Computer Science Journal of Moldova, vol.**21**, no.1(61), 2013

Developments in Networks of Evolutionary Processors *

Artiom Alhazov

Abstract

Networks of evolutionary processors (NEPs) are distributed word rewriting systems typically viewed as language generators. Each node contains a set of words, a set of operations (typically insertion, deletion or rewriting of one symbol with another one), an input filter and an output filter. The purpose of this paper is to overview existing models of NEPs, their variants and developments.

In particular, besides the basic model, hybrid networks of evolutionary processors (HNEPs) have been extensively studied. In HNEPs, operations application might be restricted to specific end of the string, but the filters are random-context conditions (they were regular in the basic model). We will also cover the literature on the so-called obligatory HNEPs, i.e., ones where the operations are obligatory: the string that cannot be rewritten is not preserved.

Some specific aspects that we pay attention to are: computational universality and completeness, the topology of the underlying graph, the number of nodes, the power of filters.

1 Introduction

Insertion, deletion, and substitution are fundamental operations in formal language theory, their power and limits have obtained much at-

*The author gratefully acknowledges project STCU-5384 "Models of high performance computations based on biological and quantum approaches", and project ref. nr. 12.819.18.09A "Development of IT support for interoperability of electronic linguistic resources" from Supreme Council for Science and Technological Development of the Republic of Moldova.

tention during the years. Due to their simplicity, language generating mechanisms based on these operations are of particular interest. *Networks of evolutionary processors* (NEPs, for short), introduced in [17], are proper examples for distributed variants of these constructs. In this case, an evolutionary processor (a rewriting system which is capable to perform an insertion, a deletion, and a substitution of a symbol) is located at every node of a virtual graph which may operate over sets or multisets of words. The system functions by rewriting the collections of words present at the nodes and then re-distributing the resulting strings according to a communication protocol defined by a filtering mechanism. The language determined by the network is defined as the set of words which appear at some distinguished node in the course of the computation. These architectures also belong to models inspired by cell biology, since each processor represents a cell performing point mutations of DNA and controlling its passage inside and outside the cell through a filtering mechanism. The evolutionary processor corresponds to the cell, the generated word – to a DNA strand, and the operations insertion, deletion, and substitution of a symbol – to the point mutations. It is known that, by using an appropriate filtering mechanism, NEPs with a very small number of nodes are computationally complete computational devices, i.e. they are as powerful as the Turing machines (see, for example [12, 13]).

1.1 Basic model

Motivated by some models of massively parallel computer architectures, networks of language processors have been introduced in [19]. Such a network can be considered as a graph, where the nodes are sets of productions and at any moment of time a language is associated with a node. In a derivation step, any node derives from its language all possible words as its new language. In a communication step, any node sends those words to other nodes that satisfy an output condition given as a regular language, and any node takes those words sent by the other nodes that satisfy an input condition also given by a regular language. The language generated by a network of language processors

consists of all (terminal) words which occur in the languages associated with a given node.

Inspired by biological processes, a special type of networks of language processors was introduced in [17], called networks with evolutionary processors, because the allowed productions model the point mutation known from biology. The sets of productions have to be substitutions of one letter by another letter or insertions of letters or deletion of letters; the nodes are then called substitution node or insertion node or deletion node, respectively. Results on networks of evolutionary processors can be found e. g. in [17], [16], [15], [12]. In [16] it was shown that networks of evolutionary processors are universal in that sense that they can generate any recursively enumerable language, and that networks with six nodes are sufficient to get all recursively enumerable languages. In [12] the latter result has been improved by showing that networks with three nodes are sufficient.

In [12] one presents the proof of the computational completeness with two nodes, additionally employing a morphism. In [9] one shows that NEPs with two nodes (one insertion node and one deletion node) generate all recursively enumerable languages (in intersection with a monoid), avoiding the need for a morphism. The same paper shows that insertion and substitution characterize context-sensitive languages, while deletion and substitution characterize finite languages.

1.2 Hybrid model

Particularly interesting variants of these devices are the so-called *hybrid networks of evolutionary processors* (HNEPs), where each language processor performs only one of the above operations on a certain position of the words in that node. Furthermore, the filters are defined by some variants of random-context conditions, i.e., they check the presence/absence of certain symbols in the words. These constructs can be considered both language generating and accepting devices, i.e., generating HNEPs (GHNEPs) and accepting HNEPS (AHNEPs). The notion of an HNEP, as a language generating device, was introduced in [27] and the concept of an AHNEP was defined in [26].

In [18] it was shown that, for an alphabet V, GHNEPs with $27 + 3 \cdot card(V)$ nodes are computationally complete. A significant improvement of the result can be found in [6], where it was proved that GHNEPs with 10 nodes (irrespectively of the size of the alphabet) obtain the universal power. For accepting HNEPs, in [24] it was shown that for any recursively enumerable language there exists a recognizing AHNEP with 31 nodes; the result was improved in [25] where the number of necessary nodes was reduced to 24. Furthermore, in [25] the authors demonstrated a method to construct for any NP-language L an AHNEP with 24 nodes which decides L in polynomial time.

At last in [7] it was proved that any recursively enumerable language can be generated by a GHNEP having 7 nodes (thus, the result from [6] is improved) and in [8] the same authors showed that any recursively enumerable language can be accepted by an AHNEP with 7 nodes (thus, the result from [25] is improved significantly). An improvement of the accepting result to 6 nodes has been obtained in [23], by simulating Tag systems. In [8] also it was showed that the families of GHNEPs and AHNEPs with 2 nodes are not computationally complete.

In [18] it was demonstrated that a GHNEP with one node can generate only regular language, while in [14] a precise form of the generated language was presented, also considering one case omitted in the previous proof. Tasks of characterization of languages generated by a GHNEP with two nodes and languages accepting by an AHNEP with two nodes are still open.

1.3 Obligatory operations

A variant of HNEPs, called Obligatory HNEPs (OHNEP for short) was introduced in [3]. The differences between HNEP and OHNEP are the following:

1. in deletion and substitution: a node discards a string if no operations in the node are applicable to the string (in HNEP case, this string remains in the node),

2. the underlying graph is a directed graph (in HNEP case, this graph is undirected); this second difference disappears when we consider complete networks.

These differences make OHNEPs universal [3] with 1 operation per node, no filters and only left insertion and right deletion.

In [5] complete OHNEPs were considered, i.e., OHNEPs with complete underlying graph. One may now regard complete OHNEP as a set of very simple evolutionary processors "swimming in the environment" (i.e., once a string leaves a node, it is not essential for the rest of the computation which node it left). In [5] it is proved that the complete OHNEPs with very simple evolutionary processors, i.e., evolutionary processors with only one operation (obligatory deletion, obligatory substitution and insertion) and filters containing not more than 3 symbols are computationally complete. We recall that the filters are either single symbols or empty sets, while the **sum** of weights has been counted.

In [4] one considers OHNEPs without substitution. It is not difficult to notice that in complete OHNEPs without substitution there is no control on the number of insertion or deletion of terminal symbols (i.e., those symbols which appear in output words). Therefore, the definition of OHNEPs needed to be modified in order to increase their computational power. In [4] one shows that it is possible to avoid substitution using modified operations of insertion and deletion in evolutionary processors similar to "matrix" rules in formal language theory. By using such techniques a small universal complete OHNEP with 182 nodes without substitution is constructed.

Several open questions were posed in [4], in particular the question about the minimal total complexity of filters of evolutionary processor in computationally complete OHNEPs and the question about universal complete OHNEP without substitution with the minimal number of nodes. In [1] one considers a model of OHNEP allowing the use of all three molecular operations: insertion, deletion and substitution, and provides a very unexpected result. OHNEPs are computationally complete even if the total power of the filters of each node does not exceed 1! This means that in any node, all four filters are empty, except

possibly one, being a single symbol.

In the following we describe selected results in details.

2 Prerequisites

We first recall some basic notions from formal language theory that we shall use in the paper. An alphabet is a finite and non-empty set of symbols. The cardinality of a finite set A is denoted by $card(A)$. A sequence of symbols from an alphabet V is called a word (or a string) over V. The set of all words over V is denoted by V^*; the empty word is denoted by ε; and we define $V^+ = V^* \setminus \{\varepsilon\}$. The length of a word x is denoted by $|x|$, and we designate the number of occurrences of a letter a in a word x by $|x|_a$. For each non-empty word x, $alph(x)$ denotes the smallest alphabet Σ such that $x \in \Sigma^*$.

For a word $u \in V^*$, we define the sets of proper prefixes, proper suffixes and non-empty suffixes of u by

$$\begin{aligned} PPref(u) &= \{x \mid u = xy, |y| \geq 1\}, \\ PSuf(u) &= \{y \mid u = xy, |x| \geq 1\}, \\ NSuf(u) &= \{y \mid u = xy, |y| \geq 1\}, \text{ respectively.} \end{aligned}$$

The shuffle operation is defined on two words $x, y \in V^*$ by

$$\text{⧢}(x, y) = \{x_1 y_1 x_2 y_2 \dots x_n y_n \mid n \geq 1,\ x_i, y_j \in V^*, \\ x = x_1 x_2 \dots x_n,\ y = y_1 y_2 \dots y_n\}.$$

Let $L_1, L_2 \in V^*$ are two languages. Then

$$\text{⧢}(L_1, L_2) = \bigcup_{x \in L_1, y \in L_2} \text{⧢}(x, y).$$

A *type-0 generative grammar* is a quadruple $G = (N, T, S, P)$, where N and T are disjoint alphabets, called the nonterminal and terminal alphabet, respectively, $S \in N$ is the start symbol or the axiom, and P is a finite set of productions or rewriting rules of the form $u \to v$, where $u \in (N \cup T)^* N (N \cup T)^*$ and $v \in (N \cup T)^*$. For two strings x and y in $(N \cup T)^*$, we say that x directly derives y in G, denoted by

$x \Longrightarrow_G v$, if there is a production $u \to v$ in P such that $x = x_1ux_2$ and $y = x_1vx_2$, $x_1, x_2 \in (N \cup T)^*$ holds. The transitive and reflexive closure of $\Longrightarrow_G$ is denoted by $\Longrightarrow_G^*$. The language $L(G)$ generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow_G^* w\}$.

We recall now a concept dual to a type-0 generative grammar, called a *type-0 analytic grammar* [28]. A type-0 analytic grammar $G = (N, T, S, P)$ is a quadruple, where N, T, S are defined in the same way as for a generative grammar, and P is a finite set of productions of the form $u \to v$, where $u \in (N \cup T)^*$ and $v \in (N \cup T)^*N(N \cup T)^*$. The derivation relation is defined for a type-0 analytic grammar analogously to the derivation relation for a type-0 generative grammar. The language $L(G)$ recognized or accepted by a type-0 analytic grammar $G = (N, T, S, P)$ is defined as $L(G) = \{w \in T^* \mid w \Longrightarrow_G^* S\}$.

It is well-known that for the type-0 analytic grammar G' obtained from a type-0 generative grammar G with interchanging the left and the right hand sides of the productions in G, it holds that $L(G') = L(G)$.

A type-0 generative grammar $G = (N, T, S, P)$ is in Kuroda normal form if every rule in P is one of the following forms: $A \longrightarrow a$, $A \longrightarrow \varepsilon$, $A \longrightarrow BC$, $AB \longrightarrow CD$, where $A, B, C, D \in N$ and $a \in T$.

Analogously, we can say that a type-0 analytic grammar $G = (N, T, S, P)$ is in Kuroda-like normal form if every production in P is one of the following forms: $a \longrightarrow A$, $\varepsilon \longrightarrow A$, $AB \longrightarrow C$, $AB \longrightarrow CD$, where $A, B, C, D \in N$ and $a \in T$.

It is well-known that the type-0 generative grammars in Kuroda normal form determine the class of recursively enumerable languages and it can immediately be seen that the same statement holds for the type-0 analytic grammars in Kuroda-like normal form.

In the sequel, following the terminology in [18], we recall the necessary notions concerning evolutionary processors and their hybrid networks. These language processors use so-called evolutionary operations, simple rewriting operations which abstract local gene mutations.

2.1 Circular Post machines

Circular Post Machines (CPMs) were introduced in [21], where it was shown that all introduced variants of CPMs are computationally complete, and moreover, the same statement holds for CPMs with two symbols. In [22], [10] several universal CPMs of variant 0 (CPM0) having small size were constructed[1], among them in [10] a universal CPM0 with 6 states and 6 symbols. In this article we use the deterministic variant of CPM0s.

A *Circular Post Machine* is a quintuple $(\Sigma, Q, \mathbf{q}_0, \mathbf{q}_f, R)$ with a finite alphabet Σ, where 0 is the blank, a finite set of states Q, the initial state $\mathbf{q}_0 \in Q$, the final state $\mathbf{q}_f \in Q$, and a finite set of instructions R with all instructions having one of the forms $\mathbf{p}x \to \mathbf{q}$ (erasing the symbol read by deleting a cell), $\mathbf{p}x \to y\mathbf{q}$ (overwriting and moving to the right), $\mathbf{p}0 \to y\mathbf{q}0$ (overwriting and creating a blank cell), where $x, y \in \Sigma$ and $\mathbf{p}, \mathbf{q} \in Q$, $\mathbf{p} \neq \mathbf{q}_f$. We also refer to all instructions with $\mathbf{q}_f$ in the left hand side as halt instructions.

The storage of this machine is a circular tape, the read and write head moves only in one direction (to the right), and with the possibility to delete a cell or to create and insert a new cell with a blank.

2.2 Evolutionary processors

For an alphabet V, we say that a rule $a \to b$, with $a, b \in V \cup \{\varepsilon\}$ is a *substitution rule* if both a and b are different from ε; it is a *deletion rule* if $a \neq \varepsilon$ and $b = \varepsilon$; and, it is an *insertion rule* if $a = \varepsilon$ and $b \neq \varepsilon$. The set of all substitution rules, deletion rules, and insertion rules over an alphabet V is denoted by Sub_V, Del_V, and Ins_V, respectively. Given such rules π, ρ, σ, and a word $w \in V^*$, we define the following *actions* of σ on w: If $\pi \equiv a \to b \in Sub_V$, $\rho \equiv a \to \varepsilon \in Del_V$, and $\sigma \equiv \varepsilon \to a \in Ins_V$, then

$$\pi^*(w) = \begin{cases} \{ubv : \exists u, v \in V^*(w = uav)\}, \\ \{w\}, \quad \text{otherwise} \end{cases} \tag{1}$$

[1]Other variants of CPMs use slightly different instruction sets, which may make a difference for the size of small universal machines.

$$\rho^*(w) = \begin{cases} \{uv : \exists u, v \in V^*(w = uav)\}, \\ \{w\}, \quad \text{otherwise} \end{cases} \tag{2}$$

$$\rho^r(w) = \begin{cases} \{u : \quad w = ua\}, \\ \{w\}, \quad \text{otherwise} \end{cases} \tag{3}$$

$$\rho^l(w) = \begin{cases} \{v : \quad w = av\}, \\ \{w\}, \quad \text{otherwise} \end{cases} \tag{4}$$

$$\sigma^*(w) = \{uav : \exists u, v, \in V^*(w = uv)\}, \tag{5}$$

$$\sigma^r(w) = \{wa\},\ \sigma^l(w) = \{aw\}. \tag{6}$$

Symbol $\alpha \in \{*, l, r\}$ denotes the way of applying an insertion or a deletion rule to a word, namely, at any position ($a = *$), in the left-hand end ($a = l$), or in the right-hand end ($a = r$) of the word, respectively. Note that a substitution rule can be applied at any position. For every rule σ, action $\alpha \in \{*, l, r\}$, and $L \subseteq V^*$, we define the $\alpha - action$ of σ on L by $\sigma^\alpha(L) = \bigcup_{w \in L} \sigma^\alpha(w)$. For a given finite set of rules M, we define the $\alpha - action$ of M on a word w and on a language L by $M^\alpha(w) = \bigcup_{\sigma \in M} \sigma^\alpha(w)$ and $M^\alpha(L) = \bigcup_{w \in L} M^\alpha(w)$, respectively.

An evolutionary processor consists of a set of evolutionary operations and a filtering mechanism.

For two disjoint subsets P and F of an alphabet V and a word over V, predicates $\varphi^{(1)}$ and $\varphi^{(2)}$ are defined as follows:

$$\varphi^{(1)}(w; P, F) \equiv P \subseteq alph(w) \wedge F \cap alph(w) = \emptyset$$

and

$$\varphi^{(2)}(w; P, F) \equiv alph(w) \cap P \neq \emptyset \wedge F \cap alph(w) = \emptyset.$$

The construction of these predicates is based on *random-context conditions* defined by the two sets P (*permitting contexts*) and F (*forbidding contexts*).

For every language $L \subseteq V^*$ we define $\varphi^i(L, P, F) = \{w \in L \mid \varphi^i(w; P, F)\}$, $i = 1, 2$.

An *evolutionary processor over* V is a 5-tuple (M, PI, FI, PO, FO), where:

- Either $M \subseteq Sub_V$ or $M \subseteq Del_V$ or $M \subseteq Ins_V$. The set M represents the set of evolutionary rules of the processor. Notice that every processor is dedicated to only one type of the evolutionary operations.

- $PI, FI \subseteq V$ are the *input* permitting/forbidding contexts of the processor, while $PO, FO \subseteq V$ are the *output* permitting/forbidding contexts of the processor.
The set of evolutionary processors over V is denoted by EP_V.

2.3 Hybrid networks

Definition 1 *A* hybrid network of evolutionary processors *(an HNEP, shortly) is a 7-tuple* $\Gamma = (V, H, \mathcal{N}, C_0, \alpha, \beta, i_0)$*, where the following conditions hold:*

- V *is the alphabet of the network.*

- $H = (X_H, E_H)$ *is an undirected graph with set of vertices or nodes* X_H *and set of edges* E_H*. H is called the underlying graph of the network.*

- $\mathcal{N} : X_H \longrightarrow EP_V$ *is a mapping which associates the evolutionary processor* $\mathcal{N}(x) = (M_x, PI_x, FI_x, PO_x, FO_x)$ *with each node* $x \in X_H$*.*

- $C_0 : X_H \longrightarrow 2^{V^*}$ *is a mapping which identifies the initial configuration of the network. It associates a finite set of words with each node of the graph H.*

- $\alpha : X_H \longrightarrow \{*, l, r\}$*;* $\alpha(x)$ *defines the action mode of the rules performed in node x on the words occurring in that node.*

- $\beta : X_H \longrightarrow \{(1), (2)\}$ *defines the type of the input/output filters of a node. More precisely, for every node,* $x \in X_H$*, we define the following filters: the input filter is given as* $\mu_x(\cdot) = \varphi^{\beta(x)}(\cdot; PI_x, FI_x)$*, and the output filter is defined as* $\tau_x(\cdot) = \varphi^{\beta(x)}(\cdot, PO_x, FO_x)$*. That is,* $\mu_x(w)$ *(resp.*τ_x*) indicates whether or not the word w can pass the input (resp. output) filter of x. More generally,* $\mu_x(L)$ *(resp.* $\tau_x(L)$*) is the set of words of L that can pass the input (resp. output) filter of x.*

- $i_0 \in X_H$ *is the output node of* Γ*.*

We say that $card(X_H)$ is the size of Γ. An HNEP is said to be a complete HNEP, if its underlying graph is a complete graph.

A configuration of an HNEP Γ, as above, is a mapping $C : X_H \longrightarrow 2^{V^*}$ which associates a set of words with each node of the graph. A component $C(x)$ of a configuration C is the set of words that can be found in the node x in this configuration, hence a configuration can be

considered as the sets of words which are present in the nodes of the network at a given moment.

A configuration can change either by an evolutionary step or by a communication step. When it changes by an evolutionary step, then each component $C(x)$ of the configuration C is altered in accordance with the set of evolutionary rules M_x associated with the node x and the way of applying these rules, $\alpha(x)$. Formally, the configuration C' is obtained in one evolutionary step from the configuration C, written as $C \Longrightarrow C'$, iff $C'(x) = M_x^{\alpha(x)}(C(x)) \quad$ for all $x \in X_H$.

When the configuration changes by a communication step, then each language processor $\mathcal{N}(x)$, where $x \in X_H$, sends a copy of its each word to every node processor, where the node is connected with x, provided that this word is able to pass the output filter of x, and receives all the words which are sent by processors of nodes connected with x, provided that these words are able to pass the input filter of x. Those words which are not able to pass the respective output filter, remain at the node. Formally, we say that configuration C' is obtained in one communication step from configuration C, written as $C \vdash C'$, iff $C'(x) = (C(x) - \tau_x(C(x))) \bigcup_{\{x,y\} \in E_H} (\tau_y(C(y)) \cap \mu_x(C(y)))$ holds for all $x \in X_H$.

2.4 Computation and result

For an HNEP Γ, the computation in Γ is a sequence of configurations C_0, C_1,$C_2, \ldots$, where C_0 is the initial configuration of Γ, $C_{2i} \Longrightarrow C_{2i+1}$ and $C_{2i+1} \vdash C_{2i+2}$, for all $i \geq 0$.

HNEPs can be considered both language generating devices (generating hybrid networks of evolutionary processors or GHNEPs) and language accepting devices (accepting hybrid networks of evolutionary processors or AHNEPs).

In the case of GHNEPs we define the generated language as the set of all words which appear in the output node at some step of the computation. Formally, the language generated by a generating hybrid network of evolutionary processors Γ is $L(\Gamma) = \bigcup_{s \geq 0} C_s(i_0)$.

In the case of AHNEPs, in addition to the components above,

we distinguish an input alphabet and a network alphabet, V and U, where $V \subseteq U$, and instead of an initial configuration, we indicate an input node i_I. Thus, for an AHNEP, we use the notation $\Gamma = (V, U, H, \mathcal{N}, i_I, \alpha, \beta, i_0)$.

The computation by an AHNEP Γ for an input word $w \in V^*$ is a sequence of configurations $C_0^{(w)}$, $C_1^{(w)}$,$C_2^{(w)}$, ..., where $C_0^{(w)}$ is the initial configuration of Γ, with $C_0^{(w)}(i_I) = \{w\}$ and $C_0^{(w)}(x) = \emptyset$, for $x \in G$, $x \neq i_I$, and $C_{2i}^{(w)} \Longrightarrow C_{2i+1}^{(w)}$, $C_{2i+1}^{(w)} \vdash C_{2i+2}^{(w)}$, for all $i > 0$.

A computation as above is said to be accepting if there exists a configuration in which the set of words that can be found in the output node i_o is non-empty. The language accepted by Γ is defined by

$$L(\Gamma) = \{w \in V^* \mid \text{ the computation by } \Gamma \text{ on } w \text{ is an accepting one}\}.$$

2.5 Obligatory networks

The model of OHNEPs is obtained from the model of HNEPs by excluding the second case, i.e., "$\{w\}$,otherwise", from (1)-(4). Hence, for a string to remain in a node, it is *obligatory* for it to evolve via some rule from Sub_V, Del_V or Ins_V.

Notice that the definition of OHNEPs is thus simpler and more uniform than that of HNEPs. In the same time, using the power of the underlying graph, it makes it possible to even reach the computational completeness with nodes only having one operation, and without filters, [3].

2.6 Basic model of NEPs

The concept of NEPs is simpler than that of HNEPs. We find it suitable here to define the former in terms of the latter, by modifying the following:

- Only the $*$ mode exists for evolutionary processors.
- A permitting filter and a forbidden filter are combined into one filter, which may be any regular language. A string passes the

filter iff it belongs to the corresponding regular language. The filtering mode β loses its meaning.

3 Selected results

We would like to point out that there is no interaction, direct or indirect, between the words of the network. Hence, the generated language is a union of languages, generated by the same system, but starting with only one word.

As for the replication, i.e., the possibility of applying multiple rules or the same rule in multiple ways, producing many words from one word, this could be viewed as a non-deterministic evolution of *one* word. In this case, distributivity simply means assigning a state to the word. Summing up, the language generated by a parallel deterministic word rewriting system may be viewed as a (union of) language(s) generated by a non-deterministic one-word rewriting system with states (without any other parallelism or distributivity).

Furthermore, the nature of the model (except the obligatory variant) leads to many cases of the "shadow" computations, in the following sense. If one carefully considers the definitions, and constructs a faithful simulation of the model, one would notice that a lot of computation in the system consists of repeatedly recomputing the same steps. This is due to the fact that if some operation $\pi \in Sub_V$ or $\rho \in Del_V$ of a node is not applicable to some word w in that node, the result is w. Taking the union over all operations of a node yields a set containing w, even if some other operation was applicable to w. Clearly, in the next step, the words obtainable from w in the same node will be recomputed. However, the system is deterministic, so nothing *new* is obtained in this way.

A careful examination reveals that, in some circumstances, the shadow computations can be avoided, modifying the definition while yielding the same generated language! Indeed, preserving w is useless (everything that is possible to derive from w in that node is derived immediately) unless w exits the node. However, at least in the case of complete networks, if w enters a node and exits it unchanged, this does

not do anything new either (if w is an initial word, it can be copied to all nodes that it can reach unchanged in communication step).

The above reason lets us claim that, e.g., any result for complete OHNEPs holds also for the usual complete HNEPs, and the associated computational burden of the simulation may be greatly reduced. If the network is not complete, then a heuristic still may be used by a simulator, by preserving unchanged words only in case if they actually move from a node into some different node.

3.1 NEPs with two nodes

Theorem 1 *For any recursively enumerable language L, there are a set T and a network N of evolutionary processors with exactly one insertion node and exactly one deletion node such that $L = L(N) \cap T^*$.[9]*

Proof. (sketch) We consider a type-0 grammar $G = (N, T, P, S)$ with $L(G) = L$. Then all rules of P have the form $u \to v$ with $u \in N^+$ and $v \in (N \cup T)^*$. Let $X = N \cup T, X' = \{a, a' \mid a \in N \cup T\}$, $T' = \{a, a' \mid a \in T\}$ and $P' = \{p_i \mid p \in P, 1 \leq i \leq 4\}$. We define a morphism $\mu : X^* \to (X')^*$ by $\mu(a) = aa'$ for $a \in X$ and set $W = \{\mu(w) \mid w \in X^*\}$. We construct the following network $\mathcal{N} = (V, (M_1, A_1, I_1, O_1), (M_2, A_2, I_2, O_2), E, 2)$ of evolutionary processors with

$$
\begin{aligned}
V &= P' \cup X', \\
M_1 &= \{\lambda \to p_i \mid p_i \in P',\ 1 \leq i \leq 4\} \cup \{\lambda \to a \mid a \in X'\}, \\
A_1 &= \{\mu(S)\}, \\
I_1 &= W \setminus (T')^*, \\
O_1 &= V * \backslash (W R_{1,1} W), \\
M_2 &= \{p_i \to \lambda \mid p_i \in P',\ 1 \leq i \leq 4\} \cup \{a \to \lambda \mid a \in X' \setminus T\}, \\
A_2 &= \emptyset, \\
I_2 &= W R_{1,2} W, \\
O_2 &= V^* \setminus (W R_{2,2} W \cup (T')^*),
\end{aligned}
$$

$$E = \{(1,2),(2,1)\}, \text{ where}$$

$$\begin{aligned}
R_{1,1} &= \bigcup_{p:u\to v\in P} (\{p_1\mu(u), p_1\mu(u)p_3, p_1p_2\mu(u)p_3, p_1p_2\mu(u)p_3p_4\} \\
&\cup \{p_1p_2\mu(u)\}PPref(\mu(v))\{p_3p_4\}) \\
&\quad \setminus\{p_1p_2\mu(u)p_3p_4 \mid p : u \to v \in P\}, \\
R_{1,2} &= \{p_1p_2\mu(uv)p_3p_4 \mid p : u \to v \in P\}, \\
R_{2,2} &= \bigcup_{p:u\to v\in P} (\{p_1p_2\}PSuf(\mu(u))\{\mu(v)p_3p_4\} \\
&\cup \{p_2\mu(v)p_3p_4, p_2\mu(v)p_4, \mu(v)p_4\}).
\end{aligned}$$

The output and input filters are defined in order to remove the garbage and communicate the strings that should change the type of operation, keeping only the strings that should continue to evolve by operations of the same type. Since the morphism $\mu(a) = aa'$ is introduced, the strings obtained by applying rules to the left or to the right of the place of application of the current rule are no longer kept in the node by the filter, and are not accepted by either node (recall that $W = aa'|a \in \{N \cup T^*\}$), so they leave the system. Claim: $L(\mathcal{N}) \setminus T^* = L$. The correct simulation of an application of a production $p : a_1 \cdots a_s \to b_1 \cdots b_t$ to a sentential form $\alpha a_1 \cdots a_s \beta$ and with $x = \mu(\alpha)$ and $y = \mu(\alpha)$ has the following form. In N_1 we have

$$\begin{aligned}
xa_1a_1' \cdots a_sa_s'y &\Rightarrow^{\lambda\to p_1} xp_1a_1a_1' \cdots a_sa_s'y \\
&\Rightarrow^{\lambda\to p_3} xp_1a_1a_1' \cdots a_sa_s'p_3y \\
&\Rightarrow^{\lambda\to p_2} xp_1p_2a_1a_1' \cdots a_sa_s'p_3y \\
&\Rightarrow^{\lambda\to p_4} xp_1p_2a_1a_1' \cdots a_sa_s'p_3p_4y \\
&\Rightarrow^{\lambda\to b_1} xp_1p_2a_1a_1' \cdots a_sa_s'b_1p_3p_4y \\
&\Rightarrow^{\lambda\to b_1'} xp_1p_2a_1a_1' \cdots a_sa_s'b_1b_1'p_3p_4y \\
&\Rightarrow^{*} xp_1p_2a_1a_1' \cdots a_sa_s'b_1b_1' \cdots b_tp_3p_4y \\
&\Rightarrow^{\lambda\to b_t'} xp_1p_2a_1a_1' \cdots a_sa_s'b_1b_1' \cdots b_tb_t'p_3p_4y
\end{aligned}$$

and in N_2 we have

$$\begin{aligned}
& xp_1p_2a_1a_1' \cdots a_sa_s'b_1b_1' \cdots b_tb_t'p_3p_4y \\
\Rightarrow^{a_1 \to \lambda} \quad & xp_1p_2a_1' \cdots a_sa_s'b_1b_1' \cdots b_tb_t'p_3p_4y \\
\Rightarrow^{a_1' \to \lambda} \quad & xp_1p_2a_2 \cdots a_sa_s'b_1b_1' \cdots b_tb_t'p_3p_4y \\
\Rightarrow^{*} \quad & xp_1p_2a_sa_s'b_1b_1' \cdots b_tb_t'p_3p_4y \\
\Rightarrow^{a_s \to \lambda} \quad & xp_1p_2a_s'b_1b_1' \cdots b_tb_t'p_3p_4y \\
\Rightarrow^{a_s' \to \lambda} \quad & xp_1p_2b_1b_1' \cdots b_tb_t'p_3p_4y \\
\Rightarrow^{p_1 \to \lambda} \quad & xp_2b_1b_1' \cdots b_tb_t'p_3p_4y \\
\Rightarrow^{p_3 \to \lambda} \quad & xp_2b_1b_1' \cdots b_tb_t'p_4y \\
\Rightarrow^{p_2 \to \lambda} \quad & xb_1b_1' \cdots b_tb_t'p_4y \\
\Rightarrow^{p_4 \to \lambda} \quad & xb_1b_1' \cdots b_tb_t'y.
\end{aligned}$$

Notice that if a production can be applied to the same sentential form in different ways (multiple productions and/or multiple places to apply them), then the corresponding number of strings is produced in the first step (inserting marker p_1 associated to the production, to the left of the application place). The rest of the simulation is deterministic in the following sense: starting from $xp_1a_1a_1' \cdots a_sa_s'y$, the result $xb_1b_1' \cdots b_tb_t'y$ is obtained according to the derivations above, while all other strings are discarded. The strings that leave one node and enter another one belong to the sets $O_1 \setminus I_2 = I_2$ and $O_2 \setminus I_1 = I_1$. All other strings that leave a node do not enter anywhere. With $p \in P$, $a \in X'$ and $A \in X' \setminus T$, the following tables illustrate the behaviour of a string (the numbers give the situation which is obtained by using the rule in question and n/a refers to nonapplicability of the rule).

n	Shape in $N1$	$\lambda \to p_1$	$\lambda \to p_3$	$\lambda \to p_2$	$\lambda \to p_4$	$\lambda \to a$
1	W	2	out	out	out	out
2	$Wp_1\mu(u)W$out	3	out	out	out	
3	$Wp_1\mu(u)p_3W$	out	out	4	out	out
4	$Wp_1p_2\mu(u)p_3W$	out	out	out	5	out
5	$Wp_1p_2\mu(u)\cdot$ $PPref(\mu(v))p_3p_4W$	out	out	out	out	5,6

n	Shape in $N1$	$p_1 \to \lambda$	$p_3 \to \lambda$	$p_2 \to \lambda$	$p_4 \to \lambda$	$A \to \lambda$
6	$Wp_1p_2\cdot$ $NSuf(\mu(u))\mu(v)p_3p_4W$	out	out	out	out	6,7
7	$Wp_1p_2\mu(v)p_3p_4W$	8	out	out	out	out
8	$Wp_2\mu(v)p_3p_4W$	n/a	9	out	out	out
9	$Wp_2\mu(v)p_4W$	n/a	n/a	10	out	out
10	$W\mu(v)p_4W$	n/a	n/a	n/a	1,11	out
11	$(T')^*$	n/a	n/a	n/a	n/a	11

These tables illustrate the fact that if a symbol is inserted or deleted in a way that does not follow the "correct" simulation, than the string leaves the system. Finally, consider $L(\mathcal{N}) \cap (T')^*$. It is the set of all strings obtained in N_2 without nonterminal symbols, without markers and without pre-terminals (i. e., primed versions of terminals). Hence, all of them are obtained from shape 5 of N_2 by deleting the marker p_4, reaching shape 6 if the string only has terminals and pre-terminals. It is easy to see that in several computation steps all pre-terminal symbols will be deleted. This exactly corresponds to the set of terminal strings w produced by the underlying grammar G, all letters being represented by a double repetition, i. e., encoded by μ. Such strings remain in N_2 and all pre-terminals are deleted, obtaining w from $\mu(w)$. □

3.2 HNEPs with one node

The following theorem states the regularity result for GHNEPs with one node. Although this has already been stated in [18], their proof is certainly incomplete. They stated that while GHNEPs without insertion only generate finite languages, GHNEPs with one insertion node only generate languages I^*C_0, C_0I^*, $C_0 \coprod I^*$, for the mode $l, r, *$, respectively. In the theorem below we present a precise characterization of languages generated by GHNEP with one node and consider the case omitted in [18], when the underlying graph G has a loop.

Theorem 2 *One-node GHNEPs only generate regular languages.[14]*

Proof. As finite languages are regular, the statement holds for GHNEPs without insertion nodes. We now proceed with the case of one

insertion node. Consider such a GHNEP $\Gamma = (V, G, N_1, C_0, \alpha, \beta, 1)$, where

$$N_1 = (M, PI, FI, PO, FO).$$

Let us introduce a few notations. Inserting a symbol from I in a language C yields a language $\mathsf{ins}_I(C)$. Depending on whether $\alpha = l$, $\alpha = r$ or $\alpha = *$, $\mathsf{ins}_I(C)$ is one of IC, CI, $C \coprod I$, respectively. For inserting an arbitrary number of symbols from a set I in a language C, $\mathsf{ins}_I^*(C)$ is one of I^*C, CI^*, $C \coprod I^*$. Clearly, ins_I^* preserves regularity.

We denote the set of symbols inserted in N_1 by $I = \{a \mid \lambda \to a \in M\}$. The configuration of N_1 after one step is $C_1 = \mathsf{ins}_I(C_0)$. Assume that $\beta = 2$ (a case, when $\beta = 1$, can be considered analogously), then the conditions of passing permitting and forbidding output filter can be specified by regular languages $\pi = V^*POV^*$ and $\varphi = (V - FO)^*$, respectively. For instance, the set of words of C_1 that pass the forbidding output filter but do not pass the forbidding input filter is $C_1' = C_1 \cap \varphi \setminus \pi$. Notice that inserting symbols that belong to neither PO nor FO does not change the behavior of the filters; we denote the corresponding language by $B = \mathsf{ins}^*_{I \setminus (PO \cup FO)}(C_1)$.

Consider the case when the graph G consists of one node and no edges. Then, Γ generates the following language

$$\begin{aligned} L_1 = L_1(\Gamma) &= C_0 \cup C_1 \cup \mathsf{ins}_I^*(C_1 \setminus \varphi) \cup B \\ &\quad \cup \mathsf{ins}_{I \cap PO \setminus FO}(B) \cup \mathsf{ins}_I^*(\mathsf{ins}_{I \cap FO}(B)), \qquad (7) \\ B &= \mathsf{ins}^*_{I \setminus (PO \cup FO)}(C_1), \\ C_1 &= \mathsf{ins}_I(C_0). \end{aligned}$$

Indeed, this is a union of six languages:

1. initial configuration,
2. configuration after one insertion,
3. all words that can be obtained from a word from C_1 if it is trapped in N_1 by the forbidding filter,
4. B represents the words that pass the forbidding filter but not the permitting filter,

5. words obtained by inserting one permitting and not forbidden symbol into B, and

6. words obtained by inserting one forbidden symbol into B, and then by arbitrary insertions.

Consider the case when the graph G has a loop. The set of words leaving the node (for the first time) is $D = (C_1 \cap \varphi \cap \pi) \cap \mathsf{ins}_{I \cap PO \setminus FO}(B)$. The conditions of the permitting and forbidding input filters can be specified by regular languages $\pi' = V^*PIV^*$ and $\varphi' = (V - FI)^*$, respectively. Some of words from D return to N_1, namely $D \cap \pi' \cap \varphi'$. Notice that further insertion of symbols that belong neither to FO nor to FI causes the words to continuously exit and reeenter N_1. The associated language is $B' = \mathsf{ins}^*_{I \setminus (FO \cup FI)}(D \cap \pi' \cap \varphi')$. Finally, we give the complete presentation of the language generated by Γ in this case:

$$\begin{aligned} L_1' = L_1(\Gamma) &= L_1 \cup B' \cup \mathsf{ins}^*_I(\mathsf{ins}_{I \cap FO}(B')) \cup \mathsf{ins}_{I \cap FI \setminus FO}(B'), \quad (8) \\ B' &= \mathsf{ins}^*_{I \setminus (FO \cup FI)}(D \cap \pi' \cap \varphi'), \\ D &= (C_1 \cap \varphi \cap \pi) \cap \mathsf{ins}_{I \cap PO \setminus FO}(B), \\ C_1 &= \mathsf{ins}_I(C_0). \end{aligned}$$

Indeed, this is a union of four languages:

1. words that never reenter N_1, as in the case when G has no edges,

2. B' represents the words that once leave and reenter N_1, and keep doing so after subsequent insertions,

3. words obtained by inserting a symbol from FO into B', and then by arbitrary insertions,

4. words obtained by inserting a symbol from $FI \setminus FO$ into B'.

□

3.3 HNEPs with 7 nodes

Theorem 3 *Any recursively enumerable language can be generated by a complete HNEP of size 7. [7, 8]*

Proof. Let $L \subseteq T^*$ be a language generated by a type-0 grammar $G = (N, T, S, P)$ in Kuroda normal form.

We construct a complete HNEP $\Gamma = (V, H, \mathcal{N}, C_0, \alpha, \beta, 7)$ of size 7 which simulates the derivations in G and only that, by using the so-called rotate-and-simulate method. The rotate-and-simulate method means that the words in the nodes are involved in either the rotation of their leftmost symbol (the leftmost symbol of the word is moved to the end of the word) or the simulation of a rule of P. In order to indicate the end of the word when rotating its symbols and thus to guarantee the correct simulation, a marker symbol, #, different from any element of $(N \cup T)$ is introduced. Let $N \cup T \cup \{\#\} = A = \{A_1, A_2, \dots A_n\}$, $I = \{1, 2, \dots, n\}$, $I' = \{1, 2, \dots, n-1\}$, $I'' = \{2, 3 \dots, n\}$, $I_0 = \{0, 1, 2, \dots, n\}$, $I'_0 = \{0, 1, 2, \dots, n-1\}$, $B_0 = \{B_{j,0} \mid j \in I\}$, $B'_0 = \{B'_{j,0} \mid j \in I\}$, $\# = A_n$, $T' = T \cup \#$. Let us define the alphabet V of Γ as follows:

$$
\begin{aligned}
V &= A \cup B \cup B' \cup C \cup C' \cup D \cup D' \cup E \cup E' \cup F \cup G \cup \{\varepsilon'\},\\
B &= \{B_{i,j} \mid i \in I,\ j \in I_0\}, B' = \{B'_{i,j} \mid i \in I, j \in I_0\},\\
C &= \{C_i \mid i \in I\}, C' = \{C'_i \mid i \in I\}, D = \{D_i \mid i \in I_0\},\\
D' &= \{D'_i \mid i \in I\},\\
E &= \{E_{i,j} \mid i, j \in I\}, E' = \{E'_{i,j} \mid i, j \in I\},\\
F &= \{F_j \mid j \in I\}, G = \{G_{i,j} \mid i, j \in I\}.
\end{aligned}
$$

Let H be a complete graph with 7 nodes, let $\mathcal{N}, C_0, \alpha, \beta$ be presented in Table 1, and let node 7 be the output node of HNEP Γ.

