statistical models of appearance for computer vision. Technical report, University of Manchester, 2001.

[7] T. F. Cootes, C. J. Taylor, D. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision Image Understanding*, 61(1):38–59, 2001.

[8] H. Delingette and J. Montagnat. Shape and topology constraints on parametric active contours. *Computer Vision and Image Understanding*, (83):140–171, 2001.

[9] D. Terzopoulos G. Hamarneh, T. McInerney. Deformable organisms for automatic medical image analysis. In *Proc. Third International Conference on Medical Image Computing and Computer Assisted Interventions*, volume 1, pages 66–75, 2003.

[10] Steve R. Gunn and Mark S. Nixon. A model based dual active contour. Technical report, University of Southampton, 2002.

[11] W. Haerdle, Y. Mori, and J. Symanzik. *Computational Statistics.* Springer Verlag, 2004.

[12] J. Huang and H. Wechsler. Eye location using genetic algorithm. In *2nd International Conference on Audio and Video-Based Biometric Person Authentication (AVBPA)*, 1999.

[13] C. English J. M. Coughlan, D. Snow and A.L. Yuille. Feature extraction from faces using deformable templates. *Computer Vision and Image Understanding*, 5(11):303–319, 2000.

[14] Y. Shirai M. Etoh and M. Asada. Contour extraction by mixture density description obtained from region clustering. In *Proceedings of European Conference on Computer Vision*, pages 24–32, 1992.

[15] J. L. Prince and C. Xu. A new external force model for snakes. In *IEEE Proc. Conf. on Comp. Vis. Patt. Recog. (CVPR'97)*, pages 30–31, 2001.

[16] J. A. Sethian. Level set methods and fast marching methods. *Cambridge University Press*, 1999.

[17] A.L. Yuille. Cccp algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2004.

Oliviu Matei

Received October 11, 2006

Technical University of Cluj-Napoca,
E–mail: *oliviu.matei@holisun.com*

Computer Science Journal of Moldova, vol.**14**, no.3(42), 2006

Nash equilibria set computing in finite extended games

Valeriu Ungureanu

Abstract

The Nash equilibria set (NES) is described as an intersection of graphs of best response mappings. The problem of NES computing for multi-matrix extended games is considered. A method for NES computing is studied.

Mathematics Subject Classification 2000: 91A05, 91A06, 91A10, 91A44, 90C05, 90C31, 90C90.

Keywords and phrases: Noncooperative game, normal form game, Nash equilibrium (NE), Nash equilibria set (NES), graph of best response mapping, intersection, method, algorithm, computational complexity.

1 Introduction

The Nash equilibria set (NES) is determined as an intersection of graphs of best response mappings [5, 8, 9]. This idea yields a natural method for NES computing in mixed extended two-player $m \times n$ games and n-player $m_1 \times m_2 \times \cdots \times m_n$ games.

Consider a noncooperative finite strategic game:

$$\Gamma = \langle N, \{S_p\}_{p\in N}, \{a^p_{\mathbf{s}} = a^p_{s_1 s_2 \ldots s_n}\}_{p\in N}\rangle,$$

where $N = \{1, 2, ..., n\}$ is a set of players, $S_p = \{1, 2, \ldots, m_p\}$ is a set of strategies of player $p \in N$, $\#S_p = m_p < +\infty$, $p \in N$ and $a^p_{\mathbf{s}} = a^p_{s_1 s_2 \ldots s_n}$ is a player's $p \in N$ payoff function defined on the Cartesian product $S = \underset{p\in N}{\times} S_p$ (payoff of the player $p \in N$ is done by n dimensional matrix

$A^p[m_1 \times m_2 \times \cdots \times m_n]$). Elements $\mathbf{s} = (s_1, s_2, ..., s_n) \in S$ are named outcomes of the game (situations or strategy profiles).

The mixed extension of Γ is

$$\widetilde{\Gamma} = \langle X_p, f_p(\mathbf{x}), p \in N \rangle,$$

where

$$f_p(\mathbf{x}) = \sum_{s_1=1}^{m_1} \sum_{s_2=1}^{m_2} \cdots \sum_{s_n=1}^{m_n} a^p_{s_1 s_2 \ldots s_n} x^1_{s_1} x^2_{s_2} \ldots x^n_{s_n} = \\ = \sum_{s_1=1}^{m_1} \sum_{s_2=1}^{m_2} \cdots \sum_{s_n=1}^{m_n} a^p_{\mathbf{s}} \prod_{p=1}^{n} x^p_{s_p},$$

$$\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n) \in X = \underset{p \in N}{\times} X_p \subset R^m,$$

$$m = m_1 + m_2 + \cdots + m_n,$$

$$X_p = \left\{ \mathbf{x}^p = (x^p_1, \ldots, x^p_{m_p}) : \begin{array}{l} x^p_1 + \cdots + x^p_{m_p} = 1, \\ x^p_1 \geq 0, \ldots, x^p_{m_p} \geq 0 \end{array} \right\}$$

is a set of mixed strategies of the player $p \in N$.

Definition. *The outcome $\hat{\mathbf{x}} \in X$ of the game is a Nash equilibrium [4] if*

$$f_p(\mathbf{x}^p, \hat{\mathbf{x}}^{-p}) \leq f_p(\hat{\mathbf{x}}^p, \hat{\mathbf{x}}^{-p}), \forall \mathbf{x}^p \in X_p, \ \forall p \in N,$$

where

$$\hat{\mathbf{x}}^{-p} = (\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \ldots, \hat{\mathbf{x}}^{p-1}, \hat{\mathbf{x}}^{p+1}, \ldots, \hat{\mathbf{x}}^n),$$

$$\hat{\mathbf{x}}^{-p} \in X_{-p} = X_1 \times X_2 \times \cdots \times X_{p-1} \times X_{p+1} \times \cdots \times X_n,$$

$$\hat{\mathbf{x}}^p || \hat{\mathbf{x}}^{-p} = (\hat{\mathbf{x}}^p, \hat{\mathbf{x}}^{-p}) = (\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \ldots, \hat{\mathbf{x}}^{p-1}, \hat{\mathbf{x}}^p, \hat{\mathbf{x}}^{p+1}, \ldots, \hat{\mathbf{x}}^n) = \hat{\mathbf{x}} \in X.$$

It is well known that not all the games in pure strategies have NE, but all the extended games have NE [4], i.e. $NE(\widetilde{\Gamma}) \neq \emptyset$.

There are diverse alternative formulations of a NE [2]: as a fixed point of the best response correspondence, as a fixed point of a function, as a solution of a nonlinear complementarity problem, as a solution of a stationary point problem, as a minimum of a function on a polytope,

as a semi-algebraic set. The NES may be considered as an intersection of graphs of best response multivalued mappings (correspondences) [5, 8, 9] $\operatorname{Arg}\max\limits_{\mathbf{x}^p\in X_p} f_p\left(\mathbf{x}^p,\mathbf{x}^{-p}\right) : X_{-p} \multimap X_p,\ p=\overline{1,n}$:

$$NE(\widetilde{\Gamma}) = \bigcap_{p\in N} Gr_p,$$

$$Gr_p = \left\{ (\mathbf{x}^p,\mathbf{x}^{-p}) \in X : \begin{array}{l} \mathbf{x}^{-p}\in X_{-p}, \\ \mathbf{x}^p \in \operatorname{Arg}\max\limits_{\mathbf{x}^p\in X_p} f_p\left(\mathbf{x}^p,\mathbf{x}^{-p}\right) \end{array} \right\},\ p\in N.$$

The simplest solvable problems of NES determination are problems in the mixed extension of two-person 2×2 game [2, 3, 5], 2×3 game [9], and three-person $2\times 2\times 2$ game [8]. In this paper a method for NES computing in mixed extended $m\times n$ games and multi-matrix mixed extended games is analyzed.

According to [6]: *"The computational complexity of finding one equilibrium is unclear... Gilboa and Zemel [1] show that finding an equilibrium of a bi-matrix game with maximum payoff sum is NP-hard, so for this problem no efficient algorithm is likely to exist. The same holds for other problems that amount essentially to examining all equilibria, like finding an equilibrium with maximum support".* Consequently, the problem of Nash equilibria set computing is NP-hard. And so, from the complexity point of view proposed algorithms are admissible.

As it is easy to see, the algorithms for NES computing in multi-matrix extended games contain particularly algorithm that computes NES in extended $m\times n$ two-matrix games. But, two-matrix game has peculiar features that permit to give a more expedient algorithm. Examples have to give the reader the opportunity to easy and prompt grasp of the paper.

2 NES in two-player mixed extended $m\times n$ games

Consider a two-matrix $m\times n$ game Γ with matrices:

$$A=(a_{ij}),\ B=(b_{ij}),\ i=\overline{1,m},\ j=\overline{1,n}.$$

Let $A^i, i = \overline{1,m}$ denote the lines of matrix A,

$\mathbf{b}^j, j = \overline{1,n}$ denote the columns of matrix B,

$X = \{\mathbf{x} \in R^m : x_1 + x_2 + ... + x_m = 1, \mathbf{x} \geq 0\}$,

$Y = \{\mathbf{y} \in R^n : y_1 + y_2 + ... + y_n = 1, \mathbf{y} \geq 0\}$,

$$f_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_i y_j = (A^1\mathbf{y})x_1 + (A^2\mathbf{y})x_2 + ... + (A^m\mathbf{y})x_n,$$

$$f_2(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} x_i y_j = (\mathbf{x}^T\mathbf{b}^1)y_1 + (\mathbf{x}^T\mathbf{b}^2)y_2 + ... + (\mathbf{x}^T\mathbf{b}^n)y_n.$$

The game $\widetilde{\Gamma} = \langle X, Y;\ f_1, f_2 \rangle$ is the mixed extension of Γ.

If the strategy of the second player is fixed, then the first player has to solve a linear programming problem:

$$f_1(\mathbf{x}, \mathbf{y}) \to \max, \mathbf{x} \in X. \tag{1}$$

Evidently, this problem is a linear programming parametric problem with the parameter-vector $\mathbf{y} \in Y$.

Analogically, the second player has to solve the linear programming parametric problem:

$$f_2(\mathbf{x}, \mathbf{y}) \to \max, \mathbf{y} \in Y, \tag{2}$$

with the parameter-vector $\mathbf{x} \in X$.

Denote $\mathbf{e}\mathbf{x}^T = (1, \ldots, 1) \in R^m, \mathbf{e}\mathbf{y}^T = (1, \ldots, 1) \in R^n$. The solution of linear programming problem is realized on the vertices of polytopes of feasible solutions. In the problems (1) and (2) the sets X and Y have m and, respectively, n vertices — the axis unit vectors $\mathbf{e}^{x_i} \in R^m, i = \overline{1,m}$, and $\mathbf{e}^{y_j} \in R^n, j = \overline{1,n}$. Thus, in accordance with the simplex method and its optimality criterion [7], in the parametric problem (1) the parameter set Y is partitioned into such m subsets

$$Y^i = \left\{ \mathbf{y} \in R^n : \begin{array}{l} (A^k - A^i)\mathbf{y} \leq 0, k = \overline{1,m}, \\ \mathbf{e}\mathbf{y}^T\mathbf{y} = 1, \\ \mathbf{y} \geq 0 \end{array} \right\}, \quad i = \overline{1,m},$$

for which one of the optimal solution of the linear programming problem (1) is $\mathbf{e}^{x_i}$ — the corresponding x_i axis unit vector.

Let $U = \{i \in \{1, 2, ..., m\} : Y^i \neq \emptyset\}$. In conformity with the optimality criterion of the simplex method $\forall i \in U$ and $\forall I \in 2^{U \setminus \{i\}}$ all the points of

$$\mathbf{Conv}\{\mathbf{e}^{x_k}, k \in I \cup \{i\}\} = \left\{ \mathbf{x} \in R^m : \begin{array}{l} \mathbf{e}\mathbf{x}^T\mathbf{x} = 1, \\ \mathbf{x} \geq \mathbf{0}, \\ x_k = 0, k \notin I \cup \{i\} \end{array} \right\}$$

are optimal for parameters

$$\mathbf{y} \in Y^{iI} = \left\{ \mathbf{y} \in R^n : \begin{array}{l} (A^k - A^i)\mathbf{y} = 0, k \in I, \\ (A^k - A^i)\mathbf{y} \leq 0, k \notin I \cup \{i\}, \\ \mathbf{e}\mathbf{y}^T\mathbf{y} = 1, \\ \mathbf{y} \geq 0. \end{array} \right\}$$

Evidently $Y^{i\emptyset} = Y^i$. Hence,

$$Gr_1 = \bigcup_{i \in U,\ I \in 2^{U \setminus \{i\}}} \mathbf{Conv}\{\mathbf{e}^{x_k}, k \in I \cup \{i\}\} \times Y^{iI}.$$

In the parametric problem (2) the parameter set X is partitioned into such n subsets

$$X^j = \left\{ \mathbf{x} \in R^m : \begin{array}{l} (\mathbf{b}^k - \mathbf{b}^j)\mathbf{x} \leq 0, k = \overline{1, n}, \\ \mathbf{e}\mathbf{x}^T\mathbf{x} = 1, \\ \mathbf{x} \geq 0, \end{array} \right\}, \quad j = \overline{1, n},$$

for which one of the optimal solution of the linear programming problem (2) is $\mathbf{e}^{y_j}$ — the corresponding y_j axis unit vector.

