Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Solving Problems in Various Domains by Hybrid Models of High Performance Computations

Yurii Rogozhin Artiom Alhazov Lyudmila Burtseva
Svetlana Cojocaru Alexandru Colesnicov
Ludmila Malahov

Abstract

This work presents a hybrid model of high performance computations. The model is based on membrane system (P system) where some membranes may contain quantum device that is triggered by the data entering the membrane. This model is supposed to take advantages of both biomolecular and quantum paradigms and to overcome some of their inherent limitations. The proposed approach is demonstrated through two selected problems: SAT, and image retrieving.

1 Introduction

The present paper concerns definition and investigation of new computational models based on combination of biomolecular and quantum approaches. This new approach springs from practical needs of several disciplines delivering the hard tasks. Both existing quantum- and bio- models of calculation are widely used to solve hard tasks being not always satisfactory. Each paradigm has its own advantages and disadvantages. The proposed hybrid model supposes to compensate restrictions of existing models and to expose their benefits.

We will use membrane computing, or P systems [1] that was motivated by the structure and functioning of a living cell. The model is

based on a cell-like hierarchical arrangement of membranes. (There is a variation named the tissue model that uses a non-hierarchical arrangement.) Membranes delimit compartments where objects presented by multisets, numbers, or strings evolve according to the given evolution rules. Many variants dependent on permitted rules and operations exist; we will not restrict ourselves by a particular type.

Quantum computing uses quantum properties to represent data and perform operations over them [2]. Each quantum computation inevitably includes non-quantum steps. Quantum algorithms always begin with the (non-quantum) preparation of the initial observable (classical) state. Then a sequence of quantum operations is applied to the system whose states are unobservable during the process. At the end, a (non-quantum) measurement is performed, and the quantum system collapses to its final observable state.

One of the first hybrid computational models was proposed by A. Leporati [3]. His hybrid of membrane and quantum systems develops previously introduced UREM (Unit Rules and Energy assigned to Membranes) P systems. The former adds "energy" to the objects in a membrane system, and rules can be applied to objects inside a membrane only if there is enough energy to do so. Quantum UREM P system changes objects and rules: objects are represented as pure states of a quantum system, and rules are quantum operators. The result is a hybrid computation device, a membrane system with quantum operations.

We propose a different variant of hybrid model that keeps the entire power of P systems. Our hybrid model is the classical P system framework in which two types of membranes coexist: classical membranes, and quantum membranes, the latter containing a quantum device inside. Entrance of an object into a quantum membrane triggers the quantum computation, while the entered object is available as data in the initial state of the quantum device.

The process of hybrid calculation is illustrated by solutions of two problems. These problems represent the opposite sides of problems range: from theoretical computing (SAT problem) to everyday practical application (medical image retrieval). Presentation of such different

problems intends to demonstrate general potential of the proposed hybrid model.

This is an introductory paper dedicated to the proposed hybrid model of calculations. The article demonstrates the applicability and operability of the model on two selected examples. More applications and other aspects like estimations of efficiency and consumed resources, synchronization, hybrid simulator, etc., are subjects of further works.

2 P Systems and Hybrid Model

The first problem we selected to illustrate our construction is the SAT problem. The second selected problem is the image retrieval problem.

Transitional P systems with inhibitors. For SAT problem, we use non-cooperative transitional P systems with atomic inhibitors [4]. Although this class of systems is not even computationally complete [5], it fits well to illustrate the power of the hybrid model. We introduce here only the necessary definitions.

A non-cooperative transitional P system with atomic inhibitors with input is defined as a tuple

$$\Pi = (O, \Sigma, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0),$$

where: O is a finite alphabet; $\Sigma \subseteq O$ is the input subalphabet; μ is a membrane structure (a rooted tree, traditionally represented by bracketed expression, e.g., $[\ [\ ]_2\ ]_1$ denotes membrane 2 in membrane 1, and the set of labels of membranes from μ is $H = 1, \ldots, m$); w_i, $i \in H$, are the initial multisets associated to regions i (directly inside the corresponding membrane), traditionally represented by strings over O (only the multiplicities of symbols being relevant, not their order); R_i, $i \in H$, are the sets of rules associated to regions i; and i_0 is the label of input membrane (an input multiset over Σ is added to the initial multiset in region i_0 into the starting configuration). In this paper, multisets w_i and sets R_i are omitted if i is a label of a quantum membrane.

The rules of the corresponding model are of the forms $a \to u$ or $a \to u|_{\neg b}$, where $a, b \in O$, $u \in (O \times Tar)^*$, $Tar = \{here, out\} \cup \{in_j \mid j\}$,

and in this case j denotes a label of immediately inner membrane. The effect of a rule is replacing object a with a multiset of objects specified in the right side, in the regions specified by target indications (*here* may be omitted). A rule with inhibitor b is applicable whenever b absent. A transition step consists in parallel application of applicable rules to all possible objects (non-deterministically if there is a choice). The computation stops when no rules are applicable.

Symport/antiport tissue P systems. For the image retrieval problem, we use tissue P systems with symport/antiport rules. A tissue P system with symport/antiport rules with input is defined by a tuple

$$\Pi = (O, \Sigma, E, d, w_1, \cdots, w_d, R, i_0, o_0),$$

where: O is the alphabet; Σ is the input subalphabet; E is the set of objects occurring in the environment in infinitely many copies; d is the degree of the system ($H = \{1, \ldots, d\}$ is the set of labels of regions called cells, and the environment region is labeled by 0), w_i, $i \in H$, is the initial content of cell i; R is the set of rules; i_0 is the label of the input region (where a multiset over Σ is additionally placed in the beginning of the computation); o_0 is the label of the output region.

The rules have the form $(i, u/v, j)$, meaning that a multiset u may move from region i to region j, coupled with moving of a multiset v from region j to region i. The rules are applied non-deterministically, in the maximally parallel mode (i.e., no further rules can be applicable to the idle objects).

The structure of the tissue is deduced from rules. A rule $(i, u/v, j)$ means that the i-th and the j-th cells are neighboring.

Hybrid model. We now present the formal description of hybrid model

$$\beta = (\Pi, H_Q, N_Q, Inp, Outp, Q_1, \ldots, Q_m).$$

A hybrid system β is defined by a membrane system Π (let us denote its membrane/cell label set as H and assume in the membrane case that membranes with labels in $H_E \subseteq H$ are elementary), where quantum devices Q_j are associated to the elements j of a subset of elementary membranes/cells $H_Q \subseteq H_E$. No membrane rules are as-

sociated to the objects inside the quantum membranes. N_Q is the maximum number of qubits in the quantum part of the model.

Special objects are used to transfer data in the quantum membrane and to obtain result of quantum calculation. They are collected in alphabets Inp and $Outp$. See Sec. 3 for details of this interaction.

3 Quantum Membranes for the Hybrid Computational Model

Our model is characterized by the existence of quantum membranes. We describe below their internal construction.

Notations. Our quantum device explores function $y = f(x)$. Suppose that x is an integer, $0 \leq x < 2^N$, that is, the argument of the function takes N qubits.

M is the size of the result: y is an integer, $0 \leq y < 2^M$.

We are provided with the initial data. Suppose that the membrane is entered by a K-bit integer z_0 ($0 \leq z_0 < 2^K$). The quantum device begins to work as z_0 enters the membrane.

Suppose that intermediate data takes R qubits.

Quantum registers. Quantum calculation is to be reversible while function f is not always bijective. Therefore, we need quantum registers both for argument and for result, and we demand that the argument was restored by quantum implementation F of function f. The standard demand is

$$\mathsf{F}\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle\,, \tag{1}$$

where $|x\rangle_N$ is the argument register (the index denotes the size of N qubits), $|y\rangle_M$ is the result, $\oplus$ is the modulo 2 bitwise addition (bitwise **xor**).

Condition (1) implies the reversibility of transformation F. Moreover, F is its own inverse because:

$$\mathsf{FF}\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x) \oplus f(x)\rangle = |x\rangle\,|y\rangle\,. \tag{2}$$

The quantum register $|z\rangle_K$ keeps initial data, and $|w\rangle_R$ keeps ancillary (intermediate) data. Because of inevitable entanglement, we can regard the quantum memory as one register of $N+M+K+R$ qubits.

The linearity and the reversibility of quantum transformations permit to organize the calculation in such a manner that we can ignore ancillary data $|w\rangle$ in our deductions, as we will see below.

Initialization. Before the quantum calculation starts, each qubit is to be set in one of basis states $|0\rangle_1$, or $|1\rangle_1$. The measurement operation is already embedded to be used after calculation, so we can apply it to our qubits.

As we use the measurement before the calculation, the qubits collapse in the basis states. Measurement is irreversible, therefore it is a classical operation.

Now we are to set qubits in the initial states. For example, if we want to set them in the state $|0\rangle$, we are to check the state of each qubit and invert $|1\rangle$. This is not a quantum transformation because it glues two orthogonal vectors together. (In other words: because of linearity, $\alpha|0\rangle + \beta|1\rangle$ would be transformed to $(\alpha+\beta)|0\rangle$ that implies $|\alpha+\beta|^2 = 1$, and $\alpha\beta = 0$. The operation is not linear, or it cannot be applied to non-basis states.)

Usually, all qubits are initially set to $|0\rangle_1$. (Several quantum algorithms use different initial values though.) In our case, we will use non-quantum tools to prepare the state $|x\rangle|y\rangle|z\rangle|w\rangle = |0\rangle_N|0\rangle_M|z_0\rangle_K|0\rangle_R$. Here z_0 is the number that entered the quantum membrane and initiated the process.

As the last step of the initialization, all qubits from register $|x\rangle$ are at once transformed by one-qubit Hadamard transformation

$$\mathsf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{3}$$

As the result, register $|x\rangle$ gets the state

$$|x\rangle = \frac{1}{2^{N/2}} \sum_{0 \le x < 2^N} |x\rangle, \tag{4}$$

that corresponds to the equal probability of all possible values of argument x. This ends the initialization.

Interaction with membrane environment. The quantum membrane needs a binary number to be initialized. P systems are mostly supposed to work over multisets of objects. If this is the case we can use in the membrane part objects $Z_{i,b} \in Inp$, where $0 \leq i < K$ and $b = 0, 1$, where K is the size of the initial data $|z\rangle_K$ for the quantum device. If the quantum membrane is entered by $Z_{i,0}$ ($Z_{i,1}$), then the i-th qubit ot the quantum register $|z_i\rangle$ should be initialized to $|0\rangle$ ($|1\rangle$). We have several possible techniques:

1. We can demand that exactly K objects $Z_{i,b}$ with all $0 \leq i < K$ enter the quantum membrane simultaneously as we do it in this paper.
2. We can use $|0\rangle$ as the default initialization of $|z\rangle$ in the quantum device and require only $Z_{i,1}$ for selected values of i to enter simultaneously the quantum membrane.
3. We can use an additional object *Qtrigger* whose only mission is to start quantum calculation. In this case the simultaneous input of all $Z_{i,b}$ into the quantum membrane becomes not necessary.
4. We should not demand that $Z_{i,0}$ and $Z_{i,1}$ were mutually exclusive. For example, if the quantum membrane was entered by three instances of $Z_{2,0}$ and one instance of $Z_{2,1}$, then the second qubit of $|z\rangle$ is to be initialized as $|z_2\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$.
5. We can introduce even more objects to represent initial values of all qubits in the quantum device. For example, standard initialization of $|x\rangle$ would correspond to the entrance of both $X_{i,0}$ and $X_{i,1}$ for all $0 \leq i < N$, with obvious notations.

The output of quantum result y outside the quantum membrane is performed classically producing objects from *Outp* after the final measurement.

Quantum calculation and result. Applying F, we get into $|y\rangle$ superposition of values of function $f(x)$ at all possible values of the argument:

$$\mathsf{F}\,|x\rangle\,|0\rangle = \frac{1}{2^{N/2}} \sum_{0 \leq x < 2^N} |x\rangle\,|f(x)\rangle\,. \tag{5}$$

This is the quantum parallelism.

If this finishes the calculation, the following measurement reduces qubit states to basis and we get the result $|x_1\rangle\,|f(x_1)\rangle$. Here x_1 is a random integer between 0 and $2^N - 1$.

Nobody would mess with quantum calculations were they to produce only a value of the function at a random point. However, we could not stop after transformation F. Let us use simultaneously available values of function f at all values of its argument, and perform over $|f(x)\rangle$ another transformation G that produces some important information on all these values at once: $\mathsf{G}\,|x\rangle\,|f(x)\rangle = |x\rangle\,|\text{something important}\rangle$.

The quantum programmer should elaborately select unitary linear transformations F and G. They are applied using entanglement and quantum parallelism.

During the quantum calculation we got all values of function f but they are unobservable. We can then stop and get no more than one value $f(x_1)$, or we can continue losing the information on particular values of function f but obtaining some data on its more general properties. This is the uncertainty principle in quantum calculations.

Independence of ancillary qubits. Register $|z\rangle$ of initial data and the ancillary register $|w\rangle$ could be ignored as speaking on result of the calculation. This was done, for example, in equation (5). The necessary conditions are:

- after the calculation the qubits of $|z\rangle$ and $|w\rangle$ were not entangled with qubits of $|x\rangle$ and $|y\rangle$;
- resulting values of $|z\rangle$ and $|w\rangle$ do not depend on the initial values of $|x\rangle$ and $|y\rangle$.

The entanglement of all qubits during the calculation should take place because in the opposite case the unused qubits could be deleted from the construction.

The construction of a quantum computer shown in Fig. 1 guarantees this.

$\mathsf{V}_f^\dagger = \mathsf{V}_f^{-1}$ is the inverse transformation to V_f. C_M are M standard "controlled NOT" gates, where qubits from $|f(x)\rangle_M$ are control qubits.

An additional transformation is introduced as follows: V_f is re-

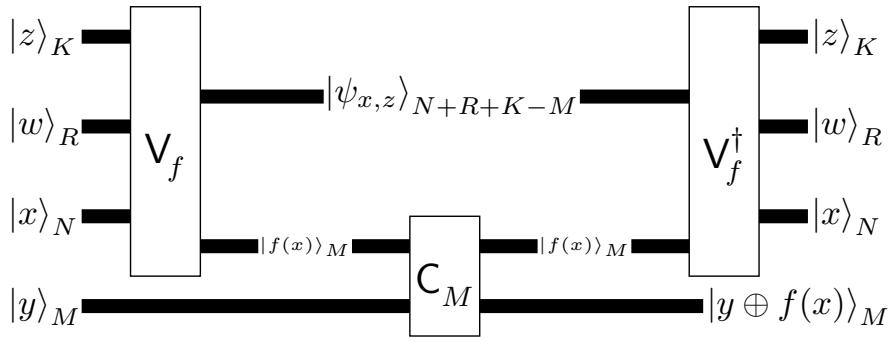


Figure 1. Quantum calculation into a membrane; initialization is not shown

placed by $\mathsf{V}_g\mathsf{V}_f$. The inverse transformations are applied in the reverse order: $\mathsf{V}_f^\dagger\mathsf{V}_g^\dagger$ (Fig. 2).

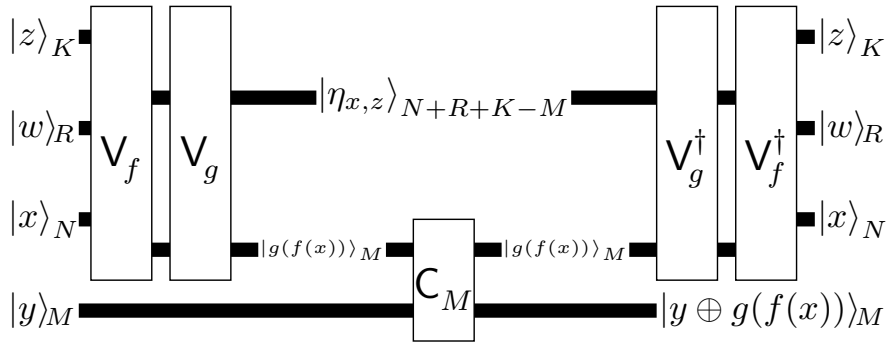


Figure 2. Quantum calculation with an additional transformation

Let us note that, in Fig. 2, transformation V_g can access only entangled qubits $|\psi\rangle$ and $|f(x)\rangle$ while the original $|x\rangle$ and $|z\rangle$ are unavailable. To correct this, we are to apply V_g exactly alike V_f after the termination of all calculations as in Fig. 1, so to speak, "as the second cascade". We need another ancillary register made from M qubits initialized by $|0\rangle$ (Fig. 3; the gray color emphasizes the part corresponding to Fig. 1).

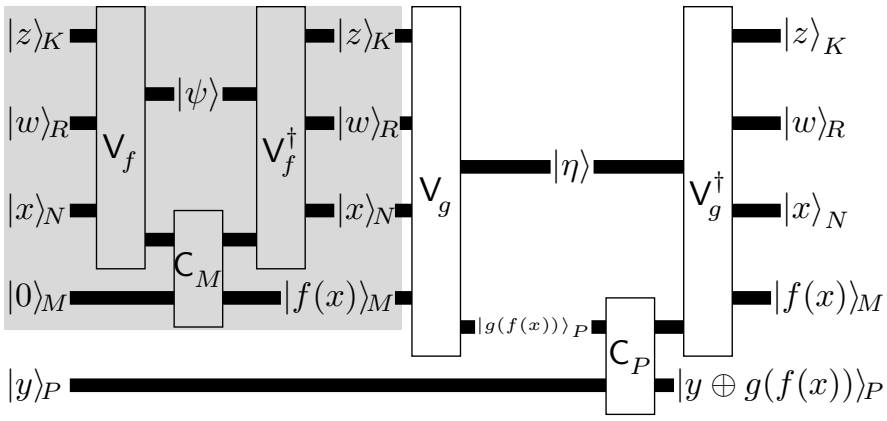


Figure 3. Additional transformation with access to the initial data

The size R of an ancillary register may grow at necessity. The size P of the result should not be equal to M. We get as the result not only $|g(f(x))\rangle$, but $|f(x)\rangle$ either. We can construct a chain of more than two transformations in this manner.

4 Satisfiability

SAT problem. A boolean formula in conjunctive normal form is an expression $\gamma = \bigvee_{1\leq j\leq m} C_j$, where $C_j = \bigwedge_{1\leq l\leq k_j} z_{l,j}$, and $z_{l,j} \in \{x_i, \neg x_i \mid 1 \leq i \leq n\}$, $1 \leq j \leq m$, $1 \leq l \leq k_j$. We assume the input is given by a set of objects from $\Sigma = \{x_{i,j}, \overline{x_{i,j}} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$, each object representing appearance of variable x_i in clause C_j without negation in case of $x_{i,j}$ and with negation in case of $\overline{x_{i,j}}$.

Remark: if x_i does not appear in clause C_j in either form, this fact also needs to be explicitly present as an input to the quantum system we consider. However, it will be the job of membrane subsystem to detect this case and produce the corresponding object.

Membrane system. For the construction we need a bijection l from pairs $(i,j) \mid 1 \leq i \leq n,\ 1 \leq j \leq m$ onto numbers $\{k \mid 1 \leq k \leq mn\}$. We define it by $l(i,j) = (j-1)m+i$. We construct the following

P system (where membrane 2 is a quantum membrane):

$$\begin{aligned}
\Pi &= (O, \Sigma, \mu = [\ [\ ]_2\ ]_1, w_1 = s, R_1, i_0 = 1),\\
O &= \Sigma \cup \{I_{k,b}, I'_{k,b} \mid 1 \le k \le 2mn,\ 0 \le b \le 1\}\\
&\cup \{s, s', \texttt{yes}, \texttt{no}\} \cup \{y_{i,j} \mid 1 \le i \le n,\ 1 \le j \le m\},\\
R_1 &= \{x_{i,j} \to I'_{2l(i,j)-1,1} I'_{2l(i,j),0}\}\\
&\cup \{\overline{x_{i,j}} \to I'_{2l(i,j)-1,1} I'_{2l(i,j),1}\}\\
&\cup \{s \to y_{1,1} \cdots y_{n,m} s'\}\\
&\cup \{y_{i,j} \to I'_{2l(i,j)-1,0} I'_{2l(i,j),0}|_{\neg I'_{2l(i,j)-1,1}}\\
&\quad \mid 1 \le i \le n,\ 1 \le j \le m\} \cup \{s' \to \lambda\}\\
&\cup \{y_{i,j} \to \lambda|_{\neg s'} \mid 1 \le i \le n,\ 1 \le j \le m\}\\
&\cup \{I'_{k,b} \to (I_{k,b}, in_2)|_{\neg s'} \mid 1 \le k \le 2mn,\ 0 \le b \le 1\}.
\end{aligned}$$

The P system above takes three steps to prepare the input for the quantum system. The first step generates the input symbols corresponding to variables appearing in some clauses. The second step generates the input symbols corresponding to variables not appearing in some clauses. The third step erases intermediate objects and sends the input into the quantum membrane. The input z_0 consists of $K = 2mn$ bits, that are grouped in m groups of $2n$ bits each. Each of m groups corresponds to a clause, and each two bytes in these groups correspond to a variable. From these bytes, the first one is 0 if the variable is absent in the clause, and it is 1 if the variable is present in the clause. The second byte is 0 for variable x_i and 1 for its negation $\bar{x}_i$. It is obvious that combination 01 is impossible for these bytes. We will denote these bytes by z_p (**p**resence) and z_g (ne**g**ation).

Quantum system. The quantum calculation is quite straightforward and implements formula:

$$C = C \vee (z_p \wedge (z_g \oplus x))$$

to calculate clause C of the propositional form γ. Here C means any of C_j, x means any of x_i, and z_p and z_g mean the corresponding pairs of bits from z_0. γ is calculated as $\gamma = \gamma \wedge C$ for each clause C.

The necessary quantum circuits are shown in Fig. 4. They use standard cNOT and Toffoli gates, and inversion (**not**). We need ancillary qubits to make **and** and **or** reversible.

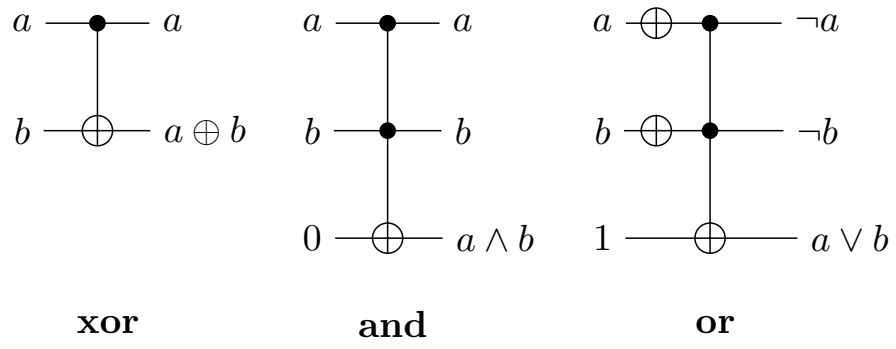


Figure 4. Quantum implementation of Boolean operations

Then Grover's algorithm is applied to search for 1 between possible values of γ. See a detailed step-by-step description in [7].

5 Image Retrieval

Many tasks of medical imaging are affected by image database size. Retrieval of images, similar to a given one, can be considered the most important one among these tasks. During the retrieval, extraction by attributes can be implemented mostly in parallel. The attribute vectors comparison can be also made in parallel for each image as we check the similarity over the whole database. Therefore, images retrieval problem is favorable for massive parallelism provided by unconventional computation, and its binary outputs are suitable for being given by quantum oracle. Basing on this, we chose image retrieval problem as relevant test example for hybrid computational model. It is obvious, however, that in general image retrieval problem is the monstrous task. In current work we only demonstrate how hybrid model can solve the essence subtask of retrieval – estimation of two images similarity.

Definitions. Let Img_{db} is one image from database of grayscale images. Img_u is the pattern image supplied by user. Both images have the size $N_w \times N_h$ pixels. The problem is to learn range of similarity of two images according to given similarity criterion C_sim. Let us assume for test purpose that $C_{sim} = N_{sim}/N_{ucont}$, where N_{sim} is the number of matching contour points, N_{ucont} is the total number of points in the contour of the user image.

Basing on general definition of hybrid model given in Sec. 2, the solution of images retrieval problem is represented by the following hybrid model:

$$\beta = (\Pi, H_Q, N_Q, Inp, Outp, Q_1, \ldots, Q_m),$$

where m is the database size as we will need m identical quantum devices for m database images.