A sentential form (a configuration) of grammar G is a word $w \in (N \cup T)^*$. When simulating the derivations in G, each sentential form w of G corresponds to a string of Γ in node 1 and having one of the forms $wB_{n,0}$ or $w''A_nw'B_{i,0}$, where $A_n = \#$, $w, w', w'' \in (N \cup T)^*$ and $w = w'A_iw''$. The start symbol $S = A_1$ of G corresponds to an initial

word $A_1\#$, represented as $A_1B_{n,0}$ in node 1 of HNEP Γ, the other nodes do not contain any word. The simulation of the application of a rule of G to a substring of a sentential form of G is done in several evolution and communication steps in Γ, through rewriting the leftmost symbol and the two rightmost or the rightmost symbol of strings. This is the reason why we need the symbols to be rotated.

In the following we describe how the rotation of a symbol and the application of an arbitrary rule of grammar G are simulated in HNEP Γ.

Rotation.

Let $A_{i_1}A_{i_2}\dots A_{i_{k-1}}B_{i_k,0} = A_{i_1}wB_{i_k,0}$ be a word found at node 1, and let $w, w', w'' \in A^*$. Then, by applying rule `1.1` we obtain $A_{i_1}A_{i_2}\dots A_{i_{k-1}}B_{i_k,0} = A_{i_1}wB_{i_k,0} \xrightarrow{1.1} \{C'_{i_1}wB_{i_k,0},\ A_{i_1}w'C'_{i_t}w''B_{i_k,0}\}$.

We note that during the simulation symbols C'_i should be transformed to ε', and this symbol can only be deleted from the left-hand end of the string (node 6). So, the replacement of C_{i_t} by its primed version in a string of the form $A_{i_1}w'C_{i_t}w''B_{i_k,0}$ results in a word that will stay in node 6 forever; thus, in the sequel, we will not consider strings with C'_i not in the leftmost position. In the communication step following the above evolution step, string $C'_{i_1}wB_{i_k,0}$ cannot leave node 1 and stays there for the next evolution step:

$$C'_{i_1}wB_{i_k,0} \xrightarrow{1.5} C'_{i_1}wB'_{i_k,0}.$$

Observe that rules `1.1` and `1.5` may be applied in any order. After then, string $C'_{i_1}wB'_{i_k,0}$ can leave node 1 and can enter only node 2. In the following steps of the computation, in nodes 1 and 2, the string is involved in evolution steps followed by communication:

$$C_{i_1-t}wB_{i_k,t} \xrightarrow{1.4} C'_{i_1-(t+1)}wB_{i_k,t} \xrightarrow{1.6} C'_{i_1-(t+1)}wB'_{i_k,t+1} \text{ (in node 1)},$$

$$C'_{i_1-t}wB'_{i_k,t} \xrightarrow{2.1} C_{i_1-(t+1)}wB'_{i_k,t} \xrightarrow{2.2} C_{i_1-(t+1)}wB_{i_k,t+1} \text{ (in node 2)}.$$

$N, \alpha, \beta, C_0,$	M, PI, FI, PO, FO
$1, *, (2),$ $\{A_1 B_{n,0}\}$	$\{\mathbf{1.1} : A_i \to C'_i \mid i \in I\} \cup \{\mathbf{1.2} : A_i \to \varepsilon' \mid i \in I',\ A_i \to \varepsilon\} \cup$ $\{\mathbf{1.3} : B_{j,0} \to B_{s,0} \mid A_j \to A_s,\ j, s \in I'\}$ $\{\mathbf{1.4} : C_i \to C'_{i-1},\ \mathbf{1.5} : B_{j,0} \to B'_{j,0},$ $\mathbf{1.6} : B_{j,k} \to B'_{j,k+1} \mid i \in I'',\ j \in I,\ k \in I'\} \cup$ $\{\mathbf{1.7} : C'_1 \to \varepsilon'\} \cup \{\mathbf{1.8} : E'_{j,k} \to E_{j,k-1},\ \mathbf{1.9} : D'_i \to D_{i+1},$ $\mathbf{1.10} : E'_{j,1} \to F_j \mid i \in I', j \in I, k \in I''\}$ $PI = \{A_n, B_{n,0}\} \cup C \cup E'$ $FI = C' \cup E \cup D \cup F \cup G \cup \{\varepsilon'\}$ $PO = C' \cup B' \cup D \cup F \cup \{\varepsilon'\}$ $FO = B \cup C \cup D' \cup E'$
$2, *, (2), \emptyset$	$\{\mathbf{2.1} : C'_i \to C_{i-1},\ \mathbf{2.2} : B'_{j,k} \to B_{j,k+1} \mid i \in I'',\ j \in I,\ k \in I'_0\} \cup$ $\{\mathbf{2.3} : C'_1 \to \varepsilon'\} \cup \{\mathbf{2.4} : E_{j,k} \to E'_{j,k-1},\ \mathbf{2.5} : D_i \to D'_{i+1},$ $\mathbf{2.6} : E_{j,1} \to F_j \mid i \in I'_0, j \in I, k \in I''\} \cup \{\mathbf{2.7} : A_n \to \varepsilon'\} \cup$ $\{\mathbf{2.8} : B_{j,0} \to A_j \mid A_j \in T\}$ $PI = \{B_{j,0} \mid A_j \in T\} \cup C' \cup E$ $FI = \{B \setminus B_{j,0} \mid A_j \in T\} \cup C \cup D' \cup E' \cup F \cup G \cup \{\varepsilon'\}$ $PO = C \cup D' \cup F \cup \{\varepsilon'\}$ $FO = \{B_{j,0} \mid A_j \in T\} \cup B' \cup C' \cup E \cup D$
$3, r, (2), \emptyset$	$\{\mathbf{3.1} : \varepsilon \to D_0\}$ $PI = B \setminus B_0 \cup B' \setminus B'_0 \cup G$ $FI = C \cup C' \cup B_0 \cup \{D_0\},\ PO = \{D_0\},\ FO = \emptyset$
$4, *, (2), \emptyset$	$\{\mathbf{4.1} : B_{j,k} \to E_{j,k},\ \mathbf{4.2} : B'_{j,k} \to E_{j,k} \mid j, k \in I\} \cup$ $\{\mathbf{4.3} : B_{j,k} \to E_{s,t},$ $\mathbf{4.4} : B'_{j,k} \to E_{s,t} \mid j, k, s, t \in I', A_j A_k \to A_s A_t\} \cup$ $\{\mathbf{4.5} : G_{j,k} \to E_{j,k} \mid j, k \in I'\}$ $PI = \{D_0\},\ FI = E,\ PO = E$ $FO = B \cup B' \cup G$
$5, *, (2), \emptyset$	$\{\mathbf{5.1} : D_j \to B_{j,0},\ \mathbf{5.2} : D'_j \to B_{j,0} \mid j \in I\} \cup$ $\{\mathbf{5.3} : F_j \to A_j \mid j \in I\} \cup \{\mathbf{5.4} : D_j \to G_{s,t},$ $\mathbf{5.5} : D'_j \to G_{s,t} \mid A_j \to A_s A_t, j, s, t \in I'\}$ $PI = D \setminus \{D_0\} \cup D'$ $FI = E \cup E' \cup \{D_0\} \cup C \cup C' \cup \{\varepsilon'\}$ $PO = \emptyset,\ FO = D \cup D' \cup F$
$6, l, (2), \emptyset$	$\{\mathbf{6.1} : \varepsilon' \to \varepsilon\}$ $PI = \{\varepsilon'\}$ $FI = B \setminus B_0 \cup B' \cup C \cup C' \cup F \cup (D \setminus \{D_0\})$ $PO = \emptyset,\ FO = \{\varepsilon'\}$
$7, *, (2), \emptyset$	$\emptyset$ $PI = T,\ FI = V \setminus T,\ PO = \emptyset,\ FO = T$

Table 1.

We note that during this phase of the computation rules `1.2`: $A_i \to \varepsilon'$ or `2.7`: $A_n \to \varepsilon'$ may be applied in nodes 1 and 2. In this case, the string leaves node 1 or 2, but cannot enter any node. So, this case will not be considered in the sequel.

The process continues in nodes 1 and 2 until subscript i of C_i or that of C'_i is decreased to 1. In this case, either rule `1.7` : $C_1 \to \varepsilon'$ in node 1 or rule `2.3` : $C'_1 \to \varepsilon'$ in node 2 will be applied and the obtained string $\varepsilon' w B'_{i_k,i_1}$ or $\varepsilon' w B_{i_k,i_1}$ is communicated to node 3. (Notice that the string is able to leave the node either if both C and B are primed or both of them are unprimed.) Then, in node 3, depending on the form of the string, either evolution step $\varepsilon' w B'_{i_k,i_1} \xrightarrow{3.1} \varepsilon' w B'_{i_k,i_1} D_0$ or evolution step $\varepsilon' w B_{i_k,i_1} \xrightarrow{3.1} \varepsilon' w B_{i_k,i_1} D_0$ is performed. Strings $\varepsilon' w B'_{i_k,i_1} D_0$ or $\varepsilon' w B_{i_k,i_1} D_0$ can enter only node 4, where (depending on the form of the string) either evolution step $\varepsilon' w B_{i_k,i_1} D_0 \xrightarrow{4.1} \varepsilon' w E_{i_k,i_1} D_0$ or evolution step $\varepsilon' w B'_{i_k,i_1} D_0 \xrightarrow{4.2} \varepsilon' w E_{i_k,i_1} D_0$ follows. The obtained word, $\varepsilon' w E_{i_k,i_1} D_0$, can enter only node 6, where evolution step $\varepsilon' w E_{i_k,i_1} D_0 \xrightarrow{6.1} w E_{i_k,i_1} D_0$ is performed. Then the string leaves the node and enters node 2.

Then, in nodes 2 and 1, a sequence of computation steps is performed, when the string is involved in evolution steps followed by communication as follows:

$$w E_{i_k,i_1-t} D_t \xrightarrow{2.4} w E'_{i_k,i_1-(t+1)} D_t \xrightarrow{2.5} w E'_{i_k,i_1-(t+1)} D'_{t+1} \text{ (in node 2).}$$

$$w E'_{i_k,i_1-t} D'_t \xrightarrow{1.8} w E_{i_k,i_1-(t+1)} D'_t \xrightarrow{1.9} w E_{i_k,i_1-(t+1)} D_{t+1} \text{ (in node 1),}$$

The process continues in nodes 1 and 2 until the second subscript of $E'_{i,j}$ or that of $E_{i,j}$ is decreased to 1. In this case, either rule `1.10` : $E'_{i_k,1} \to F_{i_k}$ in node 1 or rule `2.6` : $E_{i_k,1} \to F_{i_k}$ in node 2 is applied and the new string, $w F_{i_k} D_{i_1}$ or $w F_{i_k} D'_{i_1}$, will be present in node 5. Notice that applying rules `1.1, 1.2` and `2.7` we obtain strings that cannot enter nodes 3 – 7 and stay in nodes 1 or 2.

The next evolution steps that take place in node 5 are as follows:

$$w F_{i_k} D_{i_1} (w F_{i_k} D'_{i_1}) \xrightarrow{5.1(5.2)} w F_{i_k} B_{i_1,0} \xrightarrow{5.3} w A_{i_k} B_{i_1,0}.$$

In the following communication step, string $wA_{i_k}B_{i_1,0}$ can enter either node 1 or node 2 (if $A_{i_1} \in T$). In the first case, the rotation of symbol A_{i_1} has been successful. Let us consider the second case. Then string $wA_{i_k}B_{i_1,0}$ appears in node 2.

- Suppose that the word $wA_{i_k}B_{i_1,0}$ does not contain any nonterminal symbol except A_n. Let $wA_{i_k}B_{i_1,0} = A_n w' A_{i_k} B_{i_1,0}$, where $w = A_n w'$. So, $w' A_{i_k} A_{i_1}$ is a result and it appears in node 7. Notice that if $w = w' A_n w''$ and $w' \neq \varepsilon$, then word $w' A_n w'' A_{i_k} B_{i_1,0}$ leads to a string which will stay in node 6 forever (if rule `2.7` was applied). So, we consider the following evolution of the word $wA_{i_k}B_{i_1,0} = A_n w' A_{i_k} B_{i_1,0}$: $A_n w' A_{i_k} B_{i_1,0} \xrightarrow{2.7} \varepsilon' w' A_{i_k} B_{i_1,0} \xrightarrow{2.8} \varepsilon' w' A_{i_k} A_{i_1}$. Then, string $\varepsilon' w' A_{i_k} A_{i_1}$ will appear in node 6, where symbol ε' will be deleted by rule `6.1`. Finally, the resulted word $w' A_{i_k} A_{i_1}$ will enter node 7. This is a result.

- Suppose now that the word $wA_{i_k}B_{i_1,0}$ contains at least one nonterminal symbol different from A_n and $A_{i_1} \in T$.

  Consider the evolution of the word $wA_{i_k}B_{i_1,0} = w' A_n w'' A_{i_k} B_{i_1,0}$ in node 2:

  $$w' A_n w'' A_{i_k} B_{i_1,0} \xrightarrow{2.8} w' A_n w'' A_{i_k} A_{i_1} \xrightarrow{2.7} w' \varepsilon' w'' A_{i_k} A_{i_1}.$$

  Now, string $w' \varepsilon' w'' A_{i_k} A_{i_1}$ will enter node 6 and either it will not be able to leave it (if $w' \neq \varepsilon$) or it will not be able to enter any of the other nodes (if $w' = \varepsilon$).

In the following we will explain how the application of the rules of G are simulated in Γ.

Rule $A_i \longrightarrow \varepsilon$**.** Suppose that string $A_i w B_{j,0}$ is in node 1 and let $w, w', w'' \in A^*$. Then, by evolution, we obtain $A_i w B_{j,0} \xrightarrow{1.2} \varepsilon' w B_{j,0}$ or $A_t w' A_i w'' B_{j,0} \xrightarrow{1.2} A_t w' \varepsilon' w'' B_{j,0}$ which can enter only node 6. String $A_t w' \varepsilon' w'' B_{j,0}$ will stay in node 6 forever. By evolution $\varepsilon' w B_{j,0} \xrightarrow{6.1} w B_{j,0}$ and the resulting string, $w B_{j,0}$, enters in node 1 (and node 2, if $A_j \in T$). Thus, the application of rule $A_i \longrightarrow \varepsilon$ in G was correctly simulated.

Rule $A_i \longrightarrow A_j$**.** The evolution step performed at node 1 is $wB_{i,0} \stackrel{1.3}{\longrightarrow} wB_{j,0}$. Since string $wB_{j,0}$ is in node 1, the simulation of the rule $A_i \longrightarrow A_j$ of grammar G was done in a correct manner.

Rule $A_j \longrightarrow A_sA_t$**.** At the end of the simulation of the rotation of a symbol, in node 5 instead of applying rule $D_j \rightarrow B_{j,0}$ ($D'_j \rightarrow B_{j,0}$) a rule $D_j \rightarrow G_{s,t}$ ($D'_j \rightarrow G_{s,t}$) is applied. That is, in node 5, either evolution step $wD_j \stackrel{5.4}{\longrightarrow} wG_{s,t}$ or evolution step $wD'_j \stackrel{5.5}{\longrightarrow} wG_{s,t}$ is performed. The new string $wG_{s,t}$ can enter only node 3, where, by evolution, $wG_{s,t} \stackrel{3.1}{\longrightarrow} wG_{s,t}D_0$. String $wG_{s,t}D_0$ can enter only node 4, where evolution step $wG_{s,t}D_0 \stackrel{4.5}{\longrightarrow} wE_{s,t}D_0$ follows. The process continues as above, in the case of simulating rotation, and in several computation steps the string wF_sD_t or $wF_sD'_t$ will enter node 5. After evolution in this node, the resulting string $wA_sB_{t,0}$ will enter node 1 (and node 2, if $A_t \in T$). Thus, the application of rule $A_j \longrightarrow A_sA_t$ of G is correctly simulated.

Rule $A_iA_j \longrightarrow A_sA_t$**.** The evolutionary processor in node 4 has rules $\texttt{4.3} : B_{i,j} \rightarrow E_{s,t}$ or $\texttt{4.4} : B'_{i,j} \rightarrow E_{s,t}$. As in the case of simulating rotation, above, we will obtain string $wA_sB_{t,0}$ in node 1 (and in node 2, if $A_t \in T$).

We have demonstrated how the rotation of a symbol and the application of rules of G are simulated by Γ. By the constructions, the reader can verify that G and Γ generate the same language. □

3.4 Obligatory HNEPs

As described in the second section, obligatory operations were considered in HNEPs. The result of the evolution step now consists of all strings produced from the current ones by the operations of insertion, deletion and substitution (the current strings are no longer preserved, even if some operation is not applicable to them). Not only this yields a simpler and a more uniform definition, but also the following result is obtained.

Theorem 4 *Any CPM0 P can be simulated by an OHNEP P', where*

obligatory evolutionary processors are with empty input and output filters and only insertion and obligatory deletion operations in right and left modes are used (without obligatory substitution operations). [3, 2]

Proof. Let us consider a CPM0 P with symbols $a_j \in \Sigma,\ j \in J = \{0, 1 \ldots, n\}$, $a_0 = 0$ is the blank symbol, and states, $q_i \in Q,\ i \in I = \{1, 2, \ldots, f\}$, where q_1 is the initial state and the only terminal state is $q_f \in Q$. We suppose that P stops in the terminal state q_f on every symbol, i.e., there are instructions $q_f a_j \to Halt,\ a_j \in J$. (Notice that it is easy to transform any CPM0 P into a CPM0 P' that stops on every symbol in the final state.)

So, we consider CPM0 P with the set R of instructions of the forms $q_i a_j \longrightarrow q_l$, $q_i a_j \longrightarrow a_k q_l$, $q_i 0 \longrightarrow a_k q_m 0$, $q_f a_j \longrightarrow Halt$, where $q_i \in Q \setminus \{q_f\},\ q_l, q_m \in Q,\ a_j, a_k \in \Sigma$. We do not consider case $q_m = q_f$ in instruction $q_i 0 \longrightarrow a_k q_m 0$. Notice that it is easy to modify the program of P such that it only halts by instructions of other types.

A configuration $w = q_i a_j W$ of CPM0 P describes that P in state $q_i \in Q$ considers symbol $a_j \in \Sigma$ to the left of $W \in \Sigma^*$.

Now we construct an OHNEP P' simulating P. To simplify the description of P', we use $\langle q_f a_j \rangle$ and $\langle q_f a_j \rangle_1,\ j \in J$ as aliases of $\langle out \rangle$. Let v take values from Q and let u take values from $\in Q \cup Q \cdot \{0\}$.

$$
\begin{aligned}
P' &= (V, G, N, C_0, \alpha, \beta, i_0), \\
V &= \{q_1\} \cup \Sigma, \\
G &= (X_G, E_G), \\
X_G &= \{\langle init \rangle, \langle out \rangle\} \\
&\cup \{\langle q_i a_j \rangle \mid (q_i a_j \to v) \in R\} \\
&\cup \{\langle q_i a_j \rangle_1 \mid (q_i a_j \to a_k u) \in R\}, \\
E_G &= \{(\langle init \rangle, \langle q_1 a_j \rangle) \mid j \in J\} \\
&\cup \{(\langle q_i a_j \rangle, \langle q_l a_k \rangle) \mid (q_i a_j \to q_l) \in R,\ k \in J\} \\
&\cup \{(\langle q_i a_j \rangle, \langle q_i a_j \rangle_1) \mid (q_i a_j \to a_k u) \in R\} \\
&\cup \{(\langle q_i a_j \rangle_1, \langle q_l a_s \rangle) \mid (q_i a_j \to a_k q_l) \in R,\ s \in J\} \\
&\cup \{(\langle q_i 0 \rangle_1, \langle q_l 0 \rangle_1) \mid (q_i 0 \to a_k q_l 0), (q_l 0 \to a_s u) \in R, s \in J\},
\end{aligned}
$$

$$\cup \quad \{(\langle q_i 0\rangle_1, \langle q_p a_s\rangle) \mid (q_i 0 \to a_k q_l 0), (q_l 0 \to q_p) \in R, s \in J\},$$

$$\begin{aligned}
C_0(x) &= \{q_1 W\}, \text{ if } x = \langle init\rangle, \\
&\quad \text{where } W \text{ is the input of } P, \\
C_0(x) &= \emptyset, \ x \in X_G \setminus \{\langle init\rangle\}, \\
\beta(x) &= 2, \ x \in X_G, \\
N(x) &= (M_x, \emptyset, \emptyset, \emptyset, \emptyset), \ x \in X_G, \\
M_x &= \{q_1 \to \varepsilon\}, x = \langle init\rangle, \\
M_x &= \{a_j \to \varepsilon\}, x = \langle q_i a_j\rangle, \\
M_x &= \{\varepsilon \to a_k\}, x = \langle q_i a_j\rangle_1, \\
&\quad \text{where } (q_i a_j \to a_k u) \in R, \\
\alpha(x) &= l, \text{ if } M_x = \{a \to \varepsilon\}, \\
\alpha(x) &= r, \text{ if } M_x = \{\varepsilon \to a\}, \text{ or } M_x = \emptyset.
\end{aligned}$$

OHNEP P' will simulate every computation step performed by CPM0 P by a sequence of computation steps in P'.

Let $q_1 a_j W_0$ be the initial configuration of CPM0 P. We represent this configuration in node $\langle init\rangle$ of OHNEP P' as a word $q_1 a_j W_0$. Obligatory evolutionary processor associated with this node is $N(\langle init\rangle) = (\{(q_1 \to \varepsilon)^l\}, \emptyset, \emptyset, \emptyset, \emptyset)$. Since all other nodes also have empty filters, in the following we will skip the complete description of obligatory evolutionary processors, and will present only their obligatory evolutionary operations. The word $a_j W_0$ will be passed from node $\langle init\rangle$ to nodes $\langle q_1 a_j\rangle$, $j \in J$.

If the computation in P is finite, then the final configuration $q_f W$ of P will appear at node $\langle out\rangle$ of P' as a string W, moreover, any string W that can appear at node $\langle out\rangle$ corresponds to a final configuration $q_f W$ of P. In the case of an infinite computation in P, no string will appear in node $\langle out\rangle$ of P' and the computation in P' will never stop.

Now we describe nodes of OHNEP P', connections between them and obligatory evolutionary operations, associated with these nodes. Let $I' = I \setminus \{f\}$.

1. Node $\langle q_i a_j \rangle$ with operation $(a_j \to \varepsilon)^l,\ i \in I',\ j \in J$.
   Let word $a_t W,\ t \in J,\ W \in \Sigma^*$ appear in this node. If $j \neq t$, then this word $a_t W$ will be discarded, and in the next communication step node $\langle q_i a_j \rangle$ will send nothing. If $j = t$, then the node sends W to nodes $\{\langle q_l a_k \rangle \mid k \in J\}$ or $\langle q_i a_j \rangle_1$.

   - Instruction of P is $q_i a_j \longrightarrow q_l,\ i \in I',\ j \in J,\ l \in I$. Node $\langle q_i a_j \rangle$ is connected with nodes $\{\langle q_l a_k \rangle \mid k \in J\}$.
   - Instructions of P are $q_i a_j \longrightarrow a_k q_l$ or $q_i 0 \longrightarrow a_k q_l 0,\ i \in I',\ j,k \in J,\ l \in I$. Node $\langle q_i a_j \rangle$ is connected with node $\langle q_i a_j \rangle_1$.

2. Node $\langle q_i a_j \rangle_1,\ i \in I',\ j \in J$ with operation $(\varepsilon \to a_k)^r$ receives word W and sends word $W a_k$ to nodes $\{\langle q_l a_s \rangle \mid s \in J\}$ or $\langle q_l 0 \rangle_1$.

   - Instructions of P are $q_i a_j \longrightarrow a_k q_l,\ i \in I',\ j,k \in J,\ l \in I$. Node $\langle q_i a_j \rangle_1$ is connected with nodes $\{\langle q_l a_s \rangle \mid s \in J\}$.
   - Instruction of P is $q_i 0 \longrightarrow a_k q_l 0,\ i \in I',\ k \in J,\ l \in I$. Node $\langle q_i 0 \rangle_1$ is connected with nodes $\{\langle q_p a_s \rangle \mid s \in J\}$ if there exists an instruction of P $q_l 0 \longrightarrow q_p$, $p \in I$; and with node $\langle q_l 0 \rangle_1$ in other cases.

We repeat that in all cases, we mean $\langle out \rangle$ whenever we write $\langle q_f a_j \rangle$ or $\langle q_f a_j \rangle_1$, $j \in J$.

Now we describe simulation of instructions of CPM0 P by OHNEP P'.

Instruction $q_i a_j \longrightarrow q_l$: $q_i a_j W \xrightarrow{P} q_l W$.

Let word $a_t W$, where $t \in J,\ W \in \Sigma^*,\ i \in I'$ appear in node $\langle q_i a_j \rangle$. If $t \neq j$, string $a_t W$ will be discarded; if $t = j$, string W will be passed to nodes $\{\langle q_l a_s \rangle \mid s \in J\}$. If $l = f$, the final configuration $q_f W$ of P will appear in the output node $\langle out \rangle$ as W. This is the result. So, we simulated instruction $q_i a_j \longrightarrow q_l$ in a correct manner.

Instruction $q_i a_j \longrightarrow a_k q_l$: $q_i a_j W \xrightarrow{P} q_l W a_k$.

Let word $a_t W$, where $t \in J,\ W \in \Sigma^*,\ i \in I'$ appear in node $\langle q_i a_j \rangle$. If $t \neq j$, string $a_t W$ will be discarded; if $t = j$ string W will be passed

to node $\langle q_i a_j \rangle_1$. Node $\langle q_i a_j \rangle_1$ receives this word and sends word Wa_k to nodes $\langle q_l a_s \rangle$, $s \in J$. If $l = f$, the final configuration $q_f W a_k$ of P will appear in the output node $\langle out \rangle$ as Wa_k. This is the result. So, we simulated instruction $q_i a_j \longrightarrow a_k q_l$ in a correct manner.

Instruction $q_i 0 \longrightarrow a_k q_l 0$: $q_i 0W \xrightarrow{P} q_l 0 W a_k$.

Let word $a_t W$, where $t \in J$, $W \in \Sigma^*$, $i \in I'$ appear in node $\langle q_i 0 \rangle$. If $a_t \neq 0$, string $a_t W$ will be discarded; if $a_t = 0$, string W will be passed to node $\langle q_i 0 \rangle_1$. It receives this word and sends word Wa_k to nodes $\langle q_p a_s \rangle, s \in J$ if there is instruction of P $q_l 0 \longrightarrow q_p, p \in I'$. If there are instructions $q_l 0 \longrightarrow a_s q_p$ or $q_l 0 \longrightarrow a_s q_p 0$, then node $\langle q_i 0 \rangle_1$ is connected with node $\langle q_l 0 \rangle_1$. Thus, word Wa_k will be passed to node $\langle q_l 0 \rangle_1$, which corresponds to the configuration of P which has "just read" symbol 0 in state q_l. So, we simulated instruction $q_i 0 \longrightarrow a_k q_l 0$ in a correct manner.

So, CPM0 P is correctly modeled. We have demonstrated that the rules of P are simulated in P'. The proof that P' simulates only P comes from the construction of the rules in P', we leave the details to the reader. □

4 Conclusion

We have described the networks of evolutionary processors, their models and variants, together with the associated results. A few selected results were presented in more details. For instance, NEPs with two nodes are already computationally complete modulo the terminal alphabet. HNEPs with one node are given the precise regular characterization, HNEPs with two nodes are not computationally complete, while seven nodes are enough to reach the computational completeness of HNEPs, even with a complete graph. We should mention that a network over a complete graph (with loops, although it is not important for the last proof) may be viewed as number of agents in a common environment, acting "independently" without explicitly enforcing any

transition protocol, where a computationally complete behavior still emerges.

A particularly interesting variant is obligatory HNEPs (OHNEPs). Using a power of the underlying graph, computational completeness is obtained even without the filters. In case of a complete graph, OHNEPs are still computationally complete. Moreover, it suffices that the sum of numbers of symbols in filters of each node does not exceed one. The last proof has been obtained in [1], using a variant of circular Post machines, CPM5, introduced in [11].

References

[1] A. Alhazov, G. Bel-Enguix, I. Epifanova, Yu. Rogozhin. *About Two Models of Complete Obligatory Hybrid Networks of Evolutionary Processors.* In preparation.

[2] A. Alhazov, G. Bel-Enguix, Yu. Rogozhin. *About a New Variant of HNEPs: Obligatory Hybrid Networks of Evolutionary Processors.* In: G. Bel-Enguix, M. D. Jiménez-López (Eds), Bio-Inspired Models for Natural and Formal Languages, Cambridge Scholars Publishing, 2011, pp.191–204.

[3] A. Alhazov, G. Bel-Enguix, Yu. Rogozhin. *Obligatory Hybrid Networks of Evolutionary Processors.* International Conference on Agents and Artificial Intelligence, Porto, 2009, INSTICC Press, pp.613–618.

[4] A. Alhazov, G. Bel-Enguix, A. Krassovitskiy, Yu. Rogozhin. *About Complete Obligatory Hybrid Networks of Evolutionary Processors without Substitution.* Advances in Computational Intelligence, 11th International Work-Conference on Artificial Neural Networks, IWANN 2011, Málaga, Lecture Notes in Computer Science, **6691**, 2011, pp.441–448.

[5] A. Alhazov, G. Bel-Enguix, A. Krassovitskiy, Yu. Rogozhin. *Complete Obligatory Hybrid Networks of Evolutionary Processors.*

Highlights in Practical Applications of Agents and Multiagent Systems, Salamanca, Advances in Intelligent and Soft Computing, **89**, 2011, pp.275–282.

[6] A. Alhazov, E. Csuhaj-Varjú, C. Martín-Vide, Yu. Rogozhin. *About Universal Hybrid Networks of Evolutionary Processors of Small Size.* Pre-Proceedings of the 2nd International Conference on Language and Automata Theory and Applications, LATA 2008, GRLMC report, **36/08**, University Rovira i Virgili, Tarragona, 2008, pp.43–54. Also in: Lecture Notes in Computer Science, **5196**, Springer, 2008, pp.28–39.

[7] A. Alhazov, E. Csuhaj-Varjú, C. Martín-Vide, Yu. Rogozhin. *Computational Completeness of Hybrid Networks of Evolutionary Processors with Seven Nodes.* In: C. Campeanu, G. Pighizzini (Eds.), Descriptional Complexity of Formal Systems, DCFS 2008 Proceedings, University of Prince Edward Island, Charlottetown, 2008, pp.38–47.

[8] A. Alhazov, E. Csuhaj-Varjú, C. Martín-Vide, Yu. Rogozhin. *On the Size of Computationally Complete Hybrid Networks of Evolutionary Processors.* Theoretical Computer Science, 410, 35, 2009, pp.3188–3197.

[9] A. Alhazov, J. Dassow, C. Martín-Vide, Yu. Rogozhin, B. Truthe. *On Networks of Evolutionary Processors with Nodes of Two Types.* Fundamenta Informaticae, **91**, 1, 2009, pp.1–15.

[10] A. Alhazov, M. Kudlek, Yu. Rogozhin. *Nine Universal Circular Post Machines.* Computer Science Journal of Moldova, **10(3)**, 2002, pp.247–262.

[11] A. Alhazov, A.Krassovitsiy, Yu.Rogozhin. *Circular Post Machines and P Systems with Exo-insertion and Deletion.* Lecture Notes in Computer Science, **7184**, Springer, 2012, pp.73–86.

[12] A. Alhazov, C. Martín-Vide, Yu. Rogozhin. *On the Number of Nodes in Universal Networks of Evolutionary Processors.* Acta Informatica, **43(5)**, 2006, pp.331–339.

[13] A. Alhazov, C. Martín-Vide, Yu. Rogozhin. *Networks of Evolutionary Processors with Two Nodes Are Unpredictable.* Pre-Proceedings of the 1st International Conference on Language and Automata Theory and Applications, LATA 2007, GRLMC report, **35/07**, Rovira i Virgili University, Tarragona, 2007, pp.521–528. Also in: Technical Report, **818**, Turku Centre for Computer Science, Turku, 2007.

[14] A. Alhazov, Yu. Rogozhin. *About Precise Characterization of Languages Generated by Hybrid Networks of Evolutionary Processors with One Node.* The Computer Science Journal of Moldova, **16(3)**, 2008, pp.364–376.

[15] J. Castellanos, P. Leupold, V. Mitrana. *On the Size Complexity of Hybrid Networks of Evolutionary Processors.* Theoretical Computer Science, **330(2)**, 2005, pp.205–220.

[16] J. Castellanos, C. Martín-Vide, V. Mitrana, J. Sempere. *Networks of Evolutionary processors.* Acta Informatica, **38**, 2003, pp.517-529.

[17] J. Castellanos, C. Martín-Vide, V. Mitrana, J. Sempere. *Solving NP-complete Problems with Networks of Evolutionary Processors.* In: J. Mira, A. Prieto (Eds.), IWANN 2001, Lecture Notes in Computer Science, **2084**, Springer, 2001, pp.621–628.

[18] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana. *Hybrid Networks of Evolutionary Processors are Computationally Complete.* Acta Informatica, **41(4-5)**, 2005, 257–272.

[19] E. Csuhaj-Varjú, A. Salomaa. *Networks of Parallel Language Processors.* In: Gh. Păun, A. Salomaa, (Eds.), New Trends in formal Language Theory Lecture Notes in Computer Science, **1218**, Springer, 1997, pp.299-318.

[20] J. Dassow, B. Truthe. *On the Power of Networks of Evolutionary Processors.* In: J. O. Durand-Lose, M. Margenstern (Eds.), MCU 2007, Lecture Notes in Computer Science, **4667**, Springer, 2007, pp.158–169.

[21] M. Kudlek, Yu. Rogozhin. *Small Universal Circular Post Machines.* Computer Science Journal of Moldova, **9(1)**, 2001, pp.34–52.

[22] M. Kudlek, Yu. Rogozhin. *New Small Universal Circular Post Machines.* In: R. Freivalds (Ed.), Proc. FCT 2001, Lecture Notes in Computer Science, **2138**, Springer, 2001, pp.217–227.

[23] R. Loos, F. Manea, V. Mitrana. *Small Universal Accepting Hybrid Networks of Evolutionary Processors.* Acta Informatica, **47**, 2, 2010, pp.133–146.

[24] F. Manea, C. Martín-Vide, V. Mitrana. *On the Size Complexity of Universal Accepting Hybrid Networks of Evolutionary Processors.* Mathematical Structures in Computer Science, **17(4)** 2007, pp.753–771.

[25] F. Manea, C. Martín-Vide, V. Mitrana. *All NP-problems can be Solved in Polynomial Time by Accepting Hybrid Networks of Evolutionary Processors of Constant Size.* Information Processing Letters, **103**, 2007, pp.112–118.

[26] M. Margenstern, V. Mitrana, M.-J. Pérez-Jiménez. *Accepting Hybrid Networks of Evolutionary Processors.* in: C. Ferretti, G. Mauri, C. Zandron (Eds.), DNA 10, Lecture Notes in Computer Science, **3384**, Springer, 2005, pp.235–246.

[27] C. Martín-Vide, V. Mitrana, M. Pérez-Jiménez, F. Sancho-Caparrini. *Hybrid Networks of Evolutionary Processors.* In: E. Cantú-Paz et al. (Eds.), Proc. of GECCO 2003, Lecture Notes in Computer Science, **2723**, Springer, 2003, pp.401-412.

[28] A. Salomaa. *Formal Languages.* Academic Press, New York, 1973.

Artiom Alhazov

Received April 9, 2013

Dr. Artiom Alhazov
Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova
E–mail: artiom@math.md

Computer Science Journal of Moldova, vol.**21**, no.1(61), 2013

Towards an Automated Semiotic Analysis of the Romanian Political Discourse*

Daniela Gîfu, Dan Cristea

Abstract

As it is known, on the political scene the success of a speech can be measured by the degree in which the speaker is able to change attitudes, opinions, feelings and political beliefs in his auditorium. We suggest a range of analysis tools, all belonging to semiotics, from lexical-semantic, to syntactical and rhetorical, that integrated in the exploratory panoply of discursive weapons of a political speaker could influence the impact of her/his speeches over a sensible auditory. Our approach is based on the assumption that semiotics, in its quality of methodology and meta-language, can capitalize a situational analysis over the political discourse. Such an analysis assumes establishing the communication situation, in our case, the Parliament's vote in favour of suspending the Romanian President, through which we can describe an action of communication.

We depict a platform, the Discourse Analysis Tool (DAT), which integrates a range of natural language processing tools with the intent to identify significant characteristics of the political discourse. The tool is able to produce comparative diagrams between the speeches of two or more subjects or analysing the same subject in different contexts. Only the lexical-semantic methods are operational in the platform today, but our investigation suggests new dimensions touching the syntactic, rhetorical and coherence perspective.

Keywords: political discourse, natural language processing, president's suspension, lexical-semantic, syntax, rhetorical analysis, coherence of discourse.

* This work was supported by the POSDRU/89/1.5/S/63663 grant, and the ICT-PSP projects METANET4U #270893 and ATLAS #250467.

1 Introduction

One of the major recent developments in the evaluation of the political language and its related facets (rhetoric, political science, journalism, sociology, etc.) is the increasing attention being paid to the objectivity and relevance of the semiotic dimensions.

Theoretical approaches in the semiotics of discourses, involving pragmatic aspects (the dynamics of relations between individuals and signs), semantic (conceptual conglomerate met in the meanings of terms), and syntactic (relations between signs) showed a significant strengthening after the '80s. The current approaches in analysing the political language (the applicative dimension) are based on Natural Language Processing (NLP) techniques designed to investigate lexical-semantic aspects of the discourse. The domain of NLP includes a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like proficiency in the interpretation of language for a range of tasks or applications [12].

In this paper we start by describing a platform, the Discourse Analysis Tool (DAT), which integrates a range of language processing tools with the intent to build complex characterisations of the political discourse and show how its functionality can be prolonged with more complex features. A linguistic profile of an author is drawn by putting together features extracted from the following linguistic layers: lexicon and morphology (richness of the vocabulary, rare co-occurrences, repetitions, use of synonyms, coverage of verbs' grammatical tenses, etc.), semantics (semantic classes used) and syntax (complexity of syntactic constructions, the frequency of relative clauses, length of the sentences, number of clauses in sentences, subordinate/coordinate structures, frequent use of certain type of syntactic relations, etc.).

Among the resources used for the study of natural language syntax, of a tremendous importance are the treebanks, large collections of sentences annotated by human experts at syntactic structures. The collection described in this paper refers to the Romanian language and has been acquired with the help of an interactive graphical tool which

allowed easy annotation, visualisation and modification of syntactic trees, initially obtained as a result of an automatic parsing process.

Our purpose was to develop a computational platform able to offer to researchers in mass-media and political sciences, to political analysts, to the public at large (interested to consolidate their options before any political context analysed), and, why not, even to politicians themselves, the possibility to measure different parameters of a written political discourse.

The paper is structured as follows. Section 2 shortly describes the previous work. Section 3 discusses a number of lexical-semantic, syntactic, rhetorical and pragmatic features on which an automatic analysis is capable to manipulate values in view of drawing statistics. Section 4 presents a platform for multi-dimensional political discourse analysis. Section 5 discusses an example of comparative analysis of discourses collected during the presidential crisis of July 2012, when the Parliament voted in favour of suspending the Romanian President. Finally, section 6 highlights interpretations anchored in our analysis and presents conclusions.

2 Previous work

The aim of an interdisciplinary approach such as analysing the language of political speeches is to define and explain different discursive contexts, in this case, reflected by the online media. The studies in this direction have mainly concentrated on three tasks: the first had to do with a cognitive side and, often, with an emotional side, of how humans acquire, produce, and understand language. The second aimed at understanding the relationship between the linguistic utterance and the world, and the third – at understanding the linguistic structure of the language as a communication device. Linguistics has usually treated language as an abstract object which can be accounted for without reference to social or political concerns of any kind [19].