Let $V = \{j \in \{1, 2, ..., n\} : X^j \neq \emptyset\}$. In conformity with the optimality criterion of the simplex method $\forall j \in V$ and $\forall J \in 2^{V \setminus \{j\}}$ all the points of

$$\mathbf{Conv}\{\mathbf{e}^{y_k}, k \in J \cup \{j\}\} = \left\{ \mathbf{y} \in R^n : \begin{array}{l} \mathbf{e}\mathbf{y}^T\mathbf{y} = 1, \\ \mathbf{y} \geq \mathbf{0}, \\ y_k = 0, k \notin J \cup \{j\} \end{array} \right\}$$

are optimal for parameters

$$\mathbf{x} \in X^{jJ} = \left\{ \mathbf{x} \in R^m : \begin{array}{l} (\mathbf{b}^k - \mathbf{b}^j)\mathbf{x} = 0, k \in J, \\ (\mathbf{b}^k - \mathbf{b}^j)\mathbf{x} \leq 0, k \notin J \cup \{j\}, \\ \mathbf{e}\mathbf{x}^T\mathbf{x} = 1, \\ \mathbf{x} \geq 0. \end{array} \right\}$$

Evidently $X^{j\emptyset} = X^j$. Hence

$$Gr_2 = \bigcup_{j \in V,\ J \in 2^{V \setminus \{j\}}} X^{jJ} \times \mathbf{Conv}\left\{\mathbf{e}^{y_k}, k \in J \cup \{j\}\right\}.$$

Finally,

$$NE(\widetilde{\Gamma}) = Gr_1 \bigcap Gr_2 = \bigcup_{\substack{i \in U,\ I \in 2^{U \setminus \{i\}} \\ j \in V,\ J \in 2^{V \setminus \{j\}}}} X_{iI}^{jJ} \times Y_{jJ}^{iI},$$

where $X_{iI}^{jJ} \times Y_{jJ}^{iI}$ is a convex component of NES,

$$X_{iI}^{jJ} = \mathbf{Conv}\left\{\mathbf{e}^{x_k}, k \in I \cup \{i\}\right\} \bigcap X^{jJ},$$
$$Y_{jJ}^{iI} = \mathbf{Conv}\left\{\mathbf{e}^{y_k}, k \in J \cup \{j\}\right\} \bigcap Y^{iI},$$

$$X_{iI}^{jJ} = \left\{ \mathbf{x} \in R^m : \begin{array}{l} (\mathbf{b}^k - \mathbf{b}^j)\mathbf{x} = 0, k \in J, \\ (\mathbf{b}^k - \mathbf{b}^j)\mathbf{x} \leq 0, k \notin J \cup \{j\}, \\ \mathbf{e}\mathbf{x}^T\mathbf{x} = 1, \mathbf{x} \geq 0, \\ x_k = 0, k \notin I \cup \{i\} \end{array} \right\}$$

is a set of strategies $\mathbf{x} \in X$ with support from $\{i\} \cup I$ and for which points of $\mathbf{Conv}\left\{\mathbf{e}^{y_k}, k \in J \cup \{j\}\right\}$ are optimal,

$$Y_{jJ}^{iI} = \left\{ \mathbf{y} \in R^n : \begin{array}{l} (A^k - A^i)\mathbf{y} = 0, k \in I, \\ (A^k - A^i)\mathbf{y} \leq 0, k \notin I \cup \{i\}, \\ \mathbf{e}\mathbf{y}^T\mathbf{y} = 1, \mathbf{y} \geq 0, \\ y_k = 0, k \notin J \cup \{j\} \end{array} \right\}$$

is a set of strategies $\mathbf{y} \in Y$ with support from $\{j\} \cup J$ and for which points of $\mathbf{Conv}\{\mathbf{e}^{x_k}, k \in I \cup \{i\}\}$ are optimal.

Theorem 1. $NE(\widetilde{\Gamma}) = \bigcup_{\substack{i \in U,\ I \in 2^{U \setminus \{i\}} \\ j \in V,\ J \in 2^{V \setminus \{j\}}}} X_{iI}^{jJ} \times Y_{jJ}^{iI}.$

The proof of the theorem is performed above.

Theorem 2. *If* $X_{iI}^{j\emptyset} = \emptyset$, *then* $X_{iI}^{jJ} = \emptyset$ *for all* $J \in 2^V$.

For the proof it is sufficient to maintain that $X_{iI}^{jJ} \subseteq X_{iI}^{j\emptyset}$ for $J \neq \emptyset$.

Theorem 3. *If* $Y_{jJ}^{i\emptyset} = \emptyset$, *then* $Y_{jJ}^{iI} = \emptyset$ *for all* $I \in 2^U$.

Theorem 3 is equivalent to theorem 2.

From the above the algorithm for NES computing follows:

$$NE = \emptyset; \quad U = \{i \in \{1, 2, ..., m\} : Y^i \neq \emptyset\}; \quad UX = U;$$
$$V = \{j \in \{1, 2, ..., n\} : X^j \neq \emptyset\};$$

for $i \in U$ **do**
{
$UX = UX \setminus \{i\}$;
for $I \in 2^{UX}$ **do**
{
$VY = V$;
for $j \in V$ **do**
{
if $(X_{iI}^{j\emptyset} = \emptyset)$ **break**;
$VY = VY \setminus \{j\}$;
for $J \in 2^{VY}$ **do**
if $(Y_{jJ}^{iI} \neq \emptyset)$ $NE = NE \cup (X_{iI}^{jJ} \times Y_{jJ}^{iI})$;
}
}
}

Algorithm executes the interior operator **if** no more then

$$\begin{aligned}
&2^{m-1}(2^{n-1}+2^{n-2}+\cdots+2^1+2^0)+\\
&+2^{m-2}(2^{n-1}+2^{n-2}+\cdots+2^1+2^0)+\\
&\cdots\\
&+2^1(2^{n-1}+2^{n-2}+\cdots+2^1+2^0)+\\
&+2^0(2^{n-1}+2^{n-2}+\cdots+2^1+2^0)=\\
&=(2^m-1)(2^n-1)
\end{aligned}$$

times. So, the following theorem is true.

Theorem 4. *The algorithm examines no more then* $(2^m-1)(2^n-1)$ *polytopes of the* $X_{iI}^{jJ}\times Y_{jJ}^{iI}$ *type.*

If all the players' strategies are equivalent, then NES consists of $(2^m-1)(2^n-1)$ polytopes.

Evidently, for practical reasons algorithm may be improved by identifying equivalent, dominant and dominated strategies in pure game [5, 8, 9] with the following pure and extended game simplification. *"In a nondegenerate game, both players use the same number of pure strategies in equilibrium, so only supports of equal cardinality need to be examined"* [6]. This property may be used to minimize essentially the number of components $X_{iI}^{jJ}\times Y_{jJ}^{iI}$ examined in nondegenerate game.

Example 1. Matrices of the two person game [6] are

$$A=\begin{bmatrix}1&0&4\\0&2&3\end{bmatrix},\quad B=\begin{bmatrix}0&2&3\\6&5&3\end{bmatrix}.$$

The exterior cycle is executed for the value $i=1$.

As

$$X_{1\emptyset}^{1\emptyset}=\left\{\mathbf{x}:\ \begin{array}{l}2x_1-x_2\le 0,\\3x_1-3x_2\le 0,\\x_1+x_2=1,\\x_1\ge 0, x_2=0\end{array}\right\}=\emptyset$$

then the cycle for $j=1$ is omitted.

Since

$$X_{1\emptyset}^{2\emptyset} = \left\{ \mathbf{x}: \begin{array}{l} -2x_1 + x_2 \le 0, \\ x_1 - 2x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 \ge 0, x_2 = 0 \end{array} \right\} = \emptyset$$

then the cycle for j=2 is omitted.

As

$$X_{1\emptyset}^{3\emptyset} = \left\{ \mathbf{x}: \begin{array}{l} -3x_1 + 3x_2 \le 0, \\ -x_1 + 2x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 \ge 0, x_2 = 0 \end{array} \right\} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\} \ne \emptyset,$$

$$Y_{3\emptyset}^{1\emptyset} = \left\{ \mathbf{y}: \begin{array}{l} -y_1 + 2y_2 - y_3 \le 0, \\ y_1 + y_2 + y_3 = 1, \\ y_1 = 0, y_2 = 0, y_3 \ge 0 \end{array} \right\} = \left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\} \ne \emptyset,$$

the point $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ is a Nash equilibrium for which $f_1 = 4$, $f_2 = 3$.

$$X_{1\{2\}}^{1\emptyset} = \left\{ \mathbf{x}: \begin{array}{l} 2x_1 - x_2 \le 0, \\ 3x_1 - 3x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 \ge 0, x_2 \ge 0 \end{array} \right\} \ne \emptyset,$$

$$Y_{1\emptyset}^{1\{2\}} = \left\{ \mathbf{y}: \begin{array}{l} -y_1 + 2y_2 - y_3 = 0, \\ y_1 + y_2 + y_3 = 1, \\ y_1 \ge 0, y_2 = 0, y_3 = 0 \end{array} \right\} = \emptyset,$$

Since

$$X_{1\{2\}}^{1\{2\}} = \left\{ \mathbf{x}: \begin{array}{l} 2x_1 - x_2 = 0, \\ 3x_1 - 3x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 \ge 0, x_2 \ge 0 \end{array} \right\} = \begin{pmatrix} 1/3 \\ 2/3 \end{pmatrix} \ne \emptyset,$$

$$Y_{1\{2\}}^{1\{2\}} = \left\{ \mathbf{y}: \begin{array}{l} -y_1 + 2y_2 - y_3 = 0, \\ y_1 + y_2 + y_3 = 1, \\ y_1 \ge 0, y_2 \ge 0, y_3 = 0 \end{array} \right\} = \begin{pmatrix} 2/3 \\ 1/3 \\ 0 \end{pmatrix} \ne \emptyset,$$

the point $\begin{pmatrix} 1/3 \\ 2/3 \end{pmatrix} \times \begin{pmatrix} 2/3 \\ 1/3 \\ 0 \end{pmatrix}$ is a Nash equilibrium for which $f_1 = 2/3$, $f_2 = 4$.

$$X_{1\{2\}}^{1\{3\}} = \left\{ \mathbf{x} : \begin{array}{l} 2x_1 - x_2 \leq 0, \\ 3x_1 - 3x_2 = 0, \\ x_1 + x_2 = 1, \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right\} = \emptyset,$$

$$X_{1\{2\}}^{1\{2,3\}} = \left\{ \mathbf{x} : \begin{array}{l} 2x_1 - x_2 = 0, \\ 3x_1 - 3x_2 = 0, \\ x_1 + x_2 = 1, \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right\} = \emptyset,$$

$$X_{1\{2\}}^{2\emptyset} = \left\{ \mathbf{x} : \begin{array}{l} -2x_1 + x_2 \leq 0, \\ x_1 - 2x_2 \leq 0, \\ x_1 + x_2 = 1, \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right\} \neq \emptyset,$$

$$Y_{2\emptyset}^{1\{2\}} = \left\{ \mathbf{y} : \begin{array}{l} -y_1 + 2y_2 - y_3 = 0, \\ y_1 + y_2 + y_3 = 1, \\ y_1 = 0, y_2 \geq 0, y_3 = 0 \end{array} \right\} = \emptyset,$$

As

$$X_{1\{2\}}^{2\{3\}} = \left\{ \mathbf{x} : \begin{array}{l} -2x_1 + x_2 \leq 0, \\ x_1 - 2x_2 = 0, \\ x_1 + x_2 = 1, \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right\} = \begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix} \neq \emptyset,$$

$$Y_{2\{3\}}^{1\{2\}} = \left\{ \mathbf{y} : \begin{array}{l} -y_1 + 2y_2 - y_3 = 0, \\ y_1 + y_2 + y_3 = 1, \\ y_1 = 0, y_2 \geq 0, y_3 = 0 \end{array} \right\} = \begin{pmatrix} 0 \\ 1/3 \\ 2/3 \end{pmatrix} \neq \emptyset,$$

the point $\begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1/3 \\ 2/3 \end{pmatrix}$ is a Nash equilibrium for which $f_1 = 8$,

$f_2 = 3$.

$$X_{1\{2\}}^{3\emptyset} = \left\{ \mathbf{x} : \begin{array}{l} -2x_1 + x_2 \le 0, \\ x_1 - 2x_2 = 0, \\ x_1 + x_2 = 1, \\ x_1 \ge 0, x_2 \ge 0 \end{array} \right\} \ne \emptyset,$$

$$Y_{3\emptyset}^{1\{2\}} = \left\{ \mathbf{y} : \begin{array}{l} -y_1 + 2y_2 - y_3 = 0, \\ y_1 + y_2 + y_3 = 1, \\ y_1 = 0, y_2 = 0, y_3 \ge 0 \end{array} \right\} = \emptyset.$$

The exterior cycle is executed for the value $i = 2$.

$$X_{2\emptyset}^{1\emptyset} = \left\{ \mathbf{x} : \begin{array}{l} 2x_1 - x_2 \le 0, \\ 3x_1 - 3x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 = 0, x_2 \ge 0 \end{array} \right\} \ne \emptyset,$$

$$Y_{1\emptyset}^{2\emptyset} = \left\{ \mathbf{y} : \begin{array}{l} y_1 - 2y_2 + y_3 \le 0, \\ y_1 + y_2 + y_3 = 1, \\ y_1 \ge 0, y_2 = 0, y_3 = 0 \end{array} \right\} = \emptyset,$$

$$X_{2\emptyset}^{1\{2\}} = \left\{ \mathbf{x} : \begin{array}{l} 2x_1 - x_2 = 0, \\ 3x_1 - 3x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 = 0, x_2 \ge 0 \end{array} \right\} = \emptyset,$$

$$X_{2\emptyset}^{1\{3\}} = \left\{ \mathbf{x} : \begin{array}{l} 2x_1 - x_2 \le 0, \\ 3x_1 - 3x_2 = 0, \\ x_1 + x_2 = 1, \\ x_1 = 0, x_2 \ge 0 \end{array} \right\} = \emptyset,$$

$$X_{2\emptyset}^{1\{2,3\}} = \left\{ \mathbf{x} : \begin{array}{l} 2x_1 - x_2 = 0, \\ 3x_1 - 3x_2 = 0, \\ x_1 + x_2 = 1, \\ x_1 = 0, x_2 \ge 0 \end{array} \right\} = \emptyset.$$

Since

$$X_{2\emptyset}^{2\emptyset} = \left\{ \mathbf{x} : \begin{array}{l} -2x_1 + x_2 \le 0, \\ x_1 - 2x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 = 0, x_2 \ge 0 \end{array} \right\} = \emptyset.$$

the cycle for $j = 2$ is omitted.

$$X_{2\emptyset}^{3\emptyset} = \left\{ \mathbf{x} : \begin{array}{l} -3x_1 + 3x_2 \le 0, \\ -x_1 + 2x_2 \le 0, \\ x_1 + x_2 = 1, \\ x_1 = 0, x_2 \ge 0 \end{array} \right\} = \emptyset.$$

Thus, the NES consists of three elements — one pure and two mixed Nash equilibria.