For this task P system based calculation implements algorithm of grayscale image region-based segmentation proposed by the Spanish P system research group [6]. Thus, Π is a tissue-like P system. On account of calculation details do not concern the hybrid model functioning, let us give only brief scheme of algorithm. Each pixel is coded by integer representation of its associated grayscale value and mapped to the corresponding multiset object a_{ij} (b_{ij} – for the second image). Graphical-related basis of algorithm is the edge-based segmentation using the cross-like 4-adjacency.

We need two membranes to perform algorithm for each of two images and one H_Q membrane to proceed to retrieval, one more membrane is added to keep the answer.

$$\Pi(n, m) = (\mu, \Sigma, \varepsilon, w_{1u}, w_{2u}, w_{1db}, w_{2db}, q_1, \mathcal{R}, i_\Pi, o_\Pi),$$

where set of membranes is $\mu = [\,]_1[\,]_2[\,]_3[\,]_4[\,]_5[\,]_{collect}$; input alphabet is $\Sigma = \{a_{ij} : a \in \mathcal{C}, 1 \leq i \leq n;\ 1 \leq j \leq m\} \cup \{b_{ij} : a \in \mathcal{C}, 1 \leq i \leq n;\ 1 \leq j \leq m\}$; environment alphabet is $\varepsilon = \{\bar{a}_{ij} : a \in \mathcal{C}, 1 \leq i \leq n,\ 1 \leq j \leq m, a \in \mathcal{C}\} \cup \{A_{ij} : a \in \mathcal{C},\ 1 \leq i \leq n; 1 \leq j \leq m\, a \in \mathcal{C}\} \cup \{\bar{b}_{ij} : a \in \mathcal{C}, 1 \leq i \leq n,\ 1 \leq j \leq m, b \in \mathcal{C}\} \cup \{B_{ij} : b \in \mathcal{C}, 1 \leq i \leq n;\ 1 \leq j \leq m\, b \in \mathcal{C}\}$; $w_{1u}, w_{2u}, w_{1db}, w_{2db} = \emptyset$; $i_\Pi = 1$; $o_\Pi = collect$.

The only quantum system for one database picture is Q_1 that will be described below.

Now let us present set $\mathcal{R}$ of communication rules.

Firstly, the segmentation is implemented by P system based calculation using subset $\mathcal{R}_{psyst}$ of $\mathcal{R}$. The $\mathcal{R}_{psyst}$ is presented only for one image, because rules are identical, excepting replacement a by b. Communication rules of $\mathcal{R}_{psyst}$ are divided into types according segmentation steps.

Rules of type 1 look like $(1, a_{ij}b_{kl}/\bar{a}_{ij}A_{ij}b_{kl}, 0)$, where $a, b \in \mathcal{C}, 1 \leq i, k \leq n; 1 \leq j, l \leq m\}$. These rules identify the contour pixels by adjacency of different colors and produce the marks of the edge pixels.

After these marks appear, the rules of type 2 start to be applied in parallel with type 1 rules. Rules of type 2 are as follows:
$(1, \bar{a}_{ij}a_{ij+1}\bar{a}_{i+1j+1}b_{i+1j}/\bar{a}_{ij}\bar{a}_{ij+1}A_{ij+1}\bar{a}_{i+1j+1}b_{i+1j}, 0)$ $a, b \in \mathcal{C}, a < b, 1 \leq i \leq n-1, 1 \leq j \leq m-1$
$(1, \bar{a}_{ij}a_{i-1j+1}\bar{a}_{ij+1}b_{ij+1}/\bar{a}_{ij}\bar{a}_{i-1j}A_{i-1j}\bar{a}_{i-1j+1}b_{ij+1}, 0)$ $a, b \in \mathcal{C}, a < b, 1 \leq i \leq n; 1 \leq j \leq m-1$
$(1, \bar{a}_{ij}a_{ij+1}\bar{a}_{i-1j+1}b_{i-1j}/\bar{a}_{ij}\bar{a}_{ij+1}A_{ij+1}\bar{a}_{i-1j+1}b_{i-1j}, 0)$ $a, b \in \mathcal{C}, 1 \leq i, k \leq n; 1 \leq j, l \leq m-1$
$(1, \bar{a}_{ij}a_{i+1j}\bar{a}_{i+1j+1}b_{ij+1}/\bar{a}_{ij}\bar{a}_{i+1j}A_{i+1j}\bar{a}_{i+1j+1}b_{ij+1}, 0)$ $a, b \in \mathcal{C}, 1 \leq i, k \leq n-1; 1 \leq j, l \leq m-1$

These rules mark with the pixels that are adjacent to two pixels of the same color, which were marked by rules of type 1 but with the condition that the marked objects are adjacent to another pixel with a different color. Together with these operations the object representing the final border pixel is brought from the environment.

Finally, the rules of type 3 $(1, A_{ij}/\lambda, 2)$, for $1 \leq i \leq n, 1 \leq j \leq m$ are applied putting all the edge pixels A_{ij} in the output cells.

The only change, made in the algorithm from [6] to adopt it for retrieval, is the absence of final stage in which segmented image is restored from tissue P system. To prepare the enter in membrane of H_Q we only need the points of contours that divide segmented areas. They are stored in the resulting multiset A_{ij} (B_{ij}). Actually we have the set of contour points that are now independent from color. To solve retrieval problem, points B_{ij} obtained from Img_u have to be checked

(in fully parallel mode) only on existence of contour point (points) A_{ij} in correspondent neighborhood for Img_{db}. The question of similarity is mapped to matching of criterion C_{sim} to threshold. This fact makes the problem similar to graph isomorphism one (for case of reduced graph), that has a number of quantum-based solutions in majority applying the adaptation of classical Grover search [7].

To manage H_Q calculation of this problem, we have to apply two extensions of classical Grover search algorithm. Firstly, we need the possibility of starting from an arbitrary state. The convergence of Grover search in this case is proved in [8] and iterations number is $\frac{\pi\sqrt{N}}{4}$. The second extension is a well known one: existence of several solutions. It is proved that Grover search algorithm converges with the same number of iterations even when the number of solutions is unknown, but only if any arbitrary solution is suited [9]. This is our case because presence of any contour point in the given neighborhood is enough for a positive answer.

The next subset $\mathcal{R}_q$ of $\mathcal{R}$ prepares the input for the quantum system. $\mathcal{R}_q = (2, A_{ij}/\lambda,)5, (4, B_{ij}/\lambda,)5$, for $1 \leq i \leq n, 1 \leq j \leq m$. Inp contains $A_{ij} \cup B_{ij}$, $Outp$ is $\{yes, no\}$ i.e. similar or not. The activation of H_Q membrane has to wait until segmentation of both images finishes.

The calculation inside H_Q membrane is implemented by the following way. Both user and database images contours are coded by integers $i, 0 < i < N_{cntr}$ that represent index number of the corresponding pixel in the image left-right and top-down. The $|x\rangle$ and $|y\rangle$ registers of H_Q membranes are activated by these tuples of integer. The search is provided for each integer in $|x\rangle$ having $|y\rangle$ as work register. Starting from description in work [7] where the close task is solved, the following Grover oracle is build:

$$f(x) = \begin{cases} 1, & \text{if } i_{leftnbh} < i < i_{rightnbh}; \\ 0, & \text{otherwise.} \end{cases}$$

In general the algorithm is:

1. Obtain the integer $|x_i\rangle$ from register $|x\rangle$
2. Use Grover Algorithm with oracle described above on the elements of $|y\rangle$ to search the contour point in given neighborhood.

3. **if** Grover search gives the positive answers, **then** $N_{sim}++$
4. **if** $i++ < N_{ucont}$, **then** return to step 1, else
5. the criterion $C_{sim} = N_{sim}/N_{ucont}$ is evaluated, then the answer yes/no is generated and passed into membrane $\mu_{collect}$.

This calculation can be repeated in parallel for all database images, collecting the answers in $\mu_{collect}$.

6 Conclusions

This work introduces our version of computational paradigm that combines both quantum and biological approaches. In the field of unconventional computing, quantum and biological paradigms were developed mostly in parallel but both are considered as the tools for hard tasks solutions. Solutions of some, mostly practical, hard tasks by pure quantum or pure bio-inspired methods could be inefficient. The idea of hybrid model springs from necessity of efficient computational models for such problems. We use P systems as the starting point at the hybrid model development. The proposed computational model applies the classical P system framework in which the quantum-style algorithms are interned.

We concentrated in this paper on demonstration of functioning of the proposed hybrid model. In our following works we will provide more detailed basis as well as the quantitative characteristics of the effectiveness of hybrid calculations. Simulation of the hybrid model will be also discussed. Here and now we just intend to show viability of membrane-quantum hybrid model. For this, we choose two problems of different patterns: the first problem belongs to theoretical computing, while the second is strongly practical one. Both problems show good applicability of the proposed hybrid model.

We presented here the hybrid model where the P system (macro) level is the main frame while quantum (micro) level is represented by membrane with quantum computation. The mutual accepting of input/output by computation models consists, on the current stage, in mapping of P system multisets to quantum basis states. Quantum computation is reduced in these problems to quantum oracle that answers

yes or *no*.

We would provide in our further research mutually-inspired development of formal hybrid model description and practical solutions for more emerging and complicated tasks that can demonstrate the advantages of hybrid model.

Acknowledgment

This work was executed under the project STCU 5384 awarded by the Scientific and Technology Center in Ukraine.

References

[1] Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

[2] C. P. Williams. *Explorations in Quantum Computing.* Springer, 2008.

[3] A. Leporati. *P systems with a quantum-like behavior: Background, definition, and computational power,* in: Lecture Notes in Computer Science, 2007, vol. 4860, pp. 32–53.

[4] A. Alhazov, R. Freund. *Asynchronous and maximally parallel deterministic controlled non-cooperative P systems characterize NFIN and coNFIN,* in: Lecture Notes in Computer Science, E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, and G. Vaszil, Eds., 2013, vol. 7762, pp. 101–111.

[5] D. Sburlan. *Further results on P systems with promoters/inhibitors,* International Journal of Foundations of Computer Science, vol. 17, pp. 205–221, February 2006.

[6] H. A. Christinal, D. Diaz-Pernil, P. Real. *Region-based segmentation of 2D and 3D images with tissue-like P systems,* Pattern Recognition Letters, vol. 32(16), pp. 2206–2212, 2011.

[7] N. Volpato, A. Moura. *A fast quantum algorithm for the closest bichromatic pair problem,* Instituto de Computação, Universidade Estadual de Campinas, Tech. Rep. IC-10-03, January 2010. [Online]. Available: `http://www.ic.unicamp.br/~reltech/2010/10-03.pdf`

[8] D. Kenigsberg. (2001) *Grover's quantum search algorithm and mixed states.* Computer Science Department, The Technion – Israel Institute of Technology. [Online]. Available: `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi/2001/MSC/MSC-2001-01`

[9] E. Rieffel, W. Polak. *Quantum Computing – a Gentle Introduction.* Massachusetts Institute of Technology, 2011.

Yurii Rogozhin, Artiom Alhazov,
Lyudmila Burtseva, Svetlana Cojocaru,
Alexandru Colesnicov, Ludmila Malahov

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova

E–mails
Artiom Alhazov: `artiom@math.md`
Lyudmila Burtseva: `luburtseva@gmail.com`
Svetlana Cojocaru: `Svetlana.Cojocaru@math.md`
Alexandru Colesnicov: `acolesnicov@gmx.com`
Ludmila Malahov: `lmalahov@gmail.com`

Received March 24, 2014

Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Chromatic Polynomials Of Some $(m, l)-$Hyperwheels

Julian A. Allagan

Abstract

In this paper, using a standard method of computing the chromatic polynomial of hypergraphs, we introduce a new reduction theorem which allows us to find explicit formulae for the chromatic polynomials of some (complete) non-uniform $(m, l)-$ hyperwheels and non-uniform $(m, l)-$hyperfans. These hypergraphs, constructed through a "join" graph operation, are some generalizations of the well-known wheel and fan graphs, respectively. Further, we revisit some results concerning these graphs and present their chromatic polynomials in a standard form that involves the Stirling numbers of the second kind.

Keywords: chromatic polynomial, hyperfan, hyperwheel, Stirling numbers.

1 Basic definitions and notations

For basic definitions of graphs and hypergraphs we refer the reader to [1, 4, 10, 17, 20]. A *hypergraph* $\mathcal{H}$ of order n is an ordered pair $\mathcal{H} = (X, \mathcal{E})$, where $|X| = n$ is a finite nonempty set of vertices and $\mathcal{E}$ is a collection of not necessarily distinct non empty subsets of X called *(hyper)edges*. In this paper, all hypergraphs discussed are considered *simple* and *Sperner*, i.e., they have distinct hyperedges and no hyperedge is a subset of another.

A hypergraph $\mathcal{H}$ is $r-$*uniform*, if $|e| = r$ for each $e \in \mathcal{E}$; otherwise, $\mathcal{H}$ is said to be *non-uniform*. In the case when $r = 2$, the resulting hypergraph is called a *graph* which is often defined by $H = (V, E)$. A

hypergraph is said to be *linear* if each pair of hyperedges has at most one vertex in common. The *degree* of a vertex v, denoted by $d(v)$, is the number of hyperedges that contain v. Hypergraphs in this paper are assumed to be connected and linear unless stated otherwise.

Given a hypergraph $\mathcal{H} = (X, \mathcal{E})$, we define the *deletion* of e by $\mathcal{H} - e$, which is the hypergraph obtained from $\mathcal{H}$ by deleting some hyperedge $e \in \mathcal{E}$. The *contraction* of e defined by $\mathcal{H}.e$, is the hypergraph obtained from $\mathcal{H}$ by identifying all the vertices in e by a single vertex and removing e from $\mathcal{E}$ (clearing).

A hyperedge $e_1 \in \mathcal{E}$ is called a *hyperleaf* if there exists $e_2 \in \mathcal{E} - e_1$ such that $e \cap e_2 \subseteq e_1 \cap e_2$ for every $e \in \mathcal{E} - e_1$. In the case of linear hypergraphs, a hyperleaf is simply a hyperedge with exactly one vertex of degree greater than 1. If $P^l := v_1, e_1, v_2, e_2, \ldots, v_l, e_l, v_{l+1}$ denotes an alternating sequence of distinct hyperedges e_i and distinct intersecting vertices v_i, then $\mathcal{P}^l$ is called an $l-$*hyperpath*, for all $l \geq 1$. In the event $v_1 = v_{l+1}$ for all $l \geq 2$, the resulting hypergraph is called an $l-$*hypercycle* which we denote by $\mathcal{C}^l$. It causes no confusion to say that $\mathcal{C}^l$ is induced by the sequence of hyperedges $(e_1, e_2, \ldots, e_l)$, for all $l \geq 2$. We note that the term *elementary* hypercycle has also been used by Tomescu [15] to describe $l-$hypercycle and yet, for simplicity, we choose to use the former term. Moreover, we point out that a $2-$hypercycle induced by (e_1, e_2) when $2 < |e_1| \leq |e_2|$, is not linear, and in fact, is a $2-$hyperpath with $|e_1 \cap e_2| = 2$. For this paper, we do not make such a distinction in name, since it does not affect the results.

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hypergraphs. The *join* of $\mathcal{H}_1$ and $\mathcal{H}_2$, denoted by $\mathcal{H}_1 \vee \mathcal{H}_2$, is the hypergraph $\mathcal{H}$ whose vertex set is $X(\mathcal{H}) = X(\mathcal{H}_1) \cup X(\mathcal{H}_2)$, a disjoint union, and whose hyperedge set is $\mathcal{E}(\mathcal{H}) = \mathcal{E}(\mathcal{H}_1) \cup \mathcal{E}(\mathcal{H}_2) \cup \{x_1x_2 \mid x_1 \in X(\mathcal{H}_1),\ x_2 \in X(\mathcal{H}_2)\}$. For example, $\overline{K}_{n_1} \vee \overline{K}_{n_2} \vee \ldots \vee \overline{K}_{n_k} = K(n_1, n_2, \ldots, n_k)$ is a complete $k-$partite graph with part sizes $n_1, \ldots, n_k$. We denote a *wheel* graph by $W^l = C^l \vee v$, where C^l is a cycle on $l = n$ vertices. C^l is the *rim* of the wheel and the edges not in the rim are called *spokes*. We will call a wheel on l rim edges (or on $n + 1$ vertices), an $l-$wheel, for short. For instance, when $l = 2$, a $2-$wheel graph is a cycle $C^3 \simeq K_3$; for this reason, it is customary to define a $3-$wheel instead. Although a wheel and a cycle

are both traditionally defined on n vertices (see [20] for instance), we think it causes no confusion to substitute (where it is convenient) the number of vertices n for l, the number of edges. Further, the notation of W^l and C^l (instead of W_n and C_n, respectively) will be particularly important for us when handling hypergraphs. In each of the formula presented in this paper, one can easily replace l with the appropriate number of vertices by a simple substitution. We also denote the falling factorial $\lambda^{\underline{t}} = \lambda(\lambda-1)(\lambda-2)\dots(\lambda-t+1)$ with $\lambda^{\underline{0}} = 1$. Further, the Stirling number of the second kind is denoted by $\left\{ {n \atop k} \right\}$; it counts the number of partitions of a set of n elements into k nonempty subsets. Clearly $\left\{ {n \atop 0} \right\} = \left\{ {0 \atop n} \right\} = 0$ and $\left\{ {n \atop 1} \right\} = \left\{ {n \atop n} \right\} = \left\{ {0 \atop 0} \right\} = 1$. These notations and other combinatorial identities can be found in [12].

2 Chromatic polynomial of some graphs

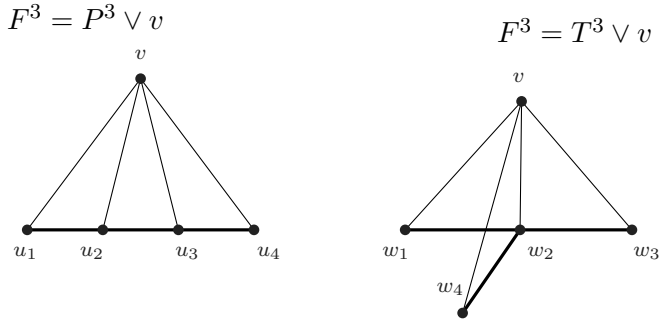
The notion of coloring the vertices of a graph has been widely studied [10, 20]. A given graph G on n vertices can be properly colored in many different ways using a sufficiently large number of colors. This property of a graph is expressed elegantly by means of a polynomial. This polynomial is called the *chromatic polynomial* of G. It is well-known that Birkhoff [5] first introduced this polynomial in 1912 in an attempt to prove the four color theorem. The value of the chromatic polynomial $P(G, \lambda) = P(G)$ of a graph with n vertices gives the number of ways to properly color the graph G, using λ or fewer colors. For instance, the chromatic polynomials of a complete graph on n vertices, a tree, and a cycle with l edges are respectively given by $P(K_n) = \lambda^{\underline{n}}$, $P(T^l) = \lambda(\lambda-1)^l$ and $P(C^l) = (\lambda-1)^l + (-1)^l(\lambda-1)$.

The following theorem of Whitney [22], which gives the chromatic polynomial of a graph in terms of "broken circuits", is often used as a standard form; this form explicitly gives a basic property of the chromatic polynomial, namely, the powers of the chromatic polynomial are consecutive and their coefficients alternate in sign.

Theorem 2.1. *(Whitney's "Broken Circuits" Theorem). Let* $P(G, \lambda) = \lambda^n - a_1\lambda^{n-1} + a_2\lambda^{n-2} - \dots + (-1)^{n-1}a_{n-1}\lambda$. *The coefficient* a_i *is equal*

to the number of $i-$subsets of edges of the graph G which contain no broken cycles, for each $i = 1, 2, \ldots, n-1$.

Although any chromatic polynomial can be written in this form, it is shown in [13] that chromatic polynomials written in terms of the Stirling numbers of the second kind have many applications. Still, we can always rewrite our results given in terms of Stirling numbers into a more standard basis, using the combinatorial identity (see [12] for instance) that $\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n$.



(a) Gem graph

(b) $\Theta(1, 2, 2, 2)$ graph

Figure1. Two non-isomorphic fan graphs

Theorem 2.2. *Suppose $\mathcal{H}$ is a (hyper)graph. Then* $P(\mathcal{H} \vee \overline{K}_n, \lambda) = \sum_{i=0}^{n} \left\{ {\lambda \atop i} \right\} \lambda^{\underline{i}} P(\mathcal{H}, \lambda - i)$.

Proof. Recall that $\left\{ {\lambda \atop i} \right\}$ counts the number of ways of partioning n vertices into i distinct classes of colors. Now, there are $\lambda^{\underline{i}}$ ways to color the vertices of each class. Since a color used in a class cannot occur on any vertex in $\mathcal{H}$, the result follows. □

Using a similar argument, we derive an alternative version of the previous theorem as:

Theorem 2.3. *(Alternative Version) Suppose $\mathcal{H}^l$ is a hypergraph with l hyperedges. Then $P(\mathcal{H}^l \vee K_n, \lambda) = \lambda^{\underline{n}} P(\mathcal{H}^l, \lambda - n)$.*

Corollary 2.3.1. *Let $G = \mathcal{T}^l \vee K_n$. The chromatic polynomial of G is given by $P(G) = \lambda^{\underline{n}}(\lambda - n)(\lambda - n - 1)^l$.*

It is easy to verify that when $l = 1$ and $n = 2$, $P(G) = \lambda^{\underline{2}}(\lambda - 2)(\lambda - 3) = \lambda^{\underline{4}} = P(K_4)$.

Corollary 2.3.2. *The chromatic polynomial of a complete $k-$partite graph $G = K(n, 1 \ldots, 1)$ is $P(G) = \sum_{i=0}^{n} \left\{ {\lambda \atop i} \right\} \lambda^{\underline{i}}(\lambda - i)^{\underline{k-1}}$.*

Proof. Clearly $G \simeq \overline{K}_n \vee K_{k-1}$ and $P(K_{k-1}, \lambda) = \lambda^{\underline{k-1}}$. The result follows from Theorem 2.2. □

Observe that if we choose to count the proper colorings of the vertices of K_{k-1} first, we can easily establish an equivalent formula that $P(G) = \lambda^{\underline{k-1}}(\lambda - k + 1)^n$, a formula that is also supported by Theorem 2.3.

Corollary 2.3.3. *Suppose $G = K(n_1, n_2)$ is a complete bipartite graph. Then $P(G) = \sum_{i=0}^{n_1} \left\{ {\lambda \atop i} \right\} \lambda^{\underline{i}}(\lambda - i)^{n_2}$.*

Proof. This result also follows from Theorem 2.2, since $G \simeq \overline{K}_{n_1} \vee \overline{K}_{n_2}$. □

We define an $(m,l)-$*wheel* and $(m,l)-$*fan* graphs respectively by $W^{m,l} \simeq \overline{K}_m \vee C^l$ and $F^{m,l} \simeq \overline{K}_m \vee P^l$, where P^l is an $l-$path, C^l an $l-$cycle, and $\overline{K}_m$ is an empty graph on m vertices. We note that though P^l may not be isomorphic to T^l for $l \geq 3$, their chromatic polynomials are the same for all l. For this reason, it even makes sense to define an $(m,l)-$*fan* graph by $F^{m,l} \simeq \overline{K}_m \vee T^l$, where T^l is an $l-$tree. When $m = 1$, a $(1,l)-$*wheel* graph is simply the usual *wheel* graph and a

$(1,l)-$*fan* graph is simply called a *fan* graph (also known as a $2-$*tree* graph in [20]); in which case, we let $W^{1,l} = W^l$ and $F^{1,l} = F^l$. Thus, $(m,l)-$wheel and $(m,l)-$fan graphs are some generalizations of the ordinary wheel and fan graphs (Figure 1), for $m \geq 1$. In section 4, we provide further generalizations of these graphs to hypergraphs.

Corollary 2.3.4. *The chromatic polynomial of an* $(m,l)-$*wheel graph is given by* $P(W^{m,l}) = \sum_{i=0}^{m} \left\{ {\lambda \atop i} \right\} \Big((\lambda - i - 1)^l + (-1)^l (\lambda - i - 1) \Big)$.

Corollary 2.3.5. *The chromatic polynomial of an* $(m,l)-$*fan graph is given by* $P(F^{m,l}) = \sum_{i=0}^{m} \left\{ {\lambda \atop i} \right\} \lambda^{\underline{i}} (\lambda - i)(\lambda - i - 1)^l$.

Corollary 2.3.6. *The chromatic polynomial of an* $l-$*wheel graph is given by* $P(W^l) = \lambda((\lambda - 2)^l + (-1)^l(\lambda - 2))$.

Corollary 2.3.7. *The chromatic polynomial of a fan graph is* $P(F^l) = \lambda(\lambda - 1)(\lambda - 2)^l$.