As we will see, one aspect of the platform that we present touches a lexical-semantic functionality, which has some similarities with the approach used in Linguistic Inquiry and Word Count (LIWC) [16]. There

are, however, important differences between the two platforms. LIWC-2007 is basically counting words and incrementing counters associated with their declared semantic classes. In the lexicon, words can be given by their long form, as a complete string of characters, or by their roots. For each text in the input, LIWC produces a set of tables, each displaying the occurrences of the word-like instances of the semantic classes defined in the lexicon, as sub-unitary values. For each semantic class, such a value is computed as the number of occurrences of the words corresponding to that class divided by the total number of words in the text. It remains in the hands of the user to interpret these figures. Also, LIWC has no support for considering lexical expressions.

A previous version of DAT [8] performs part-of-speech (POS) tagging and lemmatization of words. The lexicon contains a collection of lemmas (9.000) having the POS categories: verb, noun, adjective and adverb. In the context of the lexical semantic analysis, the pronouns, numerals, prepositions and conjunctions, considered to be semantically empty, have been left out. In contrast with LIWC-2007, which includes 64 semantic classes (classified into 4 categories: linguistic – 22 classes, psychological – 32 classes, socio-professional preoccupations – 7 classes and paralinguistic – 3 classes), DAT.v3 works with 33 semantic classes, out of which 5 are newly introduced, chosen to fit optimally with the necessities of interpreting the political discourse.

The second range of differences between the two platforms regards the user interface. In DAT, the user is served by a friendly interface, offering a lot more services: opening one or more files, displaying the file(s), modifying/editing and saving the text, functions of undo/redo, functions to edit the lexicon, visualization of the mentioning of instances of certain semantic classes in the text, etc. Then, the menus offer a whole range of output visualization functions, from tabular form to graphical representations and to printing services. Finally, another important development for semantic approach was the inclusion of a collection of formulas which can be used to make comparative studies between different subjects. The lexicon entries are coded in XML, following one of the patterns: `<word stem="`*wordStem*`" classes="`*semList*`">`, or `<word`

`lemma="`*wordLemma*`" classes="`*semList*`">`, in which *wordStem* is the stem of a word (therefore symbols optionally followed by the '*' sign), *wordLemma* is the lemma of a word, and *semList* is a list of semantic classes (each indicated by a unique identifier). The following line shows such an example of lexical entry:

```
<word lemma="deportare" classes="30,11"/>
<word stem="conspiraţioni*" classes="30,1,5,10"/>
```

3 Semiotic features of political discourse

As meta-language, the semiotics explain the evolution of different types of object-languages, from physical to linguistic (among those – the political discourse). It helps to understand the way the humans apply these systems with the intend to "designate states of possible worlds or to criticize and even change the structure of systems" [6].

The three analytical horizons are: structural analysis of the levels / hierarchical relations of (macro)sign, the triadic analysis (syntactic, semantic, pragmatic), and the analysis of the communication situation taken for investigation. In the following we will focus on one of the three horizons of analysis assumed by the semiotic methodology, namely, on the triadic analysis. Conforming to this view, any text/discourse can be analysed from three perspectives [15]: syntactic (the relation between signs), semantic (the relation between signs and reference), and pragmatic (the relation between participants in the communication). Highlighting methodological operations presumed by such a perspective offers as many (re)signifying strategies of political contexts.

We will adopt analytical techniques developed by the NLP field to a semiotic study over political texts, in the classical sense [17], that go back to the methodology proposed by Ferdinand de Saussure [20], in order to show that the results can be significantly comparable and, therefore, there are good reasons to trust the computational techniques.

3.1 The lexical-semantic perspective

A lexical-semantic perspective is supposed to focus on the following targets:

1. establishing meanings that a political speech includes, as a whole or at the level of its content units (negative/positive, affirmative/adversative, etc.); determining the correlation degree (motivation) between the orientation of the political speech and the language (code) used (adequate, partially adequate, inadequate);

2. a qualitative-semantic analysis of content units, that could be operated on two dimensions: denotative (what is said explicitly about the topic discussed), focusing on the intelligibility of the political text, by assessing its lexical-semantic connectedness [18], or by counting the originality, oddity or banality of the used lexicon, as well as the phrase length, the number of subordinate sentences, parentheses, etc.; connotative (what are the side suggestions, the sayings in-between the lines, the symbolistics of the language used), aiming to highlight the possible hidden semantic meanings of a speech and determining the most likely ones by taking into account all circumstantial factors (situational), and specifying the gap between the explicit and implicit intentions expressed;

3. a quantitative-semantic analysis focusing on determining of the frequency of key concepts encountered in the political text, highlighting the frequency of certain themes in the speech, identifying the frequency of emotionally charged terms, etc.; building a dictionary of symbols (for key-concepts) specific to the political discourse that helps to frame it in terms of semantic categories.

4. a discourse and para-language analysis considering the identification of the rhetorical aspects of the verbal language (spectacular, suggestive, allusive, emotional, metaphoric factors, etc.), and the characteristics of the nonverbal language which have a significant weight in the political discourse.

The political speaker is determined to collect empathy and to convince the public. Yet, placing himself within the general limits of the political goals, very often a skilful politician studies the public for fixing the type of vocabulary and the message to be delivered. He might exploit connections between more daring ideological categories (as is for instance the class `nationalism`) and those generally accepted (for instance, belonging to the classes `social`, `work`, `home`). The present day political language puts in value the virtues of the metaphor, its qualities to pass abruptly from complex to simple, from abstract to concrete, imposing a powerful subjective and emotional dimension to the discourse (the class `emotional`). The political metaphor may loose the virtues of poetical metaphor, becoming injurious (the class `swear`).

3.2 The syntactic perspective

Regarded as one of the most developed branches of semiotics, syntactic analysis aims at studying the relations between signs and the logical and grammatical structure at the sentence level [13]. The sentence is composed out of an ordered sequence of language signs, which are governed by a set of combinatorial rules.

From this perspective, the syntactic analysis of a text aims at: segmenting the text onto information units (sentences, clauses, phrases, words and punctuation markers), identifying the constituency structure of the sentence (recurrent levels of constituency), emphasising the dependency structure of a sentence (putting in evidence the unique syntactic head of each word and the relation linking it to its head in a tree-like dependency structure [21], etc. The syntax may reveal the level of culture, intentional persuasive attitudes towards the public, iritation or rude passion during the production of speech, etc.

Then, a combination of syntactic and semantic means of investigation could bring forward the semantic verbal roles in sentences (see, FrameNet [2]), as well as the balance between given and new or rheme and theme [10].

The final goal of a combined syntactic-semantic analysis is the inference of a logical-form of the sentence, which would give a formal

expression of the content.

3.3 The discourse-level perspective

Beyond the sentence, at the discourse level, a rhetorical analysis identifies relations or interdependencies holding between adjacent spans of text. Then, the arguments of a relation (discourse units, or spans of text) could be compared one to the other in terms of their importance (nuclearity). The rhetorical relations and their nuclearity are grouped in rhetorical schemes, as general patterns in which spans of text can be recurrently analyzed.

The main regard of discourse theories are towards explaining the structure of a text (how is a text organised in segments and these ones – in sub-segments, and how this compositional structure influences the comprehension of the meaning), its degree of difficulty (for instance, why are certain texts easier to interpret than others [9]), its cohesion (or what makes that different components of a text look like being glued together [11]) and coherence ("Intuitively, coherence is a semantic property of discourse, based on the interpretation of each individual sentence relative to the interpretation of other sentences." [22]), and, finally, what is the relationship between coherence, cohesion and discourse structure [4]. Summarisation issues are nonetheless immerged onto a discourse-level analysis.

3.4 The pragmatic perspective

The pragmatic analysis should be based on the knowledge of the political intentions (of both the speaker, and the receiver) in connection with the ideological meanings of a speech. Only in good knowledge of the political aspirations of the hearers and knowing that the speaker knows himself this spectrum of political aspirations, a human analyst would succeed in interpreting the whole range of subtleties of a political speech. It is clear that pragmatics makes a good deal of the political speeches interpretation process. It is nevertheless true that an experienced human analyst would succeed to acquire these facets of the pragmatic context of a political speech even having little direct

knowledge on them. It is like in an act of reverse engineering in which the analyst is able to infer the political ideology of the speaker and of the auditorium from the speech itself.

A closer look on a pragmatic analysis of a political discourse reveals the following aspects: interpretation of the text in terms of psychological distance between the partners, opponents, etc.; defining the transmitter's political attitude before and after the communication; determining the receptor's political attitude (i.e. being pro, against or undecided); pursuing echoes of the political communication in the audience (immediately), or in the society (after a delay), etc.; discovering the political speaker's intentions by evidencing the semantic roles of different sentence constituents (reiterations, expressions, etc.).

4 A platform for multi-dimensional political discourse analysis

In this section we briefly describe the Discourse Analysis Tool (DAT), a platform which integrates a range of language processing tools, with the intent to build complex characterisations of the public discourse. Out of the discussed perspectives of semiotic analysis, DAT (currently at version 3) implements only lexical-semantic features. The concept behind the lexical-semantic analysis in DAT is that the vocabulary used by a speaker opens a window towards the author's sensibility, towards his/her level of culture, her/his cognitive world. Some of these means of expression are persuasive, aimed to convince the public on the own opinions, while others are manipulative, aimed to induce a false perspective on an issue. Figure 1 displays a snapshot of the interface showing a semantic analysis, during a working session. The platform incorporates two alternative views for presenting the results of the lexical-semantic analysis: graphical (pie, function, columns and areas) and tabular (Microsoft Excel compatible).

The vocabulary of the 33 semantic classes (detailed in Figure 2) of DAT.v3 are considered to fulfil optimally the necessity of interpreting the political discourse of our corpus.

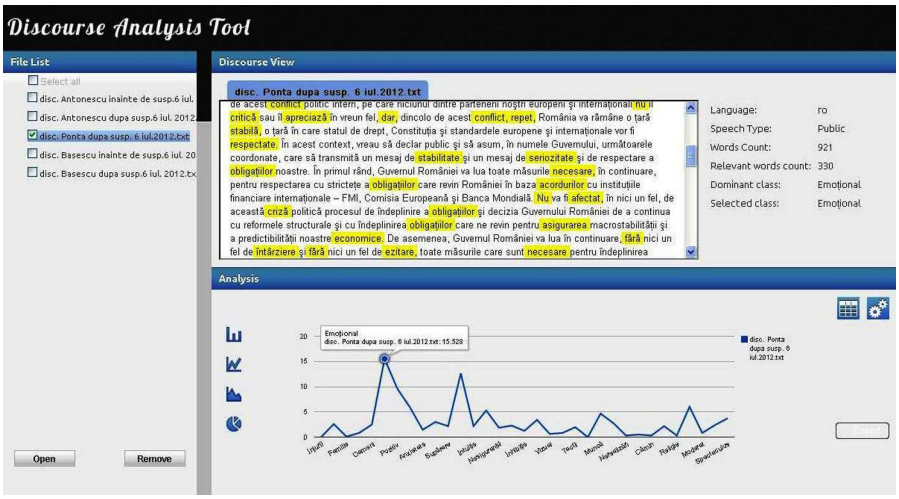


Figure 1. The DAT interface: in the left window the selected files appear, in the middle window – the text in the selected file, and in the right window – information about the text (language, word count, dominant class, etc.). Below, a plot (form chosen from a range of graphical tools) is displayed. By selecting a specific class in the middle window, all words assigned to that class are highlighted in the text.

```
<?xml version="1.0" encoding="UTF-8"?>
-<classes>
<class name="swear" id="1"/>
<class name="social" id="2"/>
<class name="family" id="3" parent="2"/>
<class name="friends" id="4" parent="2"/>
<class name="people" id="5" parent="2"/>
<class name="emotional" id="6"/>
<class name="positive" id="7" parent="6"/>
<class name="negative" id="8" parent="6"/>
<class name="anxiety" id="9" parent="8"/>
<class name="anger" id="10" parent="8"/>
<class name="sadness" id="11" parent="8"/>
<class name="cognitive" id="12"/>
<class name="intuition" id="13" parent="12"/>
<class name="determine" id="14" parent="12"/>
<class name="uncertain" id="15" parent="12"/>
<class name="certain" id="16" parent="12"/>
<class name="inhibition" id="17" parent="12"/>
<class name="perceptual" id="18"/>
<class name="see" id="19" parent="18"/>
<class name="hear" id="20" parent="18"/>
<class name="feel" id="21" parent="18"/>
<class name="sexual" id="22"/>
<class name="work" id="23"/>
<class name="achievements" id="24"/>
<class name="failure" id="25"/>
<class name="agreement" id="26"/>
<class name="home" id="27"/>
<class name="financial" id="28"/>
<class name="religion" id="29"/>
<class name="nationalism" id="30"/>
<class name="moderate" id="31"/>
<class name="firmness" id="32"/>
<class name="spectacular" id="33"/>
</classes>
```

Figure 2. Semantic classes in DAT.v3

Our interest went mainly in determining those political attitudes able to influence the voting decision of the auditorium. But the system can be parameterised to fit also other conjunctures. As such, the user can define at will her/his semantic classes, which, as can be noticed in Figure 2, are partially placed in a hierarchy.

The development of the lexicon associated with these classes was done in several phases. We started with a small vocabulary (mainly looking for translation equivalents in Romanian of a subset of the LIWC-2007 classes). Then, the words of this initial lexicon have been used as seeds in a trial to enrich the lexicon automatically by using the morphological database of DEX-online, an online electronic dictionary for Romanian language.

To prepare the integration of syntax in DAT, a dependency parser for Romanian is in the process of being trained on a dependency treebank. This corpus of syntactic trees (incorporating now over 4,000 tree structures) has been partially developed manually, by using a graphical editing tool (TreeAnnotator) and, later on, by the dependency parser itself, in a bootstrapping manner. After the corpus reached the size of 100 structures, the development of the resource continued in a bootstrapping manner: the new sentences belonging to the interim president were first parsed by the parser and then manually corrected by the first author of this paper. This way, the development of the corpus gained very much in speed. The format of the stored trees is XML, with the following elements:

- `sentence` – marking the sentences; its attributes are: a unique identifier and the name of the annotator who lastly worked over the sentence;
- `word` – marking individual words of the sentence; its attributes are: a unique identifier, the morphological tag, the lemma form of the inflected word, the ID of its parent word (the head in the dependency structure) and the name of the relation linking the word to its parent.

The following version of DAT is planned to integrate also a syntactic parser, offering to the user the possibility to identify and count relations

between different parts of speech, to put in evidence patterns of use at the semantic and syntactic level, discursive behaviours, etc.

5 A comparative study

5.1 The corpus

The corpus used for our investigation was configured to allow a comparative study over the discursive characteristics of three political leaders, Traian Băsescu, Crin Antonescu, and Victor Ponta. Traian Băsescu was the Romanian's president since 2004 (with an interruption in the summer of 2012, when he was suspended, period monitorized in this study), one of the most complex personalities of the Romanian political arena of the last decade. The second political actor, Crin Antonescu, is an ex-leader of the Liberal Party, for a short while – President of the Senate and then – the Romania's interim President (during Băsescu's suspension). The last political actor, Victor Ponta, is an ex-leader of the Social Democrat Party, the actual Romanian prime minister, and represents the new political generation. His party and Antonescu's party form the USL coalition (The Social-Liberal Union). This coalition, with a social-liberal ideology is a premiere in Romania.

We are, this way, putting on the balance three styles of political discourse that, at least in principle, are perceived as being different as ideologies (democrat-liberal, liberal, and social-democrat). But more than comparing political discourses belonging to different ideologists, the year 2012, so politically dense, offers the opportunity to study how the stress of the political battle from the edge of a crises is reflected in these major opponents' speeches, as evidenced by a semiotic analysis. Indeed, 2012 was the year of governmental changes in Romania. After the January street protests and following President Băsescu's request, the Boc Government resigns (20 January) and is replaced by the Ungureanu Government (6 February). Permanently contested and sanctioned by the public opinion, less than 3 months later, the Ungureanu Government falls, following a vote of confidence from the Parliament, put forward by the opposition block PSD-PNL-PC (27 April).

The President will designate a new premier, Victor Ponta, the head of the principal opposition party, PSD, sustained by Crin Antonescu, the liberals' head. The two politicians make the bases of a new coalition, USL, whose principal objective is the removal of the President, following thus one of the demands of the protestants. On 10 June, the local elections will completely change the political map of Romania: the governmental coalition becomes legitimate in the principal cities and districts of the country. The next step will be the relegation of Băsescu, preceded by a motion of censure (6 July), when the President is suspended. This will trigger the political crisis, around which our analisys gravitates.

For the elaboration of preliminary conclusions on the crisis process, we collected, stored and processed, partially manually, partially automatically, political texts published by three national on-line publications having similar profiles. A small part of this corpus which includes a collection of 100 political sentences, thoroughly chosen, each containing one or more clauses, has been syntactically annotated.

5.2 The lexical-semantic analyses

Apart from simply counting frequencies of mentions of semantic classes of one author, the system can also perform comparative studies between two or more authors or for the same author in different periods of time.

To exemplify, we present below different charts with two streams of data, representing the political speeches in the context of the political crisis (before Băsescu's suspension), belonging to the three important political leaders mentioned above. In fact, our analysis makes a two by two comparison of the three political actors mentioned. In each of the diagrams that follow, for each semantic class, the values corresponding to one subject are subtracted from the other. Our experience shows that an absolute difference value below the threshold of 0.5% should be considered as irrelevant and is, therefore, ignored in the interpretation. For this reason, these classes are not represented in the chart.

The graphical representation in Figure 3, in which Traian Băsescu, President of Romania before the temporary suspenssion (figured above

the Ox axis) is compared against Crin Antonescu, the President of the Senate at that time (figured below the Ox axis), should be interpreted as follows: Traian Băsescu was interested more on the labour market in Romania (the class `work`), uttered in an intuitive tone (the class `intuition`), than Crin Antonescu, whose discourse had patriotic accents (the class `nationalism`), and who developed a comparative analysis between failures (the class `failures`) and achievements (the class `achievements`) during Băsescu's presidency.

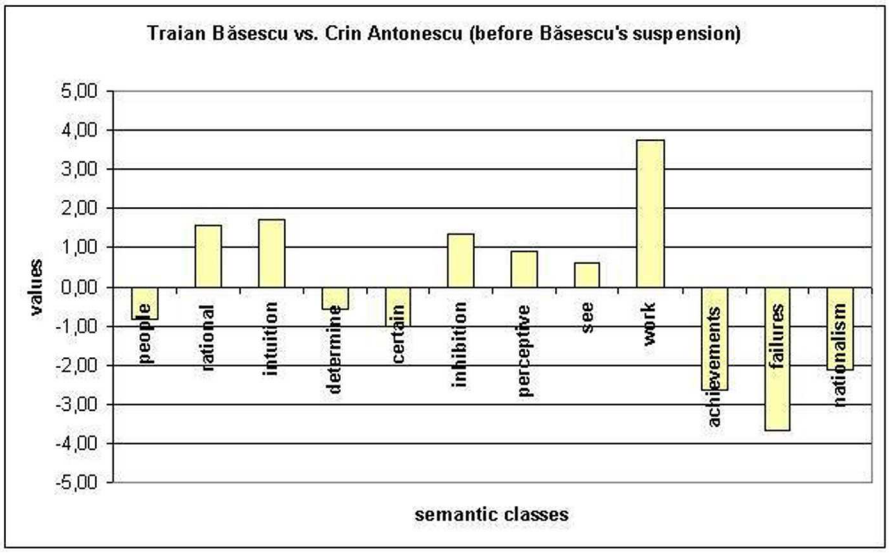


Figure 3. The average differences in the frequencies of all classes (that cumulate more than 0.5 % occurrences) in the political discourses of Traian Băsescu and Crin Antonescu, before the initiation of the crisis.

It is interesting to see how quickly the discursive spectrum changes after Băsescu's suspension: in the same day, Băsescu becomes negative, and Antonescu positive. In fact, a normal attitude... as the first subject was suspended after the vote of the Parliament, and the second subject will become the interim President, triggered by his quality of President of the Senate.

This new situation is narrated by the chart in Figure 4, which shows again two streams of data belonging to the same subjects, but this time after the moment the crisis erupted (after Băsescu's suspension). Our reading of the diagram is as follows: Traian Băsescu had a negative tone (the class `anger`), but he kept a more rational attitude (the class `intuition`) than Crin Antonescu, who becomes full of hope (the class `positive`) and who has a stronger patriotic attitude (the class `nationalism`).

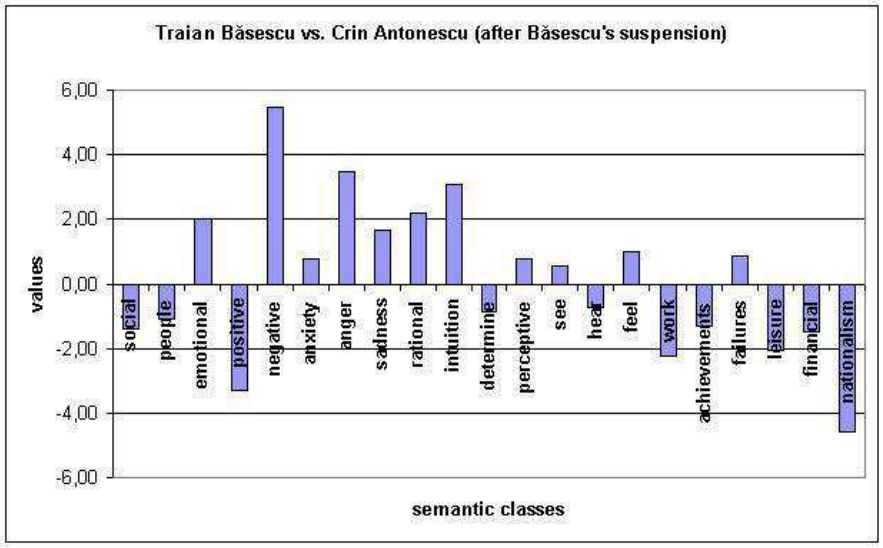


Figure 4. The average differences in the frequencies of all classes (that cumulate more than 0.5% occurrences) in the political discourses of Traian Băsescu and Crin Antonescu, after the initiation of the crisis.

The inedited element was the absence of Romanian Prime Minister, Victor Ponta, at the meeting of Parliament. He preferred to have a short statement after Băsescu's suspension.

It is also interesting to make a comparative radiography of the other two political opponents – Traian Băsescu and Victor Ponta in a critical moment, i.e. immediately after the political crisis has been fired. The

chart in Figure 5 compares the political texts of Traian Băsescu (above the Ox axis) and Victor Ponta (below the Ox axis). Our reading is the following: Traian Băsescu had a negative tone (the classes `negative`, and `anger`), but he kept a rational attitude (the classes `rational`, and `intuition`), while Victor Ponta was satisfied with the results (the class `positive`).

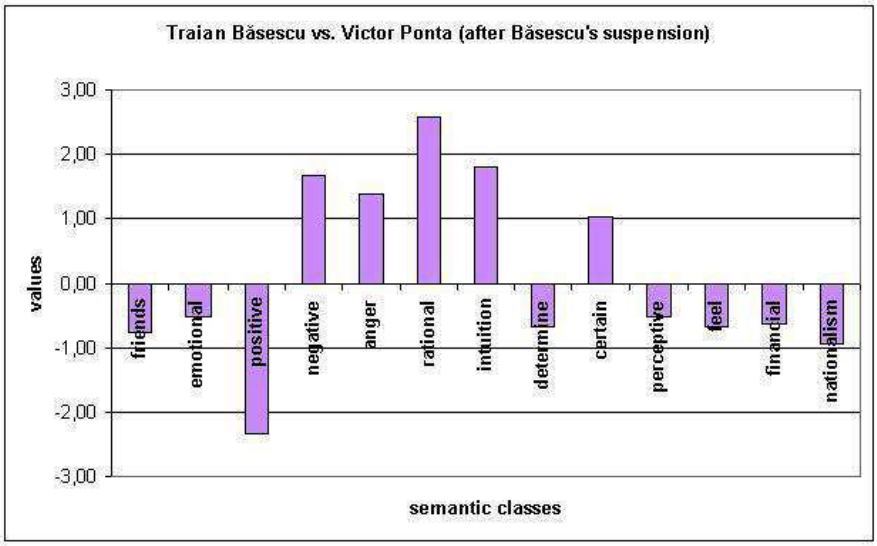


Figure 5. The average differences in the frequencies of all classes (that cumulate more than 0.5% occurrences) in the political discourses of Traian Băsescu and Victor Ponta, after the initiation of the crisis.

One of the interesting studies which we have in attention is to perform comparative studies for the same political actor in different periods of time, in our case, before and after the initiation of the crisis that resulted in the Romanian President's suspension. For exemplification, we have chosen Băsescu's speeches.

The graphical representation in Figure 6, in which the President Traian Băsescu's speech (above the Ox axis) is compared against the suspended President Traian Băsescu's speech (below the Ox axis)

should be interpreted as follows: before his suspension, the subject accentuated more on social aspects, his discourse was positive and insisted on the achievements. On the contrary, after being suspended his discourse became emotional, negative, with eruptions of anger and sadness, while still preserving a rational tone. For instance, before his suspension, Băsescu used expressions such as: "se pare că eu nu reuşesc" (*it seems that I don't succeed*), "decât atingerea scopurilor politice" (*other than attaining political purposes*), "Eu cred că este o greşeală" (*I consider being a mistake*), etc. After president's suspension, Băsescu changed the discursive tone preferring expressions, such as: "în concluzie, mergem la Referendum" (*in conclusion, we're going to Referendum*), "dar, să vedem" (*but let's see*), "Dar înainte de a merge la referendum" (*but before going to Referendum*), etc.

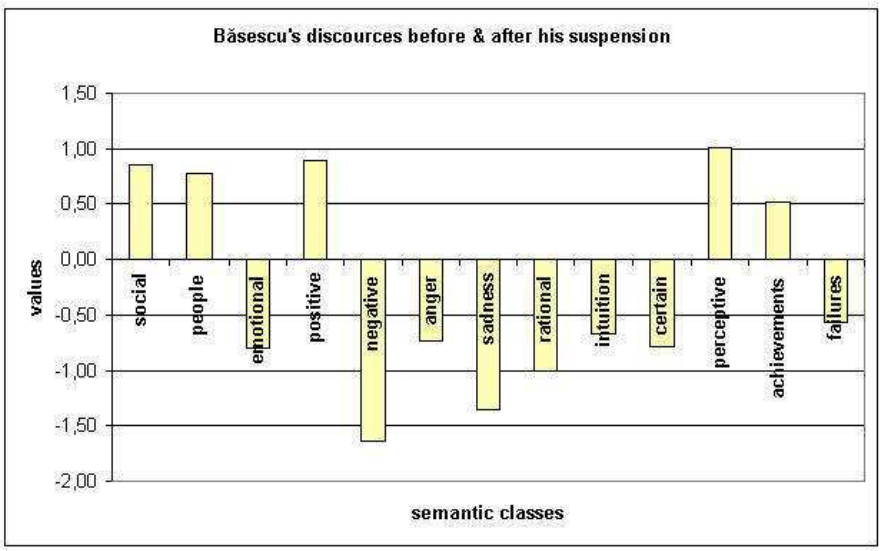


Figure 6. Băsescu's versus himself, before and after the suspension.

5.3 The syntactic analyses

In order to proceed with a syntactic level investigation, the text bodies have been pre-processed automatically by an NLP processing flow that included: sentence splitting, tokenisation, part-of-speech tagging and lemmatisation. Then, two thirds of the corpus were automatically parsed at the FDG structure, and the remaining part was manually annotated using the TreeAnnotator interface. Both resulted in heavily annotated XML files. Table 1 shows the size of the corpus used in these syntactic analysis.

Table 1. The corpus of texts annotated for syntax in Crin Antonescu's speeches

Number of sentences	Number of words	Number of annotated sentences	Number of words in the annotated sentences
123	3,960	100	3,286

We concentrated our analysis on three types of syntactic relations that we believe have a rhetoric role in the crisis context: adjectival, appositional and anacoluthic [7] (Table 2 displays absolute and relative values for all types of relations). Note that none of these relations are compulsory in the syntax of the phrase (the same as with the overtly expressed pronouns on the position of subject, in Romanian, for instance). Even more than that, the anacoluthic constructions are considered errors in a cultivated speech, although, properly mastered, they could have rhetorical value. Therefore, the use of all these relations is strictly a matter of personal choice.

The adjectival structure (marked as `a.adj`, `a.subst`, `a.vb` and `a.adv` in Table 2; 19.5% of all relations in the corpus) means adjectival, nominal, verbal and adverbial attributes: the adjectives add colour to the discourse. The orator not only that brings a contextual, albeit new, information, but enhances the enouncement by detailing it and developing it. The adjectival group is usually part of the rheme (the

Table 2. Occurrence of dependency relations for Crin Antonescu's political speeches corresponding to the crisis context

Relation	Number	Percentage
coord.	286	11.1%
prep.	320	12.4%
a.adj.	156	**6.0%**
c.d.	198	7.7%
punct.	100	3.9%
sbj.	96	3.7%
part.	120	4.6%
c.i.	76	2.9%
a.subst.	198	**7.7%**
a.vb.	112	**4.3%**
det.	90	3.5%
c.c.m.	98	3.8%
n.pred.	60	2.3%
aux.	84	3.3%
a.adv.	40	**1.5%**
refl.	120	4.6%
anacol.	98	**3.8%**
c.c.t.	40	1.5%
neg.	80	3.1%
ap.	102	3.9%
c.c.l.	46	1.8%
comp.	40	1.5%
c.c.scop.	24	0.9%
Total	2584	100

new information), not the theme (the old), being placed (in Romanian) usually after the theme element. When it is placed in the thematic position it's role is emphatic, usually associated with a particular tone, but, generally, it does not change the content of the message. The relation reveals a certain taste for belletrist culture from the part of the author.

In Figure 7 the arrows highlight the presence of two adjectival structures: "România adevărată" (*Real Romania*), "România normală" (*normal Romania*).

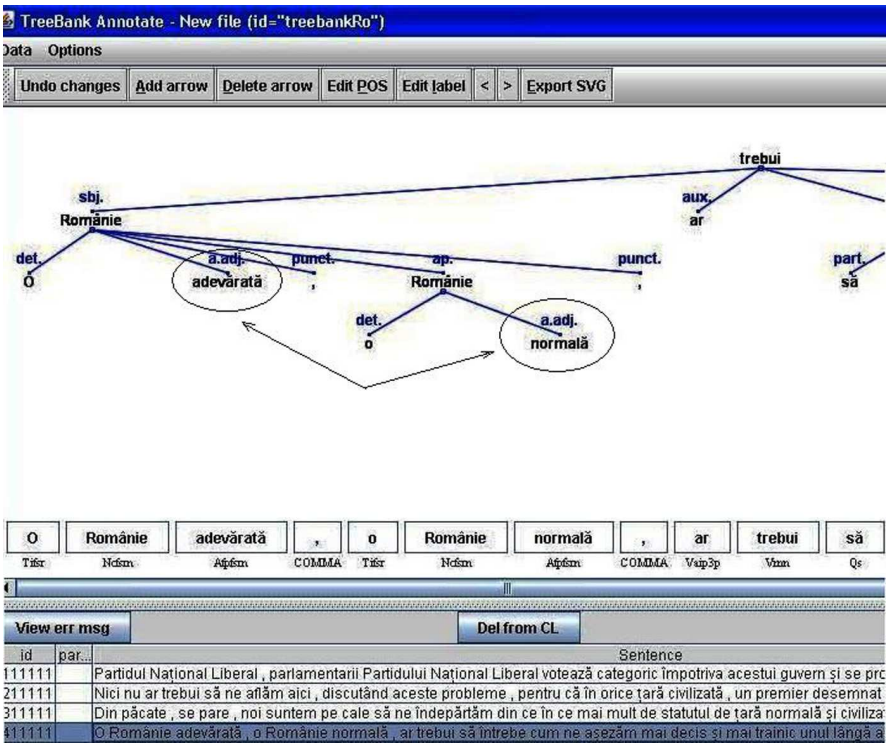


Figure 7. An adjectival structure on a dependency tree visualised with TreeAnnotator

The apposition structure (`ap.` in Table 2; 3.9%): this is the depen-

dency relation that holds between two lexical sequences, called base and apposition (the apposition being open to an unlimited number of terms), the second one giving a complementary information on the first one.

The apposition structure should be delimited from the syntactic relations of subordination and coordination, because between the base and the apposition there is no syntactic hierarchy. However, by convention, in our dependency structures, the appositive term is represented attached to the base.

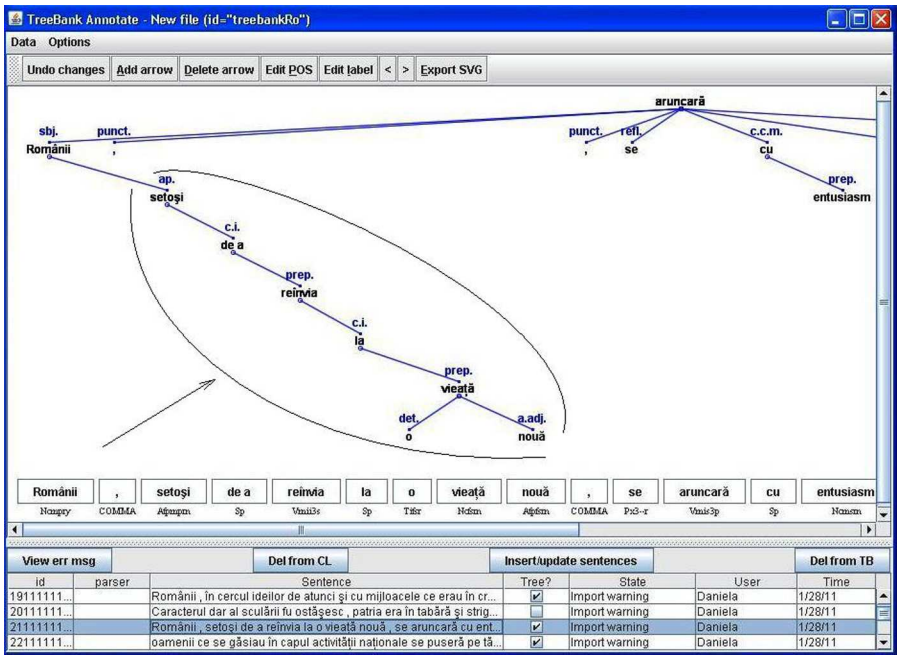


Figure 8. An apposition structure visualised with TreeAnnotator

In Figure 8, the arrow highlights an apposition structure. The sentence "Românii se aruncară cu entuziasm..." (*The Romanians jumped with enthusiasm...*) is interrupted by the apposition "setoşi de a reînvia la o viaţă nouă" (approx. *thirsty to be reborn in a new life*), which add contextual information to the main subject "Românii" (*Romanians*).

The anacoluthic structure (anacol. in Table 2; 3.8%) marks an interruption of a syntactic construction (clause, phrase) and continuation with another construction. In general, the anacoluthon is considered an error in the grammar books. So, strictly grammatical it is forbidden. To evidence it automatically in texts is extremely difficult because it is rare and a parser needs many occurrences in order to develop the ability to put it in evidence. In long sentences it is difficult even for an experienced annotator to note these intentional (or unintentional) errors, because the interspersed components have such diverse structures.

In the example in Figure 9, the principal sentence "După dânsul, veni mai târziu Regulamentul" (*After him, the Regulation came later*) is followed by the anacoluthon "căci el" (*because it*), which represents a suspended nominative (nominativus pendens) relation. The author feels the need for a change in the discourse theme, after upgrading the nominative "el" (*it*), seeming to have the function of subject near a predicate which is never uttered afterwards. The experienced political actors use anacoluthic structures strategically in communication with the intend to focus the discourse or to highlight a particular element. In this example, the politician focuses on "Regulamentul" (*the Regulation*), and the subordinate concessive sentence "deşi fu impus de străini" (*although having been imposed by foreigners*).

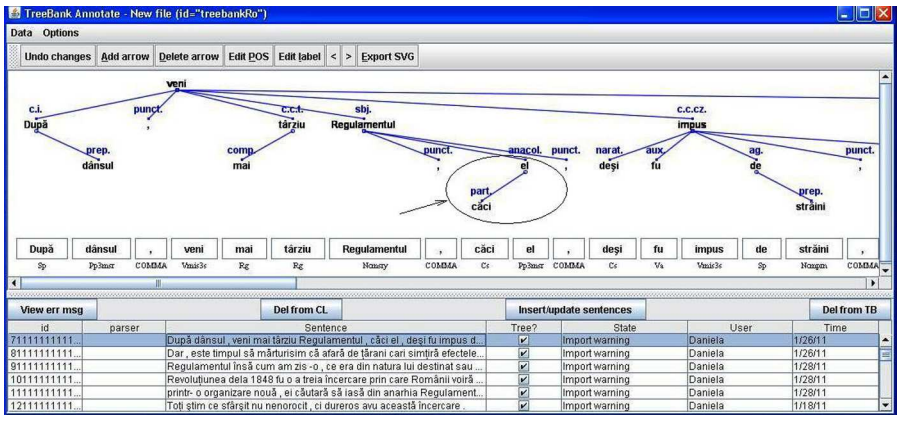


Figure 9. An anacoluthic structure, visualised with TreeAnnotator

5.4 The rhetorical and pragmatic analysis

As mentioned, a discourse-level type of analyses should reveal elements of coherence and cohesion of the text, together with the identification of the rhetorical structure of the discourse. Some of these aspects are technologically feasible with different degree of accuracy. Discourse level techniques are applied at the very end of a long processing chain, which should include: segmentation into sentences, tokenisation, pos-tagging and lemmatisation, and segmentation at the clause level. Optionally, in a more developed type of analysis, it should also include: shallow parsing (for the identification of noun-phrases), name-entity recognition, and anaphora resolution.

Counting different types of rhetorical relations in a political speech could reveal a lot over the rhetorical strategy of the author and the dynamics of the discourse. A rhetorical parser is usually trained to recognise complex rhetorical trees out of a corpus manually annotated with these structures [14]. The discourse parser developed in the NLP-Group@UAIC-FII builds rhetorical structures based on the identification of cue words and other discursive features [3, 1]. The outputted trees of the current implementation, however, miss the names of the relations, but they can retain the cue-words and the nuclearity.

Perfectly feasible with the present day technology are also the identification of some cohesion and coherence elements of a political speech, as mentioned in Section 3.3. Centering parsers (see [5, 1], for instance) could measure the coherence of a text on a scale from 0 to 4 [4]. Scaling up an exploratory tool for the purpose of our investigation would be an interesting research objective, which should take into consideration that a high quality human discourse is not always one that reaches a maximum on the coherence scale, because that one would also be very boring [5], the same as it should not be a randomly generated one, because this would be completely incoherent. It's a pharmacy chemistry that the great orators know to master, combining in proper quantities, as the discourse unfolds, the fulfilment of expectations with the unexpected and surprise.

Present day techniques make feasible the development of a number

of automatic techniques in the area of rhetorical and coherence analysis. It will be our further objective to concentrate on this type of investigation.

6 Conclusions

The analysis we proposed in this paper aims at verifying if a semiotic perspective anchored in natural language processing techniques could be of value in valuating political speeches. If this performance proves to be feasible, than semiotics would become a very applicative science, with interesting virtues in the optimization of the political discourse. Rhetorical weapons in the hands of a political actor should be: the diversity of the lexicon and a proper mastering of the semantic classes, the syntactic form, the emancipation of the expression, the coherence and the proper mastering of the comprehensibility. It is our conviction that the present day linguistic technology can successfully cover many of these facets.

However, we are aware that this study only sketches a way to go, and a lot more should be studied until a reliable discourse interpreting technology will become a tool in our hands. We should also be aware of the dangers of false interpretation. For instance, if we take as example the three orators we used in our experiments, differences at the level of lexicon and syntax, which we have evidenced as differentiating them, should be attributed only partially to their idiosyncratic rhetorical styles, because these differences could also have ideological roots. Moreover, speeches of many public actors, especially today, are the product of teams of specialists in communication and, as such, conclusions regarding their cultural universe, for instance, should be uttered with care. It remains yet to be decided the impact that the use of certain syntactic structures, such as adjectival, appositional and anacoluthic, could have over an auditory in the political discourse.

Different politicians could raise the use of these measures to the level of a rhetorical strategy, therefore exploiting them perhaps too much in the benefit of the aimed goals. In other words, this study shows that technological instruments are able to detect tendencies of manipulation

of the receiver with the evident role of detouring the attention of the audience from the actual communicated content in favour of the orator. The software allows online editing of a yet-to-be-delivered speech, in order to make it fit to the audience profile (public of large, journalists, different levels of culture).

Many interpretation facets are pertinent to the specific context a discourse is being uttered. For instance, in a crisis context a political discourse should be evaluated in function of the balance between the agenda of an orator that happens to be on the site of the political power, versus the opposite agenda. Different intensities of emotional levels could also be evidenced, and we prepare a more fined grade scale of emotional expressions. It is a known fact that the audience can be manipulated easily (e.g., the class `sadness`) by political actors when their themes are treated with excessive emotional tonalities.