The following example 2 illustrates that a simple modification in the first example of one element of the cost matrix of the first player changes the NES to a continue power set that consists of one isolated point and one segment.

Example 2. If in the first example the element a_{23} of the matrix A is modified

$$A = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 2 & \boxed{4} \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 2 & 3 \\ 6 & 5 & 3 \end{bmatrix},$$

then the NES of the obtained game consists of one distinct point $\begin{pmatrix} 1/3 \\ 2/3 \end{pmatrix} \times \begin{pmatrix} 2/3 \\ 1/3 \\ 0 \end{pmatrix}$ for which $f_1 = 10/9, f_2 = 4$ and of one distinct segment $\left[\begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ for which $f_1 = 4, f_2 = 3$.

A subsequent modification of the cost matrix of the second player in precedent game transforms the NES into non-convex continuum.

Example 3. If in the second example the first column of the matrix B is equal to the second column

$$A = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 2 & \boxed{4} \end{bmatrix}, \quad B = \begin{bmatrix} \boxed{2} & 2 & 3 \\ \boxed{5} & 5 & 3 \end{bmatrix},$$

then the NES of the such game consists of four connected segments:

- $\left[\begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right] \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 1-1/3\lambda \\ 1/3\lambda \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$,
  $\lambda \in [0;1]$, for which $f_1 = 4,\ f_2 = 3$,
- $\begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix} \times \left[\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2/3 \\ 1/3 \\ 0 \end{pmatrix}\right] \equiv \begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix} \times \begin{pmatrix} 2/3-2/3\mu \\ 1/3-1/3\mu \\ \mu \end{pmatrix}$,
  $\mu \in [0;1]$, for which $f_1 = 2/3 + 10/3\mu,\ f_2 = 3$,
- $\left[\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix}\right] \times \begin{pmatrix} 2/3 \\ 1/3 \\ 0 \end{pmatrix} \equiv \begin{pmatrix} 2/3-2/3\lambda \\ 1/3+1/3\lambda \end{pmatrix} \times \begin{pmatrix} 2/3 \\ 1/3 \\ 0 \end{pmatrix}$,
  $\lambda \in [0;1]$, for which $f_1 = 2/3,\ f_2 = 3 + 2\lambda$,
- $\begin{pmatrix} 0 \\ 1 \end{pmatrix} \times \left[\begin{pmatrix} 2/3 \\ 1/3 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\right] \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} 2/3\mu \\ 1-2/3\mu \\ 0 \end{pmatrix}$,
  $\mu \in [0;1]$, for which $f_1 = 2 - 4/3\mu,\ f_2 = 5$.

The following example 4 supplements preceding examples and illustrates that one of the NE may be strong dominant (optimal by Pareto) among all the others NE.

Example 4. The NES of the extended game with matrices:

$$A = \begin{bmatrix} 2 & 1 & 6 \\ 3 & 2 & -1 \\ -1 & 2 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 3 \\ -1 & 1 & -2 \\ 2 & -1 & 2 \end{bmatrix},$$

consists of two isolated points and one segment:

- $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, for which $f_1 = 6,\ f_2 = 3$.
- $\begin{pmatrix} 1/14 \\ 12/14 \\ 2/14 \end{pmatrix} \times \begin{pmatrix} 1/4 \\ 1/2 \\ 1/4 \end{pmatrix}$, for which $f_1 = 45/28,\ f_2 = -1/8$.

- $\left[\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}0\\3/5\\2/5\end{pmatrix}\right]\times\begin{pmatrix}0\\1\\0\end{pmatrix}\equiv\begin{pmatrix}0\\3/5+2/5\lambda\\2/5-2/5\lambda\end{pmatrix}\times\begin{pmatrix}0\\1\\0\end{pmatrix}$, $\lambda\in[0;1]$, for which $f_1=2,\ f_2=1/5+4/5\lambda$.

Evidently, the NE $\begin{pmatrix}1\\0\\0\end{pmatrix}\times\begin{pmatrix}0\\0\\1\end{pmatrix}$, for which $f_1=6,\ f_2=3$, strongly dominates all the other NE and it is more preferable than the other NE for both of the players.

Example 1 confirms the well known property that the number of Nash equilibria in the nondegenerate game is odd [2, 6]. Examples 2 – 4 illustrate that in the degenerate game the number of convex components may be both the even and the odd. Examples 3 – 4 illustrate that in the degenerate game the player cannot increase own gain by modifying his Nash strategy, but, by modifying his Nash strategy the player may essentially modify (increase or decrease) the gain of the opponent.

3 NES in n-player mixed extended $m_1\times m_2\ldots m_n$ games

Consider the n-player extended game $\widetilde{\Gamma}=\langle X_p, f_p(\mathbf{x}), p\in N\rangle$ formulated in the Introduction. The cost function of the player p is linear if the strategies of the remaining players are fixed:

$$f_p(\mathbf{x})=\left(\sum_{\mathbf{s}_{-p}\in S_{-p}} a^p_{1||\mathbf{s}_{-p}}\prod_{\substack{q=\overline{1,n}\\q\neq p}} x^q_{s_q}\right)x^p_1+$$

$$+\left(\sum_{\mathbf{s}_{-p}\in S_{-p}} a^p_{2||\mathbf{s}_{-p}}\prod_{\substack{q=\overline{1,n}\\q\neq p}} x^q_{s_q}\right)x^p_2+$$

$$+\cdots+$$
$$\left(\sum\nolimits_{\mathbf{s}_{-p}\in S_{-p}} a^p_{m_p||\mathbf{s}_{-p}} \prod_{\substack{q=\overline{1,n}\\ q\neq p}} x^q_{s_q}\right) x^p_{m_p}.$$

Thus, the player p has to solve a linear parametric problem with parameter vector $x^{-p} \in X^{-p}$:

$$f_p\left(\mathbf{x}^p, \mathbf{x}^{-p}\right) \to \max, \quad \mathbf{x}^p \in X_p, \quad p = \overline{1,n}.$$

The solution of this problem is realized on the vertices of polytope X_p that has m_p vertices — the x^p_i axis unit vectors $\mathbf{e}^{x^p_i} \in R^{m_i}, i = \overline{1,m_p}$. Thus, in accordance with the simplex method and its optimality criterion [7], the parameter set X_{-p} is partitioned into such m_p subsets

$$X_{-p}(i_p) =$$
$$= \left\{ \mathbf{x}^{-p} \in R^{m-m_p} : \begin{array}{c} \sum\limits_{\mathbf{s}_{-p}\in S_{-p}} \left(a^p_{k||\mathbf{s}_{-p}} - a^p_{i_p||\mathbf{s}_{-p}}\right) \prod\limits_{\substack{q=\overline{1,n}\\ q\neq p}} x^q_{s_q} \le 0, \\ k = \overline{1,m_p}, \\ \sum\limits_{i=1}^{m_q} x^q_i = 1, \quad q = \overline{1,n},\ q \neq p, \\ \mathbf{x}^{-p} \ge 0 \end{array} \right\},$$

$i_p = \overline{1,m_p}$, for which one of the optimal solution of the linear programming problem (1) is $\mathbf{e}^{x^p_i}$.

Let $U_p = \{i_p \in \{1,...,m_p\} : X_{-p}(i_p) \neq \emptyset\}, \mathbf{e}\mathbf{p}^T = (1,\ldots,1) \in R^{m_p}$. In conformity with the optimality criterion of the simplex method $\forall i_p \in U_p$ and $\forall I_p \in 2^{U_p\setminus\{i_p\}}$ all the points of

$$\mathbf{Conv}\left\{\mathbf{e}^{x^p_k}, k \in I_p \cup \{i_p\}\right\} = \left\{ \mathbf{x} \in R^m : \begin{array}{l} \mathbf{e}\mathbf{p}^T\mathbf{x}^p = 1, \\ \mathbf{x}^p \ge 0, \\ x^p_k = 0, k \notin I_p \cup \{i_p\} \end{array} \right\}$$

are optimal for parameters $\mathbf{x}^{-p} \in X_{-p}(i_p I_p) \subset R^{m-m_p}$, where

$X_{-p}(i_pI_p)$ is a set of solutions of the system:

$$\begin{cases} \sum\limits_{\mathbf{s}_{-p}\in S_{-p}} \left(a^p_{k||\mathbf{s}_{-p}} - a^p_{i_p||\mathbf{s}_{-p}}\right) \prod\limits_{\substack{q=\overline{1,n}\\ q\neq p}} x^q_{s_q} = 0, \quad k \in I_p, \\ \sum\limits_{\mathbf{s}_{-p}\in S_{-p}} \left(a^p_{k||\mathbf{s}_{-p}} - a^p_{i_p||\mathbf{s}_{-p}}\right) \prod\limits_{\substack{q=\overline{1,n}\\ q\neq p}} x^q_{s_q} \leq 0, \quad k \notin I_p \cup i_p, \\ \mathbf{e}\mathbf{r}^T\mathbf{x}^r = 1, \quad r=\overline{1,n},\ r\neq p, \\ \mathbf{x}^r \geq 0, \quad r=\overline{1,n},\ r\neq p. \end{cases}$$

Evidently $X_{-p}(i_p\emptyset) = X_{-p}(i_p\emptyset)$. Hence,

$$Gr_p = \bigcup_{i_p\in U_p,\ I_p\in 2^{U_p\setminus\{i_p\}}} \mathbf{Conv}\left\{\mathbf{e}^{x^p_k}, k\in I_p\cup\{i_p\}\right\}\times X_{-p}(i_pI_p).$$

Finally,

$$NE(\widetilde{\Gamma}) = \bigcap_{p=1}^{n} Gr_p = \bigcup_{\substack{i_1\in U_1,\ I_1\in 2^{U_1\setminus\{i_1\}},\\ \dots\\ i_n\in U_n,\ I_n\in 2^{U_n\setminus\{i_n\}}}} X(i_1I_1\dots i_nI_n)$$

where $X(i_1I_1\dots i_nI_n) = NE(i_1I_1\dots i_nI_n)$ is a set of solutions of the system:

$$\begin{cases} \sum\limits_{\mathbf{s}_{-r}\in S_{-r}} \left(a^r_{k||\mathbf{s}_{-r}} - a^r_{i_r||\mathbf{s}_{-r}}\right) \prod\limits_{\substack{q=\overline{1,n}\\ q\neq r}} x^q_{s_q} = 0, \quad k \in I_r, \\ \sum\limits_{\mathbf{s}_{-r}\in S_{-r}} \left(a^r_{k||\mathbf{s}_{-r}} - a^r_{i_r||\mathbf{s}_{-r}}\right) \prod\limits_{\substack{q=\overline{1,n}\\ q\neq r}} x^q_{s_q} \leq 0, \quad k \notin I_r \cup i_r, \\ r=\overline{1,n}, \\ \\ \mathbf{e}\mathbf{r}^T\mathbf{x}^r = 1, \quad \mathbf{x}^r \geq 0,\ r=\overline{1,n}, \\ x^r_k = 0, \quad k\notin I_r\cup\{i_r\},\ r=\overline{1,n}. \end{cases}$$

Theorem 5. $NE(\widetilde{\Gamma}) = \bigcup_{\substack{i_1 \in U_1,\ I_1 \in 2^{U_1}\setminus\{i_1\} \\ \cdots \\ i_n \in U_n,\ I_n \in 2^{U_n}\setminus\{i_n\}}} X\left(i_1 I_1 \ldots i_n I_n\right).$

The theorem 5 is an extension of theorem 1 to an n-player game. The proof is performed above.

The following theorem is a corollary of theorem 5.

Theorem 6. $NE(\widetilde{\Gamma})$ *consists of no more then*

$$(2^{m_1}-1)(2^{m_2}-1)\ldots(2^{m_n}-1)$$

components of the type $X(i_1 I_1 \ldots i_n I_n)$.

In game for which all the players have equivalent strategies NES is partitioned in maximal number $(2^{m_1}-1)(2^{m_2}-1)\ldots(2^{m_n}-1)$ of components.

In general, in n-player game ($n \geq 3$) components $X(i_1 I_1 \ldots i_n I_n)$ are non-convex.

An exponential algorithm for NES computing in n-player game simply follows from the expression in theorem 5. The algorithm requires to solve $(2^{m_1}-1)(2^{m_2}-1)\ldots(2^{m_n}-1)$ finite systems of multilinear ($n-1$-linear) and linear equations and inequalities in m variables. The last problem is itself a difficult one.