3 Chromatic polynomial of some hypergraphs

In 1966 P. Erdös and A. Hajnal extended the notion of *proper coloring* of a graph to the coloring of a hypergraph [11]. Thus, the *chromatic polynomial* of a hypergraph $\mathcal{H}$, first denoted in [16] by $P(\mathcal{H}, \lambda) = P(\mathcal{H})$, is the function that counts the number of *proper* $\lambda-$*colorings*, which are mappings, $f\colon X \to \{1, 2, \ldots, \lambda\}$ with the condition that every hyperedge has at least two vertices with distinct colors. We encourage the reader to refer to [1, 6, 13, 14, 17] for detailed information about chromatic polynomials, research, and applications of hypergraph colorings.

This next theorem will be instrumental to streamline the arguments we make in several upcoming proofs for the remaining of this paper. Similar versions of this theorem can be found in [6, 19]. However, we think this particular version (Reduction Theorem) is unknown as it generalizes the Fundamental Reduction Theorem for graphs found in [10, 20] for instance.

Theorem 3.1. *(Reduction Theorem) Suppose $\mathcal{H}$ is a hypergraph with $l \geq 2$ hyperedges. Then $P(\mathcal{H}) = \lambda^{|e|-2}P(\mathcal{H}-e) - P(\mathcal{H}.e)$, where e is a hyperedge with exactly two vertices of degree 2 or greater.*

Proof. Note that if $|e| = 2$, then the relation satisfies the reduction for graphs.

Let u_1 and u_2 be the 2 vertices of degree 2 in e. In any proper coloring of the hyperedge e using λ colors, the following is true:

Either (i) u_1 and u_2 have the same color, or (ii) u_1 and u_2 have different colors. We therefore count the number of such colorings for each case in turn.

Case (i) There are $\lambda^{|e|-2}-1$ ways to color the vertices in $e\backslash\{u_1, u_2\}$ so that not all receive the same color, and there are $P(\mathcal{H}.e)$ ways to color the remaining vertices so that $f(u_1) = f(u_2)$ (to see this, delete e and identify u_1 and u_2). Hence, there are $(\lambda^{|e|-2}-1)P(\mathcal{H}.e)$ colorings.

Case (ii) There are $\lambda^{|e|-2}$ colorings of the vertices in $e\backslash\{u_1, u_2\}$. For each such coloring, the number of colorings of the remaining vertices is $P(\mathcal{H}-e) - P(\mathcal{H}.e)$, since the first term counts the number of colorings where u_1 and u_2 may have the same or different colors, and the second term counts the number of colors where u_1 and u_2 may have the same color. So there are $\lambda^{|e|-2}(P(\mathcal{H}-e) - P(\mathcal{H}.e))$ colorings altogether.

By combining (i) and (ii), we obtain the result for all $|e| \geq 2$.

□

Dohmen [8] extended Whitney's "Broken Circuits" Theorem to hypergraphs with the next proposition. It denotes by $n(\mathcal{H})$, the number of vertices of $\mathcal{H}$, $m(\mathcal{H})$, the number of hyperedges of $\mathcal{H}$ and by $c(\mathcal{H})$, the number of connected components of $\mathcal{H}$.

Proposition 3.1. *Let $\mathcal{H}$ be a hypergraph. Then*

$$P(\mathcal{H}) = \sum_{S\subseteq\mathcal{H}} (-1)^{m(S)}\lambda^{n(\mathcal{H})-n(S)+c(S)}. \tag{1}$$

Later, Tomescu [15] also presented a similar result using an inclusion-exclusion principle argument in

Lemma 3.1. *Let $\mathcal{H} = (X, \mathcal{E})$ be a connected hypergraph with $|X| = n$. Denote by $N(i, j)$ the number of spanning subhypergraphs of $\mathcal{H}$ with n vertices, i components, and j hyperedges. Then*

$$P(\mathcal{H}) = \sum_{i=1}^{n} a_i \lambda^i, \tag{2}$$

where $a_i = \sum_{j \geq 0} (-1)^j N(i, j)$.

We now present some known results (see [1, 6, 19] for instance), although in different forms. Our results are more in line with the standard form (2) as they can have some added benefits for further analysis, namely, finding generating functions. In addition, these results will not only help to illustrate the Reduction Theorem but also later serve the purpose of comparing the effect of a certain "join" operation on the chromatic polynomial of some hypergraphs.

The next theorem was first presented by Walter [19] as a generalization of Dohmen's result for $r-$uniform hypertree [9]. Though the proof can be obtained by a recursion using the Reduction Theorem, it is quite simple by an induction on $l \geq 1$.

Theorem 3.2. *If $\mathcal{T}^l = (X, \mathcal{E})$ is an $l-$hypertree, then $P(\mathcal{T}^l) = \lambda \prod_{i=1}^{l} (\lambda^{|e_i|-1} - 1)$, for all $l \geq 1$.*

Proof. When $l = 1$, it is clear that there are $\lambda^{|e|} - \lambda = \lambda(\lambda^{|e|-1} - 1)$ ways to color the vertices of a hyperedge so that not all of them have the same color. So, we assume $l \geq 2$. Let e_l be a hyperleaf (there is at least one since $\mathcal{T}^l$ is acyclic). Then, for each proper coloring of $\mathcal{T}^l - e_l$, there are exactly $\lambda^{|e_l|-1} - 1$ ways to properly color the remaining (pendant) vertices of e_l, giving $(\lambda^{|e_l|-1} - 1)P(\mathcal{T}^l - e_l)$ proper colorings. Since $P(\mathcal{T}^l - e_l) = P(\mathcal{T}^{l-1})$, the result follows from the inductive hypothesis. □

Theorem 3.3. *The chromatic polynomial of an $l-$hypercycle is given by*

$$P(\mathcal{C}^l) = \sum_{i=0}^{l-1}(-1)^i\lambda^{|e_{l-i}|-2}P(\mathcal{T}^{l-i-1}),\ \textit{with}\ P(\mathcal{T}^0) = \lambda(1-\lambda^{2-|e_1|}).$$

Proof. When $l = 2$, we apply the Reduction Theorem on e_2 to obtain that $P(\mathcal{C}^2) = \lambda^{|e_2|-2}P(\mathcal{T}^1) - P(\mathcal{T}^1_*)$, where $\mathcal{T}^1_*$ is a (loop) hyperedge on $|e_1| - 1$ vertices which chromatic polynomial is $\lambda^{|e_1|-1} - \lambda = \lambda^{|e_1|-2}P(\mathcal{T}^0)$.

Further, when $l = 3$, we have that $P(\mathcal{C}^3) = \lambda^{|e_3|-2}P(\mathcal{T}^2) - P(\mathcal{C}^2) = \lambda^{|e_3|-2}P(\mathcal{T}^2) - \lambda^{|e_2|-2}P(\mathcal{T}^1) + P(\mathcal{T}^1_*)$. Using the previous result and a simple recursion, we establish the formula for all $l \geq 2$, with $P(\mathcal{T}^0) = \lambda^{2-|e_1|}P(\mathcal{T}^1_*)$. □

Corollary 3.3.1. *The chromatic polynomial of an $l-$hypercycle is given by*

$$P(\mathcal{C}^l) = \lambda\Big(\sum_{i=0}^{l-2}(-1)^i\lambda^{|e_{l-i}|-2}\prod_{j=1}^{l-i-1}(\lambda^{|e_j|-1}-1) + (-1)^{l-1}(\lambda^{|e_1|-2}-1)\Big),\ \textit{for all}\ l \geq 2.$$

Proof. This follows directly from Theorems 3.2 and 3.3. □

The following result follows when $|e| = r$, for each $e \in \mathcal{E}$.

Corollary 3.3.2. *The chromatic polynomial of any $r-$uniform $l-$hypercycle is given by*

$$P(\mathcal{C}^l_r) = \lambda\Big(\sum_{i=0}^{l-2}(-1)^i\lambda^{r-2}(\lambda^{r-1}-1)^{l-i-1} + (-1)^{l-1}(\lambda^{r-2}-1)\Big),\ \textit{for all}\ l \geq 2,\ r \geq 2.$$

The case when $r = 2$ follows as

Corollary 3.3.3. *The chromatic polynomial of any $l-$cycle is given by*

$$P(C^l) = \lambda\sum_{i=0}^{l-2}(-1)^i(\lambda-1)^{l-i-1},\ \textit{for all}\ l \geq 2.$$

We observe from this last result that we established the following: $P(C^l) = (\lambda-1)^l + (-1)^l(\lambda-1) = \lambda\sum_{i=0}^{l-2}(-1)^i(\lambda-1)^{l-i-1}$, for all $l \geq 2$.

We now present some results on some new families of hypergraphs.

4 Chromatic polynomial of some $(m,l)-$hyperwheels

Suppose $\mathcal{C}^l = (X, \mathcal{E})$ is an $l-$hypercycle induced by the set of hyperedges $(e_1, \ldots, e_l)$ and let $\overline{K}_m \simeq \{v_1, \ldots, v_m\}$ be the empty graph on m vertices. We shall call $G_1 = \overline{K}_m \vee \mathcal{C}^l$, a *complete* $(m,l)-$*hyperwheel* and $G_2 = \overline{K}_m \vee \mathcal{P}^l$, a *complete* $(m,l)-$*hyperfan*, where $\mathcal{P}^l = \mathcal{C}^{l+1} - e$ for some hyperedge e. We call G_1 and G_2 "complete" in the sense of the "join" operation. For this reason, their chromatic polynomials are easily obtained from Theorems 2.2, 3.2 and Corollary 3.3.1; these findings are presented in the next two corollaries. Further, we found that removing some of the edges of G_1 (and G_2) yields more interesting families that are less "complete". These families are better generalizations of their graphs counterparts, namely the wheel and the fan graphs. They will be called $(m,l)-$hyperwheels and $(m,l)-$hyperfans. The chromatic polynomials of $(m,l)-$hyperwheels and $(m,l)-$hyperfans remain open for all $m \geq 2$. However, we present the chromatic polynomials of the particular case when $m = 1$, that we call $l-$hyperwheels ($l \geq 2$) and $l-$hyperfans ($l \geq 1$).

Corollary 4.0.4. *The chromatic polynomial of a complete* $(m,l)-$*hyperwheel is given by*

$$P(G_1,\lambda) = \sum_{k=0}^{m} \left\{ {\lambda \atop k} \right\} \lambda^{\underline{k}}(\lambda-k)\left(\sum_{i=0}^{l-2}(-1)^i(\lambda-k)^{|e_{l-i}|-2} \prod_{j=1}^{l-i-1} \Big((\lambda-k)^{|e_j|-1} - 1 \Big) + (-1)^{l-1}\Big((\lambda-k)^{|e_l|-2} - 1 \Big) \right),$$

for all $m \geq 1$, $l \geq 2$.

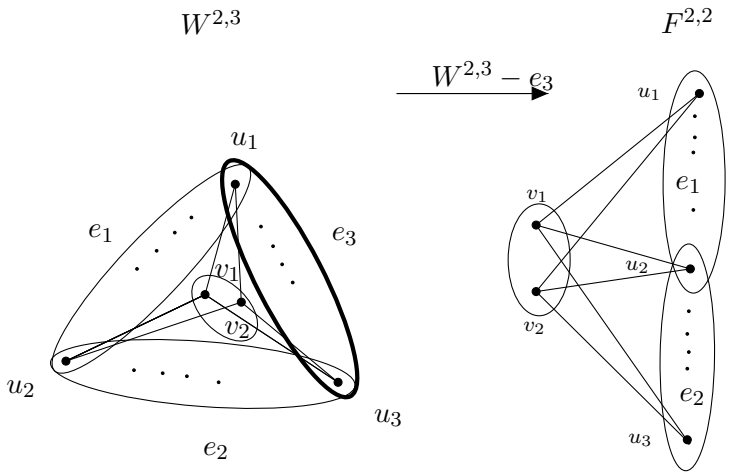


Figure 2. A $(2,3)-$hyperwheel and a $(2,2)-$hyperfan

Corollary 4.0.5. *The chromatic polynomial of a complete $(m,l)-$hyperfan is given by* $P(G_2,\lambda)=\sum_{k=0}^{m}\left\{\begin{matrix}\lambda\\k\end{matrix}\right\}\lambda^{\underline{k}}(\lambda-k)\prod_{j=1}^{l}\Big((\lambda-k)^{|e_j|-1}-1\Big)$, *for all $m\geq 1$, $l\geq 2$.*

Suppose $\mathcal{C}^l=(X,\mathcal{E})$ is an $l-$hypercycle induced by the set of hyperedges $(e_1,\ldots,e_l)$ and let $\overline{K}_m\simeq\{v_1,\ldots,v_m\}$ be the empty graph on m vertices. We define an $(m,l)-$*hyperwheel* by $\mathcal{W}^{m,l}\simeq(\overline{K}_m\vee\mathcal{C}^l)-\{uv|deg(u)=1\}$ for each $u\in X$ and $v\in V(\overline{K}_m)$. Each edge $\{u,v\}$ is referred to as a *spoke* and its endpoints u and v are called *rim* and *apex* vertices respectively. The hyperedges e_i are referred to as *rim* hyperedges as well. Figure 2 contains an example of a $(2,3)-$hyperwheel with rim hyperedges of size $|e_i|$, for $i=1,2,3$.

An $(m,l)-$*hyperfan* is defined by $\mathcal{F}^{m,l}=\mathcal{W}^{m,l+1}-e$, where $\mathcal{W}^{m,l+1}$ is an $(m,l+1)-$hyperwheel and e is a rim hyperedge. In the case when $m=1$, we write $\mathcal{W}^l$ (and $\mathcal{F}^l$) and call it an $l-$*hyperwheel* (and an

$l-$*hyperfan*). Figure 2 contains a representation of a $(2,2)-$hyperfan with rim hyperedges of size $|e_i|$, for $i = 1, 2$.

When $|e| = r$ for each rim hyperedge e, we denote an $(m,l)-$*hyperwheel* and an $(m,l)-$*hyperfan* respectively by $\mathcal{W}_r^{m,l}$ and $\mathcal{F}_r^{m,l}$. For instance $\mathcal{W}_2^l = W^l$, an $l-$wheel ($l \geq 2$) and $\mathcal{F}_2^l = F^l$, an $l-$fan ($l \geq 1$). A $1-$hyperfan (with 2 spokes) is a $3-$hypercycle (with one hyperedge and 2 edges).

Theorem 4.1. *The chromatic polynomial of an $l-$hyperfan is given by*

$$P(\mathcal{F}^l) = \lambda(\lambda-1)\prod_{i=1}^{l}(\lambda^{|e_i|-1} - \lambda^{|e_i|-2} - 1), \text{ for all } l \geq 1.$$

Proof. We proceed by induction on l, which is the number of rim hyperedges.

Note that when $l = 1$, $\mathcal{F}^1 = \mathcal{C}_*^3$, which is a $3-$hypercycle with a single hyperedge that we denote by e_1. From Corollary 3.3.1, when $l = 3$, we have that $P(\mathcal{C}_*^3) = (\lambda-1)^2(\lambda^{|e_1|-1} - 1) + (-1)^3(\lambda-1) = \lambda(\lambda-1)(\lambda^{|e_1|-1} - \lambda^{|e_1|-2} - 1)$.

For $l \geq 2$, let $e_1, \ldots, e_l$ be the rim hyperedges. Let u_l and u_{l+1} be the two vertices of e_l that are incident to v, the apex vertex. We apply the Reduction Theorem on e_l to get that $P(\mathcal{F}^l) = \lambda^{|e_l|-2}P(\mathcal{F}^l - e_l) - P(\mathcal{F}^l.e_l)$. Now, $P(\mathcal{F}^l - e_l) = P(\mathcal{F}^{l-1} \cup \{vu_{l+1}\}) = (\lambda-1)P(\mathcal{F}^{l-1})$ and $P(\mathcal{F}^l.e_l) = P(\mathcal{F}^{l-1})$. Thus, we obtain the relation that

$P(\mathcal{F}^l) = \lambda^{|e_l|-2}(\lambda-1)P(\mathcal{F}^{l-1}) - P(\mathcal{F}^{l-1}) = P(\mathcal{F}^{l-1})(\lambda^{|e_l|-1} - \lambda^{|e_l|-2} - 1)$.

Further, using $P(\mathcal{F}^1)$ as the basis of the recursion, we have that $P(\mathcal{F}^l) = P(\mathcal{F}^1)\prod_{i=2}^{l}(\lambda^{|e_i|-1} - \lambda^{|e_i|-2} - 1)$.

The result follows, since $P(\mathcal{F}^1) = P(\mathcal{C}_*^3) = \lambda(\lambda-1)(\lambda^{|e_1|-1} - \lambda^{|e_1|-2} - 1)$.

□

The following corollary is derived when each rim hyperedge of a hyperfan is of size $r \geq 2$.

Corollary 4.1.1. *The chromatic polynomial of an $l-$hyperfan with $r-$uniform rim hyperedges is $P(\mathcal{F}_r^l) = \lambda(\lambda-1)(\lambda^{r-1}-\lambda^{r-2}-1)^l$, $l \geq 1$, $r \geq 2$.*

We note that when $r = 2$, the previous formula coincides with that of Corollary 2.3.7.

Theorem 4.2. *Suppose $\mathcal{W}^l$ is a hyperwheel with rim hyperedges $e_1, e_2, \ldots, e_l$, $l \geq 2$. Then $P(\mathcal{W}^l) = \sum_{i=0}^{l-1}(-1)^i \lambda^{|e_{l-i}|-2} P(\mathcal{F}^{l-i-1})$ with $P(\mathcal{F}^0) = \lambda(\lambda-1)(1-\lambda^{2-|e_1|})$.*

Proof. Apply the Reduction Theorem on e_l, for $l \geq 2$, to obtain that $P(\mathcal{W}^l) = \lambda^{|e_l|-2}P(\mathcal{F}^{l-1}) - P(\mathcal{W}^{l-1})$. From this relation, when $l = 2$, we have that $P(\mathcal{W}^2) = \lambda^{|e_2|-2}P(\mathcal{F}^1) - P(\mathcal{W}^1)$, where $\mathcal{W}^1$ is a $2-$hyperpath with one hyperedge e_1 and one edge. Thus, $P(\mathcal{W}^1) = (\lambda-1)(\lambda^{|e_1|-1} - \lambda)$. Similarly, the result follows by a recursion with $P(\mathcal{F}^0) = \lambda^{2-|e_1|}(\lambda-1)(\lambda^{|e_1|-1}-\lambda) = \lambda(\lambda-1)(1-\lambda^{2-|e_1|})$. □

Corollary 4.2.1. *The chromatic polynomial of an $l-$hyperwheel is given by*

$$P(\mathcal{W}^l) = \lambda(\lambda-1)\Bigg(\sum_{i=0}^{l-2}(-1)^i\lambda^{|e_{l-i}|-2}\prod_{j=1}^{l-i-1}\Big(\lambda^{|e_j|-1}-\lambda^{|e_j|-2}-1\Big) + (-1)^{l-1}(\lambda^{|e_1|-2}-1)\Bigg),$$

for all $l \geq 2$.

The following result follows when $|e| = r$, for each rim hyperedge e.

Corollary 4.2.2. *The chromatic polynomial of any $l-$hyperwheel with $r-$uniform rim hyperedges is given by*

$$P(\mathcal{W}_r^l) =$$

$$= \lambda(\lambda-1)\Big(\sum_{i=0}^{l-2}(-1)^i\lambda^{r-2}(\lambda^{r-1}-\lambda^{r-2}-1)^{l-i-1} + (-1)^{l-1}(\lambda^{r-2}-1)\Big),$$

for all $l \geq 2$, $r \geq 2$.

The case when $r = 2$ follows as

Corollary 4.2.3. *The chromatic polynomial of any $l-$wheel is given by*

$$P(W^l) = \lambda(\lambda-1)\sum_{i=0}^{l-2}(-1)^i(\lambda-2)^{l-i-1}, \text{ for all } l \geq 2.$$

We observe that from Corollary 2.3.6 and Corollary 4.2.3, together we have

$$P(W^l) = \lambda((\lambda-2)^l + (-1)^l(\lambda-2)) = \lambda(\lambda-1)\sum_{i=0}^{l-2}(-1)^i(\lambda-2)^{l-i-1},$$

for all $l \geq 2$.

5 Conclusion

The purpose of this paper, as it has been for related research on the topic of chromatic polynomials, is to derive the formulae for a number of graphs and hypergraphs with the goal to further classify them. We, once again, discovered that the process of finding these formulae is of a great computational complexity. Further work is needed to determine the efficiency of our proposed Reduction Theorem. Nonetheless, we derived some nice recursive relationships which we hope can serve the purpose of further analysis such as the finding of the roots and the meaning of the coefficients of these polynomials. Our focus has been primarily on graphs and hypergraphs which are obtained as a result of a "join" graph operation. This process and the resulting new class of hypergraphs discussed in this paper are certainly worth extending to other known multivariate polynomials such as the Tutte polynomial [2, 21].

Another variation of hypergraph coloring, is the concept of *mixed* hypergraph coloring, which has been studied extensively by Voloshin [16, 18]. A *mixed hypergraph* $\mathcal{H}$ with vertex set X, is a triple $(X, \mathcal{C}, \mathcal{D})$ such that $\mathcal{C}$ and $\mathcal{D}$ are subsets of X, called $\mathcal{C}-$hyperedges and $\mathcal{D}-$hyperedges, respectively. Given the mixed hypergraph $\mathcal{H} = (X, \mathcal{C}, \mathcal{D})$, when $\mathcal{C} = \emptyset$, we write $\mathcal{H} = (X, \mathcal{D})$ and call it a

$\mathcal{D}-$hypergraph. This is the classical hypergraph discussed in this paper. It will be interesting to see these results extended to mixed hypergraphs since the chromatic polynomial of mixed $l-$hypercyles has recently been found [3].

References

[1] J.A. Allagan. *The chromatic polynomials of some linear uniform hypergraphs*, Congr. Numerantium **187** (2007), pp.156–160.

[2] J. A. Allagan. *Tutte Polynomials of multi-bridge graphs*, Computer Science Journal of Moldova, vol.21, no. **2**(62), 2013, pp.159–173.

[3] J. A. Allagan, D. Slutzky. *Chromatic Polynomials of Mixed Hypergraphs*, Australasian Journal of Combinatorics 58 **(1)** (2014), pp. 197–213.

[4] C. Berge. *Graphs and hypergraphs*, North-Holland, Amsterdam, 1973.

[5] G.D. Birkhoff. *A determinant formula for the number of ways of coloring a map*, Ann. Math. **14** (1912), pp.42–46.

[6] M. Borowiecki, E. łazuka. *Chromatic polynomials of hypergraphs*, Discuss Math., Graph Theory **20** (2000), no. 2, pp.293–301.

[7] A. Brandstdt, V.B. Le, J.P. Spinrad. *Graph Classes: A survey*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, SIAM (1987), 18.

[8] K. Dohmen. *A broken-circuits-theorem for hypergraphs*, Arch. Math. (Basel) **64** (1995), no. 2, pp.159–162.

[9] K. Dohmen. *Chromatic polynomials of graphs and hypergraphs*, (Chromatische Polynome von Graphen und Hypergraphen) Dissertation, Düsseldorf, 1993.

[10] F.M. Dong, K.M. Koh, K.L. Teo. *Chromatic polynomials and chromaticity of graphs*, Singapore: World Scientific Publishing. xxvii, 2005.

[11] P. Erdös, A. Hajnal. *On chromatic number of graphs and set-systems*, Acta Math. Acad. Sci. Hung. **17** (1966), pp.61–99.

[12] R.L. Graham, D.E. Knuth, O. Patashnik. *Concrete Mathematics*, Addison–Wesley Publishing, 1994, pp.264–265.

[13] A. Mohr, T.D. Porter. *Applications of Chromatic Polynomials Involving Stirling Numbers*, J. Combin. Math. and Comb. Computing **70** (2009), pp.57–64.

[14] R.C. Read. *An introduction to chromatic polynomials*, J. Combin. Theory **4** (1968), pp.52–71.

[15] I. Tomescu. *Chromatic coefficients of linear uniform hypergraphs*, J. Combin. Theory B **2(72)** (1998), pp.229–235.

[16] V.I. Voloshin. *The mixed hypergraphs*, Computer Science Journal of Moldova **1** (1993), no.1, pp.45–52.

[17] V.I. Voloshin. *On the upper chromatic number of a hypergraph*, Australasian Journal of Combinatorics **11** (1995), pp.25–45.

[18] V.I. Voloshin. *Coloring Mixed Hypergraphs: Theory, Algorithms and Applications*, American Mathematical Society, 2002.