We are aware that many technological aspects remain yet to be refined and enhanced. One of the most important is the determination of the senses of words and expressions in context. In the future we intend to include a word sense disambiguation module in order to determine the correct senses, in context, of those words which are ambiguous between different semantic classes, or between classes in the lexicon and outside the lexicon (in which case they would not have to be counted). Also, negations could completely reverse the semantic class a certain expression belongs to in certain contexts and need therefore special treatment.

The collection of manually annotated texts is only at beginning, a starting point for an efficient automatic annotation. In the near future we will manually correct all the automatically annotated texts, improving thus the behaviour of the parser. Another line to be continued regards the evaluation metrics, which have not received enough attention till now. We are currently studying other statistical metrics able to give a more comprehensive image on different facets of the political discourse.

We believe that the platform has a range of features that make it attractive as a tool to assist any kind of political campaigns. It can be rapidly adapted to new domains and to new languages, and its inter-

face is user-friendly and offers a good range of functionalities. It helps to outline distinctive features which bring a new and, sometimes, unexpected vision upon the discursive characteristics of political authors.

Acknowledgments: In performing this research, the first author was supported by the POSDRU/89/1.5/S/63663 grant, and the second author – by the ICT-PSP projects METANET4U # 270893 and ATLAS # 250467. Alex Moruz helped the first author to clean the DAT Romanian lexicon in an early phase. Afterwards it has been largelly extended by Radu Simionescu after importing the Romanian morphology from the DEX-online database. We are grateful to Cătălin Frâncu and Radu Borza for offering this database. The DAT platform has been developed by Mădălina Spătaru, as a post-master activity in the Faculty of Computer Science of the "Alexandru Ioan Cuza" University of Iaşi. All the Romanian NLP components mentioned in this paper were developed in the NLP-Group@UAIC-FII.

References

[1] D. Anechitei, D. Cristea, I. Dimosthenis, E. Ignat, D. Karagiozov, S.Koeva, M. Kopeć, C. Vertan. (2013, to appear). *Summarizing Short Texts Through a Discourse-Centered Approach in a Multilingual Context*. In Neustein, A., Markowitz, J.A. (eds.), Where Humans Meet Machines: Innovative Solutions to Knotty Natural Language Problems. Springer Verlag, Heidelberg/New York.

[2] C.F. Baker, C.F. Fillmore, J.B. Lowe. (1998). *The Berkeley Framenet project*. In Proceedings of the COLING-ACL 1998, Montreal, Canada.

[3] A. Belogay, D. Karagyozov, S. Koeva, C. Vertan, A. Przepiórkowski, D. Cristea, P. Raxis. (2012). *Harnessing NLP Techniques*, in Walter Daelemans, Mirella Lapata Lluis Marquez (Eds.) Processes of Multilingual Content Management, Proceedings of EACL 2012 – the 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, pp. 6–10, ISBN: 978-1-937284-19-0.

[4] D. Cristea, N. Ide, L. Romary. (1998). *Veins Theory. An Approach to Global Cohesion and Coherence*. In Proceedings of Coling/ACL '98, Montreal.

[5] D. Cristea, A. Iftene. (2011). *Grounding Coherence Properties of Discourse*. In ALEAR Final Report, vol. II. Embodied Cognitive Semantics, Berlin, April.

[6] U. Eco. (1996). *Limitele interpretării* (Limits of interpretation), Ed. Pontica, Constanţa.

[7] *Gramatica limbii române* (The Grammar of the Romanian Language). (2005). Vol. II, Enunţul (The statement), Ed. Academiei Române, Bucureşti, 105–113, 619–31, 743–747.

[8] D. Gîfu, D. Cristea. (2012). *Multi-dimensional analysis of political language*, in "Future Information Technology, Application, and Service", in James J. (Jong Hyuk) Park, Victor C.M. Leung, Cho-Li Wang, Taeshik Shon (eds.), volume 1/164, Springer Science+Business, Media Dortdrecht.

[9] B.J. Grosz, A.K. Joshi, S. Weinstein. (1995). *Centering: A framework for modeling the local coherence of discourse*. In Computational Linguistics, 12(2), 203–225.

[10] Eva Hajicŏvá, B.H. Partee, P. Sgall. (1998). *Topic–Focus Articulation, Tripartite Structures, and Semantic Content*. In Studies in Linguistics and Philosophy, 71, Dordrecht, Kluwer.

[11] M.A.K. Halliday, R. Hasan. (1976). *Cohesion in English*. Longman, London.

[12] E.D. Liddy. (2001). *Natural Language Processing*, in Encyclopaedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc.

[13] W.C. Mann, S.A. Thompson. (1988). *Rhetorical Structure Theory: Toward a functional theory of text organization*, in Text 8(3), 243–281.

[14] D. Marcu. (2000). *The theory and practice of discourse parsing and summarization*, The MIT Press, Cambridge, Massachusetts.

[15] Ch. Morris. (1938). *Foundations of the Theory of Signs*, The University of Chicago Press. Pennebaker, J. W., Francis, Martha E., Booth, R. J. (2001). Linguistic Inquiry and Word Count "LIWC2001, Mahwah, NJ, Erlbaum Publishers.

[16] J.W. Pennebaker, M.E. Francis, R.J. Booth. (2001). Linguistic Inquiry and Word Count LIWC2001, Erlbaum Publishers, Mahwah, NJ, 2001.

[17] H.F. Plett. (1983). *Ştiinţa textului şi analiza de text* (The science of text and the text analysis), Ed. Univers, Bucharest.

[18] N. Rescher. (1973). *The coherence theory of truth*, Oxford UP, London.

[19] S. Romaine. (1994). *Language in society. An Introduction to Sociolinguistics*, Oxford University Press Inc., New York.

[20] Ferdinand de Saussure. (1916). *Cours de linguistique générale*, Payot, Paris.

[21] L. Tesniére. (1959). *Elements of structural syntax*, Editions Klincksieck.

[22] T. Van Dijk. (1977). *Text and Context. Explorations in the semantics and pragmatics of discourse*, Longman, New York.

Daniela Gîfu, Dan Cristea, Received July 24, 2012

Daniela Gîfu
"Alexandru Ioan Cuza" University of Iaşi
Faculty of Computer Science
16, Berthelot St., 700483 Iaşi, Romania
Phone: +40.232.201724
E–mail: *daniela.gifu@info.uaic.ro*

Dan Cristea
"Alexandru Ioan Cuza" University of Iaşi
Faculty of Computer Science
16, Berthelot St., 700483 Iaşi, Romania
Phone: +40.232.201542
E–mail: *dcristea@info.uaic.ro*

Institute of Computer Science
Romanian Academy, the Iaşi branch
2, T. Codrescu St., 700481-Iaşi, Romania

Computer Science Journal of Moldova, vol.**21**, no.1(61), 2013

Linear discrete-time Pareto-Nash-Stackelberg control problem and principles for its solving

Valeriu Ungureanu

Abstract

A direct-straightforward method for solving linear discrete-time optimal control problem is applied to solve control problem of a linear discrete-time system as a mixture of multi-criteria Stackelberg and Nash games. For simplicity, the exposure starts with the simplest case of linear discrete-time optimal control problem and, by sequential considering of more general cases, investigation finalizes with the highlighted Pareto-Nash-Stackelberg and set valued control problems. Different principles of solving are compared and their equivalence is proved.

Mathematics Subject Classification 2010: 49K21, 49N05, 93C05, 93C55, 90C05, 90C29, 91A10, 91A20, 91A44, 91A50.

Keywords: Linear discrete-time control problem, non-cooperative game, multi-criteria strategic game, Pareto-Nash-Stackelberg control.

1 Introduction

Optimal control theory which appeared due to Lev Pontryagin [2] and Richard Bellman [3], as natural extension of calculus of variations, often doesn't satisfy all requirements and needs for modelling and solving problems of real dynamic systems and processes. A situation of this type occurs for problem of linear discrete-time system control by a decision process that evolves as Pareto-Nash-Stackelberg game with constraints – a mixture of hierarchical and simultaneous games [5, 6, 7, 8, 9]. For such system, the notion of optimal control evolves naturally to

the notion of Pareto-Nash-Stackelberg type control and to the natural principle for solving the highlighted problem by applying a concept of Pareto-Nash-Stackelberg equilibrium [9] with a direct-straightforward principle for solving.

The direct method and principle for solving linear discrete-time optimal control problem is extended to control problem of a linear system in discrete time as a mixture of multi-criteria Stackelberg and Nash games [9]. The exposure starts with the simplest case of linear discrete-time optimal control problem [1] and, by sequential considering of more general cases, finalizes with the Pareto-Nash-Stackelberg and set valued control problems. The maximum principle of Pontryagin is formulated and proved for all the considered problems. Its equivalence with the direct-straightforward principle for solving is established.

2 Linear discrete-time optimal control problem

Consider the following problem [1][1]:

$$\begin{aligned} f(x,u) = \sum_{t=1}^{T} (c^t x^t + b^t u^t) \quad &\to \max, \\ x^t = A^{t-1} x^{t-1} + B^t u^t, \quad &t = 1, ..., T, \\ D^t u^t \le d^t, \quad &t = 1, ..., T, \end{aligned} \tag{1}$$

where x^0, x^t, $c^t \in R^n$, u^t, $b^t \in R^m$, $A^{t-1} \in R^{n\times n}$, $B^t \in R^{n\times m}$, $d^t \in R^k$, $D^t \in R^{k\times n}$, $c^t x^t = \left\langle c^t, x^t \right\rangle$, $b^t u^t = \left\langle b^t, u^t \right\rangle$, $t = 1, ..., T$, $u = (u^1, \dots, u^T)$.

[1]Symbol T means discrete time horizon in this paper. Symbol of matrix translation is omitted. Left and right matrix multiplications are largely used. The reader is asked to understand by himself when column or row vector are used.

The problem (1) may be represented in the form:

$$\begin{array}{llllll}
Ex^1 & & -B^1u^1 & & & = A^0x^0,\\
-A^1x^1 + Ex^2 & & & -B^2u^2 & & = 0,\\
\cdots & & \cdots & & & \cdots\\
-A^{T-1}x^{T-1} + Ex^T & & & & -B^Tu^T & = 0,\\
 & & D^1u^1 & & & \le d^1,\\
 & & & D^2u^2 & & \le d^2,\\
 & & & \cdots & & \cdots\\
 & & & & D^Tu^T & \le d^T,\\
c^1x^1 + c^2x^2 & +\cdots+ & b^1u^1 & +b^2u^2+\cdots+b^Tu^t & & \to \max.
\end{array}$$

Its dual problem is

$$\begin{array}{llllll}
p^1 & -p^2A^1 & & & & = c^1,\\
 & p^2 - p^3A^2 & & & & = c^2,\\
 & & \cdots & & & \cdots\\
 & & p^T & & & = c^T,\\
-p^1B^1 & & +q^1D^1 & & & = b^1,\\
 & -p^2B^2 & & +q^2D^2 & & = b^2,\\
 & & & \cdots & & \cdots\\
 & & -p^TB^T & & +q^TD^T & = b^T,\\
 & & q^1 \ge 0, q^2 \ge 0, & & q^T \ge 0, & \\
p^1A^0x^0 & & +q^1d^1 & +q^2d^2+\cdots+q^Td^T & & \to \min.
\end{array}$$

From the constraints of dual problem it follows that the values of variables $p^1, p^2, \ldots, p^T$ are calculated on the bases of recurrent relation:

$$\begin{array}{rcl}
p^T & = & c^T,\\
p^t & = & p^{t+1}A^t + c^t, \quad t = T-1, ..., 1.
\end{array} \tag{2}$$

So, the dual problem is equivalent to:

$$\begin{array}{rcl}
q^1D^1 & = & p^1B^1 + b^1,\\
q^2D^2 & = & p^2B^2 + b^2,\\
 & \cdots & \\
q^TD^T & = & p^TB^T + b^T,\\
q^t & \ge & 0, t = 1, \ldots, T,\\
p^1A^0x^0 + q^1d^1 + q^2d^2 + \cdots + q^Td^T & \to & \min.
\end{array}$$

The dual of the last problem is:

$$\begin{array}{rcl} D^1u^1 & \leq & d^1, \\ D^2u^2 & \leq & d^2, \\ & \cdots & \\ D^Tu^T & \leq & d^T, \\ \sum_{t=1}^{T} \langle u^t, p^tB^T + b^t \rangle & \rightarrow & \max. \end{array} \tag{3}$$

The solution of (3) may be found by solving T linear programming problems

$$\begin{array}{rcl} D^tu^t & \leq & d^t, \\ \langle u^t, p^tB^T + b^t \rangle & \rightarrow & \max, \end{array}$$

for $t = 1, \ldots, T$. So, the solution of initial control problem (1) is identical with a sequence of solutions of T linear programming problems.

Similar results may be obtained by performing direct transformations of (1):

$$\begin{array}{rcl} x^1 & = & A^0x^0 + B^1u^1, \\ x^2 & = & A^1x^1 + B^2u^2 = A^1(A^0x^0 + B^1u^1) + B^2u^2 = \\ & = & A^1A^0x^0 + A^1B^1u^1 + B^2u^2, \\ x^3 & = & A^2x^2 + B^3u^3 = A^2(A^1A^0x^0 + A^1B^1u^1 + B^2u^2) + B^3u^3 = \\ & = & A^2A^1A^0x^0 + A^2A^1B^1u^1 + A^2B^2u^2 + B^3u^3, \\ & \cdots & \\ x^T & = & A^{T-1}x^{T-1} + B^Tu^T = \\ & = & \prod_{t=0}^{T-1} A^tx^0 + \prod_{t=1}^{T-1} A^tB^1u^1 + \prod_{t=2}^{T-1} A^tB^2u^2 + \\ & & + \cdots + \prod_{t=T-1}^{T-1} A^tB^{T-1}u^{T-1} + B^Tu^T, \end{array}$$

and by subsequent substitution in the objective function:

$$\begin{aligned}
f(x,u) = & c^1\left(A^0x^0+B^1u^1\right)+c^2\left(A^1A^0x^0+A^1B^1u^1+B^2u^2\right)+\\
& +c^3\left(A^2A^1A^0x^0+A^2A^1B^1u^1+A^2B^2u^2+B^3u^3\right)+\cdots+\\
& +c^T\left(\prod_{t=0}^{T-1}A^tx^0+\prod_{t=1}^{T-1}A^tB^1u^1+\prod_{t=2}^{T-1}A^tB^2u^2+\right.\\
& \left.+\cdots+\prod_{t=T-1}^{T-1}A^tB^{T-1}u^{T-1}+B^Tu^T\right)+\\
& +b^1u^1+b^2u^2+\cdots+b^Tu^T=\\
= & \left(c^1+c^2A^1+c^3A^2A^1+\cdots+c^TA^{T-1}A^{T-2}\ldots A^1\right)A^0x^0+\\
& +\left(c^1B^1+c^2A^1B^1+c^3A^2A^1B^1+\cdots+\right.\\
& \left.+c^TA^{T-1}A^{T-2}\ldots A^1B^1+b^1\right)u^1+\\
& +\left(c^2B^2+c^3A^2B^2+c^4A^3A^2B^2+\cdots+\right.\\
& \left.+c^TA^{T-1}A^{T-2}\ldots A^2B^2+b^2\right)u^2+\\
& +\cdots+\left(c^TB^T+b^T\right)u^T.
\end{aligned}$$

Finally, the problem obtains the form

$$\begin{aligned}
& f(u)=\\
& =\left(c^1+c^2A^1+c^3A^2A^1+\cdots+c^TA^{T-1}A^{T-2}\ldots A^1\right)A^0x^0+\\
& +\left(c^1B^1+c^2A^1B^1+c^3A^2A^1B^1+\cdots+\right.\\
& \left.+c^TA^{T-1}A^{T-2}\ldots A^1B^1+b^1\right)u^1+\\
& +\left(c^2B^2+c^3A^2B^2+c^4A^3A^2B^2+\cdots+\right.\\
& \left.+c^TA^{T-1}A^{T-2}\ldots A^2B^2+b^2\right)u^2+\\
& +\cdots+\left(c^TB^T+b^T\right)u^T\to\max,\\
& D^tu^t\le d^t,t=1,\ldots,T.
\end{aligned}\tag{4}$$

Obviously, (3) and (4) are identical. So, the solution of the last problem (4) is obtained as a sequence of solutions of T linear programming problems. Apparently, the complexity of such method is polynomial, but really it has pseudo-polynomial complexity because of possible exponential value of T on n.

Theorem 1. *Let (1) be solvable. The sequence $\bar{u}^1,\bar{u}^2,\ldots,\bar{u}^T$ forms an optimal control if and only if $\bar{u}^t$ is the solution of linear programming*

problem

$$\left(c^t B^t + c^{t+1} A^t B^t + \cdots + c^T A^{T-1} A^{T-2} \ldots A^t B^t + b^t\right) u^t \to \max,$$
$$D^t u^t \leq d^t,$$

for $t = 1, \ldots, T$.

Different particular cases may be established for (1).

Theorem 2. *If* $A^0 = A^1 = \cdots = A^{T-1} = A$, $B^1 = B^2 = \cdots = B^T = B$, *and (1) is solvable, then the sequence* $\bar{u}^1, \bar{u}^2, \ldots, \bar{u}^T$ *forms an optimal control if and only if* $\bar{u}^t$ *is the solution of linear programming problem*

$$\left(c^t B + c^{t+1} AB + c^{t+2} (A)^2 B + \cdots + c^T (A)^{T-t} B + b^t\right) u^t \to \max,$$
$$D^t u^t \leq d^t,$$

for $t = 1, \ldots, T$.

Theorem 1 establishes a principle for solving (1). By considering Hamiltonian functions

$$H_t(u^t) = \left\langle p^t B^t + b^t, u^t \right\rangle, t = T, \ldots, 1,$$

where $p^t, t = T, \ldots, 1$ are defined by (2), as it is conjectured in [1] and proved above by two ways, the maximum principle of Pontryagin [2] holds.

Theorem 3. *Let (1) be solvable. The sequence* $\bar{u}^1, \bar{u}^2, \ldots, \bar{u}^T$ *forms an optimal control if and only if*

$$H_t(\bar{u}^t) = \max_{u^t : D^t u^t \leq d^t} H_t(u^t), t = T, \ldots, 1.$$

Evidently, Theorems 1 and 3 are equivalent.

3 Linear discrete-time Stackelberg control problem

Let us modify the problem (1) by considering the control of Stackelberg type [7], that is Stackelberg game with T players [7, 8, 5, 6]. In such game, at each stage t ($t = 1, \ldots, T$) the player t selects his strategy and communicates his and all precedent selected strategies to the following $t+1$ player. After all stage strategy selections, all the players compute their gains on the resulting profile. Let us name such type of system control as Stackelberg control, and the corresponding problem – linear discrete-time Stackelberg control problem. The described decision process may be formalized as it follows:

$$\begin{aligned}
&f_1(x,u) = \sum_{t=1}^{T} \left(c^{1t}x^t + b^{1t}u^t\right) \underset{u^1}{\longrightarrow} \max,\\
&f_2(x,u) = \sum_{t=1}^{T} \left(c^{2t}x^t + b^{2t}u^t\right) \underset{u^2}{\longrightarrow} \max,\\
&\ldots\\
&f_T(x,u) = \sum_{t=1}^{T} \left(c^{Tt}x^t + b^{Tt}u^t\right) \underset{u^T}{\longrightarrow} \max,\\
&\\
&x^t = A^{t-1}x^{t-1} + B^t u^t, t = 1, ..., T,\\
&D^t u^t \le d^t, t = 1, ..., T,
\end{aligned} \tag{5}$$

where $x^0, x^t, c^{\pi t} \in R^n$, $u^t, b^{\pi t} \in R^m$, $A^{t-1} \in R^{n\times n}$, $B^t \in R^{n\times m}$, $d^t \in R^k$, $D^t \in R^{k\times n}$, $c^{\pi t}x^t = \left\langle c^{\pi t}, x^t \right\rangle$, $b^{\pi t}u^t = \left\langle b^{\pi t}, u^t \right\rangle$, $t, \pi = 1, ..., T$.

Formally, the set of strategies of player π ($\pi = 1, 2, \ldots, T$) is determined only by admissible solutions of the problem:

$$\begin{aligned}
&f_\pi\left(x, u^\pi || u^{-\pi}\right) = \sum_{t=1}^{T} \left(c^{\pi t}x^t + b^{\pi t}u^t\right) \underset{u^\pi}{\longrightarrow} \max,\\
&\\
&x^\pi = A^{\pi-1}x^{\pi-1} + B^\pi u^\pi,\\
&D^\pi u^\pi \le d^\pi.
\end{aligned}$$

In fact, as we can find out, the strategy sets of the players are interconnected and the game is not a simple normal form game. A situation similar with that in optimization theory may be established – there are problems without constraints and with constraints. So, the strategy (normal form) game may be named strategy game without constraints. Game which contains commune constraints on strategies may be named strategy game with constraints.

Player π $(\pi = 1, 2, \ldots, T)$ decision problem is defined by the linear programming problem (5). Since the controlled system is one for all players, by performing the direct transformations as above, (5) is transformed into

$$\begin{aligned}
&f_\pi\left(u^\pi || u^{-\pi}\right) = \\
&= \left(c^{\pi 1} + c^{\pi 2}A^1 + c^{\pi 3}A^2A^1 + \cdots + \right. \\
&+ \left. c^{\pi T}A^{T-1}A^{T-2}\ldots A^1\right)A^0x^0 + \\
&+ \left(c^{\pi 1}B^1 + c^{\pi 2}A^1B^1 + c^{\pi 3}A^2A^1B^1 + \cdots + \right. \\
&+ \left. c^{\pi T}A^{T-1}A^{T-2}\ldots A^1B^1 + b^{\pi 1}\right)u^1 + \\
&+ \left(c^{\pi 2}B^2 + c^{\pi 3}A^2B^2 + c^{\pi 4}A^3A^2B^2 + \cdots + \right. \\
&+ \left. c^{\pi T}A^{T-1}A^{T-2}\ldots A^2B^2 + b^{\pi 2}\right)u^2 + \\
&+ \cdots + \left(c^{\pi T}B^T + b^{\pi T}\right)u^T \underset{u^\pi}{\rightarrow} \max, \pi = 1, \ldots, T, \\
&D^t u^t \leq d^t, t = 1, \ldots, T.
\end{aligned} \tag{6}$$

From equivalence of (5) and (6) the proof of Theorem 4 follows.

Theorem 4. *Let (5) be solvable. The sequence $\bar{u}^1, \bar{u}^2, \ldots, \bar{u}^T$ forms a Stackelberg equilibrium control in (5) if and only if $\bar{u}^\pi$ is optimal solution of linear programming problem*

$$\begin{aligned}
&f_\pi(u^\pi) = \left(c^{\pi\pi}B^\pi + c^{\pi\pi+1}A^\pi B^\pi + c^{\pi\pi+2}A^{\pi+1}A^\pi B^\pi + \cdots + \right. \\
&+ \left. c^{\pi T}A^{T-1}A^{T-2}\ldots A^\pi B^\pi + b^{\pi\pi}\right)u^\pi \underset{u^\pi}{\rightarrow} \max, \\
&D^\pi u^\pi \leq d^\pi,
\end{aligned}$$

for $\pi = 1, \ldots, T$.

There are various particular cases of (5) and Theorem 4. Theorem 5 presents one of such cases.

Theorem 5. *If* $A^0 = A^1 = \cdots = A^{T-1} = A$, $B^1 = B^2 = \cdots = B^T = B$, *and (5) is solvable, then the sequence* $\bar{u}^1, \bar{u}^2, \ldots, \bar{u}^T$ *forms a Stackelberg equilibrium control if and only if* $\bar{u}^\pi$ *is the solution of linear programming problem*

$$\begin{aligned} &\left(c^{\pi\pi}B + c^{\pi\pi+1}AB + \cdots + c^{\pi T}(A)^{T-\pi}B + b^{\pi\pi}\right)u^\pi \underset{u^\pi}{\rightarrow} \max, \\ &D^\pi u^\pi \leq d^\pi, \end{aligned}$$

for $\pi = 1, \ldots, T$.

Theorem 4 establishes a principle for solving (5). The maximum principle of Pontryagin may be applied for solving (5) too. Let us consider the following recurrent relations

$$\begin{array}{rcl} p^{\pi T} & = & c^{\pi T}, \\ p^{\pi t} & = & p^{\pi t+1}A^t + c^{\pi t}, \quad t = T-1, ..., 1, \end{array} \tag{7}$$

where $\pi = 1, \ldots, T$. Hamiltonian functions are defined as

$$H_{\pi t}\left(u^t\right) = \left\langle p^{\pi t}B^t + b^{\pi t}, u^t\right\rangle, t = T, \ldots, 1, \pi = 1, \ldots, T,$$

where $p^{\pi t}, t = T, \ldots, 1, \pi = 1, \ldots, T$, are defined by (7).

Theorem 6. *Let (5) be solvable. The sequence of controls* $\bar{u}^1, \ldots, \bar{u}^T$ *forms a Stackelberg equilibrium control if and only if*

$$H_{\pi\pi}\left(\bar{u}^\pi\right) = \max_{u^\pi : D^\pi u^\pi \leq d^\pi} H_{\pi\pi}\left(u^\pi\right),$$

for $\pi = 1, \ldots, T$.

The proof of Theorem 6 may be provided by direct substitution of relations (7) in Hamiltonian functions and by comparing the final results with linear programming problems from Theorem 4. Obviously, Theorems 4 and 6 are equivalent.

From computational point of view, method for solving problem (5) established by Theorem 4 looks more preferable than the method established by Theorem 6.

4 Linear discrete-time Pareto-Stackelberg control problem

Let us modify the problem (5) by considering control of Pareto-Stackelberg type. At each stage a single player makes decision. Every player selects on his stage his strategy according to his criteria and communicates his choice and the precedent player choices to the following player. At the last stage, after all stage strategy selections, the players compute their gains. Such type of control is named Pareto-Stackelberg control, and the corresponding problem – the linear discrete-time Pareto-Stackelberg control problem.

The described decision process may be formalized in a following manner:

$$
\begin{aligned}
&f_1(x,u) = \sum_{t=1}^{T} \left(c^{1t}x^t + b^{1t}u^t\right) \underset{u^1}{\longrightarrow} \text{ef max},\\
&f_2(x,u) = \sum_{t=1}^{T} \left(c^{2t}x^t + b^{2t}u^t\right) \underset{u^2}{\longrightarrow} \text{ef max},\\
&\dots\\
&f_T(x,u) = \sum_{t=1}^{T} \left(c^{Tt}x^t + b^{Tt}u^t\right) \underset{u^T}{\longrightarrow} \text{ef max},\\
&x^t = A^{t-1}x^{t-1} + B^t u^t, t = 1, ..., T,\\
&D^t u^t \leq d^t, t = 1, ..., T,
\end{aligned}
\tag{8}
$$

where $x^0, x^t \in R^n$, $c^{\pi t} \in R^{k_\pi \times n}$, $u^t \in R^m$, $b^{\pi t} \in R^{k_\pi \times m}$, $A^{t-1} \in R^{n\times n}$, $B^t \in R^{n\times m}$, $d^t \in R^k$, $D^t \in R^{k\times n}$, $t, \pi = 1, ..., T$. Notation ef max means multi-criteria maximization.

The set of strategies of player π ($\pi = 1, \dots, T$) is determined for-

mally by the problem

$$f_\pi\left(x, u^\pi || u^{-\pi}\right) = \sum_{t=1}^{T}\left(c^{\pi t}x^t + b^{\pi t}u^t\right) \underset{u^\pi}{\longrightarrow} \text{ef max},$$

$$x^\pi = A^{\pi-1}x^{\pi-1} + B^\pi u^\pi,$$
$$D^\pi u^\pi \leq d^\pi.$$

By performing direct transformations as above, (8) is transformed into

$$\begin{array}{l} f_\pi\left(u^\pi || u^{-\pi}\right) = \\ = \left(c^{\pi 1} + c^{\pi 2}A^1 + c^{\pi 3}A^2A^1 + \cdots + \right. \\ \left. + c^{\pi T}A^{T-1}A^{T-2}\ldots A^1\right)A^0x^0 + \\ + \left(c^{\pi 1}B^1 + c^{\pi 2}A^1B^1 + c^{\pi 3}A^2A^1B^1 + \cdots + \right. \\ \left. + c^{\pi T}A^{T-1}A^{T-2}\ldots A^1B^1 + b^{\pi 1}\right)u^1 + \\ + \left(c^{\pi 2}B^2 + c^{\pi 3}A^2B^2 + c^{\pi 4}A^3A^2B^2 + \cdots + \right. \\ \left. + c^{\pi T}A^{T-1}A^{T-2}\ldots A^2B^2 + b^{\pi 2}\right)u^2 + \\ + \cdots + \left(c^{\pi T}B^T + b^{\pi T}\right)u^{\pi T} \underset{u^\pi}{\rightarrow} \text{ef max}, \pi = 1, \ldots, T, \\ D^t u^t \leq d^t, t = 1, \ldots, T. \end{array} \tag{9}$$

Equivalence of (8) and (9) proves the following Theorem 7.

Theorem 7. *Let (8) be solvable. The sequence $\bar{u}^1, \bar{u}^2, \ldots, \bar{u}^T$ forms a Pareto-Stackelberg equilibrium control in (8) if and only if $\bar{u}^\pi$ is efficient solution of multi-criteria linear programming problem*

$$\begin{array}{l} f_\pi(u^\pi) = \left(c^{\pi\pi}B^\pi + c^{\pi\pi+1}A^\pi B^\pi + c^{\pi\pi+2}A^{\pi+1}A^\pi B^\pi + \cdots + \right. \\ \left. + c^{\pi T}A^{T-1}A^{T-2}\ldots A^\pi B^\pi + b^{\pi\pi}\right)u^\pi \underset{u^\pi}{\rightarrow} \text{ef max}, \\ D^\pi u^\pi \leq d^\pi, \end{array}$$

for $\pi = 1, \ldots, T$.

As above, a particular case of (8) is examined in Theorem 7.

Theorem 8. *If $A^0 = A^1 = \cdots = A^{T-1} = A$, $B^1 = B^2 = \cdots = B^T = B$, and (8) is solvable, then the sequence $\bar{u}^1, \bar{u}^2, \ldots, \bar{u}^T$ forms a Pareto-Stackelberg equilibrium control if and only if $\bar{u}^\pi$ is the efficient solution*

of multi-criteria linear programming problem

$$\begin{aligned}&\left(c^{\pi\pi}B + c^{\pi\pi+1}AB + \cdots + c^{\pi T}(A)^{T-\pi}B + b^{\pi\pi}\right)u^{\pi} \underset{u^{\pi}}{\rightarrow} \text{ef max},\\ &D^{\pi}u^{\pi} \leq d^{\pi},\end{aligned}$$

for $\pi = 1, \ldots, T$.

Pontryagin maximum principle may be extended for (8). Let us consider the following recurrent relations

$$\begin{array}{rcl} p^{\pi T} & = & c^{\pi T}, \\ p^{\pi t} & = & p^{\pi t+1}A^{t} + c^{\pi t}, \quad t = T-1, ..., 1, \end{array} \tag{10}$$

where $\pi = 1, \ldots, T$. Hamiltonian vector-functions are defined as

$$H_{\pi t}\left(u^{t}\right) = \left\langle p^{\pi t}B^{t} + b^{\pi t}, u^{t}\right\rangle, t = T, \ldots, 1, \pi = 1, \ldots, T,$$

where $p^{\pi t}, t = T, \ldots, 1, \pi = 1, \ldots, T$ are defined by (10).

Theorem 9. *Let (8) be solvable. The sequence of controls* $\bar{u}^{1}, \ldots, \bar{u}^{T}$ *forms a Pareto-Stackelberg equilibrium control if and only if*

$$\bar{u}^{\pi} \in \underset{u^{\pi}: D^{\pi}u^{\pi} \leq d^{\pi}}{\text{Arg ef max}} H_{\pi\pi}\left(u^{\pi}\right),$$

for $\pi = 1, \ldots, T$.

By direct substitution of (10) in Hamiltonian functions and by comparing the final results with multi-criteria linear programming problems from Theorem 7 the truth of Theorem 9 arises. Theorems 7 and 9 are equivalent.

It can be remarked especially that method of Pareto-Stackelberg control established by Theorems 7–9 needs solutions of multi-criteria linear programming problems.

5 Linear discrete-time Nash-Stackelberg control problem

Let us modify the problem (5) by considering the control of Nash-Stackelberg type with T stages and $\nu_1 + \nu_2 + \cdots + \nu_T$ players, where $\nu_1, \nu_2, \ldots, \nu_T$ are the numbers of players on stages $1, 2, \ldots, T$. Every player is identified by two numbers (indices) (τ, π), where τ is the number of stage on which player selects his strategy and $\pi \in \{1, 2, \ldots, \nu_\tau\}$ is his number at stage τ. In such game, at each stage τ the players $1, 2, \ldots, \nu_\tau$ play a Nash game by selecting simultaneously their strategies and by communicating his and all precedent selected strategies to the following $\tau + 1$ stage players. After all stage strategy selections, on the resulting profile all the players compute their gains. Such type of control is named Nash-Stackelberg control, and the corresponding problem – linear discrete-time Nash-Stackelberg control problem.

The described decision process may be modelled as it follows:

$$\begin{aligned}
&f_{\tau\pi}(x, u^{\tau\pi}||u^{-\tau\pi}) = \sum_{t=1}^{T}\left(c^{\tau\pi t}x^t + \sum_{\mu=1}^{\nu_t} b^{\tau\pi t\mu}u^{t\mu}\right) \underset{u^{\tau\pi}}{\longrightarrow} \max,\\
&\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau,\\
&x^t = A^{t-1}x^{t-1} + \sum_{\pi=1}^{\nu_t} B^{t\pi}u^{t\pi}, t = 1, ..., T,\\
&D^{t\pi}u^{t\pi} \le d^{t\pi}, t = 1, ..., T, \pi = 1, \ldots, \nu_t,
\end{aligned} \tag{11}$$

where

$x^0, x^t, c^{\tau\pi t} \in R^n$,
$u^t, b^{\tau\pi t\mu} \in R^m$,
$A^{t-1} \in R^{n\times n}$,
$B^{\tau\pi} \in R^{n\times m}$,
$d^{\tau\pi} \in R^k$,
$D^{\tau\pi} \in R^{k\times n}$,
$t, \tau = 1, \ldots, T$,
$\pi = 1, \ldots, \nu_\tau$,
$\mu = 1, \ldots, \nu_t$.

By performing direct transformations

$$
\begin{aligned}
x^1 &= A^0x^0+\sum_{\pi=1}^{\nu_1}B^{1\pi}u^{1\pi},\\
x^2 &= A^1x^1+\sum_{\pi=1}^{\nu_2}B^{2\pi}u^{2\pi}=\\
&= A^1\left(A^0x^0+\sum_{\pi=1}^{\nu_1}B^{1\pi}u^{1\pi}\right)+\sum_{\pi=1}^{\nu_2}B^{2\pi}u^{2\pi}=\\
&= A^1A^0x^0+A^1\sum_{\pi=1}^{\nu_1}B^{1\pi}u^{1\pi}+\sum_{\pi=1}^{\nu_2}B^{2\pi}u^{2\pi},\\
x^3 &= A^2x^2+\sum_{\pi=1}^{\nu_3}B^{3\pi}u^{3\pi}=\\
&= A^2\left(A^1A^0x^0+A^1\sum_{\pi=1}^{\nu_1}B^{1\pi}u^{1\pi}+\sum_{\pi=1}^{\nu_2}B^{2\pi}u^{2\pi}\right)+\\
&\quad+\sum_{\pi=1}^{\nu_3}B^{3\pi}u^{3\pi}=\\
&= A^2A^1A^0x^0+A^2A^1\sum_{\pi=1}^{\nu_1}B^{1\pi}u^{1\pi}+A^2\sum_{\pi=1}^{\nu_2}B^{2\pi}u^{2\pi}+\\
&\quad+\sum_{\pi=1}^{\nu_3}B^{3\pi}u^{3\pi},\\
&\cdots
\end{aligned}
$$

$$
\begin{aligned}
x^T &= A^{T-1}x^{T-1}+\sum_{\pi=1}^{\nu_T}B^{T\pi}u^{T\pi}=\\
&= \prod_{t=0}^{T-1}A^tx^0+\prod_{t=1}^{T-1}A^t\sum_{\pi=1}^{\nu_1}B^{1\pi}u^{1\pi}+\prod_{t=2}^{T-1}A^t\sum_{\pi=1}^{\nu_2}B^{2\pi}u^{2\pi}+\\
&\quad+\cdots+\prod_{t=T-1}^{T-1}A^t\sum_{\pi=1}^{\nu_{T-1}}B^{T-1\pi}u^{T-1\pi}+\sum_{\pi=1}^{\nu_T}B^{T\pi}u^{T\pi},
\end{aligned}
$$

and by subsequent substitution in the objective/cost functions of (11),

the problem (11) is reduced to

$$
\begin{aligned}
& f(u^{\tau\pi}||u^{-\tau\pi}) = \\
& = \left(c^{\tau\pi 1} + c^{\tau\pi 2}A^1 + c^{\tau\pi 3}A^2A^1 + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^1\right)A^0x^0 + \\
& \\
& + \left(c^{\tau\pi 1}B^{11} + c^{\tau\pi 2}A^1B^{11} + c^{\tau\pi 3}A^2A^1B^{11} + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^1B^{11} + b^{\tau\pi 11}\right)u^{11} + \\
& + \left(c^{\tau\pi 1}B^{12} + c^{\tau\pi 2}A^1B^{12} + c^{\tau\pi 3}A^2A^1B^{12} + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^1B^{12} + b^{\tau\pi 12}\right)u^{12} + \\
& + \cdots + \\
& + \left(c^{\tau\pi 1}B^{1\nu_1} + c^{\tau\pi 2}A^1B^{1\nu_1} + c^{\tau\pi 3}A^2A^1B^{1\nu_1} + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^1B^{1\nu_1} + b^{\tau\pi 1\nu_1}\right)u^{1\nu_1} + \\
& \\
& + \left(c^{\tau\pi 2}B^{21} + c^{\tau\pi 3}A^2B^{21} + c^{\tau\pi 4}A^3A^2B^{21} + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^2B^{21} + b^{\tau\pi 21}\right)u^{21} + \\
& + \left(c^{\tau\pi 2}B^{22} + c^{\tau\pi 3}A^2B^{22} + c^{\tau\pi 4}A^3A^2B^{22} + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^2B^{22} + b^{\tau\pi 22}\right)u^{22} + \\
& + \cdots + \\
& + \left(c^{\tau\pi 2}B^{2\nu_2} + c^{\tau\pi 3}A^2B^{2\nu_2} + c^{\tau\pi 4}A^3A^2B^{2\nu_2} + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^2B^{2\nu_2} + b^{\tau\pi 2\nu_2}\right)u^{2\nu_2} + \\
& \\
& + \cdots + \left(c^{\tau\pi T}B^{T\nu_T} + b^{\tau\pi T\nu_T}\right)u^{T\nu_T} \underset{u^{\tau\pi}}{\rightarrow} \max, \\
& \tau = 1,\ldots,T, \pi = 1,\ldots,\nu_\tau, \\
& D^{\tau\pi}u^{\tau\pi} \leq d^{\tau\pi}, \tau = 1,\ldots,T, \pi = 1,\ldots,\nu_\tau.
\end{aligned}
\tag{12}
$$

Evidently, (12) defines a strategic game for which Nash-Stackelberg equilibrium is also Nash equilibrium and it is simply computed as a sequence of solutions of $\nu_1 + \cdots + \nu_T$ linear programming problems

$$
\begin{aligned}
& f(u^{\tau\pi}||u^{-\tau\pi}) = \\
& = \left(c^{\tau\pi\tau}B^{\tau\pi} + c^{\tau\pi\tau+1}A^\tau B^{\tau\pi} + c^{\tau\pi\tau+2}A^{\tau+1}A^\tau B^{\tau\pi} + \cdots + \right. \\
& \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^\tau B^{\tau\pi} + b^{\tau\pi\tau\pi}\right)u^{\tau\pi} \underset{u^{\tau\pi}}{\rightarrow} \max, \\
& D^{\tau\pi}u^{\tau\pi} \leq d^{\tau\pi},
\end{aligned}
\tag{13}
$$

$\tau = 1,\ldots,T, \pi = 1,\ldots,\nu_\tau$.

Equivalence of (11) and (13) proves the following Theorem 10.

Theorem 10. *Let (11) be solvable. The sequence* $\bar{u}^{11}, \bar{u}^{12}, \ldots, \bar{u}^{T\nu_T}$ *forms a Nash-Stackelberg equilibrium control in (11) if and only if* $\bar{u}^{\tau\pi}$ *is optimal in linear programming problem (13), for* $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau$.