Example 5. It is considered a three-player extended $2 \times 2 \times 2$ (diadic) game [2] with matrices:

$$a_{1**} = \begin{bmatrix} 9 & 0 \\ 0 & 3 \end{bmatrix}, \quad a_{2**} = \begin{bmatrix} 0 & 3 \\ 9 & 0 \end{bmatrix},$$

$$b_{*1*} = \begin{bmatrix} 8 & 0 \\ 0 & 4 \end{bmatrix}, \quad b_{*2*} = \begin{bmatrix} 0 & 4 \\ 8 & 0 \end{bmatrix},$$

$$c_{**1} = \begin{bmatrix} 12 & 0 \\ 0 & 2 \end{bmatrix}, \quad c_{**2} = \begin{bmatrix} 0 & 6 \\ 4 & 0 \end{bmatrix}.$$

$$f_1(\mathbf{x},\mathbf{y},\mathbf{z}) = (9y_1z_1 + 3y_2z_2)x_1 + (9y_2z_1 + 3y_1z_2)x_2,$$

$$f_2(\mathbf{x},\mathbf{y},\mathbf{z}) = (8x_1z_1 + 4x_2z_2)y_1 + (8x_2z_1 + 4x_1z_2)y_2,$$

$$f_3(\mathbf{x},\mathbf{y},\mathbf{z}) = (12x_1y_1 + 2x_2y_2)z_1 + (4x_2y_1 + 6x_1y_2)z_2.$$

Totally, we have to consider $(2^2-1)(2^2-1)(2^2-1) = 27$ components. Further, we will enumerate only nonempty components. Thus, $NE(1\emptyset1\emptyset1\emptyset) = (1;0)\times(1;0)\times(1;0)$ (for which $f_1 = 9, f_2 = 8, f_3 = 12$) is the solution of the system:

$$\begin{cases} 9y_2z_1 + 3y_1z_1 - 9y_1z_1 - 3y_2z_2 = 3(y_2-y_1)(3z_1-z_2) \le 0, \\ 8x_2z_1 + 4x_1z_2 - 8x_1z_1 - 4x_2z_2 = 4(x_2-x_1)(4z_1-z_2) \le 0, \\ 4x_2y_1 + 6x_1y_2 - 12x_1y_1 + 2x_2y_2 = 2(3x_1-x_2)(y_2-2y_1) \le 0, \\ x_1 + x_2 = 1, x_1 \ge 0, x_2 = 0, \\ y_1 + y_2 = 1, y_1 \ge 0, y_2 = 0, \\ z_1 + z_2 = 1, z_1 \ge 0, z_2 = 0. \end{cases}$$

$NE(1\emptyset2\emptyset2\emptyset) = (1;0)\times(0;1)\times(0;1)$ and $f_1 = 3, f_2 = 4, f_3 = 6$:

$$\begin{cases} 3(y_2-y_1)(3z_1-z_2) \le 0, \\ -4(x_2-x_1)(4z_1-z_2) \le 0, \\ -2(3x_1-x_2)(y_2-2y_1) \le 0, \\ x_1 + x_2 = 1, x_1 \ge 0, x_2 = 0, \\ y_1 + y_2 = 1, y_1 = 0, y_2 \ge 0, \\ z_1 + z_2 = 1, z_1 = 0, z_2 \ge 0. \end{cases}$$

$NE(2\emptyset1\emptyset2\emptyset) = (0;1)\times(1;0)\times(0;1)$ and $f_1 = 3, f_2 = 4, f_3 = 4$:

$$\begin{cases} -3(y_2-y_1)(3z_1-z_2) \le 0, \\ 4(x_2-x_1)(4z_1-z_2) \le 0, \\ -2(3x_1-x_2)(y_2-2y_1) \le 0, \\ x_1 + x_2 = 1, x_1 = 0, x_2 \ge 0, \\ y_1 + y_2 = 1, y_1 \ge 0, y_2 = 0, \\ z_1 + z_2 = 1, z_1 = 0, z_2 \ge 0. \end{cases}$$

$NE(2\emptyset 2\emptyset 1\emptyset) = (0;1) \times (0;1) \times (1;0)$ and $f_1 = 9, f_2 = 8, f_3 = 2$:

$$\begin{cases} -3(y_2 - y_1)(3z_1 - z_2) \le 0, \\ -4(x_2 - x_1)(4z_1 - z_2) \le 0, \\ 2(3x_1 - x_2)(y_2 - 2y_1) \le 0, \\ x_1 + x_2 = 1, x_1 = 0, x_2 \ge 0, \\ y_1 + y_2 = 1, y_1 = 0, y_2 \ge 0, \\ z_1 + z_2 = 1, z_1 \ge 0, z_2 = 0. \end{cases}$$

$NE(1\emptyset 1\{2\}1\{2\}) = (1;0) \times (1/3;2/3) \times (1/5;4/5)$ and $f_1 = 11/5$, $f_2 = 8/3$, $f_3 = 4$:

$$\begin{cases} 3(y_2 - y_1)(3z_1 - z_2) \le 0, \\ 4(x_2 - x_1)(4z_1 - z_2) = 0, \\ 2(3x_1 - x_2)(y_2 - 2y_1) = 0, \\ x_1 + x_2 = 1, x_1 \ge 0, x_2 = 0, \\ y_1 + y_2 = 1, y_1 \ge 0, y_2 \ge 0, \\ z_1 + z_2 = 1, z_1 \ge 0, z_2 \ge 0. \end{cases}$$

$NE(1\{2\}2\emptyset 1\{2\}) = (2/5;3/5) \times (0;1) \times (1/4;3/4)$ and $f_1 = 9/4$, $f_2 = 12/5, f_3 = 21/10$:

$$\begin{cases} 3(y_2 - y_1)(3z_1 - z_2) = 0, \\ -4(x_2 - x_1)(4z_1 - z_2) \le 0, \\ 2(3x_1 - x_2)(y_2 - 2y_1) = 0, \\ x_1 + x_2 = 1, x_1 \ge 0, x_2 \ge 0, \\ y_1 + y_2 = 1, y_1 = 0, y_2 \ge 0, \\ z_1 + z_2 = 1, z_1 \ge 0, z_2 \ge 0. \end{cases}$$

$NE(1\{2\}1\{2\}1\emptyset) = (1/2;1/2) \times (1/2;1/2) \times (1;0)$ and $f_1 = 9/2$, $f_2 = 4, f_3 = 7/2$:

$$\begin{cases} 3(y_2 - y_1)(3z_1 - z_2) = 0, \\ -4(x_2 - x_1)(4z_1 - z_2) = 0, \\ 2(3x_1 - x_2)(y_2 - 2y_1) \le 0, \\ x_1 + x_2 = 1, x_1 \ge 0, x_2 \ge 0, \\ y_1 + y_2 = 1, y_1 \ge 0, y_2 \ge 0, \\ z_1 + z_2 = 1, z_1 \ge 0, z_2 = 0. \end{cases}$$

$$NE(1\{2\}1\{2\}1\{2\}) =$$

$$\left\{ \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} \times \begin{pmatrix} 1/3 \\ 2/3 \end{pmatrix} \times \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix}, \begin{pmatrix} 2/5 \\ 3/5 \end{pmatrix} \times \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} \times \begin{pmatrix} 1/5 \\ 4/5 \end{pmatrix} \right\}$$

for which $f_1 = 9/4, f_2 = 5/2, f_3 = 8/3$ and respective $f_1 = 21/10$, $f_2 = 12/5, f_3 = 63/25$:

$$\begin{cases} 3(y_2 - y_1)(3z_1 - z_2) = 0, \\ -4(x_2 - x_1)(4z_1 - z_2) = 0, \\ 2(3x_1 - x_2)(y_2 - 2y_1) = 0, \\ x_1 + x_2 = 1, x_1 \geq 0, x_2 \geq 0, \\ y_1 + y_2 = 1, y_1 \geq 0, y_2 \geq 0, \\ z_1 + z_2 = 1, z_1 \geq 0, z_2 \geq 0. \end{cases}$$

Thus the game has 9 Nash equilibria. Remark that the last component of the NES consists of two distinct points. Hence, it is non-convex and non-connected.

4 Conclusions

The idea to consider NES as an intersection of the graphs of best response mappings yields to a simply NES representation and to a method of NES computing. Taking into account the computational complexity of the problem, the proposed exponential algorithms are pertinent.

The NES in two-matrix extended games may be partitioned into finite number of polytopes, no more then $(2^m-1)(2^n-1)$. The proposed algorithm examines, in general, a much more smaller number of sets of the type $X_{iI}^{jJ} \times Y_{jJ}^{iI}$.

The NES in multi-matrix extended games may be partitioned into finite number of components, no more then $(2^{m_1} - 1) \dots (2^{m_n} - 1)$, but they, in general, are non-convex and moreover non-polytopes. The algorithmic realization of the method is closely related with the problem of solving the systems of multilinear (n − 1-linear and simply linear) equations and inequalities, that represents in itself a serious obstacle to efficient NES computing.

References

[1] I. Gilboa, E. Zemel. *Nash and correlated equilibria: some complexity considerations,* Games and Economic Behavior, Volume 1, 1989, pp. 80–93.

[2] R.D. McKelvey, A. McLenan. *Computation of equilibria in finite games,* Handbook of Computational Economics, Volume 1, Elsevier, 1996, pp. 87–142.

[3] H. Moulen. *Théorie des jeux pour l'éqonomie et la politique,* Paris, 1981 (in Russian: Games Theory, Moscow, Mir, 1985, 200 p.).

[4] J.F. Nash. *Noncooperative game,* Annals of Mathematics, Volume 54, 1951, pp. 280–295.

[5] M. Sagaidac, V. Ungureanu. *Operational research,* Chişinău, CEP USM, 2004, 296 p. (in Romanian).

[6] von Stengel, Bernhard. *Computing equilibria for two-person games,* Handbook of Game Theory with Economic Applications, Volume 3, Elsevier, 2002, pp. 1723–1759.

[7] V. Ungureanu. *Mathematical programming,* Chişinău, USM, 2001, 344 p. (in Romanian).

[8] V. Ungureanu, A. Botnari. *Nash equilibria sets in mixed extension of* $2 \times 2 \times 2$ *games,* Computer Science Journal of Moldova, Volume 13, no. 1 (37), 2005, pp. 13–28.

[9] V. Ungureanu, A. Botnari. *Nash equilibria sets in mixed extended* 2×3 *games,* Computer Science Journal of Moldova, Volume 13, no. 2 (38), 2005, pp. 136–150.

V. Ungureanu

Received October 10, 2005

State University of Moldova,
60, A. Mateevici str.,
Chişinău, MD−2009, Moldova.
E–mail: *valungureanu@usm.md*

Computer Science Journal of Moldova, vol.**14**, no.3(42), 2006

Graph Coloring using Peer-to-Peer Networks*

Adrian Iftene Cornelius Croitoru

Abstract

The popularity of distributed file systems continues to grow in last years. The reasons they are preferred over traditional centralized systems include fault tolerance, availability, scalability and performance. In this paper, we propose a framework for analyzing peer-to-peer content distributed technologies and their applications in the cooperative solving of combinatorial optimization problems. Our approach, which follows the Content Addressable Network model, is scalable, fault-tolerant and self-organizing; we improved also load distribution at the insertion and deletion of nodes. We use this network for the classical "graph coloring" problem, in order to reduce the computational time for its cooperative solving.

Keywords: CAN, graph coloring, peer-to-peer network

1 Introduction

In the last years, Peer-to-Peer system research has grown significantly. Using a large scale a distributed network of machines has become an important element of distributed computing due to the extraordinary popularity of Peer-to-Peer services like Napster, Gnutella, Kazaa and Morpheus. Though these systems are famous mainly for file sharing, we can see how P2P (Peer-to-Peer) systems are becoming a very good area for research.

P2P systems offer a decentralized, self-sustained, scalable, fault tolerant and symmetric network of machines providing an effective balancing of storage and bandwidth resources.

*A preliminary version of this paper was presented at 5^{th} RoEduNet International Conference, Sibiu, Romania, 1-3 June 2006

Unfortunately, most of the current peer-to-peer designs are not scalable. For example, in Napster a central server stores the index of all the files available within the Napster user community and all the user queries for a specific file are sending to this computer. This server sends the answer with IP address, which has the file, and after that the user can download the file directly from this machine. In this way, the process of locating a file is very centralized and makes it very vulnerable (since there is a single point of failure). Gnutella goes one step further and decentralizes the file location process as well. Users in a Gnutella network self-organize into an application-level mesh on which requests for a file are flooded with a certain scope. Flooding on every request is clearly not scalable and, because the flooding has to be curtailed at some point, may fail to find content that is actually in the system.

In this paper we describe the design of a peer-to-peer file system *completely distributed* (it requires no form of centralized control, coordination or configuration), *scalable* (nodes maintain only a small amount of control state that is independent of the number of nodes in the system) and *fault tolerant* (nodes can route around failures) which is used to cooperative solve the graph coloring problem.

Our interest in CANs (Content Addressable Networks) is based on the belief that this abstraction would give a powerful design tool that could enable new applications and communication models.

Section 2 of this paper presents the P2P networks and their main design issues, while section 3 describes our approach based on CAN and specific operations from this class of networks: insertion and deletion of nodes. Section 4 presents the main algorithm used for graph coloring. Then, section 5 presents the results of our tests on classical graphs. Finally, section 6 discusses the feasibility of the approach in practical settings and briefly presents some further developments.

2 P2P Networks

The term Peer-to-Peer (P2P) refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner. Each node from this system has the same re-

sponsibility. Basic P2P system goals are decentralization, immediate connectivity, reduced cost of ownership and anonymity. Androutsellis-Theotokis [1] defined P2P as "*applications that take advantages of resources (storage, cycles, content, human presence) available at the edges of the Internet*".

To make ubiquitous computing become a reality, the computing devices must become reliable, resilient and have distributed access to data. The P2P architecture can help reduce storage system costs and allow cost sharing by using existing infrastructure and bundling resources from different sites. Considering these factors, the P2P model should be very useful in designing the future generation distributed file systems.

2.1 Design Issues in P2P Networks

Peer-to-Peer systems have basic properties that separate them from conventional distributed systems. We describe in this section different design issues and their effect in system performance [2].

Symmetry: Symmetry comes among the roles of the participant nodes. We assume that all nodes have the same characteristics. In client-server systems usually the servers are more powerful than the clients. In our network each node has the ability to be server and client in the same time.

Decentralization: P2P systems are decentralized by nature, and they have mechanisms for distributed storage, processing, information sharing. In this way scalability, resilience to faults and availability are increased. But, in the same time our tries to see global system and its behavior are without success.

Operations with Volunteer Participants: An important design issue is that the participants can neither be expected nor enforced. They haven't a manager or a centralized authority and can be inserted or removed from the system at any time. The system should be robust enough to survive the removal or failure of nodes at any moment.