[19] M. Walter. *Some Results on Chromatic Polynomials of Hypergraphs*, Electronic Journal of Combinatorics **16** (2009), R94.

[20] D. West. *Introduction to Graph Theory*, Prentice Hall, 2001, 229.

[21] J. A. White. *On multivariate chromatic polynomials of hypergraphs and hyperedge elimination*, Electronic Journal of Combinatorics **18** (2011), 120.

[22] H. Whitney. *A logical expansion in mathematics*, Bull. Amer. Math. **38** (1932), pp.572–579.

Julian A. Allagan Received July 12, 2013

Julian A. Allagan,
University of North Georgia, Watkinsville, GA, United States.
Department of Mathematics
E–mail: *julian.allagan@ung.edu*

Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

A New Full-Newton Step $O(n)$ Infeasible Interior-Point Algorithm for $P_*(\kappa)$-horizontal Linear Complementarity Problems

Soodabeh Asadi, Hossein Mansouri*

Abstract

In this paper, we first present a brief review about the feasible interior-point algorithm for $P_*(\kappa)$-horizontal linear complementarity problems (HLCPs) based on new directions. Then we present a new infeasible interior-point algorithm for these problems. The algorithm uses two types of full-Newton steps which are called feasibility steps and centering steps. The algorithm starts from strictly feasible iterations of a perturbed problem, and feasibility steps find strictly feasible iterations for the next perturbed problem. By accomplishing a few centering steps for the new perturbed problem, we obtain strictly feasible iterations close enough to the central path of the new perturbed problem and prove that the same result on the order of iteration complexity can be obtained.

Keywords: Horizontal linear complementarity problem, infeasible interior-point method, central path.

1 Introduction

Interior-point methods (IPMs) have been studied for decades by many researchers. After Karmarkar's pioneer work on interior-point polynomial algorithm for linear programming (LP) [1], interior-point polynomial algorithms have been investigated by many researchers. For example, Ye and Tse [2] extended Karmarkar's algorithm for convex

*This author is a corresponding author

quadratic programming (CQP) and proved that it has polynomial complexity bound $O\left(n \log\left(\frac{1}{\varepsilon}\right)\right)$. IPMs also are the powerful tools to solve some widely used mathematical problems such as, semidefinite optimization (SDO) [3, 4] and linear complementarity problem (LCP) [5, 6, 7, 8]. These methods are so-called feasible IPMs. Feasible IPMs start with a strictly feasible interior point and keep feasibility during the solution process. Infeasible IPMs (IIPMs) start with an arbitrary positive point and feasibility is reached as optimality is approached. The choice of the starting point in IIPMs is crucial for the performance. Very recently, in [9, 10], Mansouri et al. presented the first full-Newton step IIPM for LCPs, which is an extension of the work for LP [11, 12, 13]. These algorithms use an intermediate problem which is a suitable perturbation of the given original problem so that at any stage the iterations are strictly feasible for the current perturbed problem. In each iteration the size of the perturbation decreases at the same speed as the barrier parameter μ. When μ changes to a smaller value, the perturbed problem also changes, and hence also the current central path. The iterations are kept feasible for the new perturbed problem and close to its central path. To achieve this, the algorithm uses a so-called feasibility step. This step serves to get iterations that are strictly feasible for the new perturbed problem and belong to the region of quadratic convergence of its μ^+-center, where μ^+ is the barrier parameter after updating. Now the algorithm can start from the point obtained in the feasibility step and perform few centering steps to obtain iterations that are close enough to the μ^+-center of the new perturbed problem. This process continues until the algorithm finds an ε-solution or detects that the problem has no optimal solution with zero duality gap. In this paper, we discuss an extension to HLCP of the just described algorithm, using Darvay's method [14]. We show that whose complexity is at least as good as the best known complexity of IIPMs. We use the results of analysis of the centering full Newton steps in [15].

The paper is organized as follows: in Section 2 we first recall some tools in the analysis of a feasible IPM that we use also in the analysis of IIPMs proposed in this paper. In Section 3 we describe an IIPM

for HLCP. The analysis of the feasibility step of our method, the most tedious part of the analysis, is carried out in Section 4. In Section 5 we will derive a complexity bound for our algorithm. In section 6 some numerical results are presented. Finally, some concluding remarks follow in Section 7.

Some notations used throughout the paper are as follows. Vectors are denoted by lower-case Latin letters and matrices by capital Latin letters. $\mathbb{R}^n_+(\mathbb{R}^n_{++})$ is the nonnegative (positive) orthant of $\mathbb{R}^n$. Further, X is the diagonal matrix whose diagonal elements are the coordinates of the vector x, i.e., $X = \text{diag}\,(x)$, and I denotes the identity matrix of appropriate dimension. The vector $xs = Xs$ is the componentwise product (Hadamard product) of the vectors x and s, and for $\alpha \in \mathbb{R}$ the vector x^α denotes the vector whose i-th component is x_i^α . We denote the vector of ones by e. As usual, $\|\cdot\|$ denotes the 2-norm for vectors and matrices. $\min(x)$ (or $\max(x)$) denotes the smallest (or largest) value of the components of x. C^1 is the set of all continuously differentiable functions in $\mathbb{R}$. Finally, if $z \in \mathbb{R}^n_+$ and $f : \mathbb{R}_+ \to \mathbb{R}_+$, then $f(z)$ denotes the vector in $\mathbb{R}^n_+$ whose i-th component is $f(z_i)$, with $1 \leq i \leq n$. We write $f(x) = O(g(x))$ if $f(x) \leq cg(x)$ for some positive constant c.

2 Preliminaries

2.1 The HLCP problem

In the HLCP, we seek a vector pair $(x,\, s) \in \mathbb{R}^{2n}$ that satisfies the conditions

$$Qx + Rs = b, \quad (x,\, s) \geq 0, \quad x^T s = 0, \tag{P}$$

where $b \in \mathbb{R}^n$, and $Q, R \in \mathbb{R}^{n\times n}$. The standard (monotone) linear complementarity problem (SLCP) is obtained by taking $R = -I$, and Q positive semidefinite matrix. The class P_* matrices are introduced by Kojima et al. [16]. The matrix pair (Q, R) belongs to P_* if there exists a constant $\kappa \geq 0$ such that

$$Qu + Rv = 0 \;\Rightarrow\; (1 + 4\kappa) \sum_{i\in\mathcal{I}^+} u_i v_i + \sum_{i\in\mathcal{I}^-} u_i v_i \geq 0 \quad \forall u, v \in \mathbb{R}^n,$$

where $\mathcal{I}^+ = \{i : u_i v_i > 0\}$ and $\mathcal{I}^- = \{i : u_i v_i < 0\}$. Then we say that the pair (Q, R) is a $P_*(\kappa)$-pair or equivalently the HLCP is called a $P_*(\kappa)$-HLCP. For $\kappa = 0$, $P_*(0)-$HLCP is called the monotone HLCP.

2.2 Central path for the $P_*(\kappa)$-HLCP

Because of nonnegativity of x and s in (P), solving HLCP is equivalent to finding a solution of the following system of equations:

$$\begin{array}{rcll} Qx + Rs & = & b, & x \geq 0, \\ xs & = & 0, & s \geq 0. \end{array} \tag{1}$$

The classical path-following IPMs consist in introducing a positive parameter μ. One considers the nonlinear system parameterized by μ:

$$\begin{array}{rcll} Qx + Rs & = & b, & x \geq 0, \\ xs & = & \mu e, & s \geq 0, \end{array} \tag{2}$$

where e denotes the all-one vector. It is shown in [17] that under interior-point condition (IPC), i.e., the existence of a vector pair $(x, s) > 0$ with $Qx + Rs = b$, there exists one unique solution $(x(\mu), (s(\mu))$. The path $\mu \to (x(\mu), (s(\mu))$ is called the central path. It is known that when $\mu \to 0$, $(x(\mu), (s(\mu))$ goes to a solution of (P).

2.3 Feasible full-Newton step

In this section we briefly present the feasible IPM in [15]. Consider the function

$$\varphi \in C^1,\ \varphi : \mathbb{R}_+ \to \mathbb{R}_+,$$

and suppose that the inverse function φ^{-1} exists. The system of equations in (2) can be written equivalently in the following form:

$$\begin{array}{rcll} Qx + Rs & = & b, & x \geq 0, \\ \varphi\left(\frac{xs}{\mu}\right) & = & \varphi(e), & s \geq 0. \end{array} \tag{3}$$

If we use Newton's method for linearizing the system (3), we get the following system for the search directions Δx and Δs:

$$\begin{array}{rcll} Q\Delta x + R\Delta s & = & 0, & x \geq 0, \\ \frac{s}{\mu}\varphi'\left(\frac{xs}{\mu}\right)\Delta x + \frac{x}{\mu}\varphi'\left(\frac{xs}{\mu}\right)\Delta s & = & \varphi(e) - \varphi\left(\frac{xs}{\mu}\right), & s \geq 0, \end{array}$$

which is equivalent to the following system:

$$\begin{array}{l} Q\Delta x + R\Delta s = 0, \qquad x \geq 0, \\ s\Delta x + x\Delta s = \mu\left(\varphi'\left(\frac{xs}{\mu}\right)\right)^{-1}\left(\varphi(e) - \varphi\left(\frac{xs}{\mu}\right)\right), \quad s \geq 0. \end{array} \tag{4}$$

We define the following notations:

$$v = \sqrt{\frac{xs}{\mu}}, \quad d_x = \frac{v\Delta x}{x}, \qquad d_s = \frac{v\Delta s}{s}. \tag{5}$$

Then we have

$$\mu v\left(d_x + d_s\right) = x\Delta s + s\Delta x, \tag{6}$$

and

$$d_x d_s = \frac{\Delta s \Delta x}{\mu}. \tag{7}$$

Consequently we have the scaled Newton-system as follows:

$$\begin{array}{rcl} \bar{Q}d_x + \bar{R}d_s & = & 0, \\ d_x + d_s & = & p_v, \end{array} \tag{8}$$

where

$$\bar{Q} = QXV^{-1}, \quad \bar{R} = RSV^{-1}, \quad p_v = \frac{\varphi(e) - \varphi\left(v^2\right)}{v\varphi'\left(v^2\right)}\ .$$

If $\varphi(t) = t$, then $p_v = v - v^{-1}$, and we obtain the standard algorithm. Now we take $\varphi(t) = \sqrt{t}$ based on Darvay's technique for LP [14]. So we have

$$p_v = 2(e - v). \tag{9}$$

Then the systems (4) and (8) are equivalent to the following systems, respectively:

$$\begin{array}{rcll} Q\Delta x + R\Delta s & = & 0, & x \geq 0, \\ s\Delta x + x\Delta s & = & 2\mu v(e-v), & s \geq 0, \end{array} \tag{10}$$

and

$$\begin{array}{rcl} \bar{Q}d_x + \bar{R}d_s & = & 0, \\ d_x + d_s & = & 2\,(e-v)\,. \end{array} \tag{11}$$

We derive the new search directions d_x and d_s by solving (11) and then we compute Δx and Δs via (5). The new iterations are given by

$$\begin{array}{rcl} x^+ & = & x + \Delta x, \\ s^+ & = & s + \Delta s. \end{array} \tag{12}$$

In the analysis of the algorithm we use the norm-based proximity measure to the central path as follows:

$$\delta(v) := \delta(x, s; \mu) = \frac{\|p_v\|}{2} = \|e - v\|\,. \tag{13}$$

The algorithm starts with (x^0, s^0) such that $\delta(x^0, s^0; \mu^0) < \tau := \frac{1}{2(1+4\kappa)}$. In each iteration the search directions at the current iterations with respect to the current value of μ be computed and then (12) be applied to get new iterations. The algorithm terminates with a point in τ-neighborhood of the central path that satisfies $n\mu \leq \varepsilon$. The following lemmas are crucial in the analysis of the algorithm. We recall them without proof.

Lemma 2.1 (Lemma 7 in [15]). *Let $\delta := \delta(x, s; \mu) \leq \frac{1}{\sqrt{1+4\kappa}}$ and $\mu^+ = (1-\theta)\mu$, where $0 < \theta < 1$. Then*

$$\delta(x, s; \mu^+) \leq \frac{\theta\sqrt{n} + (1+4\kappa)\,\delta^2}{1 - \theta\ + \sqrt{(1-\theta)\,(1-(1+4\kappa)\,\delta^2)}}.$$

Algorithm 1. Feasible IPM for $P_*(\kappa)$-HLCPs

Input:
Accuracy parameter $\varepsilon > 0$;
threshold parameter $\tau < 1$;
barrier update parameter θ, $0 < \theta < 1$;
feasible pair $\left(x^0,\, s^0\right)$ with $(x^0)^T s^0 = n\mu^0$ and $\mu^0 > 0$ such that $\delta(x^0, s^0; \mu^0) \leq \tau$.

begin
$x := x^0;\ s := s^0;\ \mu := \mu^0;$
while $n\mu \geq \varepsilon$ **do**
begin
$(x,\, s) := (x,\, s) + (\Delta x,\, \Delta s);$
$\mu := (1-\theta)\mu;$
end
end

Lemma 2.2 (Lemma 6 in [15]). *After a full Newton-step, one has*

$$(x^+)^T s^+ \leq n\mu.$$

Lemma 2.3 (Lemma 5 in [15]). *Let $\delta := \delta(x, s; \mu) \leq \frac{1}{\sqrt{1+4\kappa}}$. Then*

$$\delta(x^+, s^+; \mu) < \frac{(1+4\kappa)\delta^2}{1+\sqrt{1-(1+4\kappa)\delta^2}}.$$

Thus $\delta(x^+, s^+; \mu) \leq \left(\sqrt{1+4\kappa}\ \delta\right)^2$, which shows the quadratic convergence of the algorithm.

The following result establishes a polynomial iteration bound of the above described algorithm; it easily follows from the above lemmas.

Theorem 1 (Theorem 1 in [15]). *If* $\theta = \frac{1}{2(1+4\kappa)\sqrt{n}}$, *then the algorithm requires at most*

$$4(1+4\kappa)\sqrt{n}\log\frac{n\mu^0}{\varepsilon}$$

iterations.

3 Infeasible full-Newton step IPM

In this section we present an infeasible interior-point algorithm that generates an ε-solution of $P_*(\kappa)$-HLCPs.

3.1 The perturbed problem and its central path

We use a triple $\left(x^0, s^0, \mu^0\right) > 0$ with $x^0 s^0 = \mu^0 e$ for some (positive) number μ^0 to start our IIPM. We denote the value of the residual at these initial points as r^0, as

$$r^0 = b - Qx^0 - Rs^0. \tag{14}$$

Now for any ν with $0 < \nu \leq 1$, we consider the perturbed problem (P_ν), defined by

$$b - Qx - Rs \;=\; \nu r^0, \qquad (x,\, s) \geq 0. \tag{P_ν}$$

Note that if $\nu = 1$, then $(x,\, s) = \left(x^0,\, s^0\right)$ yields a strictly feasible solution of (P_ν). We conclude that if $\nu = 1$ then (P_ν) satisfies the IPC.

Lemma 3.1. *Let the original problem* (P) *be feasible, then the perturbed problem* (P_ν) *satisfies IPC.*

Proof. The proof is similar to the proof of Lemma 4.1 in [9]. □

We assume that (P) is feasible. Letting $0 < \nu \leq 1$, Lemma 3.1 implies that the problem (P_ν) satisfies the IPC, and hence its central path exists. This means that the system

$$\begin{array}{rcl} b - Qx - Rs & = & \nu r^0, \quad x \geq 0, \quad s \geq 0, \\ xs & = & \mu e, \end{array} \tag{15}$$

has a unique solution, for every $\mu > 0$. In the sequel this unique solution is denoted by $(x(\mu,\nu),\ s(\mu,\nu))$. It is the μ-center of the perturbed problem (P_ν). Note that since $x^0 s^0 = \mu^0 e$, $(x^0,\ s^0)$ is the μ^0-center of the perturbed problem (P_1). In other words, $\left(x\left(\mu^0,1\right),\ s\left(\mu^0,1\right)\right) = \left(x^0, s^0\right)$. In the sequel the parameters μ and ν always satisfy the relation $\mu = \nu\mu^0$.

3.2 New feasibility search directions

For the search directions in the feasibility step we use the pair $\left(\Delta^f x, \Delta^f s\right)$ such that the new iterations

$$\begin{array}{rcl} x^f &=& x + \Delta^f x, \\ s^f &=& s + \Delta^f s. \end{array} \tag{16}$$

be feasible for (P_{ν^+}), where $\nu^+ = (1-\theta)\nu$. According to the definition of (P_ν), we have

$$b - Q(x + \Delta^f x) - R(s + \Delta^f s) = \nu^+ r^0, \quad \left(x + \Delta^f x,\ s + \Delta^f s\right) > 0.$$

Since $(x,\ s)$ is feasible for (P_ν), it follows that $\Delta^f x$ and $\Delta^f s$ should satisfy

$$Q\Delta^f x + R\Delta^f s = \theta\nu r^0.$$

Now we propose new feasibility search directions for HLCPs based on Darvay's method in LP [14]. Consider the function

$$\varphi \in C^1\ , \qquad \varphi : \mathbb{R}_+ \to \mathbb{R}_+,$$

and suppose that the inverse function φ^{-1} exists. The system of equations in (15) can be rewritten equivalently in the following form:

$$\begin{array}{rcll} b - Qx - Rs &=& \nu r^0, & x \geq 0, \\ \varphi\left(\frac{xs}{\mu}\right) &=& \varphi(e), & s \geq 0. \end{array} \tag{17}$$

We define:

$$d_x^f = \frac{v\Delta^f x}{x}, \qquad d_s^f = \frac{v\Delta^f s}{s}, \tag{18}$$

where v is defined in (5). Let $\varphi(t) = \sqrt{t}$, so after using Newton's method and linearizing the system (17), we get the following system for the feasibility search directions $\Delta^f x$ and $\Delta^f s$:

$$\begin{array}{rcll} Q\Delta^f x + R\Delta^f s & = & \theta\nu r^0, & x \geq 0, \\ s\Delta^f x + x\Delta^f s & = & 2\mu v(e - v), & s \geq 0, \end{array} \tag{19}$$

and the following system for scaled feasibility search directions d_x^f and d_s^f:

$$\begin{array}{rcl} \bar{Q}d_x^f + \bar{R}d_s^f & = & \theta\nu r^0, \\ d_x^f + d_s^f & = & 2\,(e - v)\,, \end{array} \tag{20}$$

where

$$\bar{Q} = QXV^{-1}, \quad \bar{R} = RSV^{-1}, \quad X = \mathrm{diag}\,(x), \quad S = \mathrm{diag}\,(s).$$

We derive the new search directions d_x^f and d_s^f by solving (20) and then we compute $\Delta^f x$ and $\Delta^f s$ via (18).

3.3 Description of the algorithm

Suppose that $\nu = 1$ and $\mu = \mu^0$. Then $(x,\, s) = \left(x^0,\, s^0\right)$ is the μ-center of the perturbed problem (P_ν). The algorithm is started by these iterations. We measure proximity to the μ-center of the perturbed problem (P_ν) by the quantity $\delta(v)$ as defined in (13). We assume that at the start of each iteration, just before the μ-update, $\delta(v)$ is smaller than or equal to a (small) threshold value $\tau > 0$. So this is certainly true at the start of the first iteration.

One (main) iteration of the algorithm works as follows. Suppose that for some $\mu \in \left(0,\, \mu^0\right)$ we have (x, s) satisfying the feasibility condition (15) for $\nu = \frac{\mu}{\mu^0}$, and $\delta(x, s, \mu) \leq \tau$. We reduce ν to $\nu^+ = (1 - \theta)\,\nu$ and μ to $\mu^+ = (1 - \theta)\,\mu$, with $\theta \in (0, 1)$, and find new iterations $(x^+,\, s^+)$ that satisfy (15), with μ replaced by μ^+ and ν by ν^+, and such that $x^T s \leq n\mu^+$ and $\delta(x^+, s^+; \mu^+) \leq \tau$. In each main iteration first we accomplish one feasibility step to get iterations $\left(x^f,\, s^f\right)$

that are strictly feasible for (P_{ν^+}) and close enough to its central path, and then we perform a few centering steps to improve the closeness of $\left(x^f,\, s^f\right)$ to the central path of (P_{ν^+}) and obtain a pair $(x,\, s)$ that satisfies the condition $\delta\left(x,\, s;\, \mu\right) \leq \tau$. Since during the centering steps the iterations stay feasible for P_{ν^+}, it follows that for the analysis of the centering steps we can use the analysis for the feasible IPM, presented in Section 2.

Algorithm 2. Infeasible IPM for $P_*(\kappa)$-HLCP

Input:
Accuracy parameter $\varepsilon > 0$;
threshold parameter $\tau < 1$;
barrier update parameter θ, $0 < \theta < 1$;
feasible pair $\left(x^0,\, s^0\right)$ with $(x^0)^T s^0 = n\mu^0$ and $\mu^0 > 0$ such that $\delta\left(x^0,\, s^0,\, \mu^0\right) \leq \tau$.

begin
$x := x^0;\ s := s^0;\ \mu := \mu^0;$
while $\max(n\mu, ||r||) \geq \varepsilon$ **do**
begin
feasibility step:
$$(x,\, s) := (x,\, s) + (\Delta^f x,\, \Delta^f s);$$
μ-update:
$$\mu := (1-\theta)\mu;$$
centering steps:
while $\delta(v) \geq \tau$ **do**
begin
$$(x,\, s) := (x,\, s) + (\Delta x,\, \Delta s)\,;$$
end
end
end

3.4 Some useful tools

In this subsection we present some technical lemmas which we need in the rest of the paper.

Lemma 3.2. *One has*

$$\min(v) \geq 1 - \delta(v).$$

Proof. Using (13), one has

$$\delta(v) = \|e - v\| \geq |1 - \min(v)| \geq 1 - \min(v).$$

which results the lemma. □

Lemma 3.3. *One has*

$$\|v\| \leq \sqrt{n} + \delta(v).$$

Proof. Due to Cauchy-Schwartz inequality, one has

$$e^T v \leq |e^T v| \leq \|e\| \, \|v\| .$$

Using the above inequality and (13), one has

$$\delta^2(v) = \sum_{i=1}^{n} (1 - v_i)^2 = \|v\|^2 - 2e^T v + n \geq (\|v\| - \|e\|)^2 ,$$

which completes the proof. □

4 Analysis of the feasibility step

In this section we present some conditions for strict feasibility of the feasibility step and an upper bound for the proximity function after a feasibility step.

Lemma 4.1. *The new iterations* $\left(x^f, s^f\right)$ *are strictly feasible if*

$$\left\| d_x^f d_s^f \right\|_\infty < 1 - \delta^2(v).$$

Proof. The proof of this Lemma is similar to the proof of Lemma 10 in [18] and therefore is omitted. □

To simplify the notation in the sequel we introduce

$$\omega(v) := \frac{1}{2}\left(\left\|d_x^f\right\|^2 + \left\|d_s^f\right\|^2\right).$$

Lemma 4.2. *If $\omega(v) < 1 - \delta^2(v)$, then the iterations $\left(x^f, s^f\right)$ are strictly feasible.*

Proof. The proof is similar to the proof of lemma 11 in [18]. □

Assuming $\omega(v) < 1 - \delta^2(v)$, which guarantees the strict feasibility of the iterations $\left(x^f, s^f\right)$. The next lemma gives an upper bound for $\delta\left(x^f, s^f; \mu\right)$.

Theorem 4.3. *If the new iterations $\left(x^f, s^f\right)$ are strictly feasible, then*

$$\delta\left(x^f, s^f; \mu^+\right) \leq \frac{\theta\sqrt{n} + \delta^2(v) + \omega(v)}{\sqrt{1-\theta}\left(\sqrt{1-\theta} + \sqrt{1 - \delta^2(v) - \omega(v)}\right)}. \tag{21}$$

Proof. The proof is similar to the proof of Theorem 2 in [18] and therefore is omitted. □

Note that in the algorithm for using the quadratically convergence of the centering steps after doing the feasibility step we need that the following condition is satisfied

$$\delta\left(x^f, s^f; \mu^+\right) < \frac{1}{\sqrt{1+4\kappa}}. \tag{22}$$

Using Theorem 4.3, this is certainly true, if

$$\frac{\theta\sqrt{n} + \delta^2(v) + \omega(v)}{\sqrt{1-\theta}\left(\sqrt{1-\theta} + \sqrt{1 - \delta^2(v) - \omega(v)}\right)} < \frac{1}{\sqrt{1+4\kappa}}. \tag{23}$$

By some elementary calculations, we obtain that if

$$\begin{aligned} \omega(v) &< \frac{1}{4\sqrt{1+4\kappa}}, && (24)\\ \delta(v) &< \tau < \frac{1}{4\sqrt{1+4\kappa}}, && (25)\\ 0 \leq \theta &< \frac{1}{2n(1+4\kappa)}, && (26) \end{aligned}$$

then the inequality (23) is satisfied. In other words, the inequalities (24), (25) and (26) imply that the iterations $\left(x^f,\, s^f\right)$ are strictly feasible and lie in the quadratic convergence neighborhood with respect to the μ^+-center of (P_{ν^+}). We proceed by considering the value $\omega(v)$ in more detail.