Various particular cases of (11) may be examined in Theorem 10, e.g. Theorem 11.

Theorem 11. *If* $A^0 = A^1 = \cdots = A^{T-1} = A$, $B^{11} = B^{12} = \cdots = B^{T\nu_T} = B$, *and (11) is solvable, then the sequence* $\bar{u}^{11}, \bar{u}^{12}, \ldots, \bar{u}^{T\nu_T}$ *forms a Nash-Stackelberg equilibrium control if and only if* $\bar{u}^{\tau\pi}$ *is optimal in linear programming problem*

$$\begin{aligned} & f(u^{\tau\pi}||u^{-\tau\pi}) = \\ & = \left(c^{\tau\pi\tau}B + c^{\tau\pi\tau+1}AB + c^{\tau\pi\tau+2}(A)^2B + \cdots + \right. \\ & \left. + c^{\tau\pi T}(A)^{T-\tau}B + b^{\tau\pi\tau\pi}\right) u^{\tau\pi} \underset{u^{\tau\pi}}{\rightarrow} \max, \\ & D^{\tau\pi}u^{\tau\pi} \leq d^{\tau\pi}, \end{aligned}$$

for $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau$.

Pontryagin maximum principle may be extended for (11). Let us consider the following recurrent relations

$$\begin{array}{rcl} p^{\tau\pi T} & = & c^{\tau\pi T}, \\ p^{\tau\pi t} & = & p^{\tau\pi t+1}A^t + c^{\tau\pi t}, \quad t = T-1, ..., 1, \end{array} \tag{14}$$

where $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau$. Hamiltonian functions are defined as

$$H_{\tau\pi t}\left(u^{\tau\pi}\right) = \left\langle p^{\tau\pi t}B^{\tau\pi} + b^{\tau\pi\tau\pi}, u^{\tau\pi}\right\rangle, t = T, \ldots, 1,$$

where $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau$ and $p^{\tau\pi t}, t = T, \ldots, 1, \tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau$ are defined by (14).

Theorem 12. *Let (11) be solvable. The sequence* $\bar{u}^{11}, \bar{u}^{12}, \ldots, \bar{u}^{T\nu_T}$ *forms a Nash-Stackelberg equilibrium control if and only if*

$$H_{\tau\pi t}\left(\bar{u}^{\tau\pi}\right) = \max_{u^{\tau\pi}: D^{\tau\pi}u^{\tau\pi} \leq d^{\tau\pi}} H_{\tau\pi t}\left(u^{\tau\pi}\right),$$

for $t = T, \ldots, 1, \tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau$.

Theorems 10 and 12 are equivalent.

6 Linear discrete-time Pareto-Nash-Stackelberg control problem

Let us integrate the problems (8) and (11) by considering the control of Pareto-Nash-Stackelberg type with T stages and $\nu_1 + \cdots + \nu_T$ players, where $\nu_1, \ldots, \nu_T$ are the correspondent numbers of players on stages $1, \ldots, T$. Every player is identified by two numbers as above in Nash-Stackelberg control: τ – stage on which player selects his strategy and π – player number at stage τ. In such game, at each stage τ the players $1, 2, \ldots, \nu_\tau$ play a Pareto-Nash game by selecting simultaneously their strategies according to their criteria ($k_{\tau 1}, k_{\tau 2}, \ldots, k_{\tau\nu_\tau}$ are the numbers of criteria of respective players) and by communicating his and all precedent selected strategies to the following $\tau + 1$ stage players. After all stage strategy selections, all the players compute their gains on the resulting profile. Such type of control is named Pareto-Nash-Stackelberg control, and the corresponding problem – linear discrete-time Pareto-Nash-Stackelberg control problem.

The decision control process may be modelled as:

$$\begin{aligned}
&f_{\tau\pi}(x, u^{\tau\pi}||u^{-\tau\pi}) = \sum_{t=1}^{T}\left(c^{\tau\pi t}x^t + \sum_{\mu=1}^{\nu_t} b^{\tau\pi t\mu}u^{t\mu}\right) \underset{u^{\tau\pi}}{\longrightarrow} \text{ef}\max,\\
&\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau,\\
&x^t = A^{t-1}x^{t-1} + \sum_{\pi=1}^{\nu_t} B^{t\pi}u^{t\pi}, t = 1, ..., T,\\
&D^{t\pi}u^{t\pi} \le d^{t\pi}, t = 1, ..., T, \pi = 1, \ldots, \nu_t,
\end{aligned} \tag{15}$$

where $x^0, x^t \in R^n$, $c^{\tau\pi t\mu} \in R^{k_{tp}\times n}$, $u^{\tau\pi} \in R^m$, $b^{\tau\pi t\mu} \in R^{k_{tp}\times n}$, $A^{t-1} \in R^{n\times n}$, $B^{\tau\pi} \in R^{n\times m}$, $d^{\tau\pi} \in R^k$, $D^{\tau\pi} \in R^{k\times n}$, $t, \tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau, \mu = 1, \ldots, \nu_t$.

By performing similar direct transformation as above, (15) is reduced to a sequence of multi-criteria linear programming problems

$$\begin{array}{l} f(u^{\tau\pi}||u^{-\tau\pi}) = \\ = \left(c^{\tau\pi\tau}B^{\tau\pi} + c^{\tau\pi\tau+1}A^{\tau}B^{\tau\pi} + c^{\tau\pi\tau+2}A^{\tau+1}A^{\tau}B^{\tau\pi} + \cdots + \right. \\ \left. + c^{\tau\pi T}A^{T-1}A^{T-2}\ldots A^{\tau}B^{\tau\pi} + b^{\tau\pi\tau\pi}\right) u^{\tau\pi} \underset{u^{\tau\pi}}{\rightarrow} \text{ef max}, \\ D^{\tau\pi}u^{\tau\pi} \leq d^{\tau\pi}, \end{array} \tag{16}$$

$\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_{\tau}$.

Equivalence of (15) and (16) proves the following Theorem 13.

Theorem 13. *Let (15) be solvable. The sequence $\bar{u}^{11}, \bar{u}^{12}, \ldots, \bar{u}^{T\nu_T}$ forms a Pareto-Nash-Stackelberg equilibrium control in (15) if and only if $\bar{u}^{\tau\pi}$ is an efficient solution of multi-criteria linear programming problem (16), for $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_{\tau}$.*

As a corollary, Theorem 14 follows.

Theorem 14. *If $A^0 = A^1 = \cdots = A^{T-1} = A$, $B^{11} = B^{12} = \cdots = B^{T\nu_T} = B$, and (15) is solvable, then the sequence $\bar{u}^{11}, \bar{u}^{12}, \ldots, \bar{u}^{T\nu_T}$ forms a Pareto-Nash-Stackelberg equilibrium control if and only if $\bar{u}^{\tau\pi}$ is an efficient solution of multi-criteria linear programming problem*

$$\begin{array}{l} f(u^{\tau\pi}||u^{-\tau\pi}) = \\ = \left(c^{\tau\pi\tau}B + c^{\tau\pi\tau+1}AB + c^{\tau\pi\tau+2}(A)^2B + \cdots + \right. \\ \left. + c^{\tau\pi T}(A)^{T-\tau}B + b^{\tau\pi\tau\pi}\right) u^{\tau\pi} \underset{u^{\tau\pi}}{\rightarrow} \text{ef max}, \\ D^{\tau\pi}u^{\tau\pi} \leq d^{\tau\pi}, \end{array}$$

for $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_{\tau}$.

Pontryagin maximum principle may be generalized for (15). By considering recurrent relations

$$\begin{array}{rcl} p^{\tau\pi T} & = & c^{\tau\pi T}, \\ p^{\tau\pi t} & = & p^{\tau\pi t+1}A^t + c^{\tau\pi t}, \quad t = T-1, ..., 1, \end{array} \tag{17}$$

where $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_{\tau}$, Hamiltonian vector-functions are defined as

$$H_{\tau\pi t}\left(u^{\tau\pi}\right) = \left\langle p^{\tau\pi t}B^{\tau\pi} + b^{\tau\pi\tau\pi}, u^{\tau\pi}\right\rangle, t = T, \ldots, 1,$$

where $\tau = 1, \ldots, T, \pi = 1, \ldots, \nu_{\tau}$ and $p^{\tau\pi t}, t = T, \ldots, 1, \tau = 1, \ldots, T, \pi = 1, \ldots, \nu_{\tau}$. Remark the vector nature of (17) via (14).

Theorem 15. *Let (15) be solvable. The sequence* $\bar{u}^{11}, \bar{u}^{12}, \ldots, \bar{u}^{T\nu_T}$ *forms a Pareto-Nash-Stackelberg equilibrium control if and only if*

$$\bar{u}^{\tau\pi} \in \underset{u^{\tau\pi}: D^{\tau\pi}u^{\tau\pi} \leq d^{\tau\pi}}{\operatorname{Arg\,ef\,max}} H_{\tau\pi t}\left(u^{\tau\pi}\right),$$

for $t = T, \ldots, 1, \tau = 1, \ldots, T, \pi = 1, \ldots, \nu_\tau$.

Theorems 13 and 12 are equivalent.

7 Linear discrete-time set-valued optimal control problem

The state of controlled system is described above by a point. Indeed, real systems may be treated as n-dimension body, the state of which is described by a set of points in every time moment. Evidently, the initial state of the system is described by initial set $X^0 \subset R^n$. Naturally, the following problem arises

$$\begin{aligned} F(X,U) = \sum_{t=1}^{T} (c^t X^t + b^t U^t) &\to \max, \\ X^t = A^{t-1} X^{t-1} + B^t U^t, &\quad t = 1, ..., T, \\ D^t U^t \leq d^t, &\quad t = 1, ..., T, \end{aligned} \tag{18}$$

where $X^0, X^t \subset R^n$, $c^t \in R^n$, $U^t \subset R^m$, $b^t \in R^m$, $A^{t-1} \in R^{n\times n}$, $B^t \in R^{n\times m}$, $d^t \in R^k$, $D^t \in R^{k\times n}$, $c^t X^t = \left\langle c^t, X^t \right\rangle$, $b^t U^t = \left\langle b^t, U^t \right\rangle$, $t = 1, ..., T$. Linear set operations in (18) are defined obviously (see, e.g., [4]): $AX = \{Ax : x \in X\}$, $\forall X \subset R^n$, $\forall A \in R^{n\times n}$.

The objective set-valued map $F : X \times Y \multimap R$, $F(X,Y) \subset R$ represents a summation of intervals. That is, the optimization of the objective map in problem (11) needs interval arithmetic treatment.

By direct transformations, (18) is transformed into

$$\begin{aligned}
&F(U) = \\
&= \left(c^1 + c^2A^1 + c^3A^2A^1 + \cdots + \right. \\
&\left. + c^TA^{T-1}A^{T-2}\ldots A^1\right)A^0X^0 + \\
&+ \left(c^1B^1 + c^2A^1B^1 + c^3A^2A^1B^1 + \cdots + \right. \\
&\left. + c^TA^{T-1}A^{T-2}\ldots A^1B^1 + b^1\right)U^1 + \\
&+ \left(c^2B^2 + c^3A^2B^2 + c^4A^3A^2B^2 + \cdots + \right. \\
&\left. + c^TA^{T-1}A^{T-2}\ldots A^2B^2 + b^2\right)U^2 + \\
&+ \cdots + \left(c^TB^T + b^T\right)U^T \to \max, \\
&D^tU^t \le d^t, t = 1, \ldots, T.
\end{aligned} \tag{19}$$

The equivalence of the problems (18) and (19) and the form of objective map proves that optimal solution doesn't depend on initial point. The cardinality of every control set $U^1, \ldots, U^T$ is equal to 1. Thus, the Theorem 1 is true for problem (18).

Theorem 16. *Let (18) be solvable. The sequence $\bar{u}^1, \bar{u}^2, \ldots, \bar{u}^T$ forms an optimal control if and only if $\bar{u}^t$ is the solution of linear programming problem*

$$\left(c^tB^t + c^{t+1}A^tB^t + \cdots + c^TA^{T-1}A^{T-2}\ldots A^tB^t + b^t\right)u^t \to \max,$$
$$D^tu^t \le d^t,$$

for $t = 1, \ldots, T$.

Analogous conclusions are true for all problems and theorems considered above.

8 Concluding remarks

There are different types of processes control: optimal control, Stackelberg control, Pareto-Stackelberg control, Nash-Stackelberg control, Pareto-Nash-Stackelberg control, etc.

The direct-straightforward, dual and classical principles (Pontryagin and Bellman) may be applied for determining the desired control of dynamic processes. These principles are the bases for pseudo-polynomial methods, which are exposed as a consequence of theorems for linear discrete-time Pareto-Nash-Stackelberg control problems.

The direct-straightforward principle is applied for solving the problem of determining the optimal control of set-valued linear discrete-time processes. Pseudo-polynomial method of solving is constructed.

The results obtained for different types of set-valued control will be exposed in a future paper.

References

[1] S.A. Ashmanov, A.V. Timohov. *The optimization theory in problems and exercises,* Moscow, Nauka, 1991, pp. 142–143.

[2] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, E.F. Mishchenko. *Mathematical theory of optimal processes,* Moscow, Nauka, 1961 (in Russian).

[3] R. Bellman. *Dynamic Programming,* Princeton, New Jersey, Princeton University Press, 1957.

[4] T. Rockafellar. *Convex Analysis,* Princeton University Press, 1970.

[5] J. von Neuman, O. Morgenstern. *Theory of Games and Economic Behavior,* Annals Princeton University Press, Princeton, NJ, 1944, 2nd ed. 1947.

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

[7] H. Von Stackelberg. *Marktform und Gleichgewicht (Market Structure and Equilibrium),* Springer Verlag, Vienna, 1934.

[8] G. Leitmann. *On Generalized Stackelberg Strategies,* Journal of Optimization Theory and Applications, Vol. 26, 1978, pp.637–648.

[9] V. Ungureanu. *Solution principles for simultaneous and sequential games mixture,* ROMAI Journal, Vol. 4, No.1, 2008, pp. 225–242.

V. Ungureanu

Received December 8, 2012

State University of Moldova,
60, A. Mateevici str.,
Chişinău, MD−2009, Moldova.
E–mail: *v.ungureanu@ymail.com*

Computer Science Journal of Moldova, vol.**21**, no.1(61), 2013

Robust Geometric Programming Approach to Profit Maximization with Interval Uncertainty

Hossein Aliabadi and Maziar Salahi

Abstract

Profit maximization is an important issue to the firms that pursue the largest economic profit possible. In this paper, we consider the profit-maximization problem with the known Cobb-Douglas production function. Its equivalent geometric programming form is given. Then due to the presence of uncertainties in real world modeling, we have assumed interval uncertainties on the model parameters. The robust counterpart is not known to be considered as a geometric program and efficiently solvable using interior point algorithms. Thus using piecewise convex linear approximations, an approximate equivalent of the robust counterpart is given, which is in the form of a geometric programming problem. Finally an example is presented showing the impact of uncertainties.

Keywords: Economic Profit; Geometric Program; Robust Optimization.

1 Introduction

Economic profit is the difference between revenue from selling output and the cost of acquiring the factors necessary to produce it. A profit maximizing firm chooses both its inputs and outputs to achieve maximum economic profits. In other words, the firm seeks to maximize the difference between its total revenue and its total economic costs. If firms are strict profit maximizers, they will adjust those variables that can be controlled until it is impossible to increase profits further [9-14].

Most production functions in the profit maximization problem are represented as power functions. When the production function is represented as a power function, the profit-maximization problem can be treated as a geometric program, a class of nonlinear program which is efficiently solvable using interior point methods [8], while traditionally, the profit-maximization problem is solved by classical method of calculus [1, 5].

In real world applications, model parameters usually involve certain level of uncertainties and thus the original model does not apply anymore. Robust optimization is a new framework which takes into account the parameters uncertainties of the model and solves the underlying problem in the worst case. Several uncertainty sets have been considered in the literature. In this paper, we consider interval uncertainties on parameters of the model [2]. The robust counterpart is not known to be in the form of a tractable geometric programming problem [7]. Thus we give upper and lower piecewise convex linear approximations of it that are efficiently solvable using interior point methods [3, 8]. Finally an illustrative example is presented to show the importance of the model parameters uncertainties.

2 Mathematical Model and Robust Counterpart

Let us consider the following short-run profit maximization problem

$$\begin{aligned} \max \quad & p(Ax_1^{\alpha}x_2^{\beta}) - v_1x_1 - v_2x_2 \\ s.t. \quad & x_1 \leq k, \end{aligned}$$

where $Ax_1^{\alpha}x_2^{\beta}$ is the known Cobb-Douglas production function, p is the market price per unit, A is the scale of production, α and β are the output elasticities, x_i and v_i are ith input quantity and output price, respectively and k is a given constant that restricts the quantity of x_1 [4,9-14]. To solve this problem using efficient algorithms like interior point methods, we may write it as follows

$$\begin{aligned} \max \quad & \pi \\ s.t. \quad & pAx_1^{\alpha}x_2^{\beta} - v_1x_1 - v_2x_2 \geq \pi, \\ & x_1 \leq k, \end{aligned} \tag{1}$$

or in the geometric programming form

$$\begin{aligned} \min \quad & \pi^{-1} \\ s.t. \quad & \pi p^{-1}A^{-1}x_1^{-\alpha}x_2^{-\beta} + v_1p^{-1}A^{-1}x_1^{1-\alpha}x_2^{-\beta} + v_2p^{-1}A^{-1}x_1^{-\alpha}x_2^{1-\beta} \leq 1, \\ & k^{-1}x_1 \leq 1, \end{aligned} \tag{2}$$

where π, x_1, x_2 and p, A, v_1, v_2, k, α, β are variables and parameters respectively. Moreover, by the change of variables

$$z_1 = \log(\pi),\ z_2 = \log(x_1),\ z_3 = \log(x_2)$$

and taking logarithm of the objective function and constraints and finally using the following notation

$$lse(z_1, \ldots, z_k) = v \log(e^{z_1} + \cdots + e^{z_k})$$

we have the following equivalent convex form of (1) which is efficiently solvable by Mehrotra's predictor-corrector interior point method [8]:

$$\begin{aligned} \min \quad & [-1 \quad 0 \quad 0]z \\ s.t.\ & lse([1 \quad -\alpha \quad -\beta]z + b_1,\ [0 \quad 1-\alpha \quad -\beta]z + b_2, \\ & \qquad [0 \quad -\alpha \quad 1-\beta]z + b_3) \leq 0, \\ & [0 \quad 1 \quad 0]z + b_4 \leq 0, \end{aligned} \tag{3}$$

where $z = (z_1 \quad z_2 \quad z_3)^T$ are variables and

$$b_1 = \log(p^{-1}A^{-1}),\ b_2 = \log(v_1p^{-1}A^{-1}),$$

$$b_3 = \log(v_2p^{-1}A^{-1}),\ b_4 = \log(k^{-1}).$$

Now suppose that parameters α, β, p, v_1, v_2, k are subject to interval uncertainties, namely

$$\begin{aligned}
\alpha - \varepsilon_1 \le \alpha \le \alpha + \varepsilon_1 \quad &\text{or} \quad \alpha + u_1\varepsilon_1 \quad |u_1| \le 1,\\
\beta - \varepsilon_2 \le \beta \le \beta + \varepsilon_2 \quad &\text{or} \quad \beta + u_2\varepsilon_2 \quad |u_2| \le 1,\\
p - \varepsilon_3 \le p \le p + \varepsilon_3 \quad &\text{or} \quad p + u_3\varepsilon_3 \quad |u_3| \le 1,\\
v_1 - \varepsilon_4 \le v_1 \le v_1 + \varepsilon_4 \quad &\text{or} \quad v_1 + u_4\varepsilon_4 \quad |u_4| \le 1,\\
v_2 - \varepsilon_5 \le v_2 \le v_2 + \varepsilon_5 \quad &\text{or} \quad v_2 + u_5\varepsilon_5 \quad |u_5| \le 1,\\
k - \varepsilon_6 \le k \le k + \varepsilon_6 \quad &\text{or} \quad k + u_6\varepsilon_6 \quad |u_6| \le 1.
\end{aligned}$$

In the sequel we show how these uncertainties affect $b_1, \ldots, b_4$. We have

$$b_1 = -\log p - \log A,\ b_2 = b_1 + \log v_1,\ b_3 = b_1 + \log v_2,\ b_4 = -\log k.$$

Let us consider the following case:

$$p - \varepsilon_3 \le p \le p + \varepsilon_3 \text{ implies } \log(p - \varepsilon_3) \le \log p \le \log(p + \varepsilon_3)$$

or

$$\log p - \log p + \log(p - \varepsilon_3) \le \log p \le \log p - \log p + \log(p + \varepsilon_3)$$

or

$$\log p + \log p^{-1} + \log(p - \varepsilon_3) \le \log p \le \log p + \log p^{-1} + \log(p + \varepsilon_3).$$

Thus we have

$$\log p + \log(1 - p^{-1}\varepsilon_3) \le \log p \le \log p + \log(1 + p^{-1}\varepsilon_3).$$

Since p, $\varepsilon_3 \ge 0$, then we have

$$1 + p^{-1}\varepsilon_3 \le \frac{1}{1 - p^{-1}\varepsilon_3} = (1 - p^{-1}\varepsilon_3)^{-1},$$

thus

$$\log(1 + p^{-1}\varepsilon_3) \le -\log(1 - p^{-1}\varepsilon_3).$$

Using this at the previous inequality we have

$$\begin{aligned}\log p + \log(1 - p^{-1}\varepsilon_3) \leq \log p \leq \log p + \log(1 + p^{-1}\varepsilon_3)\\ \leq \log p - \log(1 - p^{-1}\varepsilon_3)\end{aligned}$$

or

$$\log p - (-\log(1 - p^{-1}\varepsilon_3)) \leq \log p \leq \log p + (-\log(1 - p^{-1}\varepsilon_3)).$$

Thus the uncertainty for $\log p$ approximately is as follow

$$\log p - \delta_1 \leq \log p \leq \log p + \delta_1, \quad \delta_1 = -\log(1 - p^{-1}\varepsilon_3).$$

Analogously, for the other parameters we have

$$\begin{aligned}
&b_1 - \delta_1 \leq b_1 \leq b_1 + \delta_1,\ \delta_1 = -\log(1 - p^{-1}\varepsilon_3) \Rightarrow b_1 + u_7\delta_1,\ |u_7| \leq 1\\
&b_2 - \delta_2 \leq b_2 \leq b_2 + \delta_2,\ \delta_2 = \delta_1 - \log(1 - v_1^{-1}\varepsilon_4) \Rightarrow b_2 + u_8\delta_2,\ |u_8| \leq 1\\
&b_3 - \delta_3 \leq b_3 \leq b_3 + \delta_3,\ \delta_3 = \delta_1 - \log(1 - v_2^{-1}\varepsilon_5) \Rightarrow b_3 + u_9\delta_3,\ |u_9| \leq 1\\
&b_4 - \delta_4 \leq b_4 \leq b_4 + \delta_4,\ \delta_4 = -\log(1 - k^{-1}\varepsilon_6) \Rightarrow b_4 + u_{10}\delta_4,\ |u_{10}| \leq 1.
\end{aligned}$$

Now the approximate robust counterpart of (3) is as follows:

$$\begin{aligned}
&\min \quad \bar{c}^T t\\
&\sup_{u\in U} lse \begin{pmatrix} \begin{bmatrix}1 & -\alpha + u_1\varepsilon_1 & -\beta + u_2\varepsilon_2 & b_1 + u_7\delta_1\end{bmatrix} t,\\ \begin{bmatrix}0 & (1-\alpha) + u_1\varepsilon_1 & -\beta + u_2\varepsilon_2 & b_2 + u_8\delta_2\end{bmatrix} t,\\ \begin{bmatrix}0 & -\alpha + u_1\varepsilon_1 & (1-\beta) + u_2\varepsilon_2 & b_3 + u_9\delta_3\end{bmatrix} t \end{pmatrix} \leq 0,\\
&\sup_{u\in U} \left(\begin{bmatrix}0 & 1 & 0 & b_4 + u_{10}\delta_4\end{bmatrix} t\right) \leq 0,
\end{aligned} \tag{4}$$

where $\bar{c}^T = \begin{bmatrix}-1 & 0 & 0 & 0\end{bmatrix}$, $|u_i| \leq 1$, $t = \begin{pmatrix} z \\ w \end{pmatrix} \in \mathbb{R}^4$, $w = 1$.

In general it is not known whether (4) can be put in a geometric programming form, thus in the sequel we give an approximate model of (4). To do so, the first three-term constraint is approximated by two

two-term constraints as follows [3]:

$$\begin{aligned}
\min \quad & c^T y \\
& \sup_{|u_i|\le 1} lse([1 \quad -\alpha+u_1\varepsilon_1 \quad -\beta+u_2\varepsilon_2 \quad b_1+u_7\delta_1 \quad 0]y, \\
& \quad [0 \quad 0 \quad 0 \quad 0 \quad 1]y) \le 0, \qquad (5) \\
& \sup_{|u_i|\le 1} lse([0 \quad (1-\alpha)+u_1\varepsilon_1 \quad -\beta+u_2\varepsilon_2 \quad b_2+u_8\delta_2 \quad -1]y, \\
& \quad [0 \quad -\alpha+u_1\varepsilon_1 \quad (1-\beta)+u_2\varepsilon_2 \quad b_3+u_9\delta_3 \quad -1]y) \le 0, \\
& \sup_{|u_i|\le 1} lse([0\ 1 \quad 0 \quad b_4+u_{10}\delta_4 \quad 0]y) \le 0,
\end{aligned}$$

where

$$\begin{aligned}
& c^T = [-1 \quad 0 \quad 0 \quad 0 \quad 0],\ \bar{a}_1 = [1 \quad -\alpha \quad -\beta \quad b_1 \quad 0], \\
& \bar{a}_2 = [0 \quad 1-\alpha \quad -\beta \quad b_2 \quad -1], \bar{a}_3 = [0 \quad -\alpha \quad 1-\beta \quad b_3 \quad -1], \\
& \bar{a}_4 = [0 \quad 1 \quad 0 \quad b_4 \quad 0],\ B_1 = [0 \quad \varepsilon_1 \quad \varepsilon_2 \quad \delta_1 \quad 0]^T, \\
& B_2 = [0 \quad \varepsilon_1 \quad \varepsilon_2 \quad \delta_2 \quad 0]^T,\ B_3 = [0 \quad \varepsilon_1 \quad \varepsilon_2 \quad \delta_3 \quad 0]^T, \\
& B_4 = [0 \quad 0 \quad 0 \quad \delta_4 \quad 0]^T,\ |u_i| \le 1,\ y = \begin{pmatrix} t \\ s \end{pmatrix} \in \mathbb{R}^5.
\end{aligned}$$

Since (5) still is not in the form of a problem which could be easily solved, thus we assume both the lower and upper convex piecewise linear approximations of the constraints to satisfy the same inequality [3]. For example the lower three-term approximation of a two-term constraint is as follows:

$$\sup_{u\in U} lse(x, y) = \max\{x,\ y,\ 0.5x + 0.5y + 0.693\}$$

Thus we consider to have

$$\begin{aligned}
& \sup_{u\in U}([0 \quad (1-\alpha)+u_1\varepsilon_1 \quad -\beta+u_2\varepsilon_2 \quad b_2+u_8\delta_2 \quad -1]y) = \\
& \quad \sup_{u\in U}(\bar{a_2}y + u_1\varepsilon_1 y_2 + u_2\varepsilon_2 y_3 + u_8\delta_2 y_4) \le \\
& \bar{a_2}y + |\varepsilon_1 y_2| + |\varepsilon_2 y_3| + |\delta_2 y_4| = \bar{a_2}y + \sum_{i=1}^{5} |(B_2)_i y_i| \le 0,
\end{aligned}$$

$$\sup_{u\in U}(0.5 * [0 \quad (1-\alpha) + u_1\varepsilon_1 \quad -\beta + u_2\varepsilon_2 \quad b_2 + u_8\delta_2 \quad -1]y +$$
$$0.5 * [0 \quad -\alpha + u_1\varepsilon_1 \quad (1-\beta) + u_2\varepsilon_2 \quad b_3 + u_9\delta_3 \quad -1]y) + 0.693 \leq$$
$$0.5\bar{a}_2 y + 0.5\bar{a}_3 y + 0.5\left(\sum_{i=1}^{5}|(B_2)_i y_i| + \sum_{i=1}^{5}|(B_3)_i y_i|\right) + 0.693 \leq 0,$$

and

$$\sup_{u\in U}([0 \quad -\alpha + u_1\varepsilon_1 \quad (1-\beta) + u_2\varepsilon_2 \quad b_3 + u_9\delta_3 \quad -1]y) =$$
$$\sup_{u\in U}(\bar{a}_3 y + u_1\varepsilon_1 y_2 + u_2\varepsilon_2 y_3 + u_9\delta_3 y_4) \leq$$
$$\bar{a}_3 y + |\varepsilon_1 y_2| + |\varepsilon_2 y_3| + |\delta_3 y_4| = \bar{a}_3 y + \sum_{i=1}^{5}|(B_3)_i y_i| \leq 0.$$

Moreover, to have more accurate results, we have used 25 term piecewise convex linear approximations that are derived using the algorithm in [3]. In Table 1, we have given several best lower piecewise convex linear approximations of $lse(x, y)$ function. It is worth to note that in [3] up to five term approximations are reported.

Furthermore, in the following table the errors of piecewise convex linear approximations up to 25 terms are given for two levels of uncertainties.

3 Example

Let us consider the following values for the parameters in (1)

$$p = 20,\ A = 40,\ \alpha = 0.1,\ \beta = 0.4,\ v_1 = 10,\ v_2 = 35,\ k = 30.$$

Using Mehrotra's predictor-corrector interior point algorithm, the optimal objective value of (1) is 3399.55. However, if we take the following uncertainty set parameters

$$|u_i| \leq 0.5,\ \varepsilon_1 = 0.003,\ \varepsilon_2 = 0.007,\ \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = \varepsilon_6 = 0.5,$$

Table 1. Best lower piecewise linear approximations of $lse(x, y)$

r Number of terms	Error of approximation	The best lower r term piecewise linear approximation of $lse(x, y)$
3	0.223	$\max\{x, y, 0.5x + 0.5y + 0.693\}$
4	0.109	$\max \begin{cases} x, 0.271x + 0.729y + 0.584, \\ 0.729x + 0.271y + 0.584, y \end{cases}$
5	0.065	$\max \begin{cases} x,\ 0.167x + 0.833y + 0.450, \\ 0.5x + 0.5y + 0.693, \\ 0.833x + 0.167y + 0.450,\ y \end{cases}$
25	0.002076	$\max \begin{cases} x, 0.0056x + 0.9944y + 0.0348, 0.0056y + 0.9944x + 0.0348, \\ 0.0195x + 0.9805y + 0.0962, 0.0195y + 0.9805x + 0.0962, \\ 0.0414x + 0.9586y + 0.1724, 0.0414y + 0.9586x + 0.1724, \\ 0.0709x + 0.9291y + 0.2560, 0.0709y + 0.9291x + 0.2560, \\ 0.1075x + 0.8925y + 0.3413, 0.1075y + 0.8925x + 0.3413, \\ 0.1506x + 0.8494y + 0.4238, 0.1506y + 0.8494x + 0.4238, \\ 0.1995x + 0.8005y + 0.4997, 0.1995y + 0.8005x + 0.4997, \\ 0.2534x + 0.7466y + 0.5660, 0.2534y + 0.7466x + 0.5660, \\ 0.3114x + 0.6886y + 0.6202, 0.3114y + 0.6886x + 0.6202, \\ 0.3725x + 0.6275y + 0.6603, 0.3725y + 0.6275x + 0.6603, \\ 0.4357x + 0.5643y + 0.6849, 0.4357y + 0.5643x + 0.6849, \\ 0.5x + 0.5y + 0.693,\ y \end{cases}$

Table 2. Error of piecewise convex linear approximation

Errors	$\|u_i\| \leq 0.5$	$\|u_i\| \leq 1$
Error of 3 terms approximation	0.9184	0.9184
Error of 4 terms approximation	0.2876	0.3104
Error of 5 terms approximation	0.1986	0.1986
.....	.....	.....
Error of 16 terms approximation	0.0138	0.0138
.....	.....	.....
Error of 25 terms approximation	0.0055	0.0058

and solve its upper and lower approximations, then the optimal solutions of the upper and lower 25 term piecewise convex linear approximation of (5) are 2980.7 and 2964.3, respectively. Thus the lower bound for the optimal solution of (4) is 2964.3. As one can see, the range to which the optimal solution of the robust problem belongs, [2964.3, 2980.7], is significantly different than the original optimal objective value. Thus a slight uncertainty in the input parameter might lead to significant change of the optimal objective value. Moreover, if we solve the approximate robust model of (5) for the case, where

$$|u_i| \leq 1,\ \varepsilon_1 = 0.003,\ \varepsilon_2 = 0.007,\ \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = \varepsilon_6 = 1,$$

then the values of the upper and lower 25 term piecewise convex linear approximations are 2797.6 and 2781.6, respectively. Thus the lower bound for the optimal solution of (4) is 2781.6. A similar observation as in the previous case holds here as well. We should note that all computations are done in MATLAB 7.8 and we have used cvx software package [6] to solve problem (5).

4 Conclusions

Extensive research specially in the last decade shows that robust optimization can alleviate sensitivity of a given problem to its data un-

certainty by incorporating explicitly data uncertainty into the problem. In this paper, we consider the robust counterpart of the profit-maximization problem which is in the form of a geometric programming problem. Since it is not known in general that a robust geometric programming problem can be reformulated as a tractable optimization problem that interior point or other algorithms can efficiently solve, then using piecewise convex linear approximations, an approximate equivalent of the robust counterpart is given, which is in the form of a geometric programming problem. Moreover, an illustrative example is given which shows the importance and impact of the uncertainties in the model parameters for different level of uncertainties. Due to the presence of uncertainty in economical model parameters, the idea might be useful to be applied to other models.

5 Acknowledgment

The authors would like to thank the reviewer for his/her valuable comments on the early version of this paper.

References

[1] C.S. Beightler, D.T. Philips, *A*pplied Geometric Programming, John Wiley & Sons, New York, 1976.

[2] A. Ben-Tal and A. Nemirovski, Robust convex optimization, *Mathematics of Operations Research,* **23(4)** 1998, 769–805.

[3] S. Boyd, S. J. Kim, L. Vandenberghe, and A. Hassibi, A tutorial on geometric programming, *Optimization and Engineering,* **8(1)** 2007, 67–127.

[4] P.H. Douglas, The Cobb-Douglas production function once again: its history, its testing, and some new empirical values, *Journal of Political Economy,* **84(5)** 1976, 903–916.

[5] R.J. Duffin, E.L. Peterson, and C. Zener, *G*eometric Programming: Theory and Applications, John Wiley & Sons, New York, 1967.

[6] M. Grant and S. Boyd,Cvx users, guide for cvx version 1.21, www.Stanford.edu/ boyd/cvx, 2010.

[7] K.L. Hsiung, S.K. Kim, S. Boyd, Tractable approximate robust geometric programming, *Optimimization and Engineering*, **9** 2008, 95–118.

[8] K. Kortanek, X. Xu, Y. Ye, An infeasible interior-point algorithm for solving primal and dual geometric programs, *Mathematical Programming,* **76(1)** 1997, 155–181.

[9] S.T. Liu, Using geometric programming to profit maximization with interval coefficients and quantity discount, *Applied Mathematics and Computation,* **209(2)** 2009, 259–265.

[10] S.T. Liu, Posynomial geometric programming with interval exponents and coefficients, *European Journal of Operational Research,* **186(1)** 2008, 17–27.

[11] S.T. Liu, A computational method for the maximization of long-run and short-run profit, *Applied Mathematics and Computation,* **186(2)** 2007, 1104–1112.

[12] S.T. Liu, Profit maximization with quantity discount: An application of geometric programming, *Applied Mathematics and Computation,* **190(2)** 2007, 1723–1729.

[13] S.T. Liu, A geometric programming approach to profit maximization, *Applied Mathematics and Computation,* **182(2)** 2006, 1093–1097.

[14] S.T. Liu, Posynomial geometric programming with parametric uncertainty, *European Journal of Operational Research,* **168(2)** 2006, 345–353.

Hossein Aliabadi[1], Maziar Salahi[2] Received February 10, 2012

[1] Department of Applied Mathematics, Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran,
E–mail: *hosein.aliabadi@gmail.com*

[2] Department of Applied Mathematics,
Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran,
E–mails: *salahim@guilan.ac.ir, salahi.maziar@gmail.com.*

Computer Science Journal of Moldova, vol.**21**, no.1(61), 2013

ECO-generation for some restricted classes of compositions

Jean-Luc Baril, Phan-Thuan Do

Abstract

We study several restricted classes of compositions by giving one-to-one maps between them and different classes of restricted binary strings or pattern avoiding permutations. Inspired by the ECO method [8], new succession rules for these classes are presented. Finally, we obtain generating algorithms in Constant Amortized Time (CAT) for theses classes.

Keywords : Composition of an integer, ECO method, succession rule, generating tree, pattern avoiding permutation.

1 Introduction

A *composition* of an integer n is an ordered collection of one or more positive integers whose sum is n. So, a composition c of n can be written $c = (c_1, c_2, \ldots, c_k)$ with $c_1 + c_2 + \cdots + c_k = n$ and $c_i \geq 1$ for all $i \leq k$. The integer k corresponds to the number of *parts* of the composition. Let $C(n)$ be the set of compositions of n. It is well known that the cardinality of $C(n)$ is 2^{n-1} and there is a one-to-one correspondence between $C(n)$ and binary strings of length $n-1$ (see Definition 1). There are many studies about enumeration of compositions and their restrictions: $(1,p)$-compositions, *i.e.,* compositions whose parts are 1 or p have been introduced in [12, 15, 16]; compositions with no occurrence of part p have been studied in [17]; see also [1, 14, 24, 25, 27, 28, 30, 32]. However, a very few articles deal with their exhaustive generations. Some Gray codes are given for compositions of a positive integer n in [31, 37]; for compositions with parts of

size smaller than p in [36]; or for $(1,p)$-compositions in [12, 15]. These papers mostly study the classes of compositions in terms of binary strings. On the other hand, in [30], some results are provided using restricted permutations for a few classes of compositions, but they cannot be considered as avoidance patterns. More recently, a generalization of the Simion-Schmidt injection [35] gave a bijection between binary strings and pattern avoiding permutations [29] which creates a natural link between compositions and pattern avoiding permutations. For example, the class of compositions is in one-to-one correspondence with the class of permutations avoiding 321 and 312 [4, 26]; the set of compositions of n with all parts of sizes smaller than $(p+1)$ is enumerated by the p-generalized Fibonacci numbers, see [4, 11, 27, 28] and there is a bijection between this set and permutations avoiding the patterns 321, 312 and $234\cdots(p+1)1$.

In this paper, we use the ECO method [8] (Enumeration Combinatorial Object method) in order to generate some restricted classes of compositions represented as binary strings or pattern avoiding permutations. The *ECO method* is a recursive description of a combinatorial object class which explains how an object of size n can be reached from one and only one object of smaller size (see for example [2, 3, 5, 6, 7, 13, 18, 19, 21, 22, 23]). It consists to define a system of *succession rules* for a combinatorial object class which induces a generating tree such that each node is labeled by the number of its successors. In fact, the set of successions rules describes for each node the label of its successors. More formally, the root of the generating tree is labeled (b), $b \in \mathbb{N}^+$, and we define the rules Ω:

$$\{(k) \rightsquigarrow (e_1(k))(e_2(k))\cdots(e_k(k)), k \in \mathbb{N}\},$$

where $e_i : \mathbb{N}^+ \longrightarrow \mathbb{N}^+$. This means that each node labeled (k) has k successors labeled $(e_1(k))$, $(e_2(k))$,..., $(e_k(k))$. For $\ell \geq 1$, the symbol $\overset{\ell}{\rightsquigarrow}$ means that the succession rule transforms an element of size n into another one of size $n+\ell$. For $\ell = 1$ we frequently omit the superscript ℓ over $\rightsquigarrow$.

By coding each node of the generating tree with either a binary string or a permutation, we deduce new bijections between classes of

restricted compositions, pattern avoiding permutations and restricted binary strings.

This paper is organized as follows. Section 2 recalls the definition of pattern avoiding permutations, and gives existing links between compositions, binary strings and pattern avoiding permutations. Sections from 3 to 7 present succession rules for compositions with a given number of parts, compositions with at most p parts, $(1, p)$-compositions and compositions without parts of a given size. Moreover, each induced generating tree will be encoded by binary strings and pattern avoiding permutations. Finally, we deduce efficient algorithms (Constant Amortized Time algorithms) for generating all these classes (Constant Amortized Time means that the total amount of computation divided by the number of objects is bounded by a constant independent of the size of objects).