Fast Resource Location: One of the most important P2P design is the method used for resource location. Because resources are distributed in diverse peers, an efficient mechanism for object location becomes the deciding factor in the performance of such system. Currently, object location and routing systems include Chord, Pastry, Tapestry and CAN. Pastry and Tapestry use Plaxton trees, basing their routing on address prefixes that is a generalization of hypercube routing. However, Pastry and Tapestry add robustness, dynamism and self-organizing properties to the Plaxton scheme. Chord uses the numerical difference with the destination address to route messages. The Content Addressable Network (CAN) uses a d-dimensional space to route messages. Another important location strategy used in few systems is Distributed Hash Table (DHT). It uses hashing of file or resource names to locate the object.

Load Balancing: Load balancing is very important in one robust P2P system. The system must have optimal distribution of resources based on capability and availability of node resources. The system should prevent the building of locations where the load is high and also should be possible to re-balance the system based on usage patterns.

Anonymity: With scope to prevent censorship we need to have anonymity in our system. Our network must assure anonymity for producer and for consumer of information.

Scalability: The number of peers from network should be of any value (hundreds, thousands or millions), and that should not affect the network behavior. Unfortunately actual systems are not scalable over few hundreds or thousands of nodes.

Persistence of Information: Methods from our system should be able to provide persistent data to consumers, the data stored in the system is safe, protected against destruction, and highly available in a transparent manner.

Security: Security from attacks and system failure are design goals for every system. The system should be able to assure data security, and this can be achieved with encryption, different coding schemes, etc.

2.2 Existing Systems

Designing a system that can implement all properties from above is very difficult. Most of existing systems utilize specific properties or mechanisms, and that involves some advantages and also some disadvantages.

FreeNet: This system presented in [3] is an adaptive P2P that enables the publication, replication and retrieval of data while protecting the anonymity of the authors, readers and data location. It uses probabilistic routing for that, and it works over many individual computers that allow files to be inserted, stored and requested. The file identification is made with a hash function. Any request may be locally processed, or on failure may be routed to the closest neighbor accordingly to the routing table. Communications between Freenet nodes are encrypted.

CFS: Cooperative File System (CFS) [4] is a peer-to-peer system developed by MIT with the following design goals: provable for efficiency, robustness, load balancing and scalability. CFS uses DHT for storage of blocks. The file system is being built like a set of blocks distributed over the CFS servers. CFS has three layers: FS, which interprets blocks as files and presents an interface to applications; Dhash - the distributed hash table that stores unstructured data blocks; and *Chord*, which maintains routing tables for lookup and query manager. CFS is read only system from user's perspective and authentication is based on keys.

OceanStore: OceanStore [5] is a prototype system to provide distributed access at persistent data in a uniform way. It is designed using a cooperative utility model in which consumers pay the service providers to ensure access to persistent storage. Using mainly untrusted servers, OceanStore caches data anywhere in the network, with encryption. This provides high availability and prevents attacks. Persistent objects are uniquely identified by a Global ID (GUID) and are located by either a non-deterministic but fast algorithm or a slower deterministic algorithm.

Farsite: Farsite [6] is a symbiotic, server-less, distributed file system.

It works between the cooperating clients, but not completely trusting the each other. Data stored in the servers is encrypted and replicated in a non-Byzantine way. Farsite first encrypts the contents of the files to prevent unauthorized reads. Digital signatures are used to prevent an unauthorized user to write a file. Replication provides reliability through long-term data persistence and immediate availability of requested file data.

3 CAN

The term CAN stands for *Content Addressable Network* [7], which is used like a distributed hash table. The basic operations performed on a CAN are the insertion, lookup and deletion of nodes. Our implementation provides a completely distributed system (we haven't centralized control, coordination or administration), scalable (every node has a small number of neighbors that is independent of the total numbers of nodes from system), and fault-tolerant (we support node failures).

3.1 CAN Construction

Our design is over a virtual *d-dimensional* Cartesian coordinate space. This space is completely logical and we haven't any relation between it and any physical system. In this *d-dimensional* coordinate space, two nodes are neighbors if their coordinates concur overlap along d-1 dimensions and differ over one dimension. In this way, if all zones from this space are approximatively the same, every node has $2d$ neighbors. To allow the CAN to grow, a new node that joins this space must receive its own portion (zone) of the coordinate space.

3.2 CAN Insertion

The process has three steps:

- The new node must find a node that is already in CAN (*source node*).

- After that, it knows CAN dimensions and it generates a *d-point* in this space (this point is in a node zone - *destination node*). We route from *source node* to *destination node* following the straight-line path through the Cartesian space. The *destination node* then splits its zone in half and assigns one half to the new node.

- In the last step, the neighbors of the split zone must be notified about new node from CAN.

Insertion - Main Procedure

The new computer is *wil* and the destination computer is $nameMin$.

void CanInsertion(WriteLocalImpl wil){
  if ($nameMin$ is available){
    Let $maxCoordinate$ be the first coordinate with max width;
    Create a new hyper-parallelepiped for the new node *writeImplLocal* with the following dimensions:
      * all dimensions except $maxCoordinate$ are those of the destination computer;
      * $maxCoordinate$ is half of the destination computer;
    $neighborhoodsListRestore()$;
  }
}

Interesting is the *neighborhoodsListRestore* procedure which inserts (at beginning of the list) all neighbors, after obvious positive tests.

3.3 CAN Deletion

When nodes leave the CAN, we need to ensure that their zones are taken by the remaining neighbors. If the zone of a neighbor can be merged with the node's zone to produce a valid single zone, then this is done. If not, then the zone is splitted in zones accordingly with neighborhoods structure. In some cases, it is possible that this zone to remain non-allocated, but at the first occasion it will be used.

```
void CanDeletion(WriteLocalImpl wil){
  deleted = false;
  for(int i = wil.getNeighborhoodsNumber(); i >= 1; i--){
      if(wil.getNeighborhoodIsAccesible(i) == true){
          nameServer = wil.getNeighborhoodName(i);
          wis = WriteLocalHelper.narrow(ncRefAux.resolve_
          str(numeServer));
          wis.delNeighborhoods(nameComp);
          neighborhoodsListRestore();
      }
  }
}
```

4 Graph Coloring

The Graph Coloring Problem (GCP) can be simply stated as follows: *given a graph $G(X, E)$, assign a color to each vertex in X such that no two adjacent vertices have the same color and such that the number of colors used is as least as possible.* This minimum possible number of colors is known as the chromatic number of the graph G and it is denoted by χ(G).

The GCP is well studied and has many applications including scheduling, timetabling and the solution of sparse linear systems. However it is also known to be one of the most difficult combinatorial optimization problems. Not only is the problem of finding χ(G) NP-hard, but Lund and Yannakakis [8] have even shown that, for some $\epsilon > 0$, approximate graph coloring within a factor of N^{ϵ} is also NP-hard.

We want to improve existing parallel solutions (e.g. [9]) with a new objective: speed in finding solutions. We consider a peer-to-peer cooperative approach to the graph coloring problem and look at its potential to aid the search for good solutions in reasonable time. Our algorithm divides the initial problem in sub-problems and involves all neighbors of main node in the solution search.

4.1 Algorithm Phases

In *first phase*, the input vertex set X of the graph $G = (X, E)$ is partitioned into p sub-sets $\{X_1, X_2, ..., X_p\}$, cardinality balanced. The subgraphs induced by each subset are sending then to p available neighbors of the main node, for solving.

In *second phase*, every node that receives one subgraph solves directly the problem (if the size of the problem is less than one specific value) or, recursively, splits the problem and sends the sub-problems to its available neighbors.

In *third phase*, the main node receives the partial solutions from its neighbors and tries to combine them into final solution. In this phase the main node finds "conflicts".

Last phase solves the "conflicts" and builds the final solution (the solution for the initial problem).

4.1.1 Phase I

For splitting, we used three methods:
1) Separator Theorem (Tarjan & Lipton, 1979) ([10]):
For every planar graph G with n vertices, there is a partition of $X(G)$ in disjoint sets A, B, S with the following properties:

- A and B are $S-$separated: in $G - S$ there is no edge linking a vertex from A to a vertex from B.
- $|A| \leq \frac{n}{2}, |B| \leq \frac{n}{2}$
- $|S| \leq 4\sqrt{n}$

We start from node s and we perform a breadth first search (BFS) on this graph. We mark every node with its level from BFS tree. $L(t), 0 \leq t \leq l + 1$, denotes the set with nodes from level t, and t_1 is the middle level (the level containing the node $\frac{n}{2}$).

Let t_0 be the higher node with $|L(t_0)| \leq \sqrt{n}$ and $t_0 \leq t_1$. Let t_2 be the lower node with $|L(t_2)| \leq \sqrt{n}$ and $t_1 \leq t_2$. Consider $C = \cup_{t<t_0} L(t), D = \cup_{t_0<t<t_2} L(t), E = \cup_{t_2<t} L(t)$ (like in the following figure).

Figure 1. *Node Separation with Tarjan & Lipton*

If $|D| \leq \frac{2}{3}n$ we can take $S = L(t_0) \cup L(t_2)$, A – the set of maximum elements from C, D, E and B – the union of the other two. If $|D| > \frac{2}{3}n$, then we must find a new separator for D.

With this method, we can split our initial problem only into three problems, and we lose a lot of time with splitting. The main advantage for this method is actually in the following phases, because we don't have many edges between our sets of nodes.

2) Separation based on neighbor's number

In this case the input vertex set X of the graph $G(X, E)$ is split into p sub-sets as $\{X_1, X_2, ..., X_p\}$ with the same number of nodes ($X_1 = \{1, .., \frac{n}{p}\}, X_2 = \{\frac{n}{p}+1, ..2\frac{n}{p}\}, ..$). p is the number of neighbors of current node, which can be involved in problem resolution.

The separation is made very fast, but we have a lot of work to do in the following phases, since, usually, we have many edges between our sets of nodes.

3) Random separation

In this case the input vertex set X of the graph $G(X, E)$ is split into

p sub-sets as $\{X_1, X_2, ..., X_p\}$, where p is the number of neighbors of current node, which can be involved in problem resolution. Every set has random nodes, but their cardinalities are balanced.

The separation is done very fast, but in the following phases we have omnibus-volume of work that depends of the current step.

For each of the above methods we have situations in which this method is better and obtains the minimal coloring.

Phase II

For coloring we use **Greedy Coloring Algorithm** ([11]): visit the list $X(G)$ from the left to the right and for each current node assign a color with the minimum value unused by its neighbors.

Phase III

Two nodes are in "conflicts" if they are in different sub-sets and receive the same color, and are adjacent in the initial graph.

Phase IV

"Conflicts" elimination can be done in two ways:
-**On-line**, when a conflict is detected, it is solved immediately;
-**Off-line**, only after all conflicts are detected, they are solved one by one.

Each method has advantages and disadvantages and depends on the input graph. For the *on-line* method we have the advantage that it destroys future conflicts. The disadvantage appears because number of graph colors is increased and the solution decreases in quality. The same kind of problems could appear for the *off-line* method.

4.2 Algorithm Code

We used a program that generates random graphs of large order and provides the output in XML format. The main procedure for coloring is the following:

public void graphColoring(graph G){
    if(G is too big){
        $graphSeparation(nNeighbors)$;
        for($int\ j = 1; j <= nNeighbors; j++$){

```
                graphColoring(G_j);
            }
        }
        else{
            graphColoring();
        }
        solutionsCombination();
        conflictsElimination();
}
```

This code is for the "off-line" case, in the "on-line" case procedure *conflictsElimination* appears inside of procedure *solutionsCombination*.

4.3 Tests and Results

In order to evaluate our algorithms we used a number of classical graphs, from the DIMCS Challenge([12]). We can see that the first method of separation problems (Separation Theorem (Tarjan & Lipton, 1979)) is the better method in comparison with second method of separation (Table 1), but sometimes the random method is able to give us the lower number of colors.

However, for every method of separation we have a graph in which it provides the better solution. We have the same behavior for "on-line" and "off-line" methods.

We work in a 2-dimensional CAN and we remark that the number of final colors depends on the peer-to-peer network structure in linear way and also, depends on the number of neighbors involved in resolution. In Table 1 we put with bold the results for random case which appear often.

Last two columns from Table 1 contain the execution time results obtained: first – "*Local*" – doesn't use our network advantages and solves the problem locally, and the second – "*P2P*" – uses a 2-dimensional network for solving.

Our algorithm solutions are the optimal solutions for small graphs, but aren't in almost all cases for big graphs. First off all, because **Greedy Coloring Algorithm** doesn't offer the optimal solution, and

Table 1. Test Results

ID	Nodes	Edges	χ	Theorem	Order	Random	Local	P2P
Myciel4	23	71	5	5	6	**5**, 6	0s	0s
Myciel5	47	236	6	6	7	**6**, 7	0s	0s
Myciel7	95	755	7	7	8	7, **8**	0s	0s
Anna	138	493	11	12	12	11, **12**	1s	2s
David	87	406	11	12	12	11, **12, 13**	0s	0s
Huck	74	301	11	11	11	11	0s	0s
Jean	80	254	10	11	11	**10**, 11	0s	0s
Games120	120	638	9	10	9	**9**, 10	1s	1s
Queen5-5	25	160	5	7	8	6, 7, **8**, 9	0s	1s
Queen6-6	36	290	7	9	13	**9**, 10, 11	0s	1s
Queen7-7	49	476	7	12	13	10, **11**, 12	0s	1s
Queen8-12	96	1368	12	15	17	**15**, 16, 17	0s	1s
Queen8-8	64	728	9	13	16	**12, 13**, 14	0s	1s
Queen9-9	81	2112	10	14	16	13, **14**, 15	0s	1s
Queen10-10	128	3216	12	18	17	14, **15**, 16	1s	2s
Queen16-16	256	12640	16	25	32	25	30s	27s
Le450-5a	450	5714	5	13	15	13, **14**	21s	14s
Le450-5b	450	5734	5	13	15	14	20s	13s
Le450-5d	450	9757	5	18	18	18	44s	26s
Le450-15c	450	16680	15	30	33	32	110s	79s
Le450-15d	450	16750	15	30	32	32	129s	85s
Le450-25c	450	17343	25	36	37	39	145s	90s
Le450-25d	450	17425	25	35	36	38	127s	78s

secondly, when we combine solutions and we have conflicts, the procedure $conflictsElimination$ can increase the number of colors.