4.1 An upper bound for $\omega(v)$

We start by finding some bounds for the solution of a linear system of the form

$$\begin{aligned} su + xv &= a,\\ Qu + Rv &= \tilde{b}. \end{aligned} \qquad (27)$$

Lemma 4.4 (Lemma 3.3 in [19]). *If HLCP be $P_*(\kappa)$, then for any $z = (x, s) \in \mathbb{R}^{2n}_{++}$ and any $a, \tilde{b} \in \mathbb{R}^n$ the linear system (27) has a unique solution $w = (u, v)$ and the following inequality is satisfied:*

$$\|w\|_z \leq \sqrt{1+2\kappa}\,\|\tilde{a}\| + \left(1+\sqrt{2+4\kappa}\right)\xi(z,\tilde{b}),$$

where

$$\tilde{a} = (xs)^{-\frac{1}{2}}\, a, \quad \|w\|_z^2 = \|(u,v)\|_z^2 = \|Du\|^2 + \left\|D^{-1}v\right\|^2,$$
$$D = X^{-\frac{1}{2}} S^{\frac{1}{2}},$$

and

$$\xi(z,\tilde{b})^2 = \min\{\|(\tilde{u},\tilde{v})\|_z^2 : Q\tilde{u} + R\tilde{v} = \tilde{b}\}.$$

Comparing system (27) with (19) and considering $w = (u, v) = (\Delta^f x, \Delta^f s)$, $a = 2\mu v(e - v)$, $\tilde{b} = \theta\nu r^0$, $z = (x, s)$ in the system (27), we have

$$\left\|D\Delta^f x\right\|^2 + \left\|D^{-1}\Delta^f s\right\|^2 \leq \tag{28}$$
$$\leq \left(2\sqrt{1+2\kappa}\left\|(xs)^{-1/2}\left(\mu v(e-v)\right)\right\| + (1+\sqrt{2+4\kappa})\xi(z,\theta\nu r^0)\right)^2.$$

Note that

$$\left\|(xs)^{-\frac{1}{2}}(\mu v(e-v))\right\| = \sqrt{\mu}\,\|e-v\| = \sqrt{\mu}\delta(v).$$

Also by definition of $\xi(z, \tilde{b})$, we have

$$\xi(z, \theta\nu r^0) = \theta\nu\xi(z, r^0).$$

By definitions of d_x^f and d_s^f, we obtain $D\Delta^f x = \sqrt{\mu}\ d_x^f$ and $D^{-1}\Delta^f s = \sqrt{\mu}\ d_s^f$. Substituting the above equations in (28), we have

$$\omega(v) \leq \frac{1}{\mu}\left(\sqrt{2\mu(1+2\kappa)}\delta(v) + \frac{1}{\sqrt{2}}\left(1+\sqrt{2+4\kappa}\right)\theta\nu\xi(z,r^0)\right)^2. \tag{29}$$

To proceed, we have to specify our initial iterates $\left(x^0,\, s^0\right)$. We assume that ρ_p and ρ_d are such that

$$\|x^*\| \leq \rho_p, \quad \|s^*\| \leq \rho_d, \tag{30}$$

for some optimal solutions $(x^*,\, s^*)$ of (P), and as usual we start the algorithm with

$$x^0 = \rho_p\, e, \quad s^0 = \rho_d\, e, \quad \mu^0 = \rho_p\, \rho_d. \tag{31}$$

Note that for such starting points we have clearly

$$x^* - x^0 \leq \rho_p e, \tag{32}$$

$$s^* - s^0 \leq \rho_d e. \tag{33}$$

Now we find an upper bound for $\xi(z, r^0)$.

Lemma 4.5. *Let $\xi(\cdot,\cdot)$ be as defined in Lemma 4.4. Then we have*

$$\xi(z,r^0) \leq \sqrt{\frac{\rho_p^2}{\mu v_{\min}^2}\|s\|_1^2 + \frac{\rho_d^2}{\mu v_{\min}^2}\|x\|_1^2}\ .$$

Proof. By definition of $\xi(z,\tilde{b})$, we have

$$\begin{aligned}\xi(z,r^0)^2 &= \min\{\|(\tilde{u},\tilde{v})\|_z^2 : Q\tilde{u}+R\tilde{v}=r^0\}\\ &= \min\{\|D\tilde{u}\|^2 + \left\|(D)^{-1}\tilde{v}\right\|^2 : Q\tilde{u}+R\tilde{v}=r^0\}.\end{aligned}$$

We also have

$$\begin{aligned}r^0 = b - Qx^0 - Rs^0 &= Qx^* + Rs^* - Qx^0 - Rs^0\\ &= Q(x^*-x^0) + R(s^*-s^0),\end{aligned}$$

thus by applying (32) and (33) the following inequalities are satisfied

$$\begin{aligned}\xi(z,r^0)^2 &\leq \left\|D(x^*-x^0)\right\|^2 + \left\|D^{-1}(s^*-s^0)\right\|^2 \leq\\ &\leq \|\rho_p De\|^2 + \left\|\rho_d D^{-1}e\right\|^2 =\\ &= \rho_p^2\left\|\sqrt{\frac{s}{x}}\right\|^2 + \rho_d^2\left\|\sqrt{\frac{x}{s}}\right\|^2 \leq \rho_p^2\left\|\sqrt{\frac{s}{x}}\right\|_1^2 + \rho_d^2\left\|\sqrt{\frac{x}{s}}\right\|_1^2 =\\ &= \frac{\rho_p^2}{\mu}\left\|\sqrt{\frac{\mu}{xs}}\ s\right\|_1^2 + \frac{\rho_d^2}{\mu}\left\|\sqrt{\frac{\mu}{xs}}\ x\right\|_1^2 = \frac{\rho_p^2}{\mu}\left\|\frac{s}{v}\right\|_1^2 + \frac{\rho_d^2}{\mu}\left\|\frac{x}{v}\right\|_1^2 \leq\\ &\leq \frac{\rho_p^2}{\mu v_{\min}^2}\|s\|_1^2 + \frac{\rho_d^2}{\mu v_{min}^2}\|x\|_1^2\ .\end{aligned}$$

The proof is completed. □

In what follows we obtain some upper bounds for $\|x\|_1$ and $\|s\|_1$.

Lemma 4.6. *Let $(x,\ s)$ be feasible for the perturbed problem (P_ν) and $\left(x^0,\ s^0\right)$ as defined in (31). Then for any optimal solution $(x^*,\ s^*)$, we have*

$$\begin{aligned}&\nu\left(x^Ts^0 + s^Tx^0\right) \leq\\ &\quad\leq (1+4\kappa)\left(\nu^2 n\mu^0 + \nu(1-\nu)\left((x^*)^Ts^0 + (x^0)^Ts^*\right) + x^Ts\right).\end{aligned}$$

Proof. Since $r^0 = b - Qx^0 - Rs^0$ and $b - Qx - Rs = \nu r^0$, by definition of perturbed problem, we have

$$\begin{aligned} & Q\left(\nu x^0 + (1-\nu)x^* - x\right) + R(\nu s^0 + (1-\nu)s^* - s) = \\ = \;& \nu(Qx^0 + Rs^0) + (1-\nu)(Qx^* + Rs^*) - (Qx + Rs) = \\ = \;& \nu(b - r^0) + (1-\nu)b - (b - \nu r^0) = 0. \end{aligned}$$

Thus if

$$\mathcal{I}^+ = \{i : \left(\nu x^0 + (1-\nu)x^* - x\right)_i \left(\nu s^0 + (1-\nu)s^* - s\right)_i > 0\},$$

and

$$\mathcal{I}^- = \{i : \left(\nu x^0 + (1-\nu)x^* - x\right)_i \left(\nu s^0 + (1-\nu)s^* - s\right)_i < 0\},$$

then the $P_*(\kappa)$ property implies that

$$\begin{aligned} & (1+4\kappa)\sum_{\mathcal{I}^+}(\nu x^0 + (1-\nu)x^* - x)_i(\nu s^0 + (1-\nu)s^* - s)_i + \\ & \quad + \sum_{\mathcal{I}^-}(\nu x^0 + (1-\nu)x^* - x)_i(\nu s^0 + (1-\nu)s^* - s)_i \geq 0. \end{aligned}$$

So we have

$$\begin{aligned} & \sum_{\mathcal{I}^+}(\nu x^0 + (1-\nu)x^* - x)_i(\nu s^0 + (1-\nu)s^* - s)_i + \\ & \quad + \sum_{\mathcal{I}^-}(\nu x^0 + (1-\nu)x^* - x)_i(\nu s^0 + (1-\nu)s^* - s)_i \geq \\ & \quad \geq -4\kappa \sum_{\mathcal{I}^+}(\nu x^0 + (1-\nu)x^* - x)_i(\nu s^0 + (1-\nu)s^* - s)_i. \end{aligned}$$

Thus we obtain

$$\begin{aligned} & [\nu x^0 + (1-\nu)x^* - x]^T[\nu s^0 + (1-\nu)s^* - s] \geq \\ & \quad \geq -4\kappa \sum_{\mathcal{I}^+}(\nu x^0 + (1-\nu)x^* - x)_i(\nu s^0 + (1-\nu)s^* - s)_i \geq \\ & \quad \geq \sum_{\mathcal{I}^+}\left(\nu^2 x_i^0 s_i^0 + \nu(1-\nu)(x_i^* s_i^0 + x_i^0 s_i^*) + x_i s_i\right) \geq \\ & \quad \geq -4\kappa\left(\nu^2 (x^0)^T(s^0) + \nu(1-\nu)((x^*)^T s^0 + (x^0)^T s^*) + x^T s\right). \end{aligned}$$

Since $(x^*)^T s^* = 0$, $s^T x^* + x^T s^* \geq 0$ and $s^T x^0 + x^T s^0 \geq 0$, we deduce that

$$\begin{aligned}
&-4\kappa \left(\nu^2 (x^0)^T (s^0) + \nu(1-\nu)((x^*)^T s^0 + (x^0)^T s^*) + x^T s\right) \leq \\
&\quad \leq [\nu x^0 + (1-\nu)x^* - x]^T [\nu s^0 + (1-\nu)s^* - s] = \\
&\quad = \nu^2 n\mu^0 + \nu(1-\nu)((x^*)^T s^0 + (x^0)^T s^*) - \nu(s^T x^0 + x^T s^0) + \\
&\quad + x^T s - (1-\nu)(s^T x^* + x^T s^*) + (1-\nu)(x^*)^T s^* \leq \\
&\quad \leq \nu^2 n\mu^0 + \nu(1-\nu)((x^*)^T s^0 + (x^0)^T s^*) - \nu(s^T x^0 + x^T s^0) + x^T s.
\end{aligned}$$

Therefore we have

$$\begin{aligned}
&\nu(x^T s^0 + s^T x^0) \leq \\
&\quad \leq (1+4\kappa)\left(\nu^2 n\mu^0 + \nu(1-\nu)\left((x^*)^T s^0 + (x^0)^T s^*\right) + x^T s\right).
\end{aligned}$$

The proof is completed. □

Lemma 4.7. *Let $(x,\ s)$ be feasible for the perturbed problem (P_ν) and $\left(x^0,\ s^0\right)$ as defined in (31). Then we have*

$$\|x\|_1 \leq (1+4\kappa)\left(2n + \left(\sqrt{n} + \delta(v)\right)^2\right)\rho_p, \tag{34}$$

$$\|s\|_1 \leq (1+4\kappa)\left(2n + \left(\sqrt{n} + \delta(v)\right)^2\right)\rho_d. \tag{35}$$

Proof. Using Lemma 4.6 and Lemma 3.3, this lemma may be proved in the same way as the proof of Lemma 16 in [18]. □

Substituting (34) and (35) in Lemma 4.5 and noting Lemma 3.2 gives

$$\begin{aligned}
\xi(z, r^0) &\leq \sqrt{\frac{2\rho_p^2\rho_d^2}{\mu\left(1-\delta(v)\right)^2}(1+4\kappa)^2\left(2n + \left(\sqrt{n} + \delta(v)\right)^2\right)^2} = \\
&= \sqrt{\frac{2}{\mu}}\,\frac{(1+4\kappa)\rho_p\rho_d\left(2n + (\sqrt{n} + \delta(v))^2\right)}{(1-\delta(v))}.
\end{aligned} \tag{36}$$

Now by substituting (36) in (29), we have

$$\begin{aligned}\omega(v) \;\leq\; & \frac{1}{\mu}\Bigg(\sqrt{2\mu(1+2\kappa)}\delta(v)+\\ & + \frac{\theta\nu\left(1+\sqrt{2+4\kappa}\right)(1+4\kappa)\rho_p\rho_d\left(2n+\left(\sqrt{n}+\delta(v)\right)^2\right)}{\sqrt{\mu}\,(1-\delta(v))}\Bigg)^2 =\\ = & \Bigg(\sqrt{2(1+2\kappa)}\delta(v)+\\ & + \frac{\theta\left(1+\sqrt{2+4\kappa}\right)(1+4\kappa)\left(2n+\left(\sqrt{n}+\delta(v)\right)^2\right)}{1-\delta(v)}\Bigg)^2. \qquad (37)\end{aligned}$$

4.2 Value for θ

We have found that $\delta(v) < \frac{1}{\sqrt{1+4\kappa}}$ holds if the inequalities (24) and (25) and (26) are satisfied. Then by (37), inequality (24) holds if

$$\sqrt{2(1+2\kappa)}\delta(v)+\frac{\theta\left(1+\sqrt{2+4\kappa}\right)(1+4\kappa)\left(2n+\left(\sqrt{n}+\delta(v)\right)^2\right)}{1-\delta(v)} < \\ < \frac{1}{2\sqrt[4]{1+4\kappa}}.$$

Obviously, the left-hand side of the above inequality is increasing in $\delta(v)$. Using this, one may easily verify that the above inequality is satisfied if

$$\tau = \frac{1}{32(1+4\kappa)}, \quad \theta = \frac{1}{50n(1+4\kappa)^2}, \qquad (38)$$

which is in agreement with (24) and (25).

Note that by Lemma 4.2, to keep the iterates (x^f, s^f) be feasible, the following condition must be satisfied

$$\omega(v) < 1-\delta^2(v).$$

It follows from (37) that the above inequality certainly holds if

$$\sqrt{2(1+2\kappa)}\delta(v)+\frac{\theta\left(1+\sqrt{2+4\kappa}\right)(1+4\kappa)\left(2n+\left(\sqrt{n}+\delta(v)\right)^2\right)}{1-\delta(v)}<$$
$$<\sqrt{1-\delta^2(v)}. \quad (39)$$

It is easy to verify that, for the above inequality, the left side is monotone increasing according to $\delta(v)$, while the right hand side is monotone decreasing according to $\delta(v)$. Using (38), an upper bound for the left hand side of inequality (39) is 0.1969, while a lower bound for the right hand side of inequality (39) is 0.9995. In this case, we conclude that the iterate $(x^f,\, s^f)$ is strictly feasible.

5 Complexity analysis

Let $\delta\left(x^f,\, s^f;\, \mu^+\right) \leq \frac{1}{2(1+4\kappa)}$, which is in agreement with (22). Starting at (x^f, s^f) we repeatedly apply full Newton steps until the $k-$iterate, denoted as $(x^+,\, s^+) := (x^k, s^k)$, satisfies $\delta\left(x^k,\, s^k;\, \mu^+\right) \leq \tau$. We can estimate the required number of (centering) Newton steps by using Lemma 2.3. To simplify notations we define for the moment $\delta(v^k) = \delta\left(x^k, s^k; \mu^+\right)$, $\delta(v^0) = \delta\left(x^f, s^f; \mu^+\right)$ and $\gamma = \sqrt{1+4\kappa}$. Note that $\gamma \geq 1$. It then follows that

$$\begin{aligned}\delta\left(v^k\right) \leq \left(\gamma\delta\left(v^{k-1}\right)\right)^2 &\leq \left(\gamma\left(\gamma\delta\left(v^{k-2}\right)\right)^2\right)^2 \leq \\ &\leq \ldots \leq \gamma^{2+4+\ldots+2^k}\left(\delta\left(v^0\right)\right)^{2^k}.\end{aligned}$$

This gives

$$\delta\left(v^k\right) \leq \gamma^{2^{k+1}-2}\left(\delta\left(v^0\right)\right)^{2^k} = \gamma^{-2}\left(\gamma^2\delta\left(v^0\right)\right)^{2^k} \leq \left(\gamma^2\delta\left(v^0\right)\right)^{2^k}.$$

Using the definition of γ and $\delta\left(v^0\right) \leq \frac{1}{2(1+4\kappa)}$ we obtain

$$\gamma^2\delta\left(v^0\right) \leq \left(\sqrt{1+4\kappa}\right)^2\frac{1}{2(1+4\kappa)} = \frac{1}{2}.$$

Hence we certainly have $\delta\left(x^+, s^+; \mu^+\right) \leq \tau$ if $\left(\frac{1}{2}\right)^{2^k} \leq \tau$. Taking logarithms at both sides, this reduces to

$$2^k \log_2 \frac{1}{2} \leq \log_2 \tau.$$

Thus we find that after no more than

$$\left\lceil \log_2 \left(\log_2 \frac{1}{\tau} \right) \right\rceil \tag{40}$$

centering steps we have iterations (x^+, s^+) that satisfy $\delta\left(x^+,\ s^+;\ \mu^+\right) \leq$ $\leq \tau$. Substituting the value of τ from (38) in (40) gives that in the algorithm, at most $\log_2\left(\log_2\left(32(1+4\kappa)\right)\right)$ centering steps are needed. Note that in each main iteration both the value of $x^T s$ and the norm of the residual are reduced by the factor $1-\theta$. Hence, the total number of main iterations is bounded above by

$$\frac{1}{\theta} \log \frac{\max\left\{\left(x^0\right)^T s^0,\ \left\|r^0\right\|\right\}}{\varepsilon}.$$

Due to (38) we may take

$$\theta = \frac{1}{50n(1+4\kappa)^2}.$$

Hence the total number of inner iterations is bounded above by

$$50n\left(1+4\kappa\right)^2 \log_2\left(\log_2\left(32(1+4\kappa)\right)\right) \log \frac{\max\left\{\left(x^0\right)^T s^0,\ \left\|r^0\right\|\right\}}{\varepsilon}.$$

Thus we may state without further proof the main result of the paper.

Theorem 5.1. *If (P) has an optimal solution $(x^*,\ s^*)$ such that $\left\|x^*\right\|_\infty \leq \rho_p$ and $\left\|s^*\right\|_\infty \leq \rho_d$, then after at most*

$$50n\left(1+4\kappa\right)^2 \log_2\left(\log_2\left(32(1+4\kappa)\right)\right) \log \frac{\max\left\{\left(x^0\right)^T s^0,\ \left\|r^0\right\|\right\}}{\varepsilon}$$

iterations, the algorithm finds an ε-solution of HLCP. □

Table 1. The number of iterations for the examples.

Examples	The number of iterations
6.1	107
6.2 with n=10	111
6.2 with n=20	117
6.2 with n=30	121

6 Numerical results

The algorithm was tested on some $P_*(0)$ (monotone) linear complementarity problems. So $R = -I$. For the algorithm, the initialization parameters ρ_p and ρ_d are assumed as described in Subsection 4.1, and $\tau = 0.031$, $\varepsilon = 10^{-4}$ and $\theta = 0.1$. Table 1 shows the number of iterations to obtain $\varepsilon-$solutions of the following examples with Algorithm 2.

Example 6.1.

$$Q = \begin{bmatrix} 1 & 0 & -0.5 & 0 & 1 & 3 & 0 \\ 0 & 0.5 & 0 & 0 & 2 & 1 & -1 \\ -0.5 & 0 & 1 & 0.5 & 1 & 2 & -4 \\ 0 & 0 & 0.5 & 0.5 & 1 & -1 & 0 \\ -1 & -2 & -1 & -1 & 0 & 0 & 0 \\ -3 & -1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & 0 & 0 & 0 \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 1 \\ -5 \\ -4 \\ 1.5 \end{bmatrix}.$$

Example 6.2.

$$Q = \begin{bmatrix} 1 & 2 & 2 & \dots & 2 \\ 0 & 1 & 2 & \dots & 2 \\ 0 & 0 & 1 & \dots & 2 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

7 Concluding remarks and further research

In this paper, an infeasible full-Newton step method for solving $P_*(\kappa)$-HLCP is presented. Based on new search directions, we established that the complexity of the algorithm is at least as good as the currently best known iteration bound for infeasible methods. Future research could be done on representing the other type of infeasible interior-point algorithms by analyzing the algorithm with another function $\varphi(t) \in C^1$.

8 Acknowledgments

The authors are indebted to referees for careful reading of the manuscript and for their suggestions which helped to improve the paper. The authors also wish to thank Shahrekord University for financial support. The authors were also partially supported by the Center of Excellence for Mathematics, University of Shahrekord, Shahrekord, Iran.

References

[1] N.K. Karmarkar. *A new polynomial-time algorithm for linear programming.* Proceedings of the 16th Annual ACM Symposium on Theory of Computing, vol. 4 (1984), pp. 373–395.

[2] Y. Ye, E. Tse. *An extension of Karmarkar's projective algorithm for convex quadratic programming.* Mathematical Programming, vol. 44 (1989), pp. 157–179.

[3] J. Peng, C. Roos, T. Terlaky. *Self-regular functions and new search directions for linear and semidefinite optimization.* Mathematical Programming, vol. 93 (2002), pp. 129–171.

[4] H. Mansouri, C. Roos. *A new full-Newton step $O(n)$ infeasible interior-point algorithm for semidefinite optimization.* Numerical Algorithms, vol. 52 (2) (2009), pp. 225–255.

[5] C. Gonzaga. *The largest step path following algorithm for monotone linear complementarity problems.* Math. Program., vol. 76 (1997), pp. 309–332.

[6] M. Anitescu, G. Lesaja, F. A. Potra. *Equivalence between different formulations of the linear complementarity problem.* Optim. Method Softw., vol. 7 (3) (1997), pp. 265–290.

[7] W. B. Ai, S. Z. Zhang. *An $O(\sqrt{n}L)$ iteration primal-dual path-following method, based on wide neighborhoods and large updates for monotone linear complementarity problems.* SIAM J. Optim., vol. 16 (2) (2005), pp. 400-417.

[8] H. Mansouri, S. Asadi. *A quadratically convergent $O(\sqrt{n})$ interior-point algorithm for the $P_*(\kappa)$-matrix horizontal linear complementarity Problem.* Journal of Sciences, Islamic Republic of Iran, vol. 23(3) (2012), pp. 237–244.

[9] H. Mansouri, M. Zangiabadi, M. Pirhaji. *A full-Newton step $O(n)$ infeasible interior-point algorithm for linear complementarity problems.* Nonlinear Anal. Real World Appl., vol. 12 (2011), pp. 545–561.

[10] M. Zangiabadi, H. Mansouri. *Improved infeasible-interior-point algorithm for linear complementarity problems.* Bulletin Iranian Math. Soc., vol. 38 (2012), pp. 787–803.

[11] H. Mansouri, C. Roos. *Simplified $O(n)$ infeasible interior-point algorithm for linear optimization using full-Newton step.* Optim. Methods and Soft., vol. 22 (3) (2007), pp. 519–530.

[12] H. Mansouri. *Full-Newton step interior-point methods for conic optimization.* Ph.D. thesis, Faculty of Mathematics and Computer Science, TU Delft, NL2628 CD Delft, The Netherlands (2008).

[13] C. Roos. *A full-Newton Step $O(n)$ infeasible interior-Point Algorithm for Linear Optimization.* SIAM J. Optim. vol. 16(4) (2006), pp. 1110–1136.

[14] Z. Darvay. *New interior-point algorithms in linear programming.* Adv. Model. Optim., vol. 5 (1) (2003), pp. 51–92.

[15] S. Asadi, H. Mansouri. *Polynomial interior-point algorithm for $P_*(\kappa)$ horizontal linear complementarity problems.* Numer. Algor., vol. 63 (2013), pp. 385–398.

[16] M. Kojima, N. Megiddo, T. Noma, A. Yoshishe. *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems.* Springer, Berlin, 1991.