2 Definitions and notations

Let $\mathfrak{S}_n$ be the set of permutations on $[n] = \{1, 2, \ldots, n\}$. We represent a permutation $\pi \in \mathfrak{S}_n$ in one line notation: *i.e.,* $\pi = \pi_1\pi_2 \cdots \pi_n$, where $\pi_i = \pi(i)$ for all $i \leq n$. A permutation $\pi \in \mathfrak{S}_n$ contains the pattern $\tau \in \mathfrak{S}_k$ if and only if a sequence of indices $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ exists such that $\pi(i_1)\pi(i_2)\cdots\pi(i_k)$ is order-isomorphic to τ. We denote by $\mathfrak{S}_n(\tau)$ the set of n-length permutations avoiding the pattern τ, *i.e.,* permutations that do not contain τ. For instance, the permutation **52**3**1**64 contains the pattern 321 while $314265 \in \mathfrak{S}_6(321)$. Moreover, we consider a *barred pattern* $\bar{\tau}$, *i.e.,* a permutation in $\mathfrak{S}_k$ having a bar over one or several consecutive entries (see [34]). Let r, $1 \leq r \leq k-1$, be the number of barred elements in $\bar{\tau}$; τ be the permutation on $[k]$ identical to $\bar{\tau}$ but unbarred; and $\hat{\tau}$ be the permutation on $[k-r]$ made up of the $k-r$ unbarred elements of $\bar{\tau}$ rewritten to be a permutation on $[k-r]$. Let $b = b_1 \cdots b_k \in \{0,1\}^k$ such that $b_i = 1$ if and only if the i-th entry of $\bar{\tau}$ is barred. Then $\pi \in \mathfrak{S}_n$ avoids the pattern $\bar{\tau}$ if and only if each pattern $\hat{\tau}$ in π can be expanded into a pattern τ in π such that the positions of the extended entries correspond to the positions of 1s in b. For example, if $\bar{\tau} = 21\overline{34}$, then $b = 0011$ and

$2134 \in \mathfrak{S}_4(\bar{\tau})$ and $21435 \notin \mathfrak{S}_5(\bar{\tau})$ since 43 can not be expanded into a pattern $43ab$, where $4 < a < b$. Now we define a special pattern denoted $\dot{\tau} = 23\cdots(p-1)p\dot{1}$ (see [10]). A permutation π avoids $\dot{\tau}$ if and only if each pattern $23\cdots(p-1)1$ can be extended to a pattern $23\cdots(p-1)p1$ such that the positions of the extended values do not matter, *i.e.*, each pattern $23\cdots(p-1)1$ is contained in a pattern $23\cdots(p-1)p1$ of π. For instance, if $\dot{\tau} = 23\dot{1}$, then $2341 \in \mathfrak{S}_4(\dot{\tau})$ and $21 \notin \mathfrak{S}_2(\dot{\tau})$ while $2341 \notin \mathfrak{S}_4(23\bar{4}1)$.

It is well-known that the set $\mathcal{C}(n+1)$ of compositions of $n+1$ is in one-to-one correspondence with the set $\mathcal{B}(n)$ of binary strings of length n. The following bijection φ shows this correspondence.

Definition 1 *Let $c = (c_1, c_2, \ldots, c_k)$ be a composition of $n+1$. The bijection φ between $\mathcal{C}(n+1)$ and $\mathcal{B}(n)$ is defined by: $\varphi(c) = 1^{c_1-1}01^{c_2-1}0\cdots 1^{c_{k-1}-1}01^{c_k-1}$.*

For instance, if $c = (1,3,2,3)$, then $\varphi(c) = 0\mathbf{11}010\mathbf{11}$.

On the other hand, Juarna and Vajnovszki in [29] gave a bijection ϕ between the binary strings in $\mathcal{B}(n)$ and the permutations in $\mathfrak{S}_{n+1}(321, 312)$. This bijection is considered as a generalization of the Simion-Schmidt injection [35].

Definition 2 *Let $b = b_1b_2\cdots b_n \in \mathcal{B}(n)$. The bijection ϕ between $\mathcal{B}(n)$ and $\mathfrak{S}_{n+1}(321, 312)$ is defined by: $\pi = \phi(b) \in \mathfrak{S}_{n+1}$ which has its i-th value π_i given by the following rule: if $X_i = [n+1] \setminus \{\pi_1, \pi_2, \ldots, \pi_{i-1}\}$, then*

$$\pi_i = \begin{cases} \text{the minimum value in } X_i, \text{ if } b_i = 0, \text{ or } i = n+1 \\ \text{the second minimum value in } X_i, \text{ if } b_i = 1. \end{cases}$$

For instance, if $b = \mathbf{0}11\mathbf{0}1\mathbf{0}11$, then $\phi(b) = \mathbf{1}34\mathbf{2}6\mathbf{5}89\mathbf{7}$.

3 Compositions of n with parts of size at most p

The set $\mathcal{C}_{\leq p}(n)$ of compositions of n with parts of size at most p is enumerated by the $(n+1)$-th p-generalized Fibonacci number (see [11]).

The map φ (Section 2) induces a bijection between $\mathcal{C}_{\leq p}(n)$ and the set $\mathcal{B}_{<p}(n-1)$ of binary strings of size $n-1$ without p consecutive ones. It is proved [4, 11] that there are also one-to-one correspondences with the two classes of permutations $\mathfrak{S}_n(321, 231, (p+1)12\cdots p)$ and $\mathfrak{S}_n(321, 312, 23\cdots(p+1)1)$. These permutation classes admit known succession rules (see [4, 26] and Table 1) and they can be generated in constant amortized time (see also [11, 9, 33, 36] for Gray code listing).

4 Compositions of n with exactly p parts

The set $\mathcal{C}_p(n)$ of compositions of n with exactly p parts is enumerated by the binomial coefficient $\binom{n-1}{p-1}$. Also, $\mathcal{C}_p(n)$ is in one-to-one correspondence with the set $\mathcal{B}_{p-1}(n-1)$ of binary strings of length $n-1$ and having exactly $p-1$ zeros. The function φ (see Section 2) shows such a bijection. The following theorem gives a system of succession rules in order to generate the sets $\mathcal{B}_{p-1}(n-1)$ and $\mathfrak{S}_n(132, 312, (p+1)\cdots 21, 12\cdots(n-p+1)(n-p+2))$. In this part, we say that the level of a node in the generating tree is the length of the unique path between the root and this node, plus p (thus the root is on the level p).

Theorem 1 *For $p \geq 1$, a system (Ω_p) of succession rules for the set $\mathcal{C}_p(n)$ is:*

$$(\Omega_p)\begin{cases}(p)\\(k) \rightsquigarrow (1)(2)\cdots(k-1)(k).\end{cases}$$

Each level $n \geq p$ of the generating tree induced by (Ω_p) can be coded by the binary strings of $\mathcal{B}_{p-1}(n-1)$ or by the permutations in $\mathfrak{S}_n(132, 312, (p+1)\cdots 21, 12\cdots(n-p+1)(n-p+2))$. A node other than the root and labeled (k) is coded by a binary string of the form $b = b'10^{k-1}$ (resp. a permutation $\pi = \pi' n(k-1)(k-2)\cdots 21$) and its successors are obtained from b (resp. π) either by inserting 1 (resp. $n+1$) between two entries of the suffix 10^{k-1} (resp. $n(k-1)(k-2)\cdots 21$) or by appending 1 (resp. $n+1$) on the right (see Figure 1).

Proof. We attach the binary string 0^{p-1} to the root of the generating tree obtained by (Ω_p) and we proceed by induction on the level n of

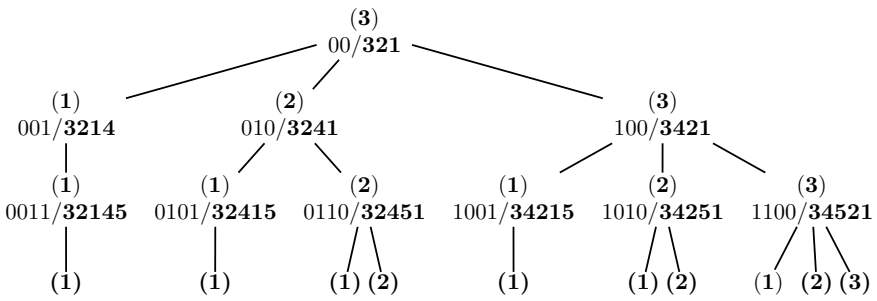


Figure 1. The first levels of the generating tree (Ω_3) (the level of the root is 3). Each node on the level n is coded by one binary string in $\mathcal{B}_2(n-1)$ or by one permutation in $\mathfrak{S}_n(132, 312, 4321, 12\cdots(n-2)(n-1))$.

the tree (the root being on the level p by convenience). So we assume that the level $(n-1)$ generates once each binary string of $\mathcal{B}_{p-1}(n-2)$. Let $b \in \mathcal{B}_{p-1}(n-2)$ such that $b = b'0^k$, $k \leq p-1$, where b' is either empty or has 1 on its right. Therefore, by inserting 1 on the right of b, or between two entries of the suffix 0^k, or on the left of 0^k, we produce $k+1$ binary strings of $\mathcal{B}_{p-1}(n-1)$ and each binary string $c = b'10^{k-\ell+1}10^{\ell-1}$ obtained by this process has ℓ successors labeled (1), (2), $\ldots$, (ℓ). Conversely, each binary string of $\mathcal{B}_{p-1}(n-1)$ can be uniquely obtained from an element of $\mathcal{B}_{p-1}(n-2)$ by this construction.

Now, we define a map ϕ' from $\mathcal{B}_{p-1}(n-1)$ to $\mathfrak{S}_n(132, 312)$.

Let $b = b_1b_2\cdots b_{n-1} \in \mathcal{B}_{p-1}(n-1)$. If $\pi = \phi'(b_1b_2\cdots b_{n-1})$, then $\pi_1 = p$ and for $i \geq 2$,

$$\pi_i = \begin{cases} p-\ell & \text{if } b_{i-1} \text{ is the } \ell\text{-th 0 from the left} \\ p+\ell & \text{if } b_{i-1} \text{ is the } \ell\text{-th 1 from the left.} \end{cases}$$

For instance, if $p = 5$ and $n = 9$, then $\phi'(1\mathbf{00}11\mathbf{0}1\mathbf{0}) = \mathbf{5}6\mathbf{4}3\mathbf{7}8\mathbf{2}\mathbf{9}\mathbf{1}$. In fact, the image by ϕ' of an element in $\mathcal{B}_{p-1}(n-1)$ is a permutation of $\mathfrak{S}_n(132, 312)$ verifying $\pi_1 = p$, or equivalently a permutation of $\mathfrak{S}_n(132, 312)$ that avoids the two patterns $(p+1)\cdots 21$ and $12\cdots(n-$

$p+1)(n-p+2)$. Now let us prove that ϕ' is a bijection from $\mathcal{B}_{p-1}(n-1)$ to $\mathfrak{S}_n(132, 312, (p+1)\cdots 21, 12\cdots(n-p+1)(n-p+2))$. Indeed, a permutation π in $\mathfrak{S}_n(132, 312, (p+1)\cdots 21, 12\cdots(n-p+1)(n-p+2))$ verifies that π_i is either $\max\{\pi_j, j \leq i\}$ or $\min\{\pi_j, j \leq i\}$ which are respectively represented by 1 and 0 in order to obtain a binary string b of length $n-1$ (we do not consider the bit corresponding to π_1). Obviously, if π avoids $(p+1)p\cdots 21$ (resp. $12\cdots(n-p+1)(n-p+2)$) then b does not contain p zeros (resp. $n-p+1$ ones) which means that b contains exactly $p-1$ zeros. Moreover, if $c = b'10^\ell 10^{k-\ell}$ is obtained from $b = b'10^k \in \mathcal{B}_{p-1}(n-2)$ by inserting 1, then $\phi'(c)$ is obtained from $\phi'(b)$ by inserting n on the same position from the right. □

Notice that the set $\mathfrak{S}_n(132, 312, (p+1)\cdots 21, 12\cdots(n-p+1)(n-p+2))$ depends on a pattern of length $n-p+2$. However, it remains the open question: is it possible to find a finite basis B (independent of n) such that $\mathfrak{S}_n(B)$ is enumerated by $\binom{n-1}{p-1}$?

5 Compositions of n with at most p parts

The set $\mathcal{C}_{\#p}(n)$ of compositions of n with at most p parts is enumerated by $\sum_{k=1}^{p}\binom{n-1}{k-1}$. Moreover, $\mathcal{C}_{\#p}(n)$ is in one-to-one correspondence with the set $\mathcal{B}_{\#(p-1)}(n-1)$ of binary strings of length $n-1$ and having at most $p-1$ zeros. The function φ (see Section 2) shows such a bijection. The following theorem gives succession rules in order to generate $\mathcal{B}_{\#(p-1)}(n-1)$. Since $\mathcal{B}_{\#(p-1)}(n-1) = \bigcup_{i=0}^{p-1}\mathcal{B}_i(n-1)$, these rules are obtained by a simple adaptation of the rules described in the previous section. Here we say that the level of a node in the generating tree is the length of the unique path between the root and this node (the root is on the level 0).

Theorem 2 *For $p \geq 1$, a system $(\Omega_{\#p})$ of succession rules for the set*

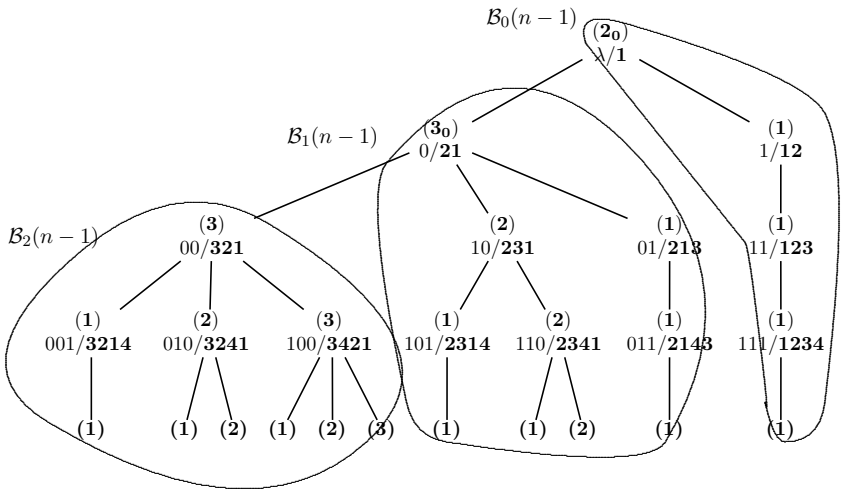


Figure 2. The first levels of the generating tree $(\Omega_{\#3})$. Each node on the level n is coded by one permutation in $\mathfrak{S}_{n+1}(132, 312, 4321)$ or by one binary string in $\mathcal{B}_{\#2}(n)$. Encircled subtrees correspond to the subsets $\mathcal{B}_i(n-1)$ for $0 \leq i \leq 2$.

$\mathcal{C}_{\#p}(n)$ *is:*

$$(\Omega_{\#p}) \begin{cases} (2_0) & \\ (k_0) & \rightsquigarrow ((k+1)_0)(k-1)\cdots(2)(1), \quad \text{if } 2 \leq k < p \\ (p_0) & \rightsquigarrow (p)(p-1)\cdots(2)(1) \\ (k) & \rightsquigarrow (1)(2)\cdots(k), \quad \text{if } 1 \leq k \leq p. \end{cases}$$

Each level $n \geq 0$ of the generating tree induced by $(\Omega_{\#p})$ can be coded by the binary strings of $\mathcal{B}_{\#(p-1)}(n)$ or by the permutations in $\mathfrak{S}_{n+1}(132, 312, (p+1)\cdots 21)$.

- *A node labeled (k_0), $2 \leq k \leq p$, is coded by the binary string 0^{k-2} (resp. the permutation $(k-1)(k-2)\cdots 21$) and its successors are obtained either by appending 0 (resp. k) on the left or by inserting 1 (resp. k) between two zeros, on the right or on the left (resp. on the same position from the right as for binary strings).*
- *All other nodes obey to the rules described in Theorem 1.*

Proof. The proof is directly deduced from Theorem 1. Indeed, a node labeled (k_0), $2 \leq k \leq p$, produces $k-1$ nodes labeled $(1), (2), \ldots, (k-1)$ which have the same succession rules as those of Theorem 1, and either one node labeled $((k+1)_0)$ if $k \neq p$ or one node labeled (k) otherwise. This means that the subtree T rooted by a node labeled (k_0), $k \neq p$, has one subtree T_1 rooted by a node labeled $((k+1)_0)$ that generates the sets $\mathcal{B}_k(n-1)$ for $n-1 \geq k$ (see Theorem 1). Now let $T_2 = T \backslash T_1$ be the subtree of T obtained by deleting all nodes of T_1. So, T_2 generates the set $\mathcal{B}_{k-1}(n-1)$ for $n-1 \geq k-1$. Finally, the complete generating tree of $(\Omega_{\#p})$ is exactly the union of subtrees T_i for $0 \leq i \leq p-1$, where T_i generates the set $\mathcal{B}_i(n-1)$, where $n-1 \geq i$. This proves that $(\Omega_{\#p})$ generates $\mathcal{B}_{\#(p-1)}(n)$. By duality and with the same argument, it also generates all permutations in $\mathfrak{S}_{n+1}(132, 312, (p+1)\cdots 21)$ (see Figure 2). □

6 Compositions of n with parts 1 and p

Let $\mathcal{C}_{1,p}(n)$ be the set of compositions of n with parts 1 and p. The following bijection φ' (see for example [12]) gives a bijection between $\mathcal{C}_{1,p}(n)$ and the set $\mathcal{B}_{\geq p-1}(n-p+1)$ of binary strings of length $n-p+1$ with at least $p-1$ zeros between two ones.

Definition 3 *Let $c = (c_1, c_2, \ldots, c_\ell)$ be a composition of n such that $c_i \in \{1, p\}$ for all $i \leq \ell$. We define the bijection φ' between $\mathcal{C}_{1,p}(n)$ and $\mathcal{B}_{\geq p-1}(n-p+1)$ by the following algorithmical process. We initialize $b = \lambda$ (the empty string). For each i from 1 to ℓ, if $c_i = 1$, then we modify b by appending 0 on its right; otherwise (i.e., $c_i = p$), we modify b by appending 10^{p-1} on its right. Finally, we delete $p-1$ zeros on the right of b which defines a binary string b of length $n-p+1$ with at least $p-1$ zeros between two ones.*

For instance, if $n = 12$, $p = 3$ and $c = (1, 3, 1, 3, 3, 1)$, then $\varphi'(c) = 0\mathbf{100}\mathbf{100}10$.

Theorem 3 *For $p \geq 2$, a system $(\Omega_{1,p})$ of succession rules for the $(1,p)$-compositions is given by:*

$$(\Omega_{1,p}) \begin{cases} (2) \\ (2) \rightsquigarrow (2)(1_0) \\ (1_i) \rightsquigarrow (1_{i+1}), \quad \textit{for } 0 \leq i < p-2 \\ (1_{p-2}) \rightsquigarrow (2). \end{cases}$$

Each level n of the generating tree induced by $(\Omega_{1,p})$ can be coded by the binary strings in $\mathcal{B}_{\geq p-1}(n)$ (the root is on the level 0). A binary string of length n can be obtained from a string of length $n-1$ by inserting 0 or 1 on the last position (see Figure 3).

Proof. We will prove by induction that the nodes on the level k can be coded by the binary strings of the set $\mathcal{B}_{\geq p-1}(k)$ for all $k \geq 0$. Remark that this is true for the root which is coded by the empty string λ (by convenience the level of the root will be 0). So let us assume that each level $j \leq k$ is coded by the elements of $\mathcal{B}_{\geq p-1}(j)$.

Let α be a binary string of length $k+1$ on the level $k+1$ and let $\beta \in \mathcal{B}_{\geq p-1}(k)$ be its predecessor on the level k.

If α is obtained from β by inserting 0 on its right, then α also belongs to $\mathcal{B}_{\geq p-1}(k+1)$. If α is obtained by inserting 1 on the right of β, then β has two sons, so its label is (2), and its predecessor γ is labeled by (2) or (1_{p-2}), then $\beta = \gamma 0$.

(i) If γ is labeled (1_{p-2}), its predecessor γ_1 is labeled (1_{p-3}). We repeat this process until γ_{p-2}, *i.e.,* until we reach the label (1_0). Then γ_{p-2} is obtained from a binary string β' labeled (2) by inserting 1 on the right of β'.

Thus α is of the form $\alpha = \beta 1 = \beta' 10^{p-1}1$. Moreover, $\beta' \in \mathcal{B}_{\geq p-1}(k-p)$ by the recurrence hypothesis. We conclude that $\alpha \in \mathcal{B}_{\geq p-1}(k+1)$.

(ii) If γ is labeled (2), its predecessor γ_1 is labeled (1_{p-2}) or (2). If γ_1 is labeled (1_{p-2}), we return to the case (i) just above, so α has at least $p-1$ consecutive zeros between two ones. If γ_1 is labeled (2), we repeat the process by replacing γ_1 with γ and it will finish when: either we reach the label (1_{p-2}) which corresponds to the

case (i), or we reach the root labeled (2). In any case, α contains $p-1$ consecutive zeros between two ones. Then $\alpha \in \mathcal{B}_{\geq p-1}(k+1)$.

Conversely, we consider $\alpha \in \mathcal{B}_{\geq p-1}(k+1)$ and we construct a path on the generating tree $(\Omega_{1,p})$ which generates this string. We distinguish two cases:

- $\alpha = \alpha' 10^j$, where $j \leq p-1$. So $\alpha' 1 \in \mathcal{B}_{\geq p-1}(k+1-j)$. Therefore, α' is labeled (2) and α is obtained from α' with either the path $(2)/\alpha' \rightsquigarrow (1_0)/\alpha' 1 \rightsquigarrow (1_1)/\alpha' 10 \rightsquigarrow \cdots \rightsquigarrow (1_j)/\alpha' 0^j$ or $(2)/\alpha' \rightsquigarrow (1_0)/\alpha' 1 \rightsquigarrow (1_1)/\alpha' 10 \rightsquigarrow \cdots \rightsquigarrow (1_{p-2})/\alpha' 10^{p-2} \rightsquigarrow (2)/\alpha' 10^{p-1}$;
- $\alpha = \alpha'' 10^j$, where $j \geq p$. Then $\alpha' = \alpha'' 10^{p-1} \in \mathcal{B}_{\geq p-1}(k+p-j)$. So, α is obtained from α' with a path of nodes all labeled (2) in the generating tree.

We repeat the same process by replacing α with α' and we will find the path from the root of the generating tree $(\Omega_{1,p})$ to reach α. For instance, $\alpha = 010010001$, the path to reach α from the root in the generating tree $(\Omega_{1,3})$ is:

$(2)/\lambda \rightsquigarrow (2)/0 \rightsquigarrow (1_0)/01 \rightsquigarrow (1_1)/010 \rightsquigarrow (2)/0100 \rightsquigarrow (1_0)/01001 \rightsquigarrow (1_1)/010010 \rightsquigarrow (2)/0100100 \rightsquigarrow (2)/01001000 \rightsquigarrow (1_0)/010010001$.

We finally conclude that the generating tree induced by $(\Omega_{1,p})$ is coded by the set $\mathcal{B}_{\geq p-1}(n)$ of binary strings of length n with at least $p-1$ zeros between two ones. □

Theorem 4 *Each level $n \geq 0$ of the generating tree of $(\Omega_{1,p})$ can be coded by the permutations in $\mathfrak{S}_{n+1}(231, 312, 321, \overline{2134\cdots(p+1)}(p+3)(p+2))$. A permutation of length n is obtained from a permutation π of length $(n-1)$ by inserting n either on the right of π or just before its last entry (see Figure 3).*

Proof. In [12], Baril and Moreira showed there is a bijection f between $\mathcal{B}_{\geq p-1}(n)$ and $\mathfrak{S}_{n+1}(231, 312, 321, \overline{2134\cdots(p+1)}(p+3)(p+2))$. Moreover, the insertion of 0 (resp. 1) on the right of $b \in \mathcal{B}_{\geq p-1}(n)$ is equivalent to the insertion of $(n+1)$ on the right of $\pi = f(b)$ (resp. just before the last entry). This means that $\mathfrak{S}_{n+1}(231, 312, 321, \overline{2134\cdots(p+1)}(p+3)(p+2))$ also codes the generating tree $(\Omega_{1,p})$. □

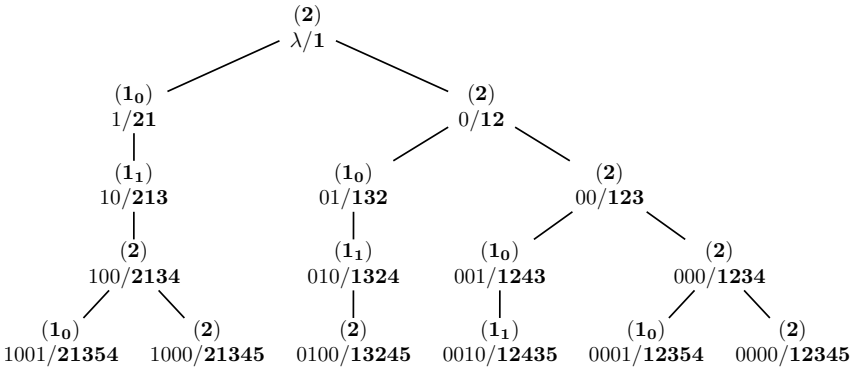


Figure 3. The first five levels of the generating tree $(\Omega_{1,3})$. Each node on the level n is coded by one permutation in $\mathfrak{S}_n(231, 312, 321, 21\overline{34}65)$ or by one binary string in $\mathcal{B}_{\geq 2}(n-1)$.

7 Compositions of n without part of size p

Let $\mathcal{C}_{\widehat{p}}(n)$ be the set of compositions of n without part of size p. The bijection φ (see Section 2) transforms $\mathcal{C}_{\widehat{p}}(n)$ into the set $\mathcal{B}_{\widehat{p-1}}(n-1)$ of the binary strings of length $(n-1)$ without run of ones of length $p-1$ knowing that a run of ones is a maximal substring of consecutive ones. For instance, the binary string $b = 0\mathbf{11}0\mathbf{111}0\mathbf{1}$ contains three runs of ones illustrated in boldface. In this section, we consider the concept of jumping succession rules introduced in [23]. This allows the construction of an element of size greater than $n+1$ from an element of size n (see Section 1).

Theorem 5 *For $p \geq 2$, a system of jumping succession rules $(\Omega_{\widehat{p}})$ for*

the compositions without part of size p is given by:

$$(\Omega_{\widehat{p}})\begin{cases}(2_0)\\ (2_i)\rightsquigarrow(2_0)(2_{i+1}), & \text{for } 0\le i\le p-3\\ (2_{p-2})\overset{1}{\rightsquigarrow}(2_0)\\ \qquad\overset{2}{\rightsquigarrow}(2_{p-1})\\ (2_{p-1})\rightsquigarrow(2_0)(2_{p-1}).\end{cases}$$

Each level n of the generating tree of $(\Omega_{\widehat{p}})$ is coded by the set $\mathcal{B}_{\widehat{p-1}}(n)$ (the root is on the level 0). Let b be a binary string in $\mathcal{B}_{\widehat{p-1}}(n)$ corresponding to a node of level n. Then b has two successors:
- if the two successors of b are on the level $n+1$, they are obtained by inserting 0 or 1 on the right of b.
- if one successor of b is on the level $n+1$ and the other on the level $n+2$, we insert 0 on the right of b in order to obtain the successor on the level $n+1$ and we insert 11 on the right of b in order to obtain that on the level $n+2$ (see Figure 4).

Proof. We proceed by induction. The root of the tree is coded by the empty string λ (by convenience the level of the root is 0). We assume that each level $k\le n$ is coded by the elements of $\mathcal{B}_{\widehat{p-1}}(k)$. Let α be the binary string of length $(n+1)$ corresponding to a node on the level $n+1$ and let β be its predecessor on the level n or $n-1$, then β belongs to $\mathcal{B}_{\widehat{p-1}}(n)$ or $\mathcal{B}_{\widehat{p-1}}(n-1)$.

(i) If β is on the level $n-1$, then β is labeled (2_{p-2}), α is labeled (2_{p-1}) and α is obtained from β by inserting 11 on its right. Thus β is obtained from its predecessor β_1 labeled (2_{p-3}) (if $p-3>0$) by inserting 1 on its right. We repeat this process until we create β_{p-2}, *i.e.,* until we reach a node labeled (2_0). Necessarily β_{p-2} is obtained from its predecessor β' by inserting 0 on its right. Thus $\alpha=\beta 11=\beta' 01^p$, and α does not contain any run of ones of length $p-1$, which implies that $\alpha\in\mathcal{B}_{\widehat{p-1}}(n+1)$.

(ii) If β is on the level n, then α is obtained from β by inserting 0 or 1 on its right.

- if α is obtained from β by inserting 0 on its right, then α obviously belongs to $\mathcal{B}_{\widehat{p-1}}(n+1)$.

- if α is obtained from β by inserting 1 on its right, then β is labeled (2) or (2_i), with $i < p-2$:

  (a) if β is labeled (2), it is obtained from its predecessor (also labeled (2)) by inserting 1 on its right. We repeat this process until we reach the label (2_{p-2}). So we retrieve the case (i) above. Then, $\alpha = \alpha' 1^\ell = \beta' 01^p 1^\ell$, with $\ell > 0$, and $\beta' \in \mathcal{B}_{\widehat{p-1}}(n-p-l)$. Therefore $\alpha \in \mathcal{B}_{\widehat{p-1}}(n+1)$.

  (b) if β is labeled (2_i), with the same process of the case (i), we have $\alpha = \beta' 1^{i+1}$, with $\beta' \in \mathcal{B}_{\widehat{p-1}}(n-i)$ with $i+1 < p-1$. Thus $\alpha \in \mathcal{B}_{\widehat{p-1}}(n+1)$.

Conversely, each string α in $\mathcal{B}_{\widehat{p-1}}(n+1)$ can be constructed on the level $n+1$ of the generating tree (Ω_p). Indeed, if $\beta = \alpha 0$, then $\beta \in \mathcal{B}_{\widehat{p-1}}(n+2)$. So β can be decomposed as $1^{c_1-1}01^{c_2-1}0\cdots 1^{c_\ell-1}0$ such that $c = (c_1, c_2, \ldots, c_\ell) \in \mathcal{C}_{\widehat{p-1}}(n+3)$ (see the bijection $\varphi : \mathcal{C}_{\widehat{p}}(n+1) \rightsquigarrow \mathcal{B}_{\widehat{p-1}}(n)$). Let $\beta = \beta' 1^{c_\ell - 1}0$, then $\beta' \in \mathcal{B}_{\widehat{p-1}}(n+2-c_\ell)$. We distinguish two cases:

- if $c_\ell < p$, then β is obtained from β' on the generating tree (Ω_p) by the path
  $(2_1) \rightsquigarrow (2_2) \rightsquigarrow \cdots \rightsquigarrow (2_{c_\ell - 1}) \rightsquigarrow (2_0)$,

- if $c_\ell > p$, then β is obtained from β' on the generating tree (Ω_p) by the path
  $(2_1) \rightsquigarrow (2_2) \rightsquigarrow \cdots \rightsquigarrow (2_{p-2}) \overset{2}{\rightsquigarrow} (2_{p-1}) \rightsquigarrow (2_{p-1})^{c_\ell - p - 1} \rightsquigarrow (2_0)$.

We repeat this process by replacing β with β' and we obtain a path from the root to β on the generating tree $(\Omega_{\widehat{p}})$. For instance, on the generating tree $(\Omega_{\widehat{3}})$, if $\alpha = 1110100$, then the path for reaching α from the root is:
$(2_0)/\lambda \rightsquigarrow (2_1)/1 \overset{2}{\rightsquigarrow} (2_2)/111 \rightsquigarrow (2_0)/1110 \rightsquigarrow (2_1)/11101 \rightsquigarrow (2_0)/111010 \rightsquigarrow (2_0)/1110100$. □

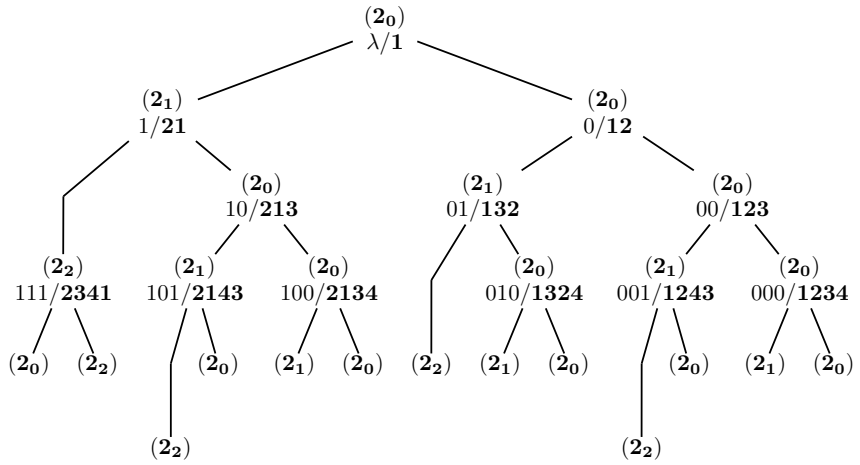


Figure 4. The first levels of the generating tree $(\Omega_{\widehat{3}})$. Each node on the level $n \geq 0$ is coded by a permutation in $\mathfrak{S}_{n+1}(312, 321, 234\dot{1})$ or by a binary string in $\mathcal{B}_{\widehat{2}}(n)$.

In order to code the generating tree by permutations avoiding patterns, we use the new pattern $23\cdots n\dot{1}$ presented in Section 2.

Theorem 6 *Each level n of the generating tree induced by $(\Omega_{\widehat{p}})$ can be coded by the permutations $\pi \in \mathfrak{S}_n(312, 321, 23\cdots(p+1)\dot{1})$ as follows:*
• *If the two successors of π belong to the level $(n+1)$, they are obtained by inserting $n+1$ on the right or just before the last entry of π.*
• *If a successor of π is on the level $(n+1)$ and the other on the level $(n+2)$, we insert $n+1$ on the right of π in order to obtain the successor on the level $n+1$, and we insert $(n+1)(n+2)$ just before the last entry of π in order to obtain that of the level $(n+2)$ (see Figure 4).*

In order to prove this theorem, we present the following proposition.

Proposition 1 *The map ϕ defined in Section 2 is a bijection from $\mathcal{B}_{\widehat{p}}(n)$ to $\mathfrak{S}_{n+1}(312, 321, 23\cdots(p+2)\dot{1})$.*

Proof. Recall that ϕ is a bijection from the set of binary strings of length n to the set of $(n+1)$-length permutations avoiding 321 and 312. So we will prove that $\phi(\mathcal{B}_{\widehat{p}}(n)) = \mathfrak{S}_{n+1}(321, 312, 23\cdots(p+2)\dot{1})$.

Let b be a binary string in $\mathcal{B}_{\widehat{p}}(n)$ and π be its image by ϕ in $\mathfrak{S}_{n+1}(321, 312)$. As $\phi(b) \in \mathfrak{S}_{n+1}(321, 312)$, there exist (see [36]) some indices $0 = k_0 < k_1 < \cdots < k_r < \cdots < k_m = n$ such that π is divided into m blocks

$$\pi = \boxed{\pi_1\pi_2\cdots\pi_{k_1}} \cdots \boxed{\pi_{k_{r-1}+1}\pi_{k_{r-1}+2}\cdots\pi_{k_r}} \cdots$$
$$\cdots \boxed{\pi_{k_{m-1}+1}\pi_{k_{m-1}+2}\cdots\pi_{k_m}}$$

satisfying the two following conditions:

(i) the rightmost elements of each block are in increasing order (*i.e.*, $1 = \pi_{k_1} < \pi_{k_2} < \cdots < \pi_{k_m}$);

(ii) each block $\pi_{k_{r-1}+1}\pi_{k_{r-1}+2}\cdots\pi_{k_r}$ having at least two elements is of the form $(a+1)(a+2)\cdots(a+k_r-k_{r-1}-1)a$ with $a = \pi_{k_r}$.

For instance, if $b = 110111$, then $\pi = \mathbf{231}5674$. It is remarkable from the definition of ϕ that if we add on the last position of b one occurrence of 0 and then divide the obtained binary string into separated blocks where each block contains exactly one occurrence of 0 at its end, then we obtain m blocks corresponding to the m blocks of $\phi(b)$. Furthermore, the number of consecutive 1s in each block of b is also the number, minus one, of elements of the respective block in $\phi(b)$ (see Definition 2). This leads us to the following claim: for $q \geq 2$, if one subsequence of $\phi(b)$ is a pattern $23\ldots q1$ (of length q), then this sequence must belong to only one block of $\phi(b)$ (since the smallest element of each block of $\phi(b)$ is greater than the largest element of the previous blocks of $\phi(b)$). Moreover, since b does not contain exactly p consecutive 1s, $\phi(b)$ does not contain any block of length $p+1$ exactly. Hence, if the subsequence $23\ldots p(p+1)1$ appears in $\phi(b)$, then we can extend it (without considering positions of extended entries) into a sequence $23\ldots(p+1)(p+2)1$, which means that $\phi(b)$ avoids $23\ldots(p+1)(p+2)\dot{1}$.

Conversely, we take a permutation π of $\mathfrak{S}_{n+1}(321, 312, 23\ldots(p+1)(p+2)\dot{1})$. We also have the decomposition of π into blocks as above. It needs to show that $\phi^{-1}(\pi)$ belongs to $B_{\widehat{p}}(n)$. This is induced by the fact

that all blocks of π considered as subsequences of π are either of length less than $p+1$ (if they do not contain the reduced pattern $23\dots p(p+1)1$) or more than $p+1$ (if they contain the extended pattern $23\dots(p+2)1$). Therefore, $\phi^{-1}(\pi)$ is a binary string without p consecutive 1s. □

Now, the proof of Theorem 6 is obtained using the following remark. The insertion of 0 (resp. 1) on the right of $b \in \mathcal{B}_{\widehat{p}}(n)$ is equivalent to the insertion of $n+1$ on the right of $\pi = \phi(b)$ (resp. just before the last entry). And also the insertion of 11 on the right of $b \in \mathcal{B}_{\widehat{p}}(n)$ is equivalent to the insertion of $(n+1)(n+2)$ just before the last entry of $\pi = \phi(b)$. This means that $\mathfrak{S}_n(312, 321, 23\cdots(p+1)\dot{1})$ also codes the generating tree $(\Omega_{\widehat{p}})$.

8 Algorithmic considerations and conclusion

In this section, we explain how all studied classes can be generated efficiently.

Let $\pi \in \mathfrak{S}_n$; the *sites* of π are the positions between two consecutive entries, before the first and after the last entry. We suppose that the sites are numbered from 1 to $n+1$ and from right to left. For example, the third site of the permutation $\pi = 463512$ is between the entries 5 and 1. Moreover, let τ be an n-length permutation (or equivalently a binary string) on a generating tree defined by a succession rule (Ω). Then, the i-th site of τ is said *active* if the element obtained from τ by the insertion of a value into this site also belongs to the generating tree. The active sites of τ are *right-justified* (see [20]) if all sites to the right of the leftmost active site are also active. If each element on the generating tree is right-justified, we will say that the generated class is *regular*. It is crucial to notice that all classes defined in this paper are *regular*.

An algorithm runs in Constant Amortized Time (CAT) if the amount of computations, after a small amount of preprocessing, is proportional to the number of generated objects. Many CAT algorithms exist in the literature, but we will take that presented in [20, 33]. This recursive algorithm acts on regular classes and enables us to ensure

that we can generate all successors of a given node in constant amortized time. Thus the total amount of computations is proportional to the number of recursive calls. Moreover, the number *obj* of generated objects is at least $\frac{c-c_1}{2}$, where c is the total number of recursive calls, and c_1 is the number of recursive calls of degree one. So, for each generating tree of this study, we will calculate c_1.

- We immediately have $c_1 = 0$ for the generating tree defined in Section 7 and $\frac{c}{obj}$ evolves as $\mathcal{O}(1)$.

- In Section 6, a simple observation gets $c_1 \leq p(c - c_1)$ and $\frac{c}{obj}$ evolves as $\mathcal{O}(p)$.

- The generating tree of Section 5 is constituted of several generating trees of Section 4. So it suffices to compute c_1 for Section 4 (see below).