According with the results, our approach is not relevant for small graphs, where the times are similar, but is very useful for large graphs with a big number of nodes and edges, where results are very promising.

5 Summary and Future Work

First phase (initial problem splitting) and last phase (conflict elimination) can be improved, but these depend on graph generation. Our algorithm used for graph generation obtains the same structure of graphs and we have not a large variation of graph types.

Also, it would be interesting to see what happens in fail-over sit-

uations: in this case the sub-problem must be sent again to another neighbor for solving. In the future, this kind of approach can help us in graph coloring with a better speed and with a solution close to the optimal.

With characteristics like decentralization, symmetry, robustness, availability and persistence of data, the P2P distributed file system is now an important part of file system research and can be involved with success in future in combinatorial solving of problems.

References

[1] A.T. Stephanos, S. Diomidis. *A survey of peer-to-peer content distribution technologies*, ACM Computing Surveys, Vol. 36, No. 4, December 2004.

[2] H. Ragib, A. Zahid. *A survey of peer-to-peer storage techniques for distributed file system*, National Center for Supercomputing Applications Department of Computer Science, April 2005.

[3] I. Clarke. *Freenet: a distributed anonymous information storage and retrieval system*, Workshop on Design Issues in Anonymity and Unobservability, pp. 46–66., 2000.

[4] F. Dabek. *Wide-area cooperative storage with CFS*, Usenix SOSP, 2001.

[5] J. Kubiatowicz. *OceanStore: an architecture for global- scale persistent storage*, ACM ASPLOS, 2000.

[6] A. Adya. *FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment*, Usenix OSDI, 2002.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A scalable content-addressable network*, In Proc. ACM SIGCOMM 2001, August 2001.

[8] C. Lund, M. Yannakakis. *On the hardness of approximating minimization problems*, Journal of the ACM, 41(5):960–981, 1994

[9] A. H. Gebremedhin, F. Manne. *Scalable parallel graph coloring algorithms*, University of Bergen, Norway, Concurrency: Pract. Exper, 2000.

[10] R.J. Lipton, R. E. Tarjan. *A separator theorem for planar graphs*, SIAM Journal on Applied Mathematics 36, 2, 177–189, 1979.

[11] J. Culberson. *Iterated greedy graph coloring and the difficulty landscape*, Tech. Rep. 92-07, Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada, 1992.

[12] Graphs used in the DIMACS Challenge (Discrete Mathematics & Theoretical Computer Science). DIMACS Center CoRE Building Rutgers, The State University of New Jersey 96 Frelinghuysen Road Piscataway, NJ 08854-8018. Graphs are available at address: http://mat.gsia.cmu.edu/COLORING02/index.html

Adrian Iftene, Cornelius Croitoru Received October 10, 2006

"Al. I. Cuza" University, Faculty of Computer Science,
General Berthelot, 16, 700483, Iasi
E-mail: *adiftene@infoiasi.ro*

Computer Science Journal of Moldova, vol.**14**, no.3(42), 2006

An Approach to Shorten Digital Signature Length

Nikolay A. Moldovyan

Abstract

A new method is proposed to design short signature schemes based on difficulty of factorizing a composite number $n = qr$, where q and r are two large primes. Using the method new digital signature schemes (DSS) with the 320-bit and 240-bit signature size are developed. The characteristic feature of the 240-bit signature DSS is the use of "three-level" verification equation, for example, $R = \beta^{\alpha^{kgH} \bmod n} \bmod p$, where $k = \left(R^{\alpha^{g} \bmod n} \bmod p\right) \bmod \delta$. The (k, g) signature corresponds to the H hash value and represents a pair of natural numbers having the size of 80 and 160 bits, respectively. The δ modulus is a prime number. The public key is the triple (α, β, p), where $p = 2n + 1$ is prime, β is the q order element modulo p, α is the γ order element modulo q. The private key is represented by the pair of two prime numbers (q, γ).

1 Introduction

One of important practical problems is developing digital signature schemes (DSS) with short signature length [2, 6]. In general the signature length depends on the required security level of the DSS, that can be estimated as number of group operations required to forge a signature. In this paper the signature length is compared for different DSS in the case of minimum security level that can be estimated at present as 2^{80} operations. The RSA and Rubin's DSS based on factorization problem use the 1024-bit signature length [3]. The DSA standard and Schnorr's DSS based on difficulty of finding discrete logarithm modulo

large prime number provide comparatively short signatures having the 320-bit length and providing the same security level [8]. The ECDSA standard also requires the use of the 320-bit signature size [1]. Recently a DSS with 171-bit signature length has been developed using Weil paring on elliptic curves [2].

In present paper we consider some ways to reduce the signature length in DSS based on difficulty of factorization of a composite number n calculated as production of two large primes q and r. In this investigation we propose some new designs of the DSS based on difficulty of factorization problem, which allows one to reduce the signature length up to 240 bits. In Section 2 we propose some short signature designs based on the signature formation mechanism described in [4]. The DSS with the 320-bit and 240-bit signature length and composite private modulus $\gamma = \gamma'\gamma''$, where γ' and γ'' are primes, are presented. Section 3 introduces DSS with 240-bit signature length and prime private modulus γ. Section 4 concludes the paper. We use notation $|x|$ to denote the length of the binary representation of the x value.

2 Short signatures from factorization problem

The well known cryptosystem RSA [7] is based on calculation modulo n that is a product of two randomly chosen prime numbers r and q: $n = rq$. The public key is represented by a pair of numbers (e, n), and the private key is a number d, which is calculated using the following formula: $ed \bmod \varphi(n) = 1$, where $\varphi(n) = (r-1)(q-1)$ is Euler phi function of n. Security of this system is based on difficulty of calculating d while φ(n) is an unknown value.The φ(n) value can be easily calculated after factorization of the modulus n, therefore divisors of n have to be kept in secret (or to be annihilated after the e and d keys have been generated). The signature corresponding to a plaintext M is a value S, which satisfies the following verification equation: $S^e \bmod n = H$, where H is the hash function value corresponding to the message to be signed. The signature generation equation is the following one: $S = H^d \bmod n$. In RSA the signature length is approximately equal to the n modulus length. At present in different practical appli-

cations the used n modulus has the length $|n| = 1024$ bits or more (this value corresponds to the mentioned above minimum security level).

To reduce the signature length in the case of DSS based on the factorization problem we use the novel signature formation mechanism [4] that can be applied while developing DSS with two-element signature denoted as (k, g). The mechanism is characterized in using a two-element public key with the structure (n, α), where α is the γ order element modulo n and in solving a system of two congruences (or one equation and one congruence) while generating signature. The private key is γ.

Some internal relation between the α and n values provides potentially some additional possibilities to factorize modulus n. This defines special requirements to the α element of the public key [4]. One should use composite γ, i. e. $\gamma = \gamma'\gamma''$, where $\gamma'|r-1$, $\gamma''|q-1$, $\gamma'' \not| r-1$ and $\gamma' \not| q-1$. To choose the size of the γ value we should take into account that the α value can be used to factorize the n modulus calculating $\gcd(\alpha^i \bmod n - 1, n)$ for $i = 1, 2, \ldots, \min\{\gamma', \gamma''\}$. Therefore we should use the 80-bit values γ' and γ''. Thus, for γ we get the following required length: $|\gamma| = 160$ bits.

A secure variant of the DSS with the 320-bit signature length is described by the following verification equation:

$$k - g = \left(\alpha^{kgH} \bmod n\right) \bmod \delta, \tag{1}$$

where $\delta \neq \gamma$ is a specified prime number and H is the hash value of the signed message. The signature generation is performed as follows: 1) generate a random number U and calculate $Z = \left(\alpha^U \bmod n\right) \bmod \delta$; 2) solve simultaneously equation $k - g = Z$ and congruence $kgh \equiv U \bmod \gamma$.

The solution gives signature elements k and g:

$$g = -\frac{Z}{2} \pm \sqrt{\frac{Z^2}{4} + \frac{U}{H}} \bmod \gamma \quad \text{and} \quad k = Z + g. \tag{2}$$

The signature elements have the size $|k| \approx |g| \approx |\gamma| \approx 160$ bits. To reduce the signature size we propose the DSS based on the following

pair of verification equations:

$$R = \alpha^{kgH} \bmod n, \quad \text{and} \quad k = (R\alpha^{g} \bmod n) \bmod \delta, \tag{3}$$

where $|\delta| = 80$ bits. The signature is generated as follows:

i) select at random U $(1 < U < \gamma)$;

ii) calculate the first signature element $k = \left(\alpha^{U} \bmod n\right) \bmod \delta$;

iii) calculate the second signature element

$$g = \frac{U}{kH + 1} \bmod \gamma. \tag{4}$$

Thus, we have a 240-bit signature: $|k| + |g| \approx |\delta| + |\gamma| = 80 + 160 = 240$ bits. The way used in the last DSS to reduce the k element length resembles the method used earlier to reduce the signature length to the 240-bit value in the DSA standard [5]. The need to calculate $(kH+1)^{-1}$ modulo γ defines the interest in constructing DSS with prime value γ.

In the next section we consider the DSS design that provides possibility to use prime private key element γ.

3 A 240-bit signature DSS with prime γ

The design idea of the DSS with prime γ consists in hiding the modulus to which the γ order belongs. If it is so, then one can use a prime modulus q, for which the α element is a generator of the γ order group: $\alpha^{\gamma} \equiv 1 \bmod q$. Implementation of this idea is connected with the following contradiction. The signature should be calculated modulo q that is secret. The verification equation should allow one to check the result of such calculation without direct use of the secret modulus q. In the DSS described below the contradiction is solved using the following "three-level" verification equations:

$$R = \beta^{\alpha^{kgH} \bmod n} \bmod p \quad \text{and} \quad k = \left(R^{\alpha^{g} \bmod n} \bmod p\right) \bmod \delta \tag{5}$$

where $p = 2n + 1$ is prime, n is product of two 512-bit primes q and r $(n = rq)$, β is the q order element modulo p, and α is a γ order element

modulo q. The prime modulus δ is a 80-bit value. The private key is represented by the (q, γ) pair, where γ is an 160-bit prime number $(\gamma | q-1)$. The public key is triple (α, β, p). The (k, g) signature corresponds to the H value and represents two natural numbers having the length $|k| = 80$ bits and $|g| = 160$ bits.

The signature generation procedure includes solving the system of two congruences: $t + g \equiv U \bmod \delta$ (i) and $t \equiv kgH \bmod \delta$ (ii), where $U < \gamma$ is selected at random, the signature element k is calculated using formula $k = \left(\beta^{\alpha^U \bmod q} \bmod p\right) \bmod \delta$, t is an auxiliary unknown. The solution of this system gives the formula (4) for calculation of the signature element g.

The proof that signature verification works is as follows. Suppose we have a valid signature (k, g) corresponding to the H hash value. Taking into account that α is the γ order element modulo q and substituting the k and g values in the verification equation we get:

$$R = \beta^{\alpha^{kgH} \bmod n} \bmod p = \beta^{\alpha^{kgH} \bmod q} \bmod p = \beta^{\alpha^{\frac{kHU}{kH+1}} \bmod q} \bmod p$$

and

$$k' = \left(R^{\alpha^g \bmod n} \bmod p\right) \bmod \delta = \left(R^{\alpha^g \bmod q} \bmod p\right) \bmod \delta =$$

$$= \left(\left(\beta^{\alpha^{\frac{kHU}{kH+1}} \bmod q} \bmod p\right)^{\alpha^g \bmod q} \bmod p\right) \bmod \delta =$$

$$= \left(\beta^{\alpha^{\frac{kHU}{kH+1} + \frac{U}{kH+1}} \bmod q} \bmod p\right) \bmod \delta = \left(\beta^{\alpha^U \bmod q} \bmod p\right) \bmod \delta.$$

Since $k' = k$ the signature verification result is positive, i.e. the verification equations work correctly.

A possible attack on this scheme is to find value $X = \log_\beta(R) \pmod p$ and calculate q as a divider of the value $\left(\alpha^{kgH} \bmod n\right) - X$. Then the secret γ can be determined dividing $q - 1$. Due to the large value of q this attack is computationally infeasible. Another attack is to find the value $X' = \log_\alpha \left(\alpha^{kgH} \bmod n\right) \pmod n$ and then to calculate γ as one of dividers of the value $kgH - X'$. The last attack defines the

following requirement to the α value: *the ϵ order of α modulo n should be large ($|\epsilon| \geq 160$ bits)*. For example, we can easily generate the α value the order of which modulo n is $\epsilon = \gamma(r-1)$. If this requirement is satisfied, then the second attack is also computationally infeasible.

4 Conclusion

Using a novel mechanism of the signature generation we have proposed new short signature schemes providing the minimum security level (2^{80} group operations) with signatures having the size of 320 bits and 240 bits. We have presented two 240-bit signature DSS, one with composite private key element γ and the other one with prime γ. Using prime γ we reduce the $\Pr(\gcd(\gamma, kH+1) \neq 1)$ probability from $\gamma'^{-1} + \gamma''^{-1}$ to γ^{-1} (if $\gcd(\gamma, kH+1) \neq 1$, then the signature generation procedure should be repeated because of the necessity to calculate $(kH+1)^{-1}$). We have attained possibility to use securely the prime private key element γ due to the use of the "three-level" verification equations. Other variants of the DSS based on the "three-level" verification equations analogous to the DSS described in section 3 are also possible. Above we have considered the signature size satisfying the minimum security level. In general case for the 2^z operations security level, where $z \geq 80$, one should use $2z$-bit private key element γ and z-bit modulus δ that defines the signature length $|k| + |g| = 3z$. At present the proposed 240-bit signature schemes provide the shortest signature length among known DSS based on factorization problem. The computational efficiency of the signature generation and verification procedures in the proposed DSS is about the same as in the DSA standard.

References

[1] ANSI X9.62 and FIPS 186-2. *Elliptic curve signature algorithm*, 1998.

[2] D.Boneh, B.Lynn, and H.Shacham. *Short signatures from the Weil pairing.* J. Cryptology. 2004, vol. 17, no 4, pp. 297–319.