[17] J. Stoer, M. Wechs. *Infeasible-interior-point paths for sufficient linear complementarity problems and their analyticity.* Math. Program. Ser., A., vol. 83 (3) (1998), pp. 407–423.

[18] L. Zhang, Y. Bai, Y. Xu. *A full-Newton step infeasible interior-point algorithm for monotone LCP based on a locally-kernel function.* Numer. Algor., DOI 10.1007/s11075-011-9530-1.

[19] F. Gurtuna, C. Petra, F. A. Potra, O. Shevehenko, A. Vancea. *Corrector-Predictor methods for sufficient linear complementarity problems.* Compute. Optim. Appl., vol. 48 (2011), pp. 453–485.

Soodabeh Asadi, Hossein Mansouri, Received April, 25 2013

Soodabeh Asadi
Department of Mathematics, Faculty of Mathematical Sciences,
University of Shahrekord,
P.O. Box 115, Shahrekord, Iran
E–mail: Sudabeasadi@yahoo.com

Hossein Mansouri
Department of Mathematics, Faculty of Mathematical Sciences,
University of Shahrekord,
P.O. Box 115, Shahrekord, Iran
Phone: +983814421622
E–mail: Mansouri@sci.sku.ac.ir

Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Search tree-based approach for the p-median problem using the ant colony optimization algorithm

Gabriel Bodnariuc, Sergiu Cataranciuc

Abstract

In this paper we present an approximation algorithm for the p-median problem that uses the principles of ant colony optimization technique. We introduce a search tree that keeps the partial solutions during the solution process of the p-median problem. An adaptation is proposed that allows ant colony optimization algorithm to perform on this tree and obtain good results in short time.

Keywords: ant colony optimization, p-median, location theory, combinatorial optimization, search tree.

1 Introduction

Let $G = (X, U)$ be an undirected graph, with the vertex set $X = \{x_1, x_2, ..., x_n\}$ and the edge set $U = \{u_1, u_2, ..., u_m\}$. We define two functions:

a) $\upsilon : X \to N$;

b) $\omega : U \to N$,

where $N = \{0, 1, 2, ...\}$.

Values $\upsilon(x_i)$ and $\omega(u_j)$ are called weights of the vertex $x_i \in X$ and of the edge $u_j \in U$, respectively. We denote by $d(x_i, x_j)$ the distance between vertices $x_i, x_j \in X$ [16] and consider the function $f : X \to N$ such that:

$$f(x_i) = \sum_{x_j \in X} \upsilon(x_j)\, d(x_i, x_j) \tag{1}$$

for $\forall x_i \in X$.

Definition 1 ([6]). *The vertex $x^* \in X$ is called median of the graph if $f(x^*) = \min\limits_{x \in X} f(x)$.*

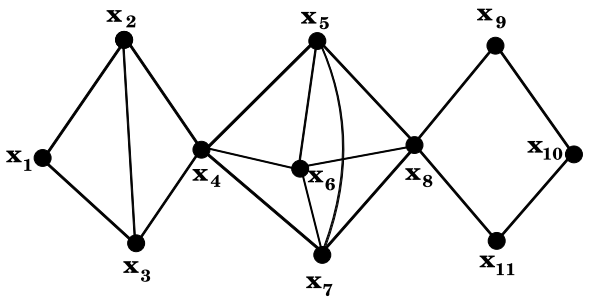


Figure 1. Vertices x_5, x_6 and x_7 are medians of the graph

According to this definition, the median of graph cannot be found univocally. For example, vertices x_5, x_6 and x_7 are the medians of the graph represented in Figure 1 (edges and vertices weights are equal to 1).

Let $A \subset X$ and an arbitrary vertex $x \in X$. We denote by $d(x, A)$ the distance between vertex x and the set A. According to [16] $d(x, A) = \min\limits_{y \in A} \{d(x, y)\}$.

We denote by $\mathcal{X}_p$ the family of all subsets of size p of the set X, $1 \leq p \leq n = |X|$, and define function $f : \mathcal{X}_p \to N$ such that:

$$f(A) = \sum_{x_j \in X} \upsilon(x_j)\, d(x_j, A) \tag{2}$$

for all $A \in \mathcal{X}_p$. Function (2) is called median function.

Definition 2 ([16]). *Set $A^* \subset X$, $|A^*| = p$, is called p-median of the graph $G = (X, U)$, if the following relation holds:*

$$f(A^*) = \min_{A \in \mathcal{X}_p} f(A) = \min_{A \in \mathcal{X}_p} \sum_{x_j \in X} p(x_j)\, d(x_j, A). \tag{3}$$

We will study the p-median problem, where vertex and edge weights have arbitrary values.

2 Methods for solving the median problem on graphs

Finding the median of an undirected graph $G = (X, U)$ is a difficult discrete optimization problem. Being a NP-complete [9], [13], this problem has stimulated the interest of many researchers for building approximation algorithms for finding graph median. In this case it is necessary to know how good is the approximation of the obtained results using these methods.

There are some well known exact algorithms for finding p-median [3], [6], [7], [18], but their efficiency is limited to a certain size of the graphs. In case of the trees, the 1-median can be found in time $O(n)$ [11] and p-median in time $O(pn^2)$ using a dynamic programming algorithm [17]. Also in polynomial time the median for d-convex simple graphs [5] can be found.

There are many other techniques for solving this problem: genetic algorithm [1], branch and cut [4], scatter search [10], variable neighborhood search [12].

In this paper we show a modification of the algorithm presented in [14] that uses the principles of the ant colony algorithms. This type of algorithm was proposed by M. Dorigo, V. Maniezzo and A. Colorni, the algorithm is described in [8]. The first problem on which the ant colony optimization algorithm was applied was the travelling salesman problem.

The ant colony optimization algorithm is based on observation of ants, which are colony organised insects. Their activity is oriented

for the benefit of the whole colony. One important aspect is the way they build short paths between colony location and food sources. Ants deposit a substance called *pheromone* while walking. Pheromone indicates the path used by other ants. Each ant usually chooses a path with high concentration of pheromone. This represents an indirect way of communication called *stigmergy*. The environment has an important role to diminish the quantity of the pheromone. This leads to changing of attractiveness of choosing different paths.

3 Tree representation of the p-median search

The problem solution is searched by starting vertex elimination from the vertex set X until p elements remain. At each step a vertex is eliminated from X according to some rules. All combinations of vertex elimination can be represented by a rooted tree, denoted by T.

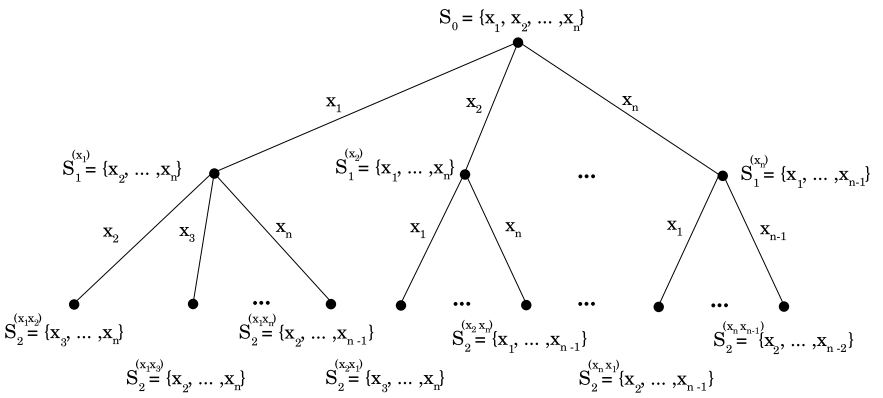


Figure 2. Tree representation of the 2-median search

Each tree node corresponds to a subset of X when from graph G a number of vertices was eliminated and the search of p-median is done on the remaining vertices. The tree root S_0 corresponds to the vertex set X. Each arc from S_0 corresponds to the elimination of a vertex from X and this vertex will not be a part of the solution. In this way, we pass from S_0 to S_1^j on the level 1. There are $n = C_n^1$ nodes on the

first level of the tree T. The process is repeated for each node on the level $i = 1, 2, ..., (n-p+1)$. An example for the case $p = n-2$ is shown in Figure 2. The bottom level nodes represent all subsets of size p of the vertex set X. The tree T contains $1 + A_n^1 + A_n^2 + ... + A_n^{n-p}$ nodes. Each node on the level k has $n-k$ descendants, where $0 \leq k \leq n-p+1$.

In the tree T a path between the root node, which corresponds to the set S_0 and any node of the level $n-p$, is considered a branch. Value of the median is among the values of median functions for the sets that correspond to the nodes of the level $n-p$.

Ant colony algorithm is used for finding a preferential branch in the tree T that will lead to choosing a set S_{n-p}^* as an approximate solution of the problem.

There are r ants in the colony. Each ant searches the p-median traversing a branch of the tree T. An iteration corresponds to the situation when r ants participate to find a branch in the tree. Obviously, some of these branches intersect other branches. At the end of an iteration, there will be some nodes with better values of the median function. These results will be used for building new preferential path in the tree T that will help to find better solutions.

4 Reduced tree representation of the p-median search

The size of the tree described above can be reduced if we take into account some specific features of the problem. This feature will optimize the solution search process performed by ant colony algorithm.

Let S_k^j, $1 \leq k \leq n-p$, be a node of the search tree, which is obtained from $S_0 = X$ after a successive elimination of vertices in the following order $a_{i_1}, a_{i_2}, ..., a_{i_k}$. In this case, we denote the node S_k^j by $S_k^{\left(a_{i_1}, a_{i_2}, ..., a_{i_k}\right)}$. The weight of this node is $\omega\left(S_k^{\left(a_{i_1}, a_{i_2}, ..., a_{i_k}\right)}\right)$ which is equal to the value of function (2) for vertex set of graph G that corresponds to the node $S_k^{\left(a_{i_1}, a_{i_2}, ..., a_{i_k}\right)}$ from the tree T. Formally, this

can be written:

$$\omega\left(S_k^{(a_{i_1},a_{i_2},...,a_{i_k})}\right) = F\left(S_k^{(a_{i_1},a_{i_2},...,a_{i_k})}\right)$$

(here $F\left(S_k^{(a_{i_1},a_{i_2},...,a_{i_k})}\right)$ represents the function value (2) for the set $A = X \setminus \{a_{i_1}, a_{i_2}, ..., a_{i_k}\}$).

Elements elimination order from set A does not affect the value of function (2) for a subset of vertices $A \subset X$, $A = X \setminus \{a_{i_1}, a_{i_2}, ..., a_{i_k}\}$, so:

Theorem 1. *Let the set $S_k^{(a_{i_1},...,a_{i_k})}$ is obtained from the set S_0 after successive elimination of elementents $a_1, a_2, ..., a_k$, and the set $S_k^{(a_{i_1^*},...,a_{i_k^*})}$ is obtained after a different elimination order of the same elements, then these two sets satisfy the following relation:*

$$F(S_k^{(a_{i_1},...,a_{i_k})}) = F(S_k^{(a_{i_1^*},...,a_{i_k^*})}).$$

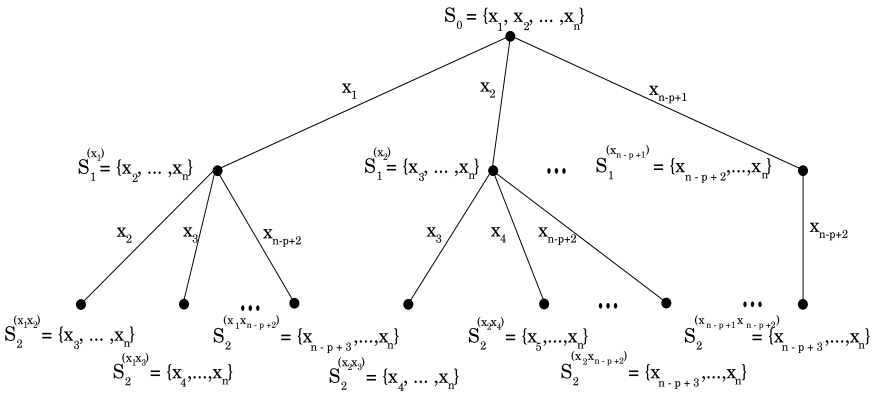


Figure 3. Reduced tree representation of the 2-median search

This gives us the possibility to build a new tree T^* which has a smaller size than T. It is obtained from T after removing some nodes that give identical results. A reduced search tree for case $p = n - 2$ is represented in Figure 3.

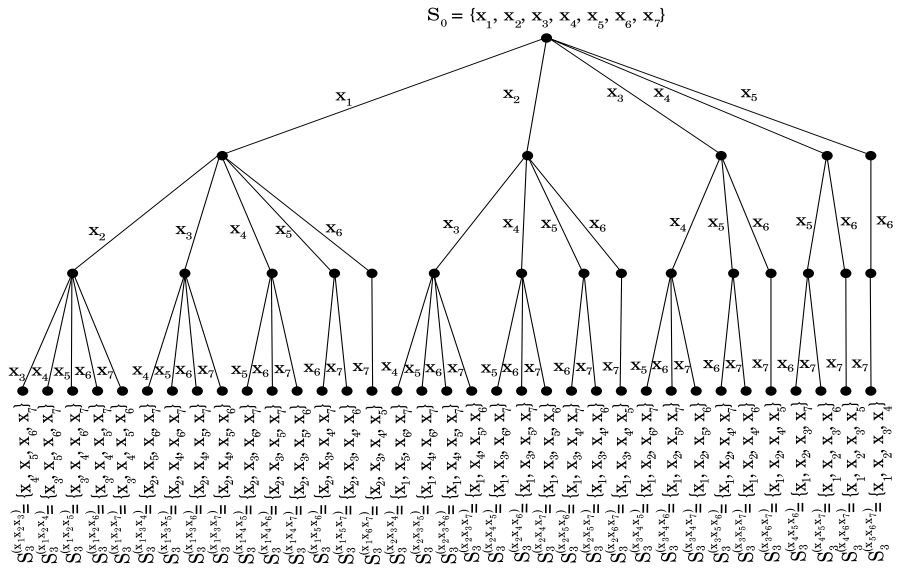


Figure 4. Reduced tree representation for the case $n = 7$ and $p = 4$

$p + 1$ arcs start from the root node $x^{T^*}_{S_0}$. These arcs correspond to exclusion of elements a_1, a_2, ..., a_{p+1} from S_0 and make connection with nodes of the level 1: $x^{T^*}_{S_1^{(a_1)}}$, $x^{T^*}_{S_1^{(a_2)}}$,..., $x^{T^*}_{S_1^{(a_{p+1})}}$ that correspond to the sets $S_1^{(a_1)} = \{a_2, a_3, ..., a_n\}$, $S_1^{(a_2)} = \{a_3, ..., a_n\}$,..., $S_1^{(a_{p+1})} = \{a_{p+2}, ..., a_n\}$, respectively.

At the tree level 1, $p + 1$ arcs start from the node $x^{T^*}_{S_1^{(a_1)}}$, and they indicate the possibility of removing of the following elements from $S_1^{(a_1)}$: a_2, a_3, ..., a_{p+2}.

It is the node $x^{T^*}_{S_1^{(a_2)}}$ from which p arcs start, and they indicate the possibility of removing of the following elements from $S_1^{(a_2)}$: a_3, ..., a_{p+2}.

It is the node $x^{T^*}_{S_1^{(a_3)}}$ from which $p - 1$ arcs start, and they indicate the possibility of removing of the following elements from $S_1^{(a_3)}$:

a_4, ..., a_{p+2}.

...

It is the node $x^{T^*}_{S_1^{(a_{p+1})}}$ from which one arc starts, and it indicates the possibility of removing of the element a_{p+2} from $S_1^{(a_{p+1})}$.

The process continues until the level $n-p$ of the tree T. There are C_n^p nodes on the last level.

There is an example in Figure 4, where $n = 7$ and $p = 4$, and in Figure 5 it is shown what happens when element x_1 is removed.

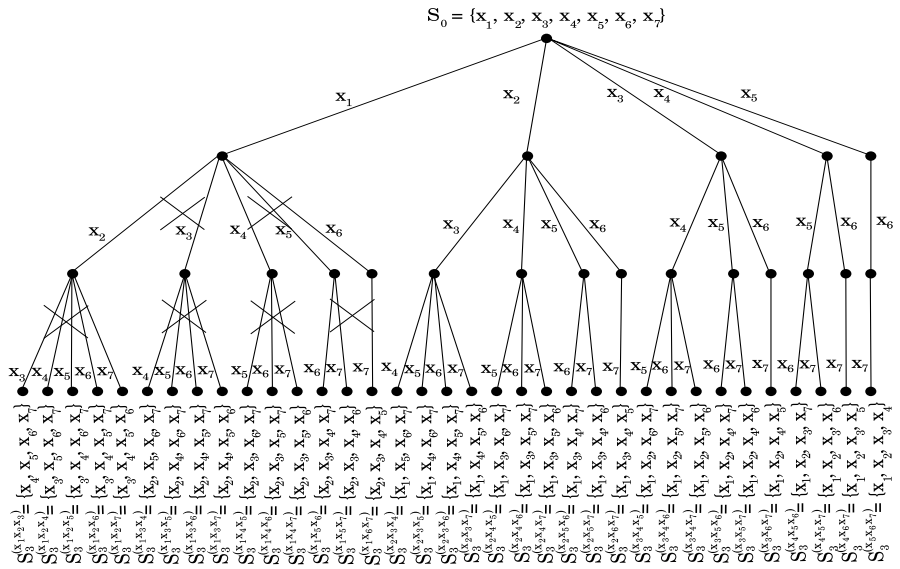


Figure 5. Prunning vertex x_1 from the search process

Theorem 2. *Let the graph $G = (X, U)$, where $|X| = n$, has the subsets $Y_k \in X$, ($|Y_k| = k, k > 1$) and $Y_{k-1} = Y_k \setminus \{x_i^*\}$, then:*

$$F(Y_k) \leq F(Y_{k-1})$$

Proof. There are two possible cases:

a) eliminated vertex x_i^* from the subset Y_k has in its neighbourhood only elements from Y_k. So, the shortest path that connects each element

of the set $X \setminus Y_k$ to one of the elements of the subset Y_k remains unchanged and the following sum does not change: $\sum_{x_i \in X \setminus Y_k} d(x_i, Y_k)$, and $F(Y_{k-1}) = F(Y_k) + \min_{x_i \in \Gamma(x_i^*)} d(x_i, x_i^*)$;

b) one of the following two relations is true: (i) $\Gamma(x_i^*) \subset X \setminus Y_k$ or (ii) $\Gamma(x_i^*) \cap \{X \setminus Y_k\} \neq \emptyset$ and $\Gamma(x_i^*) \cap \{Y_k\} \neq \emptyset$. Then paths of minimal length between some vertices from $X \setminus Y_k$ and x_i^* disappear and there is a necessity to establish new paths of minimal length, that make connections with vertices from Y_{k-1}. New paths are longer than the initial paths. A new connection between vertex x_i^* and $y_i \in Y_{k-1}$ and $y_i \in Y_{k-1}$ is built.

5 Implementation of ant colony algorithm for the p-median problem

The role of ants is to use the reduced tree described above for finding a good approximate solution. The size of tree used in the solving process should be as small as possible.

This reduced tree is denoted by T^* and it is built iteratively. At first, the tree T^* has only the root node that corresponds to the set S_0. Each ant chooses, with some probability, an element a_i, $1 \leq i \leq n-p+1$ for removing from this set. The chosen element will not be examined further by the ant as part of its solution. Eliminations are done until there are $n-p$ elements from the original set. The eliminated elements form a branch in the tree T^*. To each element $x_i \in S_0$ it is attributed a value τ_i that represents quantity of pheromone. Initially, the quantity of the pheromone is equal for every element. The pheromone influences the way how elements are chosen for removing from the set S_k.

The probability to choose element i at stage k that represents elimination of element $x_i \in S_k$, is [8]:

$$p_i(k) = \frac{[\tau_i]^\alpha [\eta_i (k+1)]^\beta}{\sum_{l=1}^{|S_k|} [\tau_l]^\alpha [\eta_l (k+1)]^\beta},$$

where $\eta_i(k+1) = 1/F(S_k \setminus \{a_i\})$.

The quantity of pheromone deposited by each ant at the end of one iteration is:

$$\Delta\tau_i^j = \begin{cases} 1/F^j(S_{n-p}) & \text{, if } x_i \in S_{n-p} \\ 0 & \text{, otherwise} \end{cases}$$

where $F^j(S_{n-p})$ is the solution obtained by the ant j.

Evaporation and addition of pheromone is realized in the following way:

$$\tau_i \leftarrow (1-\rho)\tau_i + \Delta\tau_i,$$

where $\Delta\tau_i = \sum_{j=1}^{r} \Delta\tau_i^j$, r is the number of ants and ρ is the evaporation coefficient.

According to Theorem 2, the elimination of one element from S_k leads to rising of value $F(S_{k-1})$. This helps us to build a *branch-and-bound* algorithm.

5.1 Improving the obtained solution

The solution can be improved if the vertex set X of the graph $G = (X, U)$ is partitioned into p subsets: X_i, $i = \overline{1,p}$. Let $S = \{x_{i_1}, x_{i_2}, ..., x_{i_p}\}$ be an approximate solution.

Each subset X_j, $i = \overline{1,p}$, consists of one vertex x_{i_j} contained in the solution S and the closest to it vertexes from the set $G \setminus S$. The 1-median of each subgraph $G_i = (X_i, U_i)$ is found. The set S_1 of obtained 1-medians could be considered a new approximate solution if $F(S) > F(S_1)$.

5.2 Ant colony optimization algorithm on the reduced search algorithm (ACORST)

All operations are done on the tree T^*.

1. Let $nrIterations$ be the maximal number of iterations, $nrAnts$ – the number of ants, $Rec = \infty$ – the initial value of the searched‘ solution.

2. $iteration := 0$;

3. $idAnt := 0$;

4. $k := 0$;

5. Ant $idAnt$ starts from the root node x_0;

6. If for the current node x_k there are no child nodes of the level $k+1$, then the neigbourhood set $\Gamma^+_{x_k}$ is built;

7. For each node $x_{k+1} \in \Gamma^+_{x_k}$ the value $F(S_{k+1})$ is calculated;

8. If there are nodes x_{k+1} for which $Rec < F(S_{k+1})$, then the node x_{k+1} and its subtree are removed and will not be examined further;

9. A node $x_{k+1} \in \Gamma^+_{x_k}$ is chosen with probability p described above;

10. If $\Gamma^+_{x_k} = \emptyset$, then we pass to the step 13;

11. $k := k + 1$;

12. If $k < n - p - 1$, then we pass to the step 6;

13. If $k = n - p - 1$, then improve the solution using the algorithm described in the section 5.1 and then $rec = \min\limits_{x_{k+1} \in \Gamma^+_{x_k}} F(S_{k+1})$. If $rec < Rec$, then $Rec := rec$;

14. If there are nodes x_{k+1}, for which $Rec < F(S_{k+1})$, then the node x_{k+1} and its subtree is removed and will not be examined further;

15. $k := k - 1$;

16. If $k > 0$, then we pass to the step 14;

17. $idAnt := idAnt + 1$;

18. If $idAnt < nrAnts$, then we pass to the step 4;

19. $iteration := iteration + 1$;

20. If the tree T^* has only the root node x_0, then STOP;

21. If $iteration < nrIterations$, then we pass to the step 3, else STOP.

6 Experimental results

The OR Library was chosen for tests [2]. Here the results of ACORST algorithm and the results of the implementation of ACO algorithm [14] are produced. The tests were performed on a Pentium Dual Core 2.2 GHz PC with 3 GB memory. The algorithms were implemented in C++ and the codes were compiled with gcc 4.5.0 compiler with optimization flag -O2. For both algorithms the ant colony consists of 30 ants and the number of iterations is limited to 40. In the Table 1 the best results obtained after 50 runs of the algorithm for each instance and the mean running time in seconds are produced.

Table 1. Results of ACORST and ACO algorithms

Test	n	p	Optimal	ACORST	Time	ACO	Time
pmed1	100	5	5819	5819	0.427	5819	3.214
pmed2	100	10	4093	4247	0.597	4093	3.097
pmed3	100	10	4250	4279	0.646	4273	3.11
pmed4	100	20	3034	3312	1.105	3050	2.461
pmed5	100	33	1355	1463	1.354	1357	2.211
pmed6	200	5	7824	7877	2.047	7824	24.554
pmed7	200	10	5631	5854	2.434	5645	25.641
pmed8	200	20	4445	4824	4.087	4479	23.601
pmed9	200	40	2734	3036	7.168	2797	21.064
pmed10	200	67	1255	1331	14.466	1288	16.483
pmed11	300	5	7696	7721	6.117	7696	93.177
pmed12	300	10	6634	6948	8.075	6657	86.424
pmed13	300	30	4374	4900	16.89	4449	83.907
pmed14	300	60	2968	3341	26.247	3057	74.346
pmed15	300	100	1729	1987	35.965	1773	52.85
pmed16	400	5	8162	8314	12.494	8162	245.656
pmed17	400	10	6999	7262	12.956	7010	226.541
pmed18	400	40	4809	5157	30.153	4906	213.587
pmed19	400	67	2845	3713	40.772	3319	193.144
pmed20	400	133	1789	1866	76.547	1820	132.204

The ACO algorithm has slightly better solutions than ACORST, but the running time is worse. The running time of our algorithm is proportional to p for fixed n.