- In Section 4, each node produces exactly once a node of degree one. Thus the number of nodes of degree one and on levels at most $n-1$ in the generating tree (Ω_p) is equal to

$$c_1 = \sum_{i=p-1}^{n-1} \binom{i}{p-1}.$$

A simple calculation proves that if $p \geq \lfloor \frac{n}{2} \rfloor$ and $n \geq 4$, then $c_1 \leq 2\binom{n}{p-1}$, which means that the number of nodes of degree one divided by 2 is smaller than the number of generated objects $obj = \binom{n}{p-1}$ for $p \geq \lfloor \frac{n}{2} \rfloor$. In this case, the complexity is $\mathcal{O}(1)$. The case $p \leq \lfloor \frac{n}{2} \rfloor$ is obtained *mutatis mutandis* by interchanging 0 with 1 in the binary strings of the generating tree.

Finally, this means that the total amount of computation divided by the number of objects is bounded by a constant independently to the size of the objects. Therefore all studied classes in this paper can be generated in Constant Amortized Time (CAT) using an algorithm similar to that of [20, 33]. We summarize our results in Table 1 that contains the succession rules of each studied class and their corresponding pattern avoiding permutation classes.

Table 1. Classes of compositions, corresponding succession rules and corresponding classes of pattern avoiding permutations.

Classes	Succession rules	Avoided patterns
$C(n)$	(2) $(2) \rightsquigarrow (2)(2)$	$\{321, 312\}$ [4, 26]
	(2) $(k) \rightsquigarrow (k+1)(1)^{k-1}$	$\{321, 231\}$ [4, 26]
$C_p(n)$	(p) $(k) \rightsquigarrow (1)(2)\cdots(k)$	$\{132, 312, (p+1)p\cdots 21,$ $12\cdots(n-p)(n-p+1)\}$
$C_{\#p}(n)$	(2_0) $(k_0) \rightsquigarrow ((k+1)_0)(k-1)\cdots(2)(1)$ for $2 \le k < p$ $(p_0) \rightsquigarrow (p)(p-1)\cdots(2)(1)$ $(k) \rightsquigarrow (1)(2)\cdots(k-1)(k)$ for $1 \le k \le p$	$\{132, 312, (p+1)p\cdots 21\}$
$C_{\le p}(n)$	(2_0) $(2_0) \rightsquigarrow (2_0)(2_1)$ $(2_i) \rightsquigarrow (2_0)(2_{i+1})$, for $1 \le i < p-2$ $(2_{p-2}) \rightsquigarrow (2_0)(1)$ $(1) \rightsquigarrow (2_0)$	$\{321, 312, 234\cdots(p+1)1\}$ [4, 11]
	(2) $(k) \rightsquigarrow (k+1)(1)^{k-1}$ $(p) \rightsquigarrow (p)(1)^{k-1}$	$\{312, 231, (p+1)p\cdots 321\}$ [4, 11]
$C_{1,p}(n)$	(2) $(2) \rightsquigarrow (1_0)(2)$ $(1_i) \rightsquigarrow (1_{i+1})$, for $0 \le i < p-2$ $(1_{p-2}) \rightsquigarrow (2)$	$\{231, 312, 321,$ $21\overline{34\cdots(p+1)}(p+3)(p+2)\}$
$C_{\widehat{p}}(n)$	(2_0) $(2_i) \rightsquigarrow (2_0)(2_{i+1})$, for $0 \le i \le p-3$ $(2_{p-2}) \overset{1}{\rightsquigarrow} (2_0)$ $\overset{2}{\rightsquigarrow} (2_{p-1})$ $(2_{p-1}) \rightsquigarrow (2_0)(2_{p-1})$	$\{312, 321, 23\cdots(p+1)\dot{1}\}$

References

[1] K. Alladi and V.E. Hoggatt. Compositions with ones and twos. *Fibonacci Quarterly*, 13(3):233–239, 1975.

[2] S. Bacchelli, E. Barcucci, E. Grazzini, and E. Pergola. Exhaustive generation of combinatorial objects by ECO. *Acta Informatica*, 40:585–602, 2004.

[3] C. Banderier, P. Flajolet, D. Gardy, M. Bousquet-Melou, A. Denise, and D. Gouyou-Beauchamps. Generating functions for generating trees. *Discrete Mathematics*, 246:29–55, 2002.

[4] E. Barcucci, A. Bernini, and M. Poneti. From Fibonacci to Catalan permutations. *PU.M.A.*, 2007(1-2):1–18, 2006.

[5] E. Barcucci, A. Del Lungo, and E. Pergola. Random generation of trees and other combinatorial objects. *Theoretical Computer Science*, 218(2):218–232, 1999.

[6] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. A methodology for plane tree enumeration. *Discrete Mathematics*, 180:45–64, 1998.

[7] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. Directed animals, forests of trees and permutations. *Discrete Mathematics*, 204:41–71, 1999.

[8] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. ECO: a methodology for the enumeration of combinatorial objects. *Journal of Difference Equations and Applications*, 5:435–490, 1999.

[9] J.-L. Baril. More restrictive Gray codes for some classes of pattern avoiding permutations. *Information Processing Letters*, 109:799–804, 2009.

[10] J.-L. Baril. Classical sequences revisited with permutations avoiding dotted pattern. *Electronic Journal of Combinatorics*, 18:p#178, 2011.

[11] J.L. Baril and P.T. Do. ECO-generation for p-generalized Fibonacci and Lucas permutations. *Pu.M.A.*, 17(1-2):19–37, 2006.

[12] J.L. Baril and C. Moreira Dos Santos. Gray code for compositions of n with parts 1 and p. *Advances and Applications in Discrete Mathematics*, 3(1):67–84, 2009.

[13] S. Brlek, E. Duchi, E. Pergola, and S. Rinaldi. On the equivalence problem for succession rules. *Discrete Math.*, 298:142–154, 2005.

[14] L. Carlitz. Restricted compositions. *The Fibonacci Quart.*, 14:254–264, 1976.

[15] T. Chinburg, C.D. Savage, and H.S. Wilf. Combinatorial families that exponentially far from being listable in Gray code sequence. *Transactions of the AMS*, 351:379–402, 1999.

[16] P. Chinn and S. Heubach. $(1, k)$-Compositions. *Congressus Numerantium*, 164:183–194, 2003.

[17] P. Chinn and S. Heubach. Compositions of n with no occurrence of k. *Congressus Numerantium*, 164:33–51, 2003.

[18] A. Del Lungo, A. Frosini, and S. Rinaldi. ECO method and the exhaustive generation of convex polyominoes. *DMTCS*, pages 103–116, 2003.

[19] E. Deutsch, L. Ferrari, and S. Rinaldi. Production matrices. *Advances in Applied Mathematics*, 34:101–122, 2005.

[20] P.T. Do and V. Vajnovszki. Exhaustive generation of some classes of pattern avoiding permutations using succession functions. *Conference in honor of Donald E. Knuth*, 2007.

[21] E. Duchi, A. Frosini, R. Pinzani, and S. Rinaldi. A note on rational succession rules. *Journal of Integer Sequences, Article 03.1.7*, 6, 2003.

[22] L. Ferrari, E. Pergola, R. Pinzani, and S. Rinaldi. An algebraic characterization of the set of succession rules. *Theoretical Computer Science*, 281:351–367, 2002.

[23] L. Ferrari, E. Pergola, R. Pinzani, and S. Rinaldi. Jumping succession rules and their generating functions. *Discrete Math*, 271:29–50, 2003.

[24] R.P. Grimaldi. Compositions with odd summands. *Congressus Numerantium*, 142:113–127, 2000.

[25] R.P. Grimaldi and S. Heubach. Binary strings without odd runs of zeros. *Ars Combinatoria*, 75:241–255, 2005.

[26] O. Guibert. Combinatoire des permutations motifs exclus en liaison avec mots, cartes planaires et tableaux de Young. *Ph.D Thesis, Université Bordeaux 1*, 1995.

[27] S. Heubach and T. Mansour. Combinatorics of compositions and words. *Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL.*

[28] S. Heubach and T. Mansour. Compositions of n with parts in a set. *Congressus Numerantium*, 168:127–143, 2004.

[29] A. Juarna and V. Vajnovszki. Combinatorial isomorphism between Fibonacci classes. *Journal of Discrete Mathematical Science and Cryptography*, 11(2):147–158, 2008.

[30] S. Kitaev, T. McAllister, and K. Petersen. Enumerating segmented patterns in compositions and encoding with restricted permutations. *Integers: Electronic Journal of Combinatorial Number Theory*, 6:16pp, 2006.

[31] P. Klingsberg. A Gray code for compositions. *J. Algorithms*, 3:41–44, 1982.

[32] A. Knopfmacher and H. Prodinger. On Carlitz compositions. *European Journal of Combinatorics*, 19(5):579–589, 1998.

[33] T. Mansour, W.M. Dukes, M.F. Flanagan, and V. Vajnovszki. Combinatorial Gray codes for classes of pattern avoiding permutations. *Theoretical Computer Science*, 396:35–49, 2008.

[34] L. Pudwell. Enumeration schemes for permutations avoiding barred patterns. *The electronic Journal of Combinatorics*, 17,R29, 2010.

[35] R. Simion and F.W. Schmidt. Restricted permutations. *European J. Combin.*, 6:383–406, 1985.

[36] V. Vajnovszki. A loopless generation of bitstrings without p consecutive ones. *Discrete Mathematics and Theoretical Computer Science*, 1:227–240, Springer 2001.

[37] T.R. Walsh. Loop-free sequencing of bounded integer compositions. *J. Combin. Math. Combin. Comput.*, 33:323–345, 2000.

Jean-Luc Baril, Phan-Thuan Do Received July 16, 2012

Jean-Luc Baril,
LE2I UMR-CNRS 5158, Université de Bourgogne
B.P. 47 870, 21078 DIJON-Cedex France
E–mail: *barjl@u-bourgogne.fr*

Phan-Thuan Do,
Hanoi University of Science and Technology, Vietnam
Department of Computer Science
E–mail: *thuandp@soict.hut.edu.vn*

Computer Science Journal of Moldova, vol.**21**, no.1(61), 2013

Median calculation for heterogeneous complex of abstract cubes*

Sergiu Cataranciuc

Abstract

Median problem for an arbitrary complex of abstract cubes is studied. First, a number of auxiliary results related to some special metric properties of abstract n-dimensional cube are presented. Basing on these results and results obtained in the study of homogeneous complexes [13], it is proved that median of an arbitrary complex of abstract cubes can be calculated without using metrics. Method which represents a generalization of method applied to homogeneous complex of abstract cubes is proposed.

Keywords: Abstract cube, cubic complex, median, transversal, group of homologies, convex set, class of parallel edges.

1 Introduction

In this paper we study median problem that was partially examined in [3], [5] for some particular case. For example, in a complex of abstract figures, which do not belong to any space, it is considered a finite metrizable set of points X, which is a part of a set of continuum power. Then we search the points from X with the following property: for any such point the sum of distances from this point to all other points from X to be *minimal*. These are the so-called **medians**. This problem is known as a practical one and even the corresponding applications [2], [3] are known.

We are interested to consider figures complex for which the searchable points **do not depend on the metric**, but only on the **weights** of the elements from X.

* This work was supported by the Institutional Project Ref. Nr. 11.817.08.47A

It is to mention that these simple formulations, as well as applications, require a strong theoretical foundation: *topology of multi-ary* relations [23] and their homology theory. Homology and co-homology groups for a complex of abstract cubes, as a particular case of complex of multi-ary relations, have been described and studied in [13], [15-17].

2 Complex of abstract cubes

In [14] abstract n-dimensional cube is defined. Definition is based on a complex of multi-ary relations defined on a set of elements X. Let X be an arbitrary set of elements and L – a finite subset from X. We form a finite sequence of Cartesian products $L = L^1$, $L^2 = L \times L, ..., L^{n+1} = = L^n \times L$ and consider the subset $R^m \subset L^m$, $1 \leqslant m \leqslant n+1$, called m-ary relation [23]. A case when these relations represent families of ordered sequences without repetitions of elements from L is studied in [17]. Any subsequence $(x_{j_1}, x_{j_2}, ..., x_{j_l})$ for sequence $(x_{i_1}, x_{i_2}, ..., x_{i_m} \in R^m)$, $1 \leqslant l \leqslant m$, which preserves an order of elements of $(x_{i_1}, x_{i_2}, ..., x_{i_m})$, is called a hereditary subsequence.

Definition 1. [17] *Family of relations* $\{R^1, R^2, ..., R^{n+1}\}$ *which satisfies the following conditions:*

I. $R^1 = L^1 = L$*;*

II. $R^{n+1} \neq \varnothing$*;*

III. *any hereditary subsequence with* l *elements of sequences from* R^m*,* $2 \leqslant m \leqslant n+1$*, belongs to the* l*-ary relation* R^l*;*

IV. *intersection of two sequences from* R^m*,* $1 \leqslant m \leqslant n$*, is a hereditary subsequence for each of them or is an empty set;*

is called ***finite complex of multi-ary relations*** *and denoted by* $\mathcal{R}^{n+1} = (R^1, R^2, ..., R^{n+1})$.

In general case, the sequences from sets R^m, $2 \leqslant m \leqslant n+1$, may also contain repetitions of elements. This situation leads to a complex,

called generalized complex of multi-ary relations. This case is examined in papers [15], [16].

A sequence $(x_{i_0}, x_{i_1}, ..., x_{i_m}) \in R^{m+1}$, $0 \leqslant m \leqslant n$, can be examined as **abstract simplex** with dimension m. This simplex is denoted by $S_i^m = (x_{i_0}, x_{i_1}, ..., x_{i_m})$. Let $\mathcal{S}^m$ be a set of all m-dimensional simplexes, determined by the elements of multi-ary relation. Then the complex of multi-ary relations can be seen as a family of abstract simplexes of dimension $0 \leqslant m \leqslant n$, which we will denote by $K^n = (\mathcal{S}^0, \mathcal{S}^1, \mathcal{S}^2, ..., \mathcal{S}^n)$. Any element from $\mathcal{S}^l$, $0 \leqslant m \leqslant n$, which is a hereditary subset of $S_i^m \in \mathcal{S}^m$, $0 \leqslant l \leqslant m \leqslant n$, is called **a face of dimension** l of the abstract simplex S_i^m. As it results from condition IV of the Definition 1, the intersection of any two simplexes represents an abstract simplex (face) or an empty set.

Definition 2. *A set of all simplexes from complex K^n, for which abstract simplex S_i^m is a common face, is called star of simplex S_i^m and is denoted by stS_i^m.*

Using abstract simplexes and their **vacuums**, the abstract n-dimensional cube as well as its vacuum is defined and studied in [14], [19].

Definition 3. *(Inductive definition of abstract n-dimensional cube)*

I. *Abstract 0-dimensional cube and abstract 1-dimensional cube coincide with abstract simplex of the same dimension. Vacuum of cube with size 0 (respectively 1) coincides with vacuum of simplex of the same dimension.*

II. *We consider two pairs of 0-dimensional cubes S_1^0, S_2^0 and S_3^0, S_4^0. The 2-ary relations of the pairs of cubes S_1^0, S_2^0 and S_3^0, S_4^0 determine existence of 1-dimensional cubes $S_1^1 = (S_1^0, S_2^0)$, $S_2^1 = (S_3^0, S_4^0)$, $S_3^1 = (S_1^0, S_3^0)$ $S_4^1 = (S_2^0, S_4^0)$. Further we consider 2- and 3-ary relations between these pairs of cubes, which determine a simplicial complex* [15]*, that leads to the existence of simplexes $S_5^1 = (S_1^0, S_4^0)$, $S_1^2 = (S_1^0, S_3^0, S_4^0)$, $S_2^2 = (S_1^0, S_2^0, S_4^0)$. To define notion of abstract 2-dimensional cube, we introduce firstly notion of* ***vacuum of the*** *corresponding* ***cube****, which will be denoted by $\overset{0}{I}{}^2$. This represents union of*

vacuums of simplexes $\overset{0}{I}{}^2 = \overset{0}{S}{}_1^2 \cup \overset{0}{S}{}_2^2 \cup \overset{0}{S}{}_5^1$. *The abstract* 2*-dimensional cube is defined by its vacuum as follows:* $I^2 = \bigcup_{i=1}^{4} S_i^1 \cup \overset{0}{I}{}^2$. *So, we obtain the simplicial complex that is called* ***2-dimensional pro-cube****. We will use notation* $I^2(\triangle)$. *Geometric reprezentation of 2-dimensional pro-cube is given in Figure 1.*

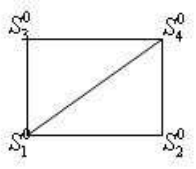


Figure 1. 2-dimensional pro-cube

III. *Assume that notion of* ***abstract*** *i-****dimensional cube*** *and* ***pro-cube****,* I^i *and* $I^i(\Delta)$, $1 \leq i \leq n-1$, *as well as notion of cube's vacuum with the same dimension* $\overset{0}{I}{}^i$ *are known.*

IV. *Let's define notion of abstract n-dimensional cube by using* $(n-1)$*-dimensional cube. Let's consider 2n cubes* $I_1^{n-1}, I_2^{n-1}, ..., I_{2n}^{n-1}$ *with dimension* $(\mathrm{n}-1)$*, corresponding pro-cubes* $I_1^{n-1}(\Delta), I_2^{n-1}(\Delta), ...,$ $I_{2n}^{n-1}(\Delta)$ *and i-ary relations,* $2 \leqslant i \leqslant n$, *among vertexes of these cubes. These i-ary relations determine some simplexes, which form an abstract simplicial complex* [15]. *Thus,* ***vacuum of n-dimensional cube****, denoted by* $\overset{0}{I}{}^n$*, consists of a union of all vacuums of n-dimensional simplexes which don't intersect pro-cubes* $I_j^{n-1}(\triangle)$, $1 \leq j \leq 2n$. ***Abstract n-dimensional cube*** *is defined as* $I^n = \bigcup_{i=1}^{2n} I_i^{n-1} \cup \overset{0}{I}{}^n$. *A simplicial complex* $I^n(\triangle)$ *will be called* ***pro-cube*** *of* I^n.

In what follows, we will consider that vacuums of abstract cubes are filled with elements of a set of cardinal of continuum (a set equivalent to the set of numbers from segment [0,1]).

Definition 4. *Non-empty and finite family of abstract cubes* $\mathbf{T}^n = \{I^m, 0 \leq m \leq n\}$, *which possesses the following properties:*

I. *for any two abstract cubes* I^s, $I^t \in \mathbf{T}^n$, $0 \leqslant s, t \leqslant n$, *the following relation takes place:* $I^s \cap I^t \in \mathbf{T}^n$, *or* $I^s \cap I^t = \varnothing$;

II. *if* I^k *is the face of the m-dimensional cube* $I^m \in \mathbf{T}^n$, $0 \leqslant k < m \leqslant n$ *then* I^k *is an element of* $\mathbf{T}^n$;

III. *family* $\mathbf{T}^n$ *contains at least one n-dimensional cube,*

is called ***abstract cubic complex with dimension*** n.

Homology groups $\square^m(\mathbf{T}^n, Z)$, $0 \leqslant m \leqslant n$, over a field of integer numbers Z for the complex of abstract cubes are defined in a similar way as the complex of multi-ary relations [13]. In terms of homology groups, we can affirm the following. If the complex of abstract cubes $\mathbf{T}^n$ respects the following conditions:

$$\square^0(\mathbf{T}^n, Z) \cong Z; \tag{1}$$

$$\square^1(\mathbf{T}^n, Z) \cong \square^2(\mathbf{T}^n, Z) \cong ... \cong \square^n(\mathbf{T}^n, Z) \cong 0, \tag{2}$$

then the corresponding complex is connected and acyclic.

In what follows we will study complexes of abstract cubes that satisfy the Definition 4, in which the homology group of rank zero is isomorphic to the group of integers Z (relation (1)), but the homology groups of rank $1, 2, 3, ..., n$ are isomorphic to zero (relation (2)). These complexes generalize the homogeneous complexes studied in [13] and are called heterogeneous complexes. Such a complex is represented in the Figure 2.

Definition 5. *Two edges, determined by 1-dimensional cubes* I^1_j *and* I^1_k *of the cubic complex* $\mathbf{T}^n$ *are called parallel if there is a sequence of 1-dimensional cubes* $I^1_j = I^1_{\alpha_1}, I^1_{\alpha_2}, ..., I^1_{\alpha_t} = I^1_k$, *so that every two neighboring cubes* $I^1_{\alpha_s}, I^1_{\alpha_{s+1}}$ *of this sequence represent the disjoint edges (opposite) of a 2-dimensional cube of* $\mathbf{T}^n$.

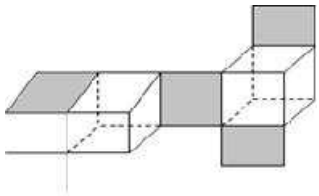


Figure 2. Heterogeneous complex of abstract cubes

Definition 6. *A maximal set of parallel edges of a complex of abstract cubes* $\mathbf{T}^n$ *is called class of parallel edges.*

According to the Definition 5 and Definition 6, in the connected and acyclic complex $\mathbf{T}^n$ every 1-dimensional cube, which is not the face of a cube with dimension $q \geqslant 2$ from $\mathbf{T}^n$, forms one class each.

It is to mention that in the case of homogeneous complexes of abstract cubes [13] every class of parallel edges contains at least 2^{n-1} of 1-dimensional cubes. For the complex represented in Figure 2 we have 11 classes of parallel edges and two of them contain only one element each.

3 Transversals of heterogeneous complex

Let $C_1, C_2, ..., C_m$ be classes of parallel edges of the complex $\mathbf{T}^n$. The set of abstract cubes of $\mathbf{T}^n$, which contains the edges of class C_i as faces, is called transversal of complex $\mathbf{T}^n$ and is denoted by $T(C_i)$, $1 \leqslant i \leqslant m$.

Theorem 1. *A transversal determined by a class of parallel edges of a connected and acyclic complex of abstract cubes* $\mathbf{T}^n$ *also represents complex of abstract cubes denoted by* $\mathbf{T}^q$, $1 \leqslant q \leqslant n$.

Proof: Let $T(C_i)$ be a transversal determined by a class of parallel edges C_i of the complex $\mathbf{T}^n$, and q – maximum dimension of abstract

cubes of $\mathbf{T}^n$ which forms transversal $T(C_i)$. Thus, we have a complex of cubes $\mathbf{T}^q$, which satisfies the conditions I - III of the Definition 4. Taking into consideration the definition of class of parallel edges (Definition 6), it results that $\mathbf{T}^q$ is a connected complex. As the homology groups of the complex $\mathbf{T}^q$ are subgroups of homology groups of complex $\mathbf{T}^n$, $\mathbf{T}^q$ is an acyclic complex of abstract cubes. ■

The border of complex $\mathbf{T}_i^q$, determined by transversal $T(C_i)$, contains two maximal disjoint complexes of dimensions $(q-1)$ that do not contain any edge of C_i. We should denote these subcomplexes through $\mathbf{T}_{i(1)}^{q-1}$ and $\mathbf{T}_{i(2)}^{q-1}$. These complexes could be disconnected.

Let $V(T(C_i))$ be the vacuum of transversal $T(C_i)$, which is a union of vacuums of cubes of complex $\mathbf{T}_i^q$, which do not belong to subcomplexes $\mathbf{T}_{i(1)}^{q-1}$ and $\mathbf{T}_{i(2)}^{q-1}$.

Theorem 2. *Transversal $T(C_i)$, $1 \leqslant i \leqslant m$, of connected and acyclic abstract cubic n-dimensional complex, divides this complex through the vacuum of the transversal $T(C_i)$ in connected and acyclic two cubic abstract complexes of dimension q, $1 \leqslant q \leqslant n$.*

Proof of the theorem results immediately from the fact that $\mathbf{T}^n$ is a connected and acyclic complex, as well as from the definition of corresponding transversals.

Let $\mathbf{T}^n$ be the n-dimensional complex of abstract cubes which satisfies the Definition 4 and contains a cube $I^r \in \mathbf{T}^n$, $0 \leqslant r \leqslant n$, which is not a face of n-dimensional cubes of $\mathbf{T}^n$. We will call such complexes heterogeneous complexes.

Let's denote by $\mathcal{Q}^t$, $1 \leqslant t \leqslant n$, the family of maximal and connected homogeneous t-dimensional subcomplexes of $\mathbf{T}^n$. Obviously, in case of heterogeneous complex $\mathbf{T}^n$, we have:

1) $\mathcal{Q}^n \neq \varnothing$;

2) there is a value of t, $1 \leqslant t \leqslant n$, such that $\mathcal{Q}^t \neq \varnothing$ ($t \geqslant 1$, as it was mentioned above, we are studying only connected and acyclic complexes).

Thereby, in a heterogeneous complex any two maximal homogeneous n-dimensional subcomplexes are united by a sequence of homo-

geneous complexes $\mathbf{T}_1^{n_1}, \mathbf{T}_2^{n_2}, ..., \mathbf{T}_q^{n_q}$, where $1 \leqslant n_1, n_2, ..., n_q < n$, with the following properties:

a) $\mathbf{T}_j^{n_j} \in \mathcal{Q}^j, 1 \leqslant j \leqslant q$;

b) $n_i \neq n_{i+1}$, for any $i \in \{1, 2, .., q-1\}$.

In Figure 3 a geometric representation of a heterogeneous complex of dimension three is given. We have three families of homogeneous subcomplexes:

$$\mathcal{Q}^1 = \{\mathbf{T}_1^3, \mathbf{T}_2^3, \mathbf{T}_3^3\},$$

$$\mathcal{Q}^2 = \{\mathbf{T}_1^2, \mathbf{T}_2^2, \mathbf{T}_3^2, \mathbf{T}_4^2\},$$

$$\mathcal{Q}^3 = \{\mathbf{T}_1^1, \mathbf{T}_2^1, \mathbf{T}_3^1\}.$$

Let's note that the homogeneous 3-dimensional complexes $\mathbf{T}_1^3$ and $\mathbf{T}_2^3$ are united by sequence of homogeneous complexes with smaller dimensions: $\mathbf{T}_2^2, \mathbf{T}_1^1, \mathbf{T}_3^2$.

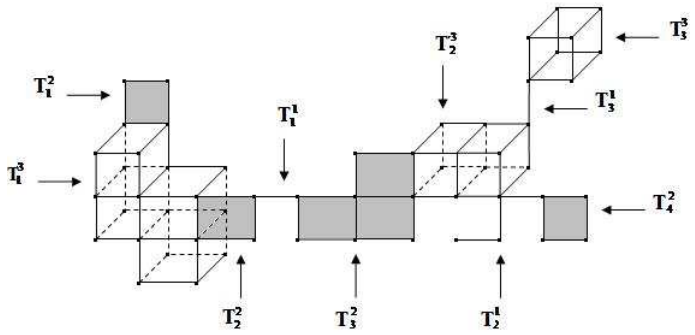


Figure 3. Heterogeneous complex with 3 families of homogeneous subcomplexes

The notion of an interior and a border of homogeneous complex of abstract cubes $\mathbf{T}^n$ was defined [13] and denoted respectively by int$\mathbf{T}^n$ and bd$\mathbf{T}^n$.

Let K^n be a heterogeneous complex of abstract cubes with families of homogeneous maximal connected complexes $\mathcal{Q}^1, \mathcal{Q}^2, ..., \mathcal{Q}^n$. In this case, for the complex K^n we use the notation: $K^n = (\mathcal{Q}^1, \mathcal{Q}^2, ..., \mathcal{Q}^n)$.

Definition 7. *A union* $\bigcup_{\mathbf{T}^n \in Q^n} int\mathbf{T}^n$ *is called an interior of complex* K^n *and is denoted by* $intK^n$. *Difference* $K^n \backslash intK^n$ *is called a border of this complex and is denoted by* bdK^n.

4 Representation of abstract cubes

Let I^n be an abstract n-dimensional cube with a set of vertexes $V = \{x_1, x_2, ..., x_{2^n}\} \subset X$, $cardX = r > 2^n$. We fix vertex x_1 and form the following sets:

$$\begin{aligned}
\Gamma_0 &= \{x_1\}, \\
\Gamma_1 &= \{x_i \in V : d(x_1, x_i) = 1\}, \\
&\cdots\cdots\cdots\cdots\cdots\cdots \\
\Gamma_k &= \{x_i \in V : d(x_1, x_i) = k\}, \\
&\cdots\cdots\cdots\cdots\cdots\cdots \\
\Gamma_n &= \{x_i \in V : d(x_1, x_i) = n\} = \{x_{2^n}\},
\end{aligned}$$

where d represents the Hamming metrics defined on this cube (see [21] and theorem 3 from [13]).

Without losing generality, we consider that x_{2^n} is the vertex of cube I^n for which $d(x_1, x_{2^n}) = n$.

Thus, the abstract n-dimensional cube I^n can be represented by the sequence of sets $\Gamma_0, \Gamma_1, ..., \Gamma_n$, which possesses the following properties:

a) $card\Gamma_i = C_n^i$;

b) any two distinct elements of the set Γ_i, $0 < i < n$, do not represent a 1-dimensional face of cube I^n,

c) $d(x_k, x_l)$ is an even number, for any two distinct elements x_k, x_l of the set Γ_i, $0 < i < n$.

Obviously, not every sequence of this type represents the n-dimensional cube.

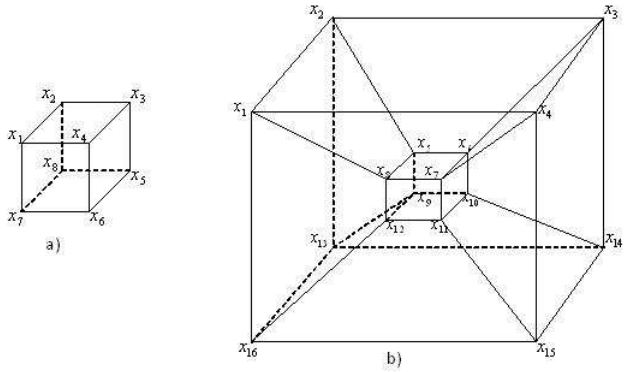


Figure 4. Geometric representation of 3-dimensional cube (a) and 4-dimensional cube (b)

In addition to classic geometric representations of n-dimensional cubes (Figure 4) we will also use the representation of those by respective sequence of sets $\Gamma_0, \Gamma_1, ..., \Gamma_n$ (Figure 5).

If for n-dimensional cube I^n there are known the sets $\Gamma_0, \Gamma_1, ..., \Gamma_n$, then we will write $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$.

Lemma 1. *The number of edges that connect any vertex of* Γ_i *with vertexes of* Γ_{i-1}, $1 \leqslant i \leqslant n-1$, *is equal to* $N^i_{i-1} = i$ *and the number of edges that connect any vertex of* Γ_i *with vertexes of* Γ_{i+1} *is equal to* $N^i_{i+1} = n - i$. *(Obviously,* $N^0_1 = N^n_{n-1} = n$.*)*

Proof. Let x_j be the element of Γ_i, and $g_{i-1}(x_j)$ be the number of elements of Γ_{i-1} adjacent to x_j. On the n-dimensional cube $\mathbf{T}^n$ we introduce Hamming metrics, considering that the sequence $(0, 0, ..., 0)$ is attributed to vertex x_1. Then, any element of Γ_i will have a sequence α with i units. This element will be adjacent to those elements of Γ_{i-1} whose sequences differ from α by exactly one element. The number of such sequences is equal exactly to i (each sequence differs from α

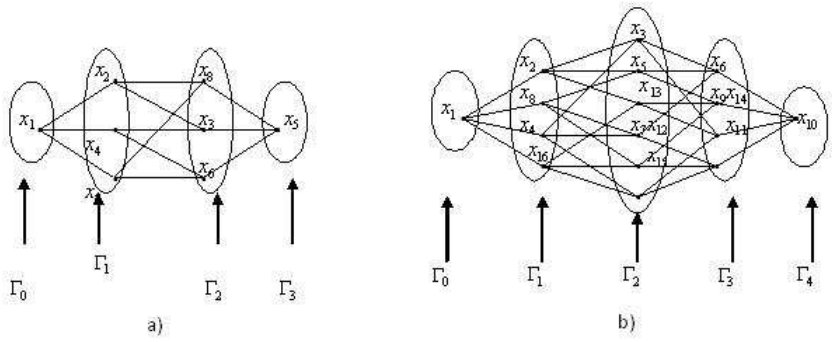


Figure 5. Representation of 3-dimensional cube by the set $\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3$ (a) and 4-dimensional cube by the set $\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ (b)

by the fact that only one from i units of α will be replaced by zero). Therefore, $q_{i-1}(x_j) = i$ and, as the x_j was chosen arbitrarily from Γ_i, we obtain $N^i_{i-1} = i$.

In the n-dimensional cube all vertexes have degree equal to n. Therefore, $N^i_{i+1} = n - i$.

If $i = 0$ and $i = n$ one must calculate only $N^0_1 = N^n_{n-1} = n$. ■

Now we will estimate the sum of distances between an arbitrary element $x \in \Gamma_k$ and the set Γ_{k+l} in case of abstract $n-$dimensional cube $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$, $0 \leqslant k \leqslant n-1$, $1 \leqslant l \leqslant n-k$.

Lemma 2. *The sum of distances between the element $x \in \Gamma_k$ and all elements of set Γ_{k+l} is equal to*

$$\sum_{i=0}^{\min\{k,n-k-l\}} (l+2i) \cdot C^i_k \cdot C^{l+i}_{n-k}.$$

Proof. We choose an arbitrary vertex $x \in \Gamma_k$ and define Hamming metrics on the cube $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$ so that the vertex $x_1 \in \Gamma_0$

is marked by the sequence (0, 0, ..., 0). Binary sequence which corresponds to vertex $x \in \Gamma_k$ will have k units and $(n-k)$ zeros. Without losing generality we will consider that it is the sequence $\widetilde{\alpha} = (\underbrace{11...1}_{k}\ \underbrace{00...0}_{n-k})$. Any element from Γ_{k+l} is marked by a sequence $\widetilde{\beta}$ which contains l units more than $\widetilde{\alpha}$. These l units can be obtained in two ways:

a) some l zeros from $\widetilde{\alpha}$ are replaced with units. The distance between the vertexes, to which $\widetilde{\alpha}$ and $\widetilde{\beta}$ correspond, will be equal to l. So the sequence $\widetilde{\beta}$ can be chosen in C_{n-k}^{l} modes;

b) some i units from $\widetilde{\alpha}$ are replaced by zeros, but some $(l+i)$ zeros are replaced by units (on condition $l+i \leqslant n-k$). As a result, we get a sequence $\widetilde{\beta}$ with $(k+l)$ units and the distances, between $\widetilde{\alpha}$ and $\widetilde{\beta}$ will be $l+2i$ (we apply Hamming distance, which is calculated according to the formula $\sum\limits_{i=1}^{n} |\alpha_i - \beta_i|$). Such a sequence $\widetilde{\beta}$ can be chosen in $C_k^i \cdot C_{n-k}^{l+i}$ modes.

As a result, we obtain that the sum of distances between the element $x \in \Gamma_k$ and all elements of set Γ_{k+1} is:

$$\sum_{i=0}^{\min\{k,n-k+l\}} (l+2i) \cdot C_k^i \cdot C_{n-k}^{l+i}. \quad \blacksquare$$

The sum from Lemma 2 is a constant, that characterizes the relation between the sets Γ_k and Γ_{k+l}. We will denote this constant by σ_k^{k+l}. Similarly it can be proved that the sum of distances between a fixed element $x \in \Gamma_{k+1}$ and all elements of the set Γ_k does not depend on the choice of x, so it is a constant value, which we denote by σ_{k+l}^{k}. It is easy to prove that $\sigma_k^{k+l} \neq \sigma_{k+l}^{k}$, for any values of k and l which verify the relations: $0 \leqslant k \leqslant n-l$, $1 \leqslant l \leqslant n$ and $k+l \neq n-k$. In case when $k+l = n-k$ we have the sets Γ_k and Γ_{n-k} for which $\sigma_k^{n-k} = \sigma_{n-k}^{k}$, i.e. $\sigma_k^{k+l} = \sigma_{k+l}^{k}$.

Corollary of Lemma 2 *For any set Γ_i, $1 \leqslant i \leqslant n$, of the n-dimensional cube $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$, the sequence $\sigma_0^i, \sigma_1^i, ..., \sigma_{i-1}^i$ is decreasing.*

Let $K^n(I^n)$ be the n-dimensional complex of abstract cubes, determined by cube I^n, i.e. $K^n(I^n)$ is formed by I^n and all its faces. From Definition 5 and Definition 6 it results that the n-dimensional transversal divides the complex $K^n(I^n)$ into two connected subcomplexes. Let I^1 be a 1-dimensional face (an edge) of the cube $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$, $\Gamma_0 = \{x_1\}$, $\Gamma_n = \{x_{2^n}\}$, incident to vertex x_{2^n}. We will denote by $T^n_{I^1}$ the transversal determined by the class of parallel edges, which contain the face I^1, and by $K(x_{2^n})$ – the subcomplex of $K^n(I^n)$, determined by $T^n_{I^1}$, which contains the vertex x_{2^n}. Respectively, by $K(\overline{x_{2^n}})$ we will denote the second subcomplex.

Lemma 3. *For any transversal $T^n_{I^1}$ and any set Γ_k, $\left[\frac{n}{2}\right] < k \leqslant n-1$, the following relation holds:*

$$|K(x_n) \cap \Gamma_k| > |K(\overline{x_n}) \cap \Gamma_k| .$$

Proof. Let I^1 be the 1-dimensional face (edge) of the n-dimensional cube I^n, incident to vertex x_{2^n}. We denote the extremities of edge I^1 by a and b, considering $b = x_{2^n}$. The transversal T_{I^1} divides the cube I^n into two $n-1$-dimensional cubes I_1^{n-1} and I_2^{n-1}, so that the vertex a belongs to I_1^{n-1} and the vertex $b = x_{2^n}$ – to cube I_2^{n-1}. Elements of set Γ_k are at the distance $(n-k)$ from the vertex b. The set Γ_k is formed by 2 subsets:

a) $S_k^{'}$ – the set of all vertexes of $(n-1)$-dimensional cube I_1^{n-1}, that is at the distance $(n-k)$ from the vertex b in the cube I^n.

b) $S_k^{''}$ – the set of all vertexes of I_2^{n-1}, that is at the distance $(n-k)$ from b in I^n;

Let's remind now that equality $d(b,z) = d(b,a) + d(a,z)$ holds in I^n for any vertex z of I^{n-1_1} that belongs to $S_k^{'}$. Taking into consideration that b $(b = x_{2^n})$ is vertex of I_2^{n-1} we obtain the inequality $\left|S_k^{'}\right| > \left|S_k^{''}\right|$. This inequality confirms the lemma's affirmation. ■

Theorem 3. *For any set Γ_k of n-dimensional cube $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$, $\left[\frac{n}{2}\right] < k \leqslant n-1$, the relation $\sigma^k_{k-l} > \sigma^k_n$ is true.*

Proof. On the basis of corollary from Lemma 2 it is sufficient to prove that for any set $\Gamma_k^{'}$, $[n/2] < k \leqslant n-1$, the relation $\sigma_{k-l}^k > \sigma_n^k$ is true.

Evidently

$$\sigma_n^k = (n-k) \cdot |S_k| = (n-k) \cdot C_n^k.$$

According to Lemma 2 we obtain:

$$\sigma_{k-1}^k = \sum_{i=0}^{\min\{k-1,n-k\}} (1+2i) \cdot C_{k-1}^i \cdot C_{n-k+1}^{1+i}.$$

Since the theorem is formulated for the sets Γ_k with indexes $k > \left[\frac{n}{2}\right]$, we will have:

$$\min\{k-1, n-k\} = n-k.$$

So for σ_{k-1}^k we get the sum:

$$\sigma_{k-1}^k = \sum_{i=0}^{n-k} (1+2i) \cdot C_{k-1}^i \cdot C_{n-k+1}^{1+i}.$$

Examining expressions σ_n^k and σ_{k-1}^k and taking into consideration the condition $\left[\frac{n}{2}\right] < k \leqslant n-1$, we obtain $\sigma_{k-l}^k > \sigma_n^k$. ■

Let $K^n(I^n)$ be an abstract cubic complex determined by n-dimensional cube $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$, i.e. $K^n(I^n)$ is a complex formed by I^n and all its faces have dimensions k, $0 \leqslant k \leqslant n-1$. We choose a subcomplex $K^q \subset K^n(I^n)$, formed of faces of cube I^n, which are generated by the sets of vertexes $\Gamma_0, \Gamma_1, ..., \Gamma_q$, where $q > \left[\frac{n}{2}\right]$. We build the n-dimensional transversals through each edge I^1 incident to vertex x_1 (x_1 is the unique element of Γ_0) and denote by $K_{I^1}^{n-1}(I^n)$ subcomplex from $K^n(I^n)$ determined by the transversal T_{I^1} and containing at least half of elements of Γ_q. We denote by $Q^1(x_1)$ the family of all 1-dimensional cubes from I^n incident to vertex x_1. Further, the cube I^n will be called the cubic closing of the complex K^q.