[3] A.J.Menezes, P.C. Van Oorschot, and S.A.Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, FL, 1997.

[4] A.A.Moldovyan, D.N.Moldovyan, L.V.Gortinskaya. *Cryptoschemes Based on New Signature Formation Mechanism.* Computer Science Journal of Moldova. 2006 (in print).

[5] D.Naccache and J.Stern. *Singing on a postcard.* In Y.Frankel, editor, Proceedings of Financial Cryptography 200, volume 1962 of LNCS, pp. 121–135, Springer-Verlag, Berlin, 2000.

[6] L.Pintsov and S.Vanstone. *Postal revenue collection in the digital age.* In Y.Frankel, editor, Proceedings of Financial Cryptography 200, volume 1962 of LNCS, pp. 105–120, Springer-Verlag, Berlin, 2000.

[7] R.L.Rivest, A.Shamir, and L.M.Adleman. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, 1978, vol. 21, no 2, pp. 120–126.

[8] N.Smart. *Cryptography: an introduction.* McGraw-Hill Publication, London, 2003.

Nikolay A. Moldovyan

Received June 19, 2006

Specialized Center of Program Systems "SPECTR",
Kantemirovskaya, 10, St.Petersburg 197342, Russia;
Phone/fax: 7-812-2453743,
E–mail: *nmold@cobra.ru*

Computer Science Journal of Moldova, vol.**14**, no.3(42), 2006

Cryptoschemes Based on New Signature Formation Mechanism

A.A.Moldovyan, D.N.Moldovyan, L.V.Gortinskaya

Abstract

Several variants of new digital signature schemes (DSS) based on the discrete logarithm and factorization problems have been proposed. Considered DSS are characterized in that a novel mechanism of the signature generation is used, in which two parameters of the (k, S) or (R, S) signature are defined after solving a system of two congruences. In the case of composite modulus additional restrictions conditions have been introduced for selection of the public key.

1 Introduction

Public key cryptosystems based on hard mathematical problems are well approved for information authorization, i.e. for generating digital signatures. For such application the most important of such problems are the following: 1) factorization of a large integer number and 2) finding discrete logarithm modulo large prime number. The first problem underlies the RSA cryptosystem and the second problem is the base of ElGamal's DSS, US standard FIPS 186, and Russian GOST P.34.10-94. The large modulo exponentiation procedure is used in these schemes. The RSA cryptosystem uses composite modulus n that is a product of two randomly chosen prime numbers r and q: $n = rq$. In the cryptosystems of the second type the prime modulus p is considered.

In RSA the public key is represented by pair of numbers (e, n), and the private key is a number d, which is calculated using the following formula: $ed \bmod n = 1$, where $\varphi(n) = (r-1)(q-1)$ is Euler phi function

of n. Security of this system is based on difficulty of calculating d while φ(n) is an unknown value. The φ(n) value can be easily calculated after factorization of the modulus n, therefore divisors of n have to be kept in secret (or to be annihilated after the e and d keys have been generated). The signature corresponding to a plaintext M is a value S, which satisfies the following verification equation: $S^e \bmod n = H(M)$, where $H(M) = H$ is the hash function value corresponding to M. The signature generation equation is the following one: $S = H^d \bmod n$. In RSA the signature length is approximately equal to the n modulus length (denoted as $|n|$). At present in different practical applications the used n modulus has the length 1024 bits or more (for example, 1024-2048 bits). In ElGamal's DSS the public key y is calculated exactly as in Diffie-Hellman's public-key distribution system: $y = \alpha^x \bmod p$, where x is private key (an integer number of sufficiently large size) and α is a primitive element modulo p. The signature is represented by a pair of numbers (R, S) that are calculated depending on the both private key and message to be signed. The signature generation procedure is described as follows:

1) generate a random number k that is coprime with $p - 1$;

2) calculate $R = \alpha^k \bmod p$;

3) calculate the S value satisfying the following equation:

$$H = (xr + kS) \bmod (p - 1)\,,$$

where H is the hash function value calculated from the plaintext.

The (R, S) signature is considered as valid one if the following signature verification equation is satisfied: $\alpha^H = y^R R^S \bmod p$. To ensure required security the modulus p size has to be equal to $|p| \geq 1024$ bits. Thus, the (R, S) signature size is 1024 bits or more.

In this paper several new DSS are considered, which are based on difficulty of the discrete logarithm and factorization problems. Additional restriction requirements to selection of the composite modulus are formulated.

2 New Signature Generation Mechanism

The ElGamal-like DSS allow one to calculate the (R, S) and H values, which satisfy the verification equation, without using the private key. In practice this theoretic attack is eliminated due to using secure hash functions, since the attack produces random values H. Indeed, the mentioned attack is based on the preliminary generating the R parameter with special structure that permits calculation of the "correct" S and H values that play a part of adjusting parameters. Representing the R parameter as $R = \alpha^u y^t \bmod p$ and using two fitting parameters attacker can avoid necessity to solve discrete logarithm problem, i. e. he can easily find "satisfactory" values S and H, but the last is a random one. While using strong hash functions, the attacker can indicate no plaintext corresponding to the H hash value calculated while attacking. Because of this fact the described attack has theoretical character and no practical significance in majority of applications. However, the possibility of mentioned attack is an unwanted property of DSS.

To eliminate such attacks we propose to use a new mechanism of the (R, S) signature generation in the DSS based on difficulty of solving DLP. The idea is to calculate simultaneously both parameters R and S each of which play a part of a variable defining the exponent value and a part of a multiplier in the signature verification equation. In new schemes the R and S parameters form two different functions $F_1(R, S)$ and $F_2(R, S)$. The signature parameters are changed in such a way that the $F_1(R, S)$ function changes, while the $F_2(R, S)$ function keeps its value constant. Due to variability of the F_1 function and invariability of the F_2 function we have potential possibility to calculate the values R and S satisfying the verification equation. Assuming that we consider only such values of the R and S signature parameters, which save the value $F_2 = Z = \text{const}$, it is potentially possible to simplify the verification equation and calculate some signature value $(R(Z), S(Z))$ depending on Z. While constructing some concrete verification equation the (F_1, F_2) pair of functions can be used, for example, in one of the following forms:

$$(R/S \bmod p, RS \bmod p),$$

$$(RS \bmod p, R/S \bmod p),$$

$$(RS^H \bmod p, RS \bmod p),$$

$$(RS \bmod p, RS^2 \bmod p),$$

$$(RS^Z \bmod p, RS \bmod p),$$

$$(RS^Z \bmod p, R/S \bmod p),$$

where $Z = RS \bmod p$, the R and S values are supposed to have the following structure: $R = \alpha^k \bmod p$ and $S = \alpha^g \bmod p$. The values k and g are to be calculated simultaneously as solution of the system of two congruences, one of the lasts defining invariability of the F_2 function and having the form: $k + g \equiv \text{const} \bmod (p-1)$. Taking into account that signature verification expression should contain multipliers depending on the hash value and public key we have the following examples of DSS based on the described signature formation mechanism:

$$R/S = y^{(RS \bmod p)^H} \alpha^{(RS \bmod p)} \bmod p,$$

$$R = Sy^{H(RS \bmod p)} \alpha^{(RS \bmod p) \bmod \delta} \bmod p,$$

$$(R/S)^{(RS \bmod p)} = y^{(RS \bmod p) \bmod \delta} \alpha^H \bmod p,$$

where δ is an arbitrary prime number such that $|\delta| \approx 0.25\,|p|$ (the mod δ operation defines a compression function depending on F_2).

Considering these concrete variants of the verification equation it is possible to note that an attacker can re-designate the signature parameters R and S and reduce the verification equation to some equivalent expression in which parameters R and S can be calculated consecutively without using the private key. For example, let us consider the DSS defined by the verification equation

$$R = Sy^{H(RS \bmod p)} \alpha^{(RS \bmod p) \bmod \delta} \bmod p.$$

Denoting $Z = RS \bmod p$ we have $Z/S = Sy^{HZ} \alpha^{Z \bmod \delta} \bmod p$. Now the signature can be forged as follows:

i) select arbitrary value Z,

ii) calculate $S^2 = Z\left(y^{HZ}\alpha^{Z \bmod \delta}\right)^{-1} \bmod p$,

iii) if the value $W = Z\left(y^{HZ}\alpha^{Z \bmod \delta}\right)^{-1} \bmod p$ is not a quadratic residue, then go to step 1, otherwise calculate $S = W^{1/2} \bmod p$ and $R = Z/S \bmod p$.

It is easy to demonstrate that the calculated signature satisfies the initial verification equation. The last attack shows that we should make one step back in our DSS design, namely we should indicate the value R in the verification equation as $\alpha^k \bmod p$ and define the signature as (k, S). In this case the verification equation can be simplified, since there is no necessity to indicate the α parameter two times (as α^k and α^H) and to use the compression function as second type function depending on variables k and S and keeping its value constant. Let us consider the following verification equation that takes into account the mentioned remarks:

$$\alpha^{k+H} \equiv S^2 y^{(\alpha^k S \bmod p)} \bmod p,$$

where $\gcd(3, p-1) = 1$. Using private key it is possible to generate the (k, S) signature as follows:

i) choose random number $U < p-1$ and calculate $Z = \alpha^U \bmod p$;

ii) solve the following system of two congruences:

$$U \equiv k + g \bmod (p-1),$$

$$k \equiv gH + x + Z \bmod (p-1),$$

where k and g are unknown parameters:

$$k = \frac{2U - H + xZ}{3} \bmod p - 1,$$

$$g = \frac{U + H - xZ}{3} \bmod p - 1$$

iii) calculate $S = \alpha^g \bmod p$.

To avoid additional restriction imposed on the p modulus one can use a generator of the γ-order group $\{\varepsilon, \varepsilon^2 \bmod p, ..., \varepsilon^{\gamma-1} \bmod p, \varepsilon^{\gamma}$

$\mathrm{mod}p\}$, where γ is a prime divisor of p and $|\gamma| = 160$-256 bit, as the α parameter in the verification equation. In this case the mentioned above system of congruences gets the form:

$$U \equiv k + g \bmod \gamma,$$

$$k \equiv gH + x + Z \bmod \gamma.$$

Therefore the k value size is $|k| = 160$-256 bit and we get reduction of the signature length. Similarly the other variants of DSS can be constructed. For example, the following verification equations define several DSS:

$$\varepsilon^{k+H} \equiv Sy^{(\varepsilon^k S \bmod p)} \bmod p,$$

$$\varepsilon^{k} \equiv S^{H}y^{(\varepsilon^k S \bmod p)} \bmod p,$$

$$\varepsilon^{Hk}S^{-1} \equiv y^{(\varepsilon^k S \bmod p)} \bmod p,$$

$$\varepsilon^{-k}S \equiv y^{H(\varepsilon^k S \bmod p)} \bmod p.$$

Concrete system of congruences, which should be solved to calculate the k and g parameters, can be easily derived from the respective verification equation.

If the prime modulus p in the described DSS design approach is replaced by RSA modulus n, factorization of which is kept secret, then security of DSS will be defined by intractability of both the DLP modulo n and the factorization modulus n. Indeed, to calculate γ it is necessary to find the $\varphi(n)$ value. The public key of such schemes is (n, y, α) or (n, y, ε). Unfortunately the DLP modulo n can be reduced to problem of finding discrete logarithms modulo r and modulo q. Therefore this replacement will be actual for practice, if some new algorithms of finding discrete logarithms that avoid the factorization problem are developed. Some DSS, based only on the factorization problem, represent current interest since they provide possibility to reduce the difficulty of signature verification procedure. In the next section we consider some of such DSS variants.

3 Schemes based on difficulty of the modulus factorization

While using the RSA modulus the verification equation can be simplified due to avoiding the y or/and ε parameters. The public key in this case is (n, ε) or n, respectively. The DSS security in this case depends on difficulty to factorize the RSA modulus. Several possible variants are present in Table 1.

Table 1. Some variants of new DSS schemes

Verification equation	System of congruences	Formulas for calculating the k and g values	Public key
$\alpha^k \equiv S^{\frac{H\alpha^k}{S} \bmod n} \bmod n$	$k - g \equiv U \bmod \gamma$ $gZ \equiv U \bmod \gamma$ $\left(Z = \left(H\alpha^U \bmod n\right) \bmod \gamma\right)$	$k = \frac{ZU}{1-Z} \bmod \gamma$ $g = \frac{U}{1-Z} \bmod \gamma$	n
$\alpha^{k-H} S^{\alpha^k S \bmod n} \equiv 1 \bmod n$	$k + g \equiv U \bmod \gamma$ $k + gZ \equiv H \bmod \gamma$ $\left(Z = \alpha^U \bmod n\right)$	$k = \frac{ZU-H}{Z-1} \bmod \gamma$ $g = \frac{H-U}{Z-1} \bmod \gamma$	(n, α)
$R = S^H \alpha^{(RS \bmod n)} \bmod n$	$k + g \equiv U \bmod \gamma$ $k - Hg \equiv Z \bmod \gamma$ $\left(Z = \alpha^U \bmod n\right)$	$k = \frac{HU+Z}{H+1} \bmod \gamma$ $g = \frac{U-Z}{H+1} \bmod \gamma$	(n, α)
$R = S\alpha^{H(RS \bmod n)} \bmod n$	$k + g \equiv U \bmod \gamma$ $k - g \equiv HZ \bmod \gamma$ $\left(Z = \alpha^U \bmod n\right)$	$k = \frac{U+HZ}{2} \bmod \gamma$ $g = \frac{U-HZ}{2} \bmod \gamma$	(n, α)

In the last two DSS variants with composite modulus we indicate directly the R value in the verification equation. The attack based on using arbitrary value Z and calculating the respective value S is prevented because the S value is to be calculated as $S = \left(\frac{Z}{\alpha^{HZ}}\right)^{1/2} \bmod n$ or as $S = \left(\frac{Z}{\alpha^Z}\right)^{\frac{1}{H+1}} \bmod n$. In both cases it is difficult to calculate S without factorizing the modulus.