7 Conclusion

We proposed an algorithm for the p-median problem using ant colony optimization technique. The algorithm is based on using a tree for keeping track of vertex removals and for pruning bad solutions. The algorithm gives good results for $p < \frac{n}{2}$ and there is a comparison table with results obtained by ACO algorithm.

References

[1] O. Alp, E. Erkut, D. Drezner. *An efficient genetic algorithm for the p-median problem*, Annals of Operations Research, vol. 122, issue 1-4, pp. 21–42, 2003.

[2] J.E. Beasley. *OR-Library: Distributing test problems by electronic mail*, Journal of the Operational Research Society, vol. 41, no. 11, pp. 1069–1072, 1990.

[3] C. Beltran, C. Tadonki, J.-Ph. Vial. *Solving the p-median problem with a semi-Lagrangian relaxation*, Computational Optimization and Applications, vol. 35, issue 2, pp.239–260, 2006.

[4] M. Boccia, A. Sforza, C. Sterle, I. Vasilyev. *A cut and branch approach for the capacitated p-median problem based on Fenchel cutting planes*, J. Math. Model. Algor., vol. 7, pp 43–58, 2008.

[5] S. Cataranciuc, N. Sur. *d-convex simple and quasi-simple graphs*, CECMI Textbook and Monograph Series, vol. 7, State University of Moldova, Chişinău, 200 p., 2009. (in Romanian)

[6] N. Christofides. *Graph Theory: an algorithmic approach*, Academic Press, 400 p., 1975.

[7] N. Christofides, J.E. Beasley. *Extensions to a Lagrangean relaxation approach for the capacitated warehouse location problem*, European Journal of Operational Research, Vol. 12, no. 1, pp. 19–28, 1983.

[8] M. Dorigo, G. Di Caro, M. Gambardella. *Ant algorithm for Discrete Optimization*, Artificial Life, vol. 5, no. 2, pp. 137–172, 1999.

[9] M.R. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 338 p., 1979.

[10] F. Garcia-Lopez, B. Melian-Batista, J.A. Moreno-Perez, J. Moreno-Vega. *Parallelization of the scatter search for the p-median problem*, Parallel Computing, vol. 29, pp 575–589, 2003.

[11] A.J. Goldman. *Optimal center location in simple networks*, Transportation Science, vol. 5, issue 2, pp. 212–221, 1971.

[12] K. Fleszar, K.S. Hindi. *An effective VNS for the capacitated p-median problem*, European Journal of Operational Research, no. 191, pp. 612–622, 2008.

[13] O. Kariv, S.L. Hakimi. *An Algorithmic Approach To Network Location Problems. II: The p-medians*, SIAM J. appl. math., vol. 37, no. 3, pp. 513–538, 1979.

[14] T.V. Levanova, M.A. Loresh. Algorithms of ant system and simulated annealing for the *p*-median problem, Avtomatika i Telemekhanika, vol. 65, no. 3, pp. 80–89, 2004. (in Russian)

[15] G.J. Lim, J. Reese, Holder G. Allen. *Fast and Robust Techniques for the Euclidean p-Median Problem with Uniform Weights*, Computers and Industrial Engineering, vol. 57, issue 3, pp. 896–905, 2009.

[16] P. Soltan. *Extremal problems on convex sets*, Ştiinţa, Chişinau, 115 p., 1976. (in Russian)

[17] A. Tamir. *An $O(pn^2)$ algorithm for the* p*-median and related problems on tree*, Operations Research Letters, vol. 19, issue 2, pp. 59–64, 1996.

[18] J.R. Weaver, R.L. Church. *A median location model with nonclosest facility service* Transportation Science, vol. 19, issue 1, pp. 58–74, 1985.

Gabriel Bodnariuc, Sergiu Cataranciuc Received June 20, 2013

Gabriel Bodnariuc
State University of Moldova, Republic of Moldova
60 A. Mateevici, MD-2009.
E–mail: gabriel.bodnariuc@gmail.com

Sergiu Cataranciuc
State University of Moldova
60 A. Mateevici, MD-2009, Republic of Moldova
E–mail: s.cataranciuc@gmail.com

Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Artificial Bee Colony with Different Mutation Schemes: A comparative study

Iyad Abu Doush, Basima Hani F. Hasan,
Mohammed Azmi Al-Betar, Eslam Al Maghayreh,
Faisal Alkhateeb, Mohammad Hamdan

Abstract

Artificial Bee Colony (ABC) is a swarm-based metaheuristic for continuous optimization. Recent work hybridized this algorithm with other metaheuristics in order to improve performance. The work in this paper, experimentally evaluates the use of different mutation operators with the ABC algorithm. The introduced operator is activated according to a determined probability called mutation rate (MR). The results on standard benchmark function suggest that the use of this operator improves performance in terms of convergence speed and quality of final obtained solution. It shows that Power and Polynomial mutations give best results. The fastest convergence was for the mutation rate value (MR=0.2).

Keywords: Artificial Bee Colony, Evolutionary Algorithms, Mutation, Meta-heuristic algorithm, Polynomial mutation.

1 Introduction

Meta-heuristic and evolutionary algorithms are computational methods that solve a problem by iteratively trying to improve a candidate solution with regard to a given fitness function. While meta-heuristics do not guarantee reaching an optimal solution if one is available, they are useful for solving combinatorial optimization problems in both science and engineering. Many algorithms that mimic natural phenomena

such as genetic algorithms [21], simulated annealing [5], ant colony optimization [8], and particle swarm optimization [8] have shown significant efficiency in solving many real-world problems.

The Artificial Bee Colony (ABC) is an optimization algorithm based on the intelligent behavior of honey bees [1, 16]. In ABC, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. In the ABC algorithm, the goal of the bees (employed bees, onlookers and scouts) is to discover the places of food sources with high nectar amount and finally the one with the highest nectar. The algorithm basically works as follows: employed bees go to their food source and return to hive. The employed bee whose food source has been abandoned becomes a scout and starts searching a new food source. Onlookers choose food sources depending on dances. Later on, the abandoned food sources are replaced with the new food sources discovered by scouts and the best food source found so far is registered. These steps are repeated until reaching the stop condition.

The ABC has been used to solve several optimization problems (e.g., forecasting stock markets [3, 18, 12], capacitated vehicle routing problem [29, 3, 9], multiproduct manufacturing system [2], and combat air vehicle path planning [13, 22, 30]). Other researchers work on modifiying the original ABC (e.g., [4, 17]).

In ABC, some scouts choose the food sources randomly based upon a randomized mutation function. In this paper, we explore the use of different mutation functions as a replacement for the original random mutation used in the ABC algorithm, specifically in the scout bees' phase, to enhance the convergence speed. In particular we used five mutation schemes (non-uniform mutation, Makinen, Periaux and Toivanen (MPT) mutation, power mutation, polynomial mutation and best-based mutation). The effectiveness of the updated ABC algorithms are evaluated by using eight benchmark functions with different characteristics (Sphere function, Step function, Schwefel's problem 2.26, Six-Hump Camel-Back, Shifted Sphere, Shifted Schwefel's problem, Shifted Rosenbrock and Shifted Rastrigin).

The remainder of this paper is organized as follows: Section 2 introduces the Artificial Bee Colony (ABC) Algorithm. Section 3 presents the different mutation methods. The experimental environment is presented in Section 4. Finally, we conclude in Section 5.

2 The Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm was first proposed by Karaboga in [14, 16]. In a real bees colony, there are different types of specialized bees performing different tasks. The main goal of the bees is to maximize the amount of nectar stored in the hive.

According to the ABC algorithm, the bees' colony involves three different types of bees. Employed bees, onlooker bees and scouts. Half of the colony are employed bees and the reset are onlookers. Employed bees exploit food sources visited previously and provide the onlooker bees with information regarding the quality of the food sources they are exploiting. The onlooker bees use the information shared with employed bees to decide where to go. Scout bees explore the environment randomly looking for new sources of food. When a food source exhausted, the corresponding employed bee becomes a scout. The main steps of the ABC algorithm are described below.

Step1: Initialize the food source positions.

In the ABC algorithm the position of a food source represents a possible solution of the optimization problem and the amount of nectar at each food source represents the fitness of the corresponding solution. In this step of the algorithm, solutions x_i (where $i = 1 \dots n$) are randomly generated within the ranges of the problem's parameters, where n is the number of food sources (one source for each employed bee). Each solution is a vector of d dimensions, where d is the number of the problem's parameters.

Step 2: Each employed bee generates a new food source and exploits the better one.

The new food source (solution) is generated according to the following formula:

$$y_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}),$$

where φ_{ij} is a random number in the range [-1,1], k is the index of the solution chosen randomly and $j = 1, \ldots, d$.

After generating the new solution y_i, the employed bee compares between it and the original solution x_i and exploits the better one.

Step 3: Each onlooker bee chooses a food source based on its quality, generates a new food source and then exploits the best one.

An onlooker bee selects a food source based on the probability value associated with it. The probability value is calculated as follows:

$$p_i = \frac{Fitness_i}{\sum_{k=1}^{n} Fitness_k},$$

where $Fitness_i$ is the fitness of solution i, and n is the number of food sources which is equal to the number of employed bees.

Step 4: Stop the exploitation of the food sources exhausted and convert its employed bees into scouts.

A solution (represented by a food source) is said to be exhausted if it has not been improved after a predetermined number of execution cycles. The employed bee of each exhausted food source is converted into a scout that performs a random search for another solution (food source) based on the following formula:

$$x_{ij} = x_j^{min} + (x_j^{max} - x_j^{min}) * rand,$$

where x_j^{max} and x_j^{min} are the upper and the lower bounds of parameter (decision variable) j.

Step 5: Keep the best solution (food source) found so far.

Step 6: Repeat steps 2 to 5 until the termination condition is satisfied.

3 Mutation Methods

The mutation method is an essential operator in EAs [7]. It normally provides a mechanism to explore unvisited regions in the search space. It operates with less consideration to the natural principle of the 'survival of the fittest'. Any successful EA should have a mutation mechanism to ensure the diversification of the search space while makes use of the accumulative search.

Genetic Algorithm can be seen as the most popular EA algorithm that is widely used for optimization problems [25, 26]. It begins with population of individuals generated randomly. Evolutionary, it selects, recombines and mutates the current population to come up with a new population hopefully better. It exploits the current population using selection and recombination operators. It also explores the search space using mutation. This is necessary to prevent getting stuck in the local optima and increasing the chance of finding the global optima.

The way of diversifying the individuals in the population have been widely studied [11, 10, 7]. The mutation operator in GA randomly and structurally changes some genes in the individual without considering the characteristics of their parents. In order to implement a mutation operator, two issues should be watched: i) the probability of using mutation over population and ii) the power of mutation represented by the perturbation obtained in an individual.

Similarly to GA, other population-based method have a mutation operator to diversify the search, e.g., the Random consideration in Harmony Search Algorithm, the scout bee in Artificial Bee Colony (ABC). Particularly in ABC, the scout bee provides a mechanism to diversify the individuals and therefore to prevent the search to fault down in a local optima trap.

Figure 1 flowcharts the scout bee process. At each iteration, all individuals, $\boldsymbol{x}_i, \forall i \in 1 \dots SN$, in the population will be examined using $\mathsf{Scout}[.]$ vector. Note that the $\mathsf{Scout}[.]$ vector contains an accumulative counter information about each individual regarding if they improved. In case the individual (say $\mathbf{x}_i = (x_{i,1}, x_{i,2} \dots, x_{i,N})$, where N is the number of genes) is improved in such iteration, the $\mathsf{Scout}[i]$ will be

initialized by 0, otherwise it will be incremented by 1 until a certain *limit* exceeded, then a Do_mutation() operator will be applied.

In general, the continuous optimization problem is formulated as follows

$$\min\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathbf{X}\},$$

where $f(\boldsymbol{x})$ is the objective function; $\boldsymbol{x} = \{x_i \mid i = 1, \ldots, N\}$ is the set of decision variables (or genes). $\mathbf{X} = \{\boldsymbol{X}_i \mid i = 1, \ldots, N\}$ is the possible value range for each gene, where $\boldsymbol{X}_i \in [L_{x_i}, U_{x_i}]$, where L_{x_i} and U_{x_i} are the lower and upper bounds for the gene x_i respectively and N is the number of genes.

In this paper, five mutation operators have been investigated in the scout bee operator. These mutation operators are controlled by a Mutation Rate (MR). The purpose of mutation is to diversify the search direction and to prevent the convergence into local optimum. Algorithm 1 shows that each gene in the selected individual will be examined for whether or not it will be changed randomly. In each mutation type, the change process is different as we will discuss below.

Algorithm 1 Scout Bee Procedure

```
for i = 1, ··· , SN do
  if Scout[i] < limit then
    for j = 1, ··· , N do
      if U(0, 1) < MR then
        DO_Mutation()
      end if
    end for
  end if
end for
```

3.1 Original mutation

The original mutation is proposed by [15], which is called random mutation. In this type of mutation, the gene ($x_{i,j}$) that met the probability of MR is changed as follows

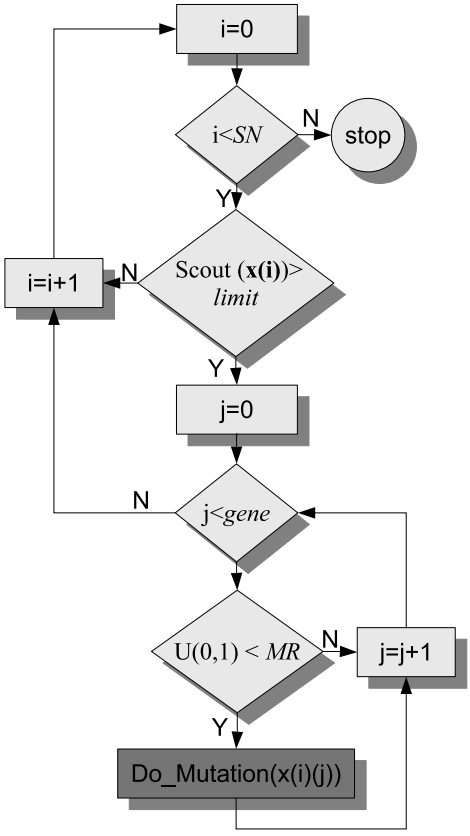


Figure 1. The flowchart Scout Bee Operation

$$x'_{i,j} = L_{x_j} + U(-1,1)(U_{x_j} - L_{x_i}).$$

In this type of mutation, the value of the gene is replaced randomly with a value within the range of decision variable [21] in the abandoned solution i. Note that $U(-1,1)$ generates a random number between -1 and 1.

3.2 Non-uniform random mutation

The non-uniform random mutation is one of popular mutation types that is widely used in GA [21, 20]. In non-uniform mutation, as the generations increase, the step size decreases, therefore making a uniform search in the initial stage of search and very little at later stages. In this type of mutation, the gene $(x_{i,j})$ that met the probability of MR is changed as follows:

$$x'_{i,j} = x_{i,j} \times (1 - U(0,1)^{(\frac{1-t}{MSN})^b}),$$

where b is a system parameter determining the degree of dependency of iteration number (in this study, the value of b is fixed to 5 as recommended by a previous study [21]), t is the generation (or iteration) number. And MSN refers to the maximum number of iterations in ABC. Note that the gene $x'_{i,j}$ is assigned with a value in a range $[0, x_{i,j}]$.

3.3 Makinen, Periaux and Toivanen (MPT) Mutation

Makinen, Periaux and Toivanen mutation, proposed by [19], is a relatively new mutation and has been applied to solve multidisciplinary shape optimization problem in addition to a large set of optimization problems with constrained nature. In this type of mutation, the gene $(x_{i,j})$ that met the probability of MR is changed as follows:

$$x'_{i,j} = (1 - \hat{t}_j) \times L_{x_j} + \hat{t}_j \times U_{x_j},$$

where

$$\hat{t}_j \leftarrow \begin{cases} t_j - (t_j) \times (\frac{t_j - r_j}{t_j})^b & r_j < t_j \\ t_j & r_j = t_j, \\ t_j + (1 - t_j) \times (\frac{r_j - t_j}{1 - t_j})^b & r_j > t_j \end{cases}$$

and

$$t_j = \frac{x_{i,j} - L_{x_j}}{U_{x_j} - x_{i,j}}.$$

Normally, the value of $b = 1$.

3.4 Power Mutation

This type of mutation operator is based on power distribution, and proposed by [7]. It is an extended version of MPT mutation. In this type of mutation, the gene ($x_{i,j}$) that met the probability of MR is changed as follows:

$$x'_{i,j} \leftarrow \begin{cases} x_{i,j} - s \times (x_{i,j} - L_{x_j}) & t < r \\ x_{i,j} - s \times (U_{x_j} - x_{i,j}) & otherwise, \end{cases}$$

where

$$t = \frac{x_{i,j} - L_{x_j}}{U_{x_j} - x_{i,j}},$$

and s is a random number generated according to the final distribution, and r is a uniform random number generated in the range 0 and 1, $r \in [0, 1]$.

3.5 Polynomial mutation

Polynomial mutation was first introduced by Deb and Agrawal in [6] which has been successfully applied for single and multi-objective optimization problems. In this type of mutation, the gene ($x_{i,j}$) that met the probability of MR is changed as follows:

$$x'_{i,j} = x_{i,j} + \delta_q \times (U_{x_j} - L_{x_j}),$$

where

$$\delta_q \leftarrow \begin{cases} [(2r) + (1-2r) + (1-\delta_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1 & r < 0.5 \\ 1 - [(2(1-r)) + (2(r-0.5)) + (1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} & otherwise, \end{cases}$$

$$\delta_1 = \frac{x_{i,j} - L_{x_j}}{U_{x_j} - L_{x_j}},$$

$$\delta_2 = \frac{U_{x_j} - x_{i,j}}{U_{x_j} - L_{x_j}}.$$

Note that r is a uniform random number generated in the range 0 and 1, $r \in [0, 1]$.

3.6 Best-based Mutation

This type of mutation is initially proposed by [27]. It is effective and powerful mutation type for unconstrained large scaled optimization problems. In this type of mutation, four individuals are randomly selected (i.e., $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}, \boldsymbol{x}_{r3}, \boldsymbol{x}_{r4})$ from the entire population. After that, the gene $(x_{i,j})$ that met the probability of MR is changed as follows:

$$x'_{i,j} = x_{best,j} + F \times (x_{r1,j} - x_{r2,j}) + F \times (x_{r3,j} - x_{r4,j}),$$

where $x_{best,j}$ is the gene in the best individual from the entire population. F is an important parameter that ensures the balance between exploration and exploitation which normally is experimentally determined, and takes a value range between 0 and 1.

4 Experimental results

We used 8 global minimization benchmark functions (Table 2) to evaluate different mutation methods used in ABC (see Table 1). These benchmark functions have been selected as one function for each set of unique characteristics (e.g, unimodal, multi-modal, and separable), as it is shown in the last column of Table 2. The benchmark functions

were implemented with a multi-dimension (N=100), with the exception to Six-Hump Camel-Back function which is two-dimensional.

We conducted four different experiements, each one with different mutation rate (MR = 0, 0.2, 0.5, and 0.8). In each experiment we tested six different mutation methods: ABC original, Non-uniform, MPT, Power, Polynomial, and Best mutation. Each experiment was repeated 30 times with different random seeds. The average of the best values obtained by the algorithms is calculated. The obtained results of the mean best values and standard deviation are shown in tables 3, 4, 5, and 6.

The experiments were executed on a P4 machines with 1 GB of RAM using C++ under Microsoft Visual Studio environment. In all the experiments the values for common parameters are as follows: the population size (NP) was 100, the food sources was 50, the stopping criteria = 10,000, and the algorithm ran for 30 times. The limit is defined using the formula D* NP*0.5 which uses the dimension of the problem and the colony size to determine the limit value. These values are similar to what has been suggested in the state of the art methods.

The results in tables 3, 4, 5, and 6 show that for the sphere function the mutation rate (MR=0.2) gives the best result, and the power mutation (M4) gives the best result. The best result for sphere function using other mutation rate values (MR= 0.5 and 0.8) was given by the polynomial mutation (M5).

On the other hand, for the Schwefel problem function the mutation rate (MR=0.2) gives the best result with MPT mutation (M3). The shifted rosenbrock gives the best result using the mutation rate (MR=0.5) with MPT mutation (M3). The best result for shifted rosenbrock function using other mutation rate values (MR= 0.2 and 0.8) was given by the Non-uniform mutation (M2).

For the rest of the functions (i.e., step, camel-back, shifted sphere, shifted schwefel, and shifted rastrigin) all the mutation methods (M1-M6) with all mutation rates reached the global optimal solution.

Figure 3 shows the effect of using the mutation rate value (MR=0.8) on the convergence speed of the six different mutation methods. The compared benchmark functions are: sphere and shifted rosenbrock.

For the two functions polynomial mutation (M5) has the fastest convergence speed. The slowest convergence speed for the sphere function was for Power mutation (M4). On the other hand, for the shifted rosenbrock, Best mutation (M6) has the slowest convergence speed.

Figure 2 compares the effect of using different mutation rate values on the convergence speed of the mutation method used (i.e., M1 to M6 in Table 1). Generally speaking the mutation rate (MR=0.2) has the fastest convergence speed for the mutation methods M1, M4, and M6 (Original ABC, Power, and Best). For the mutation methods M3 and M5 (MPT and Polynomial), the mutation rate (MR=0.8) has the fastest convergence speed. These observations confirm the mean best results obtained, which are usually using mutation rate value (MR=0.2). This value allows diversifying the population without changing the population to be far from optimal solution.

Table 1. The different mutation schemes used in the experiments.

Original ABC	Non-uniform	Makinen, Periaux and Toivanen (MPT)	Power	Polynomial	Best
M1	M2	M3	M4	M5	M6

5 Conclusion and Further Work

It is common in swarm-based algorithms to hybridize with operators from evolutionary algorithms such as mutation. Normally, the hybridization comes by adding a common to an existing algorithm. This paper investigated the performance of ABC using different mutation schemes. The mutation happened in the ABC algorithm after reaching the limit and calling the scout bees. We updated the original ABC algorithm with six different mutation schemes. The application of the mutation is performed according to a defined parameter called mutation rate (MR). We evaluated the updated algorithms of ABC with 8 global benchmark functions used in the literature. We investigated the mean best value and studied the convergence speed using the six different mutation methods parametrized with four different mutation rates.

Table 2: Benchmark functions used to evaluate HS variations

Function Name	Expression	Search Range	Optimum Value	Category [24]
Sphere function [23]	$f_1(x) = \sum_{i=1}^{N} x_i^2$	$x_i \in [-100, 100]$	$\min(f_1) = f(0, \ldots, 0) = 0$	unimodal
Step function [23]	$f_2(x) = \sum_{i=1}^{N} (\lfloor x_i + 0.5 \rfloor)^2$	$x_i \in [-100, 100]$	$\min(f_2) = f(0, \ldots, 0) = 0$	discontinuous unimodal
Schwefel's problem 2.26 [31]	$f_3(x) = -\sum_{i=1}^{N} \left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right) \right)$	$x_i \in [-500, 500]$	$\min(f_3) = f(420.9687, \ldots, 420.9687) = -12569.5$	difficult multimodal
Six-Hump Camel-Back function [23]	$f_4(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$x_i \in [-5, 5]$	$\min(f_4) = f(-0.08983, 0.7126) = -1.0316285$	low dimensional

Continuation of Table 2

Function Name	Expression	Search Range	Optimum Value	Category [24]
Shifted Sphere function [28]	$f_5(x) = \sum_{i=1}^{N} z_i^2 + f\_bias_1$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-100, 100]$	$\min(f_5) = f(o_1, \ldots, o_N) = f\_bias_1 = -450$	unimodal, shifted, separable, and scalable
Shifted Schwefel's problem 1.2 [28]	$f_6(x) = \sum_{i=1}^{N} \left(\sum_{j=1}^{i} z_j \right)^2 + f\_bias_2$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-100, 100]$	$\min(f_6) = f(o_1, \ldots, o_N) = f\_bias_6 = -450$	unimodal, shifted, non-separable, and scalable
Shifted Rosenbrock [28]	$f_7(x) = \sum_{i=1}^{N-1} (100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2) + f\_bias_6$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-100, 100]$	$\min(f_7) = f(o_1, \ldots, o_N) = f\_bias_6 = -390$	multi-modal, shifted, non-separable, and scalable
Shifted Rastrigin [28]	$f_8(x) = \sum_{i=1}^{N} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_9$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-5, 5]$	$\min(f_8) = f(o_1, \ldots, o_N) = f\_bias_9 = -330$	multi-modal, shifted, separable, and scalable

Generally speaking the results show that Power and Polynomial mutations give best results. The mutation rate value (MR=0 and 0.8) gives the slowest convergence. On the other hand, the fastest convergence was for the mutation rate value (MR=0.2).