Theorem 4. *If $I^n = (\Gamma_0, \Gamma_1, ..., \Gamma_n)$ is a cubic closing of complex of abstract cubes K^q, $[n/2] < q \leqslant n-1$, then*

a) $\left(\bigcap\limits_{I^1 \in Q^1(x_1)} K^{n-1}_{I^1}(I^n) \right) \cap K^q = \varnothing$;

b) $x_{2^n} \in \bigcap\limits_{I^1 \in Q^1(x_1)} K^{n-1}_{I^1}(I^n)$.

Proof. At first, we will prove the relation b) from the Theorem. We should mention that in the cubic closing I^n any transversal determined by the class of parallel edges, which contains an edge incident to vertex x_1, will also contain obligatory an edge incident to vertex x_{2^n}. On the basis of Lemma 3 and condition of the Theorem, it results that:

$$x_{2^n} \in \bigcap_{I^1 \in Q^1(x_1)} K^{q-1}_{I^1}.$$

Let us now prove the relation a). We should mention that in the n-dimensional cube any n-dimensional transversal divides it into two $(n-1)$-dimensional cubes, which are faces of I^n. For n-dimensional cube we can build exactly n transversals, hence, we obtain n pairs of $(n-1)$-dimensional cubes. We denote by $x_1, x_2, ..., x_{2^n}$ the vertexes of cube $I^n = (\Gamma_0 = \{x_1\}, \Gamma_1, ..., \Gamma_{n-1}, \Gamma_n = \{x_{2^n}\})$. Evidently, each transversal contains exactly one edge incident to vertex x_{2^n}. From each pair of $(n-1)$-dimensional cubes determined by transversals from I^n, we choose the cube which contains vertex x_{2^n} and forms a family of $(n-1)$-dimensional cubes $F^{n-1}_{I^n}(x_{2^n})$. The intersection of all cubes from $F^{n-1}_{I^n}(x_{2^n})$ contains only one vertex – vertex x_{2^n}.

The subcomplex $K^{n-1}_{I^1}(I^n)$ contains at least a half of the elements of set Γ_q. Since $q > [n/2]$, this subcomplex does not contain the vertex x_1 from K^q, but contains the vertex x_{2^n} from the cubic closing I^n of K^q. Hence, the affirmation a) is proved. ■

Theorem 5. *If $I^n = (\Gamma_0 = \{x_1\}, \Gamma_1, ..., \Gamma_{n-1}, \Gamma_n = \{x_{2^n}\})$ is the n-dimensional cube, and K^q is a subcomplex of $K^n(I^n)$ formed from faces of the cube I^n, that is generated by the sets $\Gamma_0, \Gamma_1, ..., \Gamma_q$, where*

$q > \left[\frac{n}{2}\right]$, then there exists a system of weights $p(x_i)$ of vertexes of complex K^q so that for any vertex x of K^n with function $f(x) = \sum_{i=1}^{2^n} p(x_i) \cdot d(x, x_i)$ we obtain:

$$f(x) > f(x_{2^n}).$$

Proof. For vertexes of complex K^q we fix the weights:

$p(x_i) = 1$ for any element $x_i \in \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_{q-1}$;

$p(x_i) = M$ for any element $x_i \in \Gamma_q$, where M is a sufficiently big number.

Number M can be chosen so that it will be bigger than the sum of distances from x_{2^n} till all vertexes of set $\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_{q-1}$.

Basing on structure of the set Γ_q and the Theorem 4, we obtain the affirmation of the Theorem 5. ■

We will study further the complexes of abstract cubes $\mathbf{T}^n$ which possess the following properties:

1) any cube $I^k \in int\mathbf{T}^n$ belongs to at least 2^{n-k} n-dimensional cubes;

2) if x is a vertex from border $bd\mathbf{T}^n$, to which exactly t edges, $3 \leqslant t \leqslant n$, are incident and the union $\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_{q-1}$ exists on this border, then the cubic closing of this subcomplex belongs to $\mathbf{T}^n$;

3) homology group of rank zero of complex $\mathbf{T}^n$ is isomorphic with the group of integer numbers, i.e.

$$\square^0(\mathbf{T}^n, Z) \cong Z;$$

4) homology groups of rank $k = 1, 2, .., n$ of complex $\mathbf{T}^n$ are isomorphic with zero, i.e.

$$\square^1(\mathbf{T}^n, Z) \cong \square 2(\mathbf{T}^n, Z) \cong ... \cong \square^n(\mathbf{T}^n, Z) \cong 0;$$

5 Interpretation of complex $\mathbf{T}^n$ in a normed space

Similarly with the case of homogeneous tree of abstract cubes, we define metric d on the set of 0-dimensional cubes (vertexes) from K^n [13].

1-dimensional skeleton of complex $\mathbf{T}^n$ is a directed graph $G = (X; E)$ with the vertexes that correspond to 0-dimensional cubes, and the arcs that correspond to 1-dimensional cubes from $\mathbf{T}^n$.

Let $C_1, C_2, ..., C_m$ be the classes of parallel edges with lengths $d_1, d_2, ..., d_m$. We consider the space R_1^m over the field of real numbers with the norm $\|x\| = \sum_{i=1}^{m} |x_i|$. We construct the segments with lengths $d_1, d_2, ..., d_m$ on axes of coordinates $OY_1, OY_2, ..., OY_m$ from origin $O \in R_1^m$. So the parallelepiped P^m is constructed univocally on these segments. The set of all k-dimensional faces of P^m forms a complex of parallelepipeds, which we will denote by $\mathcal{P}^k = \{P^k \subset P^m | 0 \leqslant k \leqslant m\}$. In case of $k = 1$ we obtain the complex $\mathcal{P}^1$, that represents a connected, metric and undirected graph. We will denote this graph by $H = (Y; V)$.

From modality of construction of complex $\mathcal{P}^m$ and definition of metric on complex, the following theorem results:

Theorem 6. *For the complex $\mathbf{T}^n$ exists an unequivocal application $\alpha : \mathbf{T}^n \longrightarrow \mathcal{P}^m$, that interprets $\mathbf{T}^n$ on the subcomplex of $\mathcal{P}^m$, so that $\alpha : G\mathcal{P}^1$ is an isometry.*

In R_1^m we denote by $Y = \{y_1, y_2, ..., y_m\} \subset \alpha(\mathbf{T}^n)$ the set of vertexes $\alpha(X)$ and consider the following function

$$f(y) = \sum_{i=1}^{m} p(y_i) \cdot \|y - y_i\|, \tag{4}$$

where y_i is the image $\alpha(x_i)$ and $p(y_i) = p(\alpha(x_i))$, $1 \leqslant i \leqslant n$.

We will study this function and prove that the point $y^* \in R_1^m$ such as

$$f(y^* = \min_{y \in R_1^m} f(y)) = \min_{y \in R_1^m} \sum_{i=1}^{m} p(y_i) \cdot \|y - y_i\|$$

is a median of graph $H = (Y; V)$.

6 Median calculation

For an arbitrary point $y = (y^1, y^2, ..., y^m) \in R_1^m$ and every point $y_j = (y_j^1, y_j^2, ..., y_j^m) \in Y$, $1 \leqslant j \leqslant m$ we form the sets

$$\begin{aligned} J^+ &= \{j : y^i - y_j^i > 0\} \\ J^0 &= \{j : y^i - y_j^i = 0\} \\ J^- &= \{j : y^i - y_j^i < 0\}. \end{aligned}$$

similarly to [22].

We denote

$$\begin{aligned} A &= \sum_{j \in J^- \cup J^0} P(y_j), \\ B &= \sum_{j \in J^+} P(y_j), \\ C &= \sum_{j \in J^-} P(y_j), \\ D &= \sum_{j \in J^- \cup J^0} P(y_j). \end{aligned} \tag{1}$$

As in case of homogeneous complex of multi-ary relations [13], the following theorem holds:

Theorem 7. *The point $y_j \in Y \subset R_1^m$ minimizes the function (4) if and only if the following relations are satisfied:*

$$A \geqslant C \quad \textit{and} \quad B \leqslant D. \tag{5}$$

We denote that the calculation of median does not depend on the edges length from classes $C_1, C_2, ..., C_m$. This suggests us the idea that the parallelepiped P^m could be replaced by a unitary cube in space R_1^m.

Thus, we can define the application $\beta : P^m \longrightarrow Q^m$, where Q^m is a unitary cube situated at the origin of coordinate system. As a result of application β, the graph H passes in a metric graph $\beta(H)$, which we will denote by $\mathcal{G} = (Z; W)$. This graph is a subgraph of 1-dimensional skeleton of the cube Q^m. As a result, we obtain that on the cube and respectively on the graph G the Hamming metric is defined. Hence, the application $\beta\alpha(C_i)$, $1 \leqslant i \leqslant n$, represents a set of edges C_i^1 in the constructed graph $\mathcal{G} = (Z; W)$. For the set of vertexes of the graph G we will keep the system of weights of the complex K^n, i.e. $p(z_i) = p(x_i)$, where $z_i = \beta\alpha(x_i), 1 \leqslant i \leqslant n$, but n represents the number of 0-dimensional cubes of K^n.

So we obtained the application:

$$\beta\alpha : G \longrightarrow \mathcal{G}.$$

Let's examine an arbitrary vertex z_i of the cube Q^m. We represent its coordinates by a sequence formed from 0 and 1. From coordinates of vertices z_i of the graph G we form a matrix A, similarly to the case of homogeneous complex of abstract cubes [13].

For the matrix A we calculate a resulting sequence $r = (r_1, r_2, ..., r_m)$ considering the following:

a) $r_j = 1$, if $\sum\limits_{i=1}^{n} z_i^j \cdot p(z_i) > 1/2 \cdot \sum\limits_{i=1}^{n} p(z_i)$;

b) $r_j = 0$, if $\sum\limits_{i=1}^{n} z_i^j \cdot p(z_i) < 1/2 \cdot \sum\limits_{i=1}^{n} p(z_i)$;

c) $r_j = 0$ or 1(indifferently), if $\sum\limits_{i=1}^{n} z_i^j \cdot p(z_i) = 1/2 \cdot \sum\limits_{i=1}^{n} p(z_i)$.

Theorem 8. *A vector $r = (r_1, r_2, ..., r_m)$ calculated according to the rules a)-c) represents a line of matrix A.*

Proof. Assume the opposite. Let there exists a vertex $r = (r_1, r_2, ..., r_m)$ in the cube Q^m, which does not belong to the graph $\mathcal{G} = \beta\alpha(\mathcal{G})$, but this vertex minimizes the function (4). Obviously, m faces of dimension $m-1$ correspond to r. Each of these faces contains the $(n-1)$-dimensional transversal of complex $\beta\alpha(\mathbf{T}^n)$. We should mention that r respects the relations (4.9) from [13]. In this complex any m transversal $\beta\alpha(T^{n-1}_{I^1_1}), ..., \beta\alpha(T^{n-1}_{I^m_1})$ has an empty intersection, because r does not belong to them. It is in contradiction to the Theorem 6. ■

References

[1] D. S. Hochbaum. *Heuristics for the fixed cost median problem*, Math. Programming, 22, 1982, pp. 148–162.

[2] N. Christofides. *Graph theory: an algorithmic approach,* Academic Press, 1975, 400p.

[3] P. Soltan, D. Zambiţchi, Ch. Prisăcaru. *Extremal problems on graphs and algorithms for their solution,* Chişinău, Ştiinţa, 1973, 90p. (in Russian).

[4] V. Soltan. *Introduction to axiomatic theory of convexity,* Chişinău, Ştiinţa, 1984, 221p. (in Russian).

[5] P. Soltan. *Extremal problems on convex sets,* Chişinău, Ştiinţa, 1976, 115p (in Russian).

[6] V. Boltyansky, P. Soltan. *Combinatorial geometry of various classes of convex sets,* Chişinău, Ştiinţa, 1978, 280p. (in Russian).

[7] K. Menger. *Untersuchungen uber allgemeine Metrik. I, II, III,* Math. Ann., nr.100, 1928, p.75–163 (in German).

[8] J. De. Groot. *Some special metrics in general topology,* Collog. Math., 6 (1958), pp. 283–286.

[9] A. D. Aleksandrov, V. A. Zalgaller. Two – dimensional manifolds of bounded curvature, *(Foundations of the intriusic geometry of surfaces)*, Trudy Math. Inst. Steklov., 63, Acad. Sci. USSR, Moscow – Leningrad, 1962, 262p. (in Russian).

[10] F. A. Toranzos. *Inmersion de espacios metricos convexos en* , Math. Notac, 21, Nos. 1-2 (1966/67), pp.29-53. (in Spanish).

[11] E. Soetens. *Convexity in Busemann spaces,* Bull. Soc. Math. Belg., 19, nr.2, 1967, pp. 194–213.

[12] T.T. Arkhipova, I. V. Sergienko. *On the formalization and solution of some problems of organizing the computing process in data processing systems,* Kibernetika (kiev), nr.5, 1973, pp. 11–18 (in Russian); English transl. in Cybernetics 9, 1973.

[13] S. Cataranciuc, P. Soltan. *Complex of abstact cubes and median problem,* Computer Science Journal of Moldova. Vol.19, nr.1(55) 2011, pp. 38–63.

[14] M. Bujac, S. Cataranciuc, P. Soltan. *On the division in cubes of abstract manifolds,* Buletinul Academiei de Ştiinţe a Rep. Moldova. Seria Matematica, nr. 2(51), Chişinău, 2006, pp. 29–34.

[15] S. Cataranciuc, P. Soltan. *Abstract complexes, their homologies and applications,* Buletinul Academiei de Ştiinţe a Rep. Moldova. Seria Matematica, nr.2(63), Chişinău, 2010, pp. 31–58.

[16] S. Cataranciuc. *G-complex of multi-ary relations,* Analele Ştiinţifice ale USM, Seria "Ştiinţe fizico-matematice", Chişinău, 2006, pp. 119–122 (in Romanian).

[17] S. Cataranciuc, P. Soltan. *Hypergraphs and their homologies,* Trends in the Development of the Information and Communication Technology in Education and Management. International Conf., March 20-21, 2003, Chişinău, pp. 294–300 (in Romanian).

[18] P. J. Hilton, S. Wylie. *Homology Theory: An Introduction to Algebraic Topology,* Cambridge University Press, New York 1960 xv+484p.

[19] M. Bujac. *Classification of abstract multidimensional orientable manifolds without borders,* Analele Ştiinţifice ale USM, Seria "Ştiinţe fizico-matematice", Chişinău, 2003, pp. 247–250 (in Romanian).

[20] V. Boltyanski, H. Martini, P. Soltan. *Excursions into Combinatorial Geometry,* Springer, Berlin-Heidelberg, 1997, 444p.

[21] W. Hamming Richard. *Error detecting and error correcting codes.* Bell. System Technical Journal Nr. 29(2), 1950, pp. 147-160.

[22] P.S. Soltan, K.F. Prisacaru. *The Steiner Problem on Graph.* Dokl. Acad. Nauk SSSR, 198(1971), Nr.1, p.46–49.

[23] A.G. Kurosh. *Lectures on general algebra,* Fizmatgiz, Moscow, 1962, 396p. (in Russian).

Sergiu Cataranciuc

Received February 2, 2013

State University of Moldova
60 A. Mateevici street, MD-2009, Chişinău, MD-2009,
E-mail: *s.cataranciuc@gmail.com*

Computer Science Journal of Moldova, vol.**21**, no.1(61), 2013

Smoke detection algorithm for intelligent video surveillance system

N. Brovko, R. Bogush, S. Ablameyko

Abstract

An efficient smoke detection algorithm on color video sequences obtained from a stationary camera is proposed. Our algorithm considers dynamic and static features of smoke and is composed of basic steps: preprocessing; slowly moving areas and pixels segmentation in a current input frame based on adaptive background subtraction; merge slowly moving areas with pixels into blobs; classification of the blobs obtained before. We use adaptive background subtraction at a stage of moving detection. Moving blobs classification is based on optical flow calculation, Weber contrast analysis and takes into account primary direction of smoke propagation. Real video surveillance sequences were used for smoke detection with utilization our algorithm. A set of experimental results is presented in the paper.

Keywords: smoke detection, video sequences, background subtraction, Weber contrast analysis

1 Introduction

Reliable and early fire detection on open spaces, in buildings, in territories of the industrial enterprises is important making any system of fire safety. Traditional fire detectors which have been widely applied in the buildings are based on infrared sensors, optical sensors, or ion sensors that depend on certain characteristics of fire, such as smoke, heat, or radiation. Such detection approaches require a position of sensor in very close proximity to fire or smoke and often give out false alarms. So they may be not reliable and cannot be applied into open spaces

and larger areas. Due to the rapid developments in digital camera technology and video processing techniques currently intelligent video surveillance systems are installed in various public places for monitoring, therefore there is a noticeable trend to use such systems for early fire detection with special software applied. Smoke detection is rather for fire alarm systems when large and open areas are monitored, because the source of the fire and flames cannot always fall into the field of view. However, smoke of an uncontrolled fire can be easily observed by a camera even if the flames are not visible. This results in early detection of fire before it spreads around.

Motion and color are two usually used important features for detecting smoke on the video sequences. Motion information provides a key as the precondition to locate the possible smoke regions. The algorithm of background subtraction is traditionally applied to movement definition in video sequence [1-4]. Common technique uses adaptive Gaussian Mixture Model to approximate the background modeling process [1, 2].

In [5], optical flow calculation is applied to smoke movement detection. Lacks of the given approach are high sensitivity to noise and high computational cost. Algorithms based on color and dynamic characteristics of a smoke are applied for classification of the given moving blobs. In [6] the algorithm comparative evaluation of the histogram-based pixel level classification is considered. In this algorithm the training set of video sequences on which there is a smoke is applied to the analysis. Methods based on preliminary training are dependent on classification quality on a training set. It demands many qualitative characteristics of processed video images. The area of decreased high frequency energy component is identified as smoke using wavelet transforms [1, 2]. However change of scene illumination can be contours degradation reason. Therefore such approach requires additional estimations.

Color information is also used for identifying smoke in video. Smoke color at different stages of ignition and depending on a burning material is distributed in a range from almost transparent white to saturated gray and black. In [1] decrease in value of chromatic components U

and V of color space YUV is estimated.

In this paper we propose an algorithm for smoke detection on color video sequences obtained from a stationary camera. Our algorithm consists of the following steps: preprocessing; slowly moving areas and pixels segmentation in a current input frame based on adaptive background subtraction; merge slowly moving areas with pixels into blobs; classification of the blobs obtained before. We use adaptive background subtraction at a stage of moving detection. Moving blobs classification is based on optical flow calculation, Weber contrast analysis and takes into account primary direction of smoke propagation.

2 Algorithm description

The proposed algorithm uses motion and contrast as the two key features for smoke detection. Motion is a primary sign and is used at the beginning for extraction from a current frame of candidate areas. In addition to consider a direction of smoke distribution the movement estimation based on the optical flow is applied. The relation of smoke intensity to background intensity above than at objects with similar behavior, such as a fog, shadows from slowly moving objects and patches of light. Therefore contrast calculated with Weber formula is a good distinctive sign for a smoke. The algorithm is a group of the following modules as it is shown in Figure 1. A consecutive frames I_{t-2},I_{t-1}, I_t and obtained from the stationary video surveillance camera are entered to an input of the preprocessing block. This block carries out some transformations which improve contrast qualities of the input frames and reduce calculations. Then adaptive background subtraction is applied to extract from the frame I_{t+1} of slowly moving areas and pixels of the so-called foreground. The background subtraction adaptive algorithm considers that a smoke gradually is mixed to a background. Then the connected components analysis is used in order to clear the foreground noise and to merge the slowly moving areas with pixels into blobs. The received connected blobs are transferred into the classification block for Weber contrast analysis. At the same time the connected blobs are entered to an input of the block for optical flow calculation.

Finally the classification block processes the information to obtain the final result of smoke detection.

2.1 Frame preprocessing

The preprocessing block applies some methods of image processing which increase the performance of the proposed detection algorithm and reduce false alarms. Frame preprocessing block comprises three steps: grayscale transformation, histogram equalization and the discrete wavelet of the current input frame. Cameras and image sensors must usually deal not only with the contrast on a scene but also with the image sensors exposure to the resulting light on that scene. Histogram equalization is a most commonly used method for improvement of contrast image characteristics. To resize the image and to remove high frequencies on horizontal, vertical and diagonal details the discrete wavelet transform to Haar basis is applied. Wavelet transform to Haar basis is the simplest and the fastest [7] algorithm that is important for systems of video processing.

2.2 Slowly moving areas and pixels segmentation

In the course of the distribution a smoke is being gradually blended to the background. Our adaptive algorithm of background subtraction considers this characteristic of a smoke and is based on the ideas stated in works [2,8]. A background image B_t at time instant t is recursively estimated from the image frame I_{t-1} and the background image B_{t-1} of the video as follows [9]:

$$B_t(x,y) = \begin{cases} \alpha B_{t-1}(x,y) + (1-\alpha) I_{t-1}(x,y), \ if \ (x,y) \ is \ moving, \\ B_{t-1}(x,y), \ if \ (x,y) \ is \ stationary, \end{cases}$$

where (x, y) represent a pixel video frame and α is an adaptation parameter between 0 and 1. As the area of a smoke frame by frame grows slowly, so that the pixels belonging to a smoke quickly are not fixed in a background, value α should be close to 1.

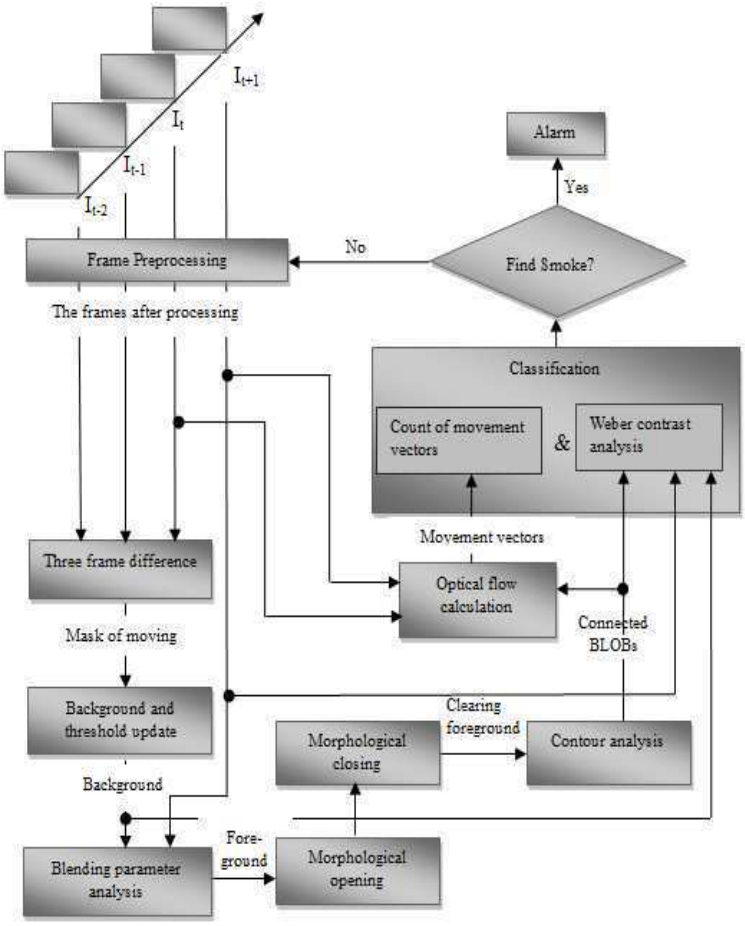


Figure 1. Flow chart of our proposed algorithm

At the initial moment of time $B_0(x,y) = I_0(x,y)$. Pixel (x,y) belongs to moving object if the following condition is satisfied [8]:

$$(|I_t(x,y) - I_{t-1}(x,y)| > T_t(x,y))\&(|I_t(x,y) - I_{t-2}(x,y)| > T_t(x,y)),$$

where $I_{t-2}(x,y)$, $I_{t-1}(x,y)$, $I_t(x,y)$ values of intensity of pixel (x,y) at time instant $t-2$, $t-1$ and t respectively; $T_t(x,y)$ is adaptive threshold for pixel (x,y) at time instant t calculated as follows:

$$T_t(x,y) = \begin{cases} \alpha T_{t-1}(x,y) + (1-\alpha)(5 \times |I_{t-1}(x,y) - B_{t-1}(x,y)|), \\ if\ (x,y)\ is\ moving, \\ T_{t-1}(x,y),\ if\ (x,y)\ is\ stationary. \end{cases}$$

At the initial moment of time $T_0(x,y) = const > 0$.

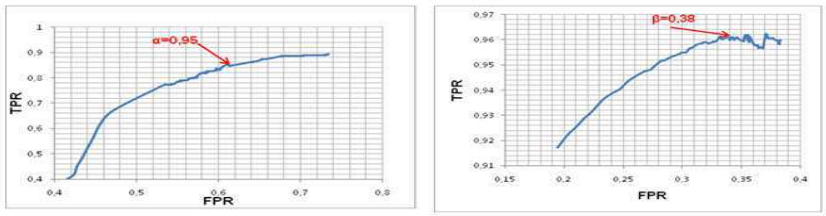
Accurate separating of a foreground object from the background is the main task of digital matting. Porter and Duff [9] introduced the blending parameter (so-called alpha channel) as a solution of this problem and a mean to control the linear combination of foreground and background components. Mathematically the current frame I_{t+1} is modeled as a combination of foreground F_{t+1} and background B_t components using the blending parameter β:

$$I_{t+1}(x,y) = \beta F_{t+1}(x,y) + (1-\beta)B_t(x,y).$$

For opaque objects the value of β is equal to 1, for transparent objects the value of β is equal to 0 and for the semitransparent objects, such as smoke, the value of β lays in a range from 0 to 1. As it is shown further in this section, we have experimentally established the optimum value for β to be equal to 0.38.

So, as soon as we have obtained B_t component on background update step, the current frame I_{t+1} and β set to 0.38, we can estimate the foreground component F_{t+1}. Then we apply the threshold processing to receive the binary foreground F_{bin}:

$$F_{bin} = \begin{cases} 1,\ if\ F_{t+1} > 245, \\ 0,\ otherwise. \end{cases}$$



(a) (b)

Figure 2. ROC curve for variable α (a) and β (b)

At the current step of algorithm we have 2 parameters α and β which are necessary to be estimated. Optimum values of α and β can be estimated using receiver operating characteristic (ROC) analysis. For estimation implementation the training set from 5 video sequences of the 200 frames length which contain and do not contain smoke were used. Using the ground truth regions which have been online marked as a smoke in the training frames, rates of true and false detection were calculated for the whole frame set. We received a background for each value of α within a range of (0, 1). After that we applied a background subtraction and thresholding to each frame from a training set. And then True Positive Rate (TPR) and False Positive Rate (FPR) were calculated as follows:

$$TPR = \frac{TP}{P}; FPR = \frac{FP}{N},$$

where TP – number of correctly classified pixels, P – number of all positive classified pixels; FP – number of incorrectly classified pixels, N – number of all negative classified pixels. For each value of α, the average TPR and FPR is evaluated on a training frame set and used in the ROC curve (Figure 2a).

Using the ROC curve, an optimum value for α can be easily selected for the smoke detection algorithm based on a pre-defined correct detection versus false detection rates. It is necessary to choose such value of α that slowly moving objects will not join a background too quickly,

i.e. that a smoke will not be fixed in a background too fast. At the given stage of algorithm high TPR is important and high enough FPR is acceptable as it is necessary to receive as much as possible pixels for the analysis, and incorrectly classified pixels should be excluded at the following stages. Therefore we have established that an α value equals to 0.95. Similarly using the training frame set, receiving a foreground component F_{t+1} and after that the foreground F_{bin} and counting FPR and TPR for all values β from a range (0,1) with the step 0.001 we build a ROC curve for β (Figure 2b). Value of β has been chosen to be equal to 0.38, because such value of β provides high TPR and low FPR.

2.3 Connected component analysis

At the next step of algorithm to clear of noise and to connect moving blobs the connected components analysis is used. This form of analysis takes in a noisy input foreground. Morphological operations are applied to reduce the noise:

$$S \circ M = (S(-)M) \oplus M,$$

where S is image, M – structuring element 3×3; morphological closing to rebuild the area of surviving components that was lost in opening is the following:

$$S \bullet M = (S \oplus M)(-)M,$$

where M – structuring element 3×3.

Then search of all contours is carried out. Then it tosses the contours that are too small and approximate the rest with polygons.

2.4 Moving blobs classification

Blocks matching approach for optical flow calculation assumes that the frame is divided into small regions called blocks. It considers a primary direction of smoke propagation. In [10] it is shown, that global direction of smoke is 0-45°. This statement allows to simplify procedure of blocks matching detection and, hence, considerably to reduce number

of calculations. Blocks are typically squares and contain some number of pixels. These blocks do not overlap. In our implementation frames of the size 320×240 pixels are divided into blocks of 2×2 pixels. Block matching algorithm attempts to divide both the previous and current frames into such blocks and then computes the motion of these blocks. Each block of size 2×2 can move in eight possible directions. Our implementation searches in three directions of the original block $q_{x,y}^{prev}$ (in the previous frame) and compares the candidate new blocks $q_{x-1,y-1}^{curr}$, $q_{x,y-1}^{curr}$ and $q_{x+1,y-1}^{curr}$ (in the current frame) with the original. This comparison is calculated as follows:

$$F(q_{x,y}^{prev}, q_{x+k,y-1}^{curr})_{x,y\in[2;N]}^{k\in\{-1;0;1\}} = \left(\frac{min(I_{i,j}^{prev}, I_{i,j}^{curr})}{max(I_{i,j}^{prev}, I_{i,j}^{curr})}\right),$$

where $I_{i,j}^{prev}$ is the intensity value of pixel on the previous frame, belonging to the block $q_{x,y}^{prev}$; $I_{i,j}^{curr}$ is the intensity value of pixel on the current frame, belonging to the block $q_{x,y}^{curr}$; N is the number of blocks into which the previous and current frame are divided.

The block $q_{x,y}^{prev}$ in the previous frame will correspond to the block in the current frame if function F has the maximum value. Optical flow calculation (function F) is done only for the blocks belonging to the foreground (Figure 3b). The result of this step is the set of vectors c_s having a direction corresponding to primary propagation of smoke (Figure 3c).

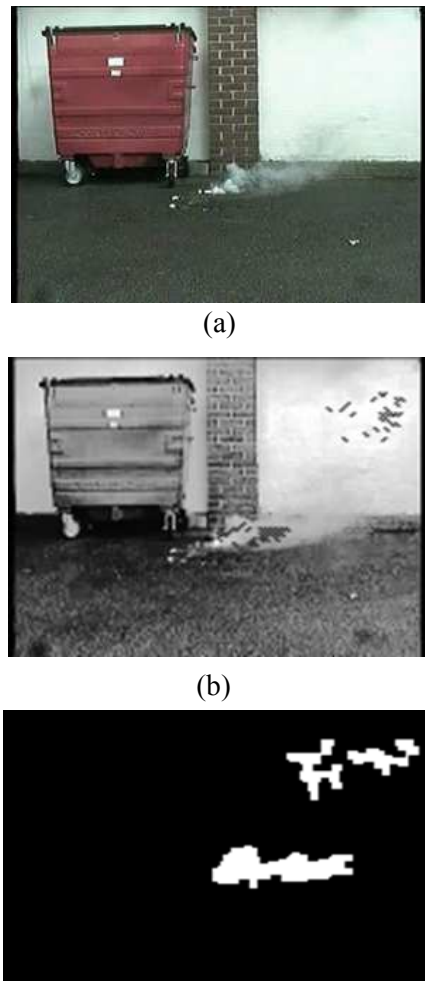
From each blob from the previous steps we calculate percentage ρ of blocks which have moved in primary direction of smoke:

$$\rho = \frac{c_s}{c}100\%,$$

where c – the total number of blocks on a current frame, and Weber contrast C_w:

$$C_w = \frac{1}{n}\sum_{i=1}^{n}\frac{I_{t+1}(x,y) - B_t(x,y)}{B_t(x,y)},$$

where $F_{t+1}(x,y)$ – value of pixel intensity (x,y) at time instant t, belonging to a blob, $B_t(x,y)$ – value of background pixel intensity (x,y)



(a)

(b)

(c)

Figure 3. The current frame (a), the clean up foreground (b) by the connected components analysis and the results of optical flow calculation (c)

at time instant t under blob, n – number of the pixels belonging to a blob.

If the blob has been successfully checked out, then we classify it as a smoke. Experimentally established values $C_w > 0,5$ and $\rho > 20\%$ allow efficient distinguishing of a smoke from objects with similar behavior: a fog, shadows from slowly moving objects and patches of light.

3 Results and discussion

The developed algorithm was tested on the real cases. Tests were run on a PC (Pentium(R) DualCore CPU T4300, 2,1 GHz, RAM 1,96GB). Our program was implemented using Visual C++ and an open source computer vision library OpenCV. The proposed algorithm has been evaluated using data set publicly available at the web address http://signal.ee.bilkent.edu.tr/VisiFire/Demo/SampleClips.html and http://www.openvisor.org. Test video sequences contain a smoke, moving people, moving transport, a complex dynamic background, and also a number of video sequences without any smoke. Figure 4 shows some examples of smoke detection.

Detection results for some of the test sequences are presented in Table 1. Processing time of a current frame depends on the blob sizes and frequency of changes occurring in a background. If the background is stable and few blobs are detected, then processing time decreases. Table 1 (the second column) contains average processing time on all frames for each test video sequence. The smoke has been found successfully out on all test video sequences with a smoke.

If at first a strongly rarefied smoke moves slowly, then it is gradually included into the background. Therefore in this case, we cannot directly find out a smoke, and the detection time increases. The performed experiments have shown that the algorithm quickly finds out a smoke on a complex dynamic scene. Smoke detection is achieved practically in real time. The processing time per frame is about 15 ms. for frames with sizes of 320 by 240 pixels. The algorithm considers both dynamic and static features of a smoke. The algorithm has a low false alarm level. False alarms on objects with properties similar to a smoke are

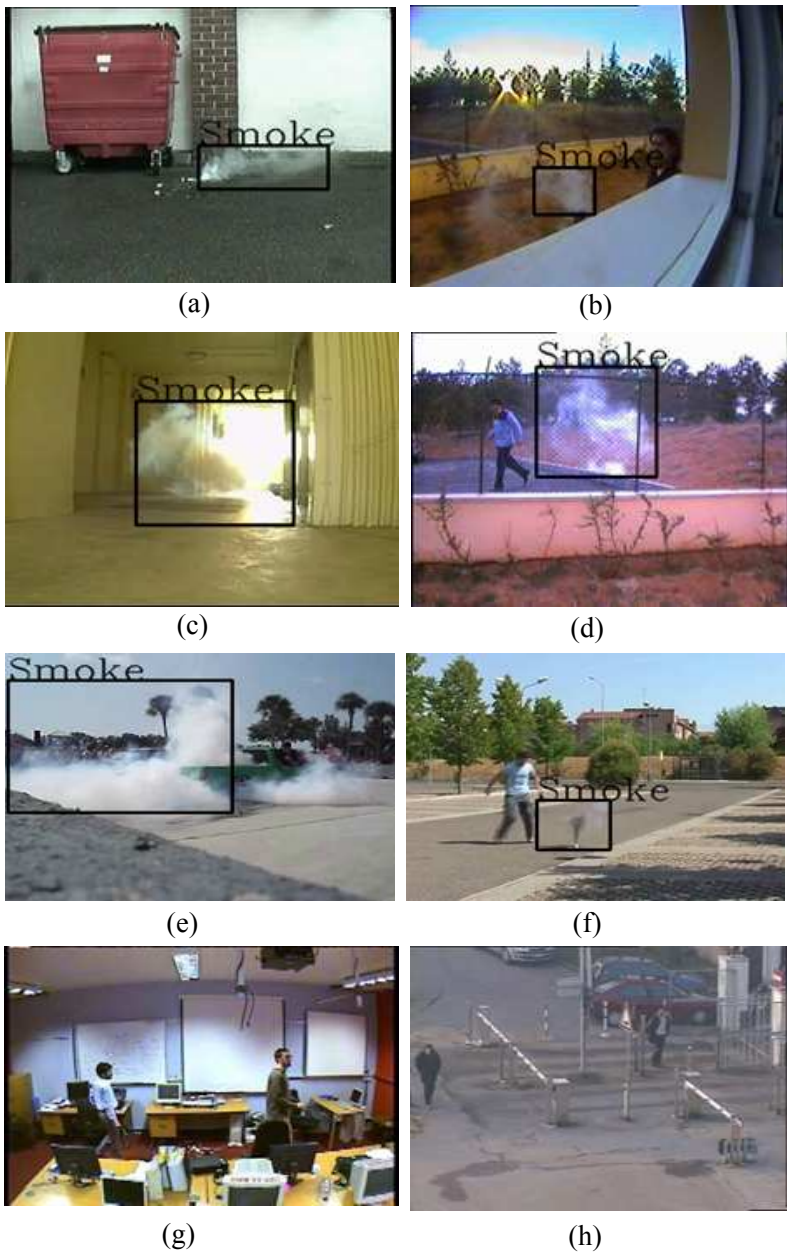


Figure 4. Smoke detection in real video sequences

sometimes possible. Tracing of smoky properties during some frames can solve this problem.

The smoke and flame are primary signs of a fire. Often there is a visible smoke development prior to flame. It can be important for early fire prevention. Therefore our algorithm can be effectively used in video surveillance systems for early detection of fire on open spaces.

Table 1. Detection results of our algorithm for some of the test sequences

Video sequences (Figure4)	The processing time per frame (ms)	The smoke was presented with / is found with (number of frame)	The number of frames on which there was false alarm / total of frames
a	12,665	10 / 12	0 / 900
b	14,786	20 / 112	0 / 244
c	14,969	80 / 87	0 / 483
d	15,003	30 / 117	0 / 630
e	15,491	360 / 388	0 / 2200
f	14,039	463 / 469	0 / 1835
g	16,346	-	0 / 1073
h	14,567	-	0 / 1179

4 Conclusion

We have presented in this paper an algorithm for smoke detection in video sequences. Our algorithm consists of the following steps: preprocessing; slowly moving areas and pixels segmentation in a current input frame based on adaptive background subtraction; merge slowly moving areas with pixels into blobs; classification of the blobs obtained before. We use adaptive background subtraction at a stage of moving

detection. Moving blobs classification is based on optical flow calculation, Weber contrast analysis and takes into account primary direction of smoke propagation. The efficiency of our approach is illustrated and confirmed by our experimental videos.

References

[1] P. Piccinini, S. Calderara, R. Cucchiara. *Reliable smoke detection system in the domains of image energy and color.* 15th International Conference on Image Processing, (2008), pp.1376–1379.

[2] B.Ugur Toreyin et al. *Wavelet based real-time smoke detection in video.*Signal Processing: Image Communication, EURASIP, Elsevier (20) (2005), pp. 255–256.

[3] DongKeun Kim, Yuan-Fang Wang. *Smoke Detection in Video.* World Congress on Computer Science and Information Engineering (2009), pp. 759–763.

[4] B. Ugur Toreyin, Yigithan Dedeoglu, A. Enis Cetin. *Contour based smoke detection in video using wavelets.* In European Signal Processing Conference (2006), pp. 123–128.

[5] F. Comez-Rodriuez et al. *Smoke Monitoring and measurement Using Image Processing. Application to Forest Fires.* Automatic Target Recognation XIII, Proceedings of SPIE (2003), pp. 404–411.

[6] D. Krstinić, D. Stipaničev, T. Jakovčević. *Histogram-Based Smoke Segmentation in Forest Fire Detection System.* Information Technology and Control 38(3) (2009), pp.237–244.

[7] E. Stolnitz, T. DeRose, D. Salesin. *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kaufmann (1996), pp. 1–272.

[8] R.T. Collins. *A System for Video Surveillance and Monitoring.* Proc. of American Nuclear Society 8th Int. Topical Meeting on Robotics and Remote Systems (1999), pp. 68–73.

[9] T. Porter, T. Duff. *Compositing digital images.* Computer Graphics, vol.18, no 3 (1984), pp. 253–259.

[10] R. Yasmin. *Detection of Smoke Propagation Direction Using Color Video Sequences.* International Journal of Soft Computing 4 (1) (2009), pp. 45–48.

N. Brovko, R. Bogush, S. Ablameyko Received December 13, 2012

N. Brovko
Polotsk State University
29, Blokhin str., Novopolotsk, Belarus, 211440
E–mail: *nadzeya.brouka@gmail.com*

R. Bogush
Polotsk State University
29, Blokhin str., Novopolotsk, Belarus, 211440
E–mail: *bogushr@mail.ru*

S. Ablameyko
Belarussian State University
4, Nezavisimosti av., Minsk, Belarus, 220050