4 Additional restriction requirements for choosing the RSA modulus

Let us consider the scheme from Table 1, where (n, α) is a public key, where α is a generator of the cyclic γ-order group, i. e. we have $\alpha^{\gamma} \equiv 1 \bmod n$. Some internal relation between the values α^{γ} and n provides potentially some additional possibilities to factorize modulus n. Let us assume that γ is a divisor of $r-1$ and α does not divide $q-1$. In practice to generate the α generator the following expression is used: $\alpha = \beta^{\varphi(n)/\gamma} \bmod n \neq 1$, where β is a random number for which we have $\gcd(\beta, n) = 1$. (Using the Euler's theorem one can demonstrate that $\alpha^{\gamma} = \beta^{\varphi(n)} \equiv 1 \bmod n$, where $\varphi(n) = (r-1)(q-1)$, i. e. if $\beta^{\varphi(n)/\gamma} \bmod n \neq 1$, then the $\beta^{\varphi(n)/\gamma} \bmod n$ value is a generator of the γ-order group). We have:

$$\alpha = \beta^{\varphi(n)/\gamma} \bmod n \equiv \left(\beta^{(q-1)}\right)^{(r-1)/\gamma} \bmod n \Rightarrow$$
$$\alpha \equiv \left(\beta^{(q-1)}\right)^{(r-1)/\gamma} \bmod q \Rightarrow \alpha \equiv 1^{(r-1)/\gamma} \bmod q \Rightarrow$$
$$\alpha \equiv 1 \bmod q \Rightarrow \alpha - 1 \equiv 0 \bmod q \Rightarrow q | \alpha - 1 \Rightarrow \gcd(\alpha - 1, n) = q$$

Thus, in the considered case it is possible to factorize modulus using extended Euclidean algorithm. Therefore some restrictions should be imposed on generation of the public key. A way preventing the described factorization method is to use such numbers r and q that both of them contain the same required large divisor γ and γ^2 does not divide neither $r-1$ no $q-1$. If this additional requirement is imposed, then the α parameter can be generated as follows: $\alpha = \beta^{L(n)/\gamma} \bmod n \neq 1$, where $L(n) = \operatorname{lcm}[r-1, q-1]$ is the generalized Euler's function. Thus, $\alpha = \beta^{uv} \bmod n \neq 1$, where $u = (r-1)/\gamma$ and $v = (q-1)/\gamma$. If we use, while generating the α value, a value that is simultaneously primitive element modulo r and primitive element modulo q as the β value (i.e. β is "double" primitive element), then we will have simultaneously $\alpha \not\equiv 1 \bmod q$ and $\alpha \not\equiv 1 \bmod r$. While using a "double" primitive element we deterministically generate a "strong" α value. But it is not strictly necessary to use "double" primitive elements. We can generate a "strong" α value selecting random β values. In this case we should

check if $\alpha \not\equiv 1 \bmod q$ and $\alpha \not\equiv 1 \bmod r$ hold. Probability that the current value is not "strong" is sufficiently low (see the next section).

The second way to generate "strong" public key is to use composite value γ, i. e. $\gamma = \gamma'\gamma''$, where $\gamma'|r-1$ and $\gamma''|q-1$ and γ' and γ'' do not divide $q-1$ and $r-1$, correspondingly. For generating the α parameter we have the following formula: $\alpha = \beta^{L(n)/\gamma} \bmod n \neq 1$, i. e $\alpha = \beta^{uv} \bmod n \neq 1$, where $u = (r-1)/\gamma'$ and $v = (r-1)/\gamma''$. Analogously to the first case, while using the β value that is "double" primitive element, we get $\alpha \not\equiv 1 \bmod q$ and $\alpha \not\equiv 1 \bmod r$.

Thus, we have two different ways to define difficulty of the n modulus factorization in the considered DSS. Unfortunately in the first way we have a problem to avoid possibility to calculate the secret parameter γ without factorizing the n modulus. Indeed, we have:

$$n-1 = (u\gamma+1)(v\gamma+1) - 1 = uv\gamma^2 + u\gamma + v\gamma = (uv\gamma + u + v)\gamma,$$

Usually the $n-1$ value can be easily factorized; therefore the secret γ can be recovered, if no new restriction requirements are imposed on selection of the n modulus. In the second way factorization of the $n-1$ value does not allow one to determine the γ secret. Thus, we have shown that the second way is preferable in practice.

The considered above restrictions are also actual for the DSS schemes, where the α parameter is not used directly in the verification equation. Indeed, parameters R and S are generated in accordance with the following formulas: $R = \alpha^k \bmod p$ and $S = \alpha^g \bmod p$, therefore, if one of the congruences $\alpha \equiv 1 \bmod r$ or $\alpha \equiv 1 \bmod q$ are valid, then we have: $R \equiv \alpha^k \equiv S \equiv a^g \equiv 1 \bmod r$ or $R \equiv \alpha^k \equiv S \equiv a^g \equiv 1 \bmod q$, correspondingly. Thus, if the additional restrictions for generating modulus n are not taken into account, then it becomes possible to factorize n as follows: $\gcd(S-1, n) = d$ or $\gcd(R-1, n) = d$, where $d = r$ or $d = q$.

5 Experiments

5.1 Portion of "double" primitive elements

Since some new restrictions have been imposed on the n modulus selection it becomes interesting if the portion of the required values is sufficiently large and the generation of the required parameters is not difficult. Some experiments have been performed to clarify this problem. We have investigated the probability of appearance of "double" primitive element modulo r and modulo q for the case $|r| = |q|$.

In the first experiment two arrays $\{q_1, q_2, ..., q_{100}\}$ and $\{r_1, r_2, ..., r_{100}\}$ of random prime numbers q and r were fixed and for all couples q_i and r_j, where $i, j = 1...100$, random values β were checked whether they were primitive elements both modulo q_i and modulo r_j. The number of cases in which the β value is a "double" primitive element were calculated. The portion of "double" primitive elements had been estimated as ratio of successful attempts to total number of the checked values β.

The second experiment was analogous to the first one except the numbers r_j had special structure such that $r_j - 1 = 2r'$, where r' was a prime (q was a random prime).

The third experiment was analogous to the first one except both of the numbers q_i and r_j had special structure such that $r_j - 1 = 2r'$ and $q_i - 1 = 2q'$, where r' and q' were primes. Results of the experiments are presented in Table 2, where probability of appearance of "double primitive element" for different r and q are shown.

Table 2. Probability of appearance of "double primitive element"

	random r and q	$r = 2r' + 1$, q is random	$r = 2r' + 1$, $q = 2q' + 1$
size16 bits	0.138	0.190	0.242
size 32 bits	0.141	0.191	0.249
size 128 bits	0.140	0.191	0.250

As you can see, the numbers r and q of special case have largest amount of "double primitive elements".

5.2 The probability to get $\gcd(\alpha - 1, n) \neq 1$

Fulfillment of the condition $\gcd(\alpha - 1, n) \neq 1$ means that the α value is not "strong" and permits easy factorization of the n modulus. We considered such primes r and q, that r and q had the following special forms: $2^k\gamma + 1$, $2^k\gamma t + 1$, $2^k\gamma wz + 1$, $2a^k\gamma + 1$, where γ, z w, a and t are primes, k is integer. For such structures of primes the following theoretic estimation holds: $\Pr[\gcd(\alpha - 1, n) \neq 1] = \Pr(r|\alpha - 1) + \Pr(q|\alpha - 1)$, where $\Pr[r|(\alpha - 1)] = \Pr[q|(\alpha - 1)] = \frac{1}{\gamma}$. In the experimental investigation we had generated 10,000 random values β for each r. For each case we calculated the value $\alpha = \beta^{L(rq)/\gamma^2} \bmod n$ and checked if the inequality $\gcd(\alpha - 1, r) \neq 1$ holds (successful attempt). The average probability to get $\gcd(\alpha - 1, r) \neq 1$ had been estimated as ratio of successful attempts to 10,000. The results are shown in Table 3.

Table 3. Probability to get $\gcd(\alpha - 1, r) \neq 1$

	$\gamma = 13$	$\gamma = 61$	$\gamma = 251$	$\gamma = 1021$	$\gamma = 4093$	$\gamma = 16381$
$r = 2^k\gamma t_r + 1$	0,07691	0,01635	0,00397	0,001	0,00024	0,00006
$r = 2^k\gamma w_r z_r + 1$	0,07694	0,01639	0,00396	0,00097	0,00024	0,00006
$r = 2a_r^k\gamma + 1$	0,077	0,01633	0,00397	0,00098	0,00026	0,00006
Theoretical estimation	0,07692	0,01639	0,00398	0,00098	0,00024	0,00006

6 Examples

This section presents numerical examples illustrating the performance of two variants of the DSS schemes based on the described method.

Example 1 corresponds to the verification equation

$$\alpha^H S = (\alpha^k y)^{(\alpha^k S \bmod p)} \bmod p,$$

where p is prime, α is a γ-order group generator, γ is a prime divisor of $p-1$.
$p = 111448094846043790046113767636511752606443770617 1$;
$\gamma = 43884286379314688609623509163491647315686 3$;
$\alpha = 89497661212609760234724389905138807616510137179$;
$H = 12345678900987654321$.
Private key x is equal to $x = 123456789421$; and public key y is $y = \alpha^x \bmod p = 70864334812129499928507069892528560686709201 3226$.

Signature generation procedure is as follows.
Choose random number $U < p-1 : U = 324567894535645$;
calculate $Z = \alpha^U \bmod p$:
$Z = 59730358980498739252487223012749133018804956266$.

Solve the following system of two congruences:

$$\begin{cases} H + g = Z(k + x) \bmod \gamma \\ k + g = U \bmod \gamma \end{cases}$$

We get

$$k = \frac{H - Zx + U}{Z + 1} \bmod \gamma =$$

$$= 24322086709251169695760096421007332936051$$

$$g = \frac{UZ - H + Zx}{Z + 1} \bmod \gamma =$$

$$= 41452077708389571640047499553847703475 6457$$

Now we can calculate $S = \alpha^g \bmod p$.
$S = 34248125602046242146432927566020390841686834125 5$.

Signature is a pair of integers

$$(k, S) = (24322086709251169695760096421007332936051,$$

$$34248125602046242146432927566020390841686834125 5).$$

The signature verification gives:

$$\alpha^H S = 18611658675032528675806572205325797943117134477 6$$

$$(\alpha^k y)^{(\alpha^k S \bmod p)} \bmod p =$$

$$= 186116586750325256758065722083257979481171344876.$$

Thus, the verification result is positive.

Example 2 corresponds to the verification equation

$$\alpha^k \bmod n = S^{H(\alpha^k S \bmod n)} \bmod n,$$

where n is RSA modulus ($n = r * q$), α is a γ-order group generator. The γ value represents a production of two primes: $\gamma = \gamma'\gamma''$, where $\gamma'|r-1$, $\gamma' \not| q-1$ and $\gamma''|q-1$, $\gamma'' \not| r-1$.

In the example we have generated the following values of the DSS parameters: $\gamma' = 304417319473$; $\gamma'' = 509801878007$; $\gamma = 155192521165192291530311$; $r = 186859595222449878933706703 9$;
$q = 207162539019283834092433954034 89$;
$n =$
$= 387103081863983561529983812356929823366698433937812474990 71$;
$\alpha =$
$= 184847879437491203098938318996018868688047509495509021823 15$;
$H = 1234567890987654321$.

The signature generation procedure is described as follows:
i) choose random number $U < \gamma$: $U = 445274222$
ii) calculate $Z = \alpha^U \bmod n$:

$Z = 797412200754415147882392318201658204428653141530458590298 6$

iii) solve the following system of two congruences:

$$\begin{cases} k = gHZ \bmod \gamma \\ k + g = U \bmod \gamma \end{cases}$$

Solving the system we get:

$$k = \frac{UHZ}{1+HZ} \bmod \gamma = 33818289091380076966133;$$

$$g = \frac{U}{1+HZ} \bmod \gamma = 121374232073812659838400$$

Now the S signature element can be calculated: $S = \alpha^g \bmod n$.
$S =$
$= 77776567119058764532311464894643184416738454486033447867 19.$

The digital signature is $(k, S) =$ (338182890913800769661 33, 77776567119058764532311464894643184416738454486033447867 19).

The signature verification gives:
$\alpha^k \bmod n =$
$= 30572854565748208790351912547491296565498163018021253755420$
$S^{H(\alpha^k S \bmod n)} \bmod n =$
$= 30572854565748208790351912547491296565498163018021253755420$
Thus, the verification result is positive.

7 Conclusion.

Using a novel mechanism of the signature generation we have proposed new signature schemes based on the DLP and factorization problem. The feature of the applied signature generation mechanism consists in simultaneous calculation of the k and g parameters that define signature (k, S) or (R, S), where $R = \alpha^k \bmod p$ and $S = \alpha^g \bmod p$, in different variants of DSS. Using the composite modulus one can simplify the verification equation, but in this case some additional (relatively ones corresponding to the RSA cryptosystem) restriction requirements to the public key should be taken into account. The fulfilled experiments have shown that the additional requirements do not introduce essential restrictions for practical use of the proposed DSS based on composite modulus.

References

[1] R.L. Rivest, A. Shamir, and L.M. Adleman. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, 1978, vol. 21, n. 2, pp. 120–126.

[2] T.A. ElGamal. *Public key cryptosystem and a signature scheme based on discrete logarithms* // IEEE Transactions on Information Theory. 1985, Vol. IT-31, No. 4. pp. 469–472.

[3] A.J. Menezes, S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996. 780 p.

[4] D.N. Moldovyan. *Digital Signature Schemes Based on Difficulty of Factorizing Modulus* // Voprosy zaschity informatsii (Moscow). 2004. No. 4 (67). pp. 6–11 (in Russian).

A.A.Moldovyan, D.N.Moldovyan, L.V.Gortinskaya Received December 6, 2005

Specialized Center of Program Systems "SPECTR",
Kantemirovskaya, 10, St.Petersburg 197342, Russia;
Phone/fax: 7-812-2453743,
E–mail: *nmold@cobra.ru*