The future work can be experimenting different mutation schemes after modifying the original ABC algorithm to apply mutation on early stages of the algorithm. Currently, the algorithm uses mutation after exceeding the limit of reaching constant optimal solution. We could benefit more from the different mutation methods presented in this paper by applying them early in the ABC algorithm.

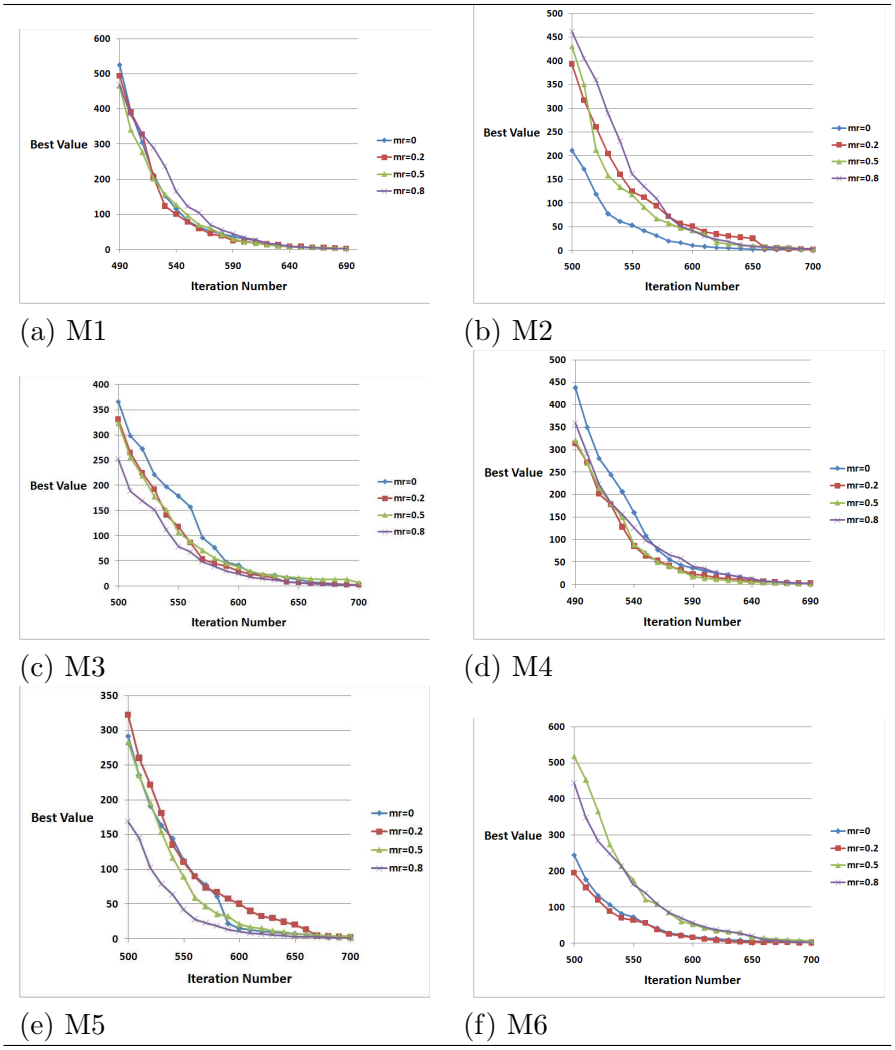
It might be interesting to adaptively select the mutation operator based on algorithm performance. But the algorithm needs to use a single best mutation rate, and mutation operator should be recommended, as it is not allowed to change this operator for each benchmark. For example, if the ABC algorithm is stuck in local optima, then use another operator in the hope to improve search capabilities and reach the global optima.

Table 3. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0

Sphere	**Step**	**Schwefel's 2.26**	**Camel-Back**	**Shifted Sphere**	**Shifted Schwe-fel's**	**Shifted Rosen-brock**	**Shifted Rast-rigin**
2.15E-15	0	-4.199724E+04	-1.03163E+00	-450	-450	3.90054E+02	-330
(1.60935E-16)	(0)	(0.25795393)	(0)	(0)	(0)	(0.102094052)	(0)

References

[1] A. Aderhold, K. Diwold, A. Scheidler, and M. Middendorf. Artificial bee colony optimization: A new selection scheme and its performance. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 283–294, 2010.



(a) M1 (b) M2

(c) M3 (d) M4

(e) M5 (f) M6

Figure 2. The convergence speed for the sphere function using different mutation rates (10,000 iterations). The figures show the portions with noticable difference, here it is within the range 0 – 600.

Table 4. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0.2

	M1	M2	M3	M4	M5	M6
Sphere	1.60E-15 (2.4049E-16)	1.63E-15 (2.50432E-16)	1.60E-15 (2.95416E-16)	**1.56E-15 (2.32707E-16)**	1.93E-15 (2.07427E-16)	1.66E-15 (2.41625E-16)
Step	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)
Schwefel's problem 2.26	-4.199731E+04 (0.223375361)	-4.199734E+04 (0.256613668)	**-4.199724E+04 (0.258221147)**	-4.199731E+04 (0.217112715)	-4.199730E+04 (0.204236734)	-4.199736E+04 (0.160781139)
Camel-Back	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)
Shifted Sphere	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Schwefel's problem 1.2	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Rosenbrock	3.90025E+02 (0.088819745)	**3.90018E+02 (0.033415342)**	3.90035E+02 (0.092817557)	3.90031E+02 (0.03157629)	3.90039E+02 (0.081382741)	3.90023E+02 (0.05924405)
Shifted Rastrigin	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)

[2] S. Ajorlou, I. Shams, and M.G. Aryanezhad. Optimization of a multiproduct conwip-based manufacturing system using artificial bee colony approach. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2, 2011.

[3] B. Akay and D. Karaboga. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, pages 1–14, 2010.

[4] Nebojsa Bacanin and Milan Tuba. Artificial bee colony (abc) algorithm for constrained optimization improved with genetic oper-

Table 5. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0.5

	M1	M2	M3	M4	M5	M6
Sphere	2.10E-15 (1.84796E-16)	2.16E-15 (1.95236E-16)	2.09E-15 (2.65087E-16)	2.13E-15 (2.10979E-16)	**1.85E-15** (**1.61943E-16**)	2.14E-15 (2.2148E-16)
Step	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)
Schwefel's problem 2.26	-4.199731E+04 (0.171671967)	-4.199736E+04 (0.195965045)	-4.199733E+04 (0.291429296)	-4.199731E+04 (0.214449308)	**-4.199728E+04** (**0.241189581**)	-4.199735E+04 (0.247377045)
Camel-Back	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)
Shifted Sphere	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Schwefel's problem 1.2	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Rosenbrock	3.90025E+02 (0.056847244)	3.90036E+02 (0.0972493)	**3.90016E+02** (**0.031588391**)	3.90019E+02 (0.03157629)	3.90039E+02 (0.066354577)	3.90032E+02 (0.083040386)
Shifted Rastrigin	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)

ators. *Studies in Informatics and Control*, 21(2):137–146, 2012.

[5] Thomas Back. *Evolutionary Algorithms in Theory and Practice*. OXFORD UNIVERSITY PRESS, New York, Oxford, 1996.

[6] K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[7] Kusum Deep and Manoj Thakur. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation*, 193(1):211 – 230, 2007.

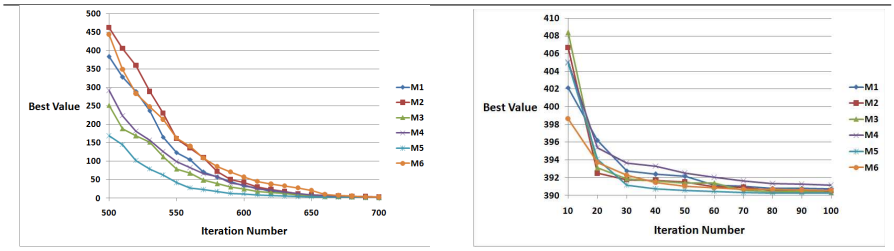
Table 6. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0.8

	M1	M2	M3	M4	M5	M6
Sphere	2.08E-15 (2.32587E-16)	2.10E-15 (1.97095E-16)	2.18E-15 (1.55962E-16)	2.16E-15 (1.62883E-16)	**1.86E-15 (1.65775E-16**)	2.09E-15 (2.15497E-16)
Step	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)
Schwefel's problem 2.26	-4.199741E+04 (0.228840154)	-4.199729E+04 (0.165952493)	-4.199728E+04 (0.277530158)	-4.199734E+04 (0.158621939)	**-4.199726E+04 (0.252550211**)	-4.199728E+04 (0.175545128)
Camel-Back	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)
Shifted Sphere	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Schwefel's problem 1.2	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Rosenbrock	3.90034E+02 (0.074387028)	**3.90020E+02 (0.041508398**)	3.90031E+02 (0.05580612)	3.90042E+02 (0.143667564)	3.90027E+02 (0.06110496)	3.90061E+02 (0.18150363)
Shifted Rastrigin	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)

[8] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[9] M. El-Abd. Black-box optimization benchmarking for noiseless function testbed using artificial bee colony algorithm. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 1719–1724. ACM, 2010.

[10] Mohammad Hamdan. A dynamic polynomial mutation for evolutionary multi-objective optimization algorithms. *International Journal on Artificial Intelligence Tools*, 20(1):209–219, 2011.



(a) Sphere (b) Shifted Rosenbrock

Figure 3. The convergence speed using different mutation methods (10,000 iterations and MR=0.8).

[11] F. Herrera and M. Lozano. Two-loop real coded genetic algorithms with adaptive control of mutation step sizes. *Applied Intelligence*, 13:187–204, 2002.

[12] T.J. Hsieh, H.F. Hsiao, and W.C. Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11(2), 2010.

[13] F. Kang, J. Li, and Z. Ma. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, 2011.

[14] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 2009.

[15] D. Karaboga and B. Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. *Foundations of Fuzzy Logic and Soft Computing*, pages 789–798, 2007.

[16] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008.

[17] Dervis Karaboga and Bahriye Akay. A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing*, 11(3):3021 – 3031, 2011.

[18] S. Kockanat, T. Koza, and N. Karaboga. Cancellation of noise on mitral valve doppler signal using iir filters designed with artificial bee colony algorithm. *Current Opinion in Biotechnology*, 22:S57–S57, 2011.

[19] Raino A.E. Makinen, Jacques Periaux, and Jari Toivanen. Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms. *International Journal for Numerical Methods in Fluids*, 30(2):149–159, 1999.

[20] Z Michalewicz, T Logan, and S Swaminathan. Evolutionary operators for continuous convex parameter space. In *Proceedings of Third Annual Conference on Evolutionary Programming*. 1994.

[21] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[22] SN Omkar, J. Senthilnath, R. Khandelwal, G. Narayana Naik, and S. Gopalakrishnan. Artificial bee colony (abc) for multi-objective design optimization of composite structures. *Applied Soft Computing*, 2009.

[23] M. G. H. Omran and M. Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656, 2008.

[24] Quan-Ke Pan, P.N. Suganthan, M. Fatih Tasgetiren, and J.J. Liang. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, 216(3):830 – 848, 2010.

[25] J.E. Smith and T.C. Fogarty. Operators and parameter adaptation in genetic algorithms. *Soft computing*, 1(2):81–87, 1997.

[26] W.M. Spears. *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993.

[27] Nadezda Stanarevic. Comparison of different mutation strategies applied to artificial bee colony algorithm. In *Proceedings of the 5th European conference on European computing conference*, ECC'11, pages 257–262, Stevens Point, Wisconsin, USA, 2011. World Scientific and Engineering Academy and Society (WSEAS).

[28] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report KanGAL Report#2005005, IIT Kanpur, India, Nanyang Technological University, Singapore, 2005.

[29] WY Szeto, Y. Wu, and S.C. Ho. An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, 2011.

[30] C. Xu, H. Duan, and F. Liu. Chaotic artificial bee colony approach to uninhabited combat air vehicle (ucav) path planning. *Aerospace Science and Technology*, 2010.

[31] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82 –102, 1999.

Iyad Abu Doush[1], Basima Hani F. Hasan[1], Mohammed Azmi Al-Betar[2], Eslam Al Maghayreh[1], Faisal Alkhateeb[1]

Received December 17, 2012

[1] Yarmouk University
Computer Science Department, Irbid, Jordan

Iyad Abu Doush
E–mail: *iyad.doush@yu.edu.jo*

[2]Jadara University
Computer Science Department, Irbid, Jordan

Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

A Note on Solvable Polynomial Algebras

Huishi Li

Abstract

In terms of their defining relations, solvable polynomial algebras introduced by Kandri-Rody and Weispfenning [J. Symbolic Comput., 9(1990)] are characterized by employing Gröbner bases of ideals in free algebras, thereby solvable polynomial algebras are completely determinable and constructible in a computational way.

Keywords: PBW basis, Monomial ordering, Gröbner basis, Solvable polynomial algebra.

1 Introduction

In the late 1980s, the Gröbner basis theory invented by Bruno Buchberger ([2], [3]) for commutative polynomial ideals was successfully generalized to one-sided ideals in enveloping algebras of Lie algebras by Apel and Lassner [1], to one-sided ideals in Weyl algebras (including algebras of partial differential operators with polynomial coefficients over a field of characteristic 0) by Galligo [6], and more generally, to one-sided and two-sided ideals in solvable polynomial algebras (or algebras of solvable type) by Kandri-Rody and Weispfenning [8]. In particular, the noncommutative Buchberger Algorithm for computing Gröbner bases of one-sided and two-sided ideals in solvable polynomial algebras has been implemented in some well-developed computer algebra systems such as SINGULAR [4].

Originally, a noncommutative solvable polynomial algebra R' was defined in [8] by first fixing a monomial ordering $\prec$ on the standard K-basis $\mathscr{B} = \{X_1^{\alpha_1} \cdots X_n^{\alpha_n} \mid \alpha_i \in \mathbb{N}\}$ of the commutative polynomial algebra $R = K[X_1, \ldots, X_n]$ in n variables $X_1, \ldots, X_n$ over a field K, and

then introducing a new multiplication $*$ on R, such that certain axioms ([8], AXIOMS 1.2) are satisfied. In the formal language of associative K-algebras, a solvable polynomial algebra can actually be defined as a finitely generated associative K-algebra $A = K[a_1, \ldots, a_n]$, that has the PBW K-basis $\mathcal{B} = \{a_1^{\alpha_1} \cdots a_n^{\alpha_n} \mid \alpha_i \in \mathbb{N}\}$ and a monomial ordering $\prec$ on $\mathcal{B}$ such that for $1 \leq i < j \leq n$, $a_j a_i = \lambda_{ji} a_i a_j + f_{ji}$ and $\mathbf{LM}(f_{ji}) \prec a_i a_j$, where $\lambda_{ji} \in K - \{0\}$, $f_{ji} \in K\text{-span}\mathcal{B}$ and $\mathbf{LM}(f_{ji})$ is the leading monomial of f_{ji} with respect to $\prec$ ([11], Definition 2.1). Full details on this definition will be recalled in the next section. In the literature, some results on the construction of solvable polynomial algebras by means of Gröbner bases for ideals in a free K-algebra $K\langle X\rangle = K\langle X_1, \ldots, X_n\rangle$ were given (see [8], Theorem 1.11; [9], CH.III, Proposition 2.2, Proposition 2.3; [10], Ch.4, Proposition 4.2), but a complete constructive characterization of solvable polynomial algebras has not been reached.

By employing Gröbner bases of ideals in free algebras, in this note we give a characterization of solvable polynomial algebras in terms of their defining relations (Section 2, Theorem 2.1), which shows that solvable polynomial algebras are completely determinable and constructible in a computational way.

Throughout this note, K denotes a field, $K^* = K - \{0\}$; $\mathbb{N}$ denotes the set of all nonnegative integers. Moreover, the Gröbner basis theory for ideals of free algebras is referred to [12] and [7].

2 The main result

We first briefly recall from ([8], [11], [9]) some basics concerning solvable polynomial algebras. Let $A = K[a_1, \ldots, a_n]$ be a finitely generated K-algebra with the set of generators $\{a_1, \ldots, a_n\}$. If, for some permutation $\tau = i_1 i_2 \cdots i_n$ of $1, 2, \ldots, n$, the set $\mathcal{B} = \{a^\alpha = a_{i_1}^{\alpha_1} \cdots a_{i_n}^{\alpha_n} \mid \alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n\}$ forms a K-basis of A, then $\mathcal{B}$ is referred to as a *PBW K-basis* of A. It is clear that if A has a PBW K-basis, then we can always assume that $i_1 = 1, \ldots, i_n = n$. Thus, we make the following convention once for all.

Convention From now on in this paper, if we say that the algebra A has the PBW K-basis $\mathcal{B}$, then it always means that

$$\mathcal{B} = \{a^\alpha = a_1^{\alpha_1} \cdots a_n^{\alpha_n} \mid \alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n\}.$$

Moreover, adopting the commonly used terminology in computational algebra, elements of $\mathcal{B}$ are referred to as *monomials* of A.

Suppose that A has the PBW K-basis $\mathcal{B}$ as presented above and that $\prec$ is a total ordering on $\mathcal{B}$. Then every nonzero element $f \in A$ has a unique expression $f = \lambda_1 a^{\alpha(1)} + \lambda_2 a^{\alpha(2)} + \cdots + \lambda_m a^{\alpha(m)}$ with $\lambda_j \in K^*$ and $a^{\alpha(j)} = a_1^{\alpha_{1j}} a_2^{\alpha_{2j}} \cdots a_n^{\alpha_{nj}} \in \mathcal{B}$, $1 \leq j \leq m$, in which $a^{\alpha(1)} \prec a^{\alpha(2)} \prec \cdots \prec a^{\alpha(m)}$. It follows that the *leading monomial*, the *leading coefficient*, and the *leading term* of f are respectively defined as $\mathbf{LM}(f) = a^{\alpha(m)}$, $\mathbf{LC}(f) = \lambda_m$, and $\mathbf{LT}(f) = \lambda_m a^{\alpha(m)}$.

Definition 1. *Suppose that the K-algebra $A = K[a_1, \ldots, a_n]$ has the PBW K-basis $\mathcal{B}$. If $\prec$ is a total ordering on $\mathcal{B}$ that satisfies the following three conditions:*

(1) *$\prec$ is a well-ordering;*

(2) *For any $a^\gamma, a^\alpha, a^\beta, a^\eta \in \mathcal{B}$, if $a^\alpha \prec a^\beta$ and $\mathbf{LM}(a^\gamma a^\alpha a^\eta)$, $\mathbf{LM}(a^\gamma a^\beta a^\eta) \notin K$, then $\mathbf{LM}(a^\gamma a^\alpha a^\eta) \prec \mathbf{LM}(a^\gamma a^\beta a^\eta)$;*

(3) *For any $a^\gamma, a^\alpha, a^\beta, a^\eta \in \mathcal{B}$, if $a^\beta \neq a^\gamma$, and $a^\gamma = \mathbf{LM}(a^\alpha a^\beta a^\eta)$, then $a^\beta \prec a^\gamma$ (thereby $1 \prec a^\gamma$ for all $a^\gamma \neq 1$),*

then $\prec$ is called a monomial ordering on $\mathcal{B}$ (or a monomial ordering on A).

Definition 2. *If the K-algebra $A = K[a_1, \ldots, a_n]$ satisfies the following two conditions:*

(S1) *A has the PBW K-basis $\mathcal{B}$;*

(S2) *There is a monomial ordering $\prec$ on $\mathcal{B}$ such that for all $a^\alpha = a_1^{\alpha_1} \cdots a_n^{\alpha_n}$, $a^\beta = a_1^{\beta_1} \cdots a_n^{\beta_n} \in \mathcal{B}$, $a^\alpha a^\beta = \lambda_{\alpha,\beta} a^{\alpha+\beta} + f_{\alpha,\beta}$, where $\lambda_{\alpha,\beta} \in K^*$, $a^{\alpha+\beta} = a_1^{\alpha_1+\beta_1} \cdots a_n^{\alpha_n+\beta_n}$, and either $f_{\alpha,\beta} = 0$ or $f_{\alpha,\beta} \in K$-span$\mathcal{B}$ with $\mathbf{LM}(f_{\alpha,\beta}) \prec a^{\alpha+\beta}$,*

then A is said to be a solvable polynomial algebra.

The results of the next proposition are summarized from ([8], Sections 2 – 5).

Proposition 2.1. *Let $A = K[a_1, \ldots, a_n]$ be a solvable polynomial algebra with the monomial ordering $\prec$ on the PBW K-basis $\mathcal{B}$ of A. The following statements hold.*

(i) *A is a (left and right) Noetherian domain.*

(ii) *Every left ideal I of A has a finite left Gröbner basis $\mathcal{G} = \{g_1, \ldots, g_t\}$ in the sense that if $0 \neq f \in I$, then there is some $g_i \in \mathcal{G}$ such that $\mathbf{LM}(g_i)|\mathbf{LM}(f)$, i.e., there is some $a^\gamma \in \mathcal{B}$ such that $\mathbf{LM}(f) = \mathbf{LM}(a^\gamma \mathbf{LM}(g_i))$, or equivalently, with $\gamma(i_j) = (\gamma_{i_1 j}, \gamma_{i_2 j}, \ldots, \gamma_{i_n j}) \in \mathbb{N}^n$, f has a left Gröbner representation:*

$$f = \sum_{i,j} \lambda_{ij} a^{\gamma(i_j)} g_j, \text{ where } \lambda_{ij} \in K^*, \ a^{\gamma(i_j)} \in \mathcal{B}, \ g_j \in \mathcal{G}, \text{ satisfying } \mathbf{LM}(a^{\gamma(i_j)} g_j) \preceq \mathbf{LM}(f) \text{ for all } (i,j).$$

(iii) *The Buchberger's Algorithm, that computes a finite Gröbner basis for a finitely generated commutative polynomial ideal, has a complete noncommutative version that computes a finite left Gröbner basis for a finitely generated left ideal $I = \sum_{i=1}^m Af_i$ of A.*

(iv) *Similar results of (ii) and (iii) hold for right ideals and two-sided ideals of A.*

□

It follows from Definition 2 that the two conditions (S1) and (S2) satisfied by a solvable polynomial algebra $A = K[a_1, \ldots, a_n]$ are completely *independent* factors. To reach the main result of this note, we also recall from the literature a constructive result for getting PBW bases.

Let $K\langle X\rangle = K\langle X_1, \ldots, X_n\rangle$ be the free K-algebra on $X = \{X_1, \ldots, X_n\}$ and $\mathbb{B} = \{1, \ X_{i_1} \cdots X_{i_s} \mid X_{i_j} \in X, \ s \geq 1\}$ the standard K-basis of $K\langle X\rangle$. For convenience, we use capital letters $U, V, W, S, \ldots$ to denote elements (monomials) of $\mathbb{B}$. Recall that a monomial ordering $\prec_X$ on $\mathbb{B}$ is a well-ordering such that for any $W, U, V, S \in \mathbb{B}$, $U \prec_X V$ implies $WU \prec_X WV$, $US \prec_X VS$ (or equivalently, $WUS \prec_X WVS$); and moreover, if $U \neq V$, then $V = WUS$ implies $U \prec_X V$ (thereby $1 \prec_X W$ for all $1 \neq W \in \mathbb{B}$). Let I be an ideal of $K\langle X\rangle$ and $\mathcal{G} \subset I$. If, with respect to some monomial ordering $\prec_X$ on $\mathbb{B}$, $\langle \mathbf{LM}(I)\rangle = \langle \mathbf{LM}(\mathcal{G})\rangle$, then $\mathcal{G}$ is said to be a Gröbner basis of I, where $\langle \mathbf{LM}(I)\rangle$ and $\langle \mathbf{LM}(\mathcal{G})\rangle$ are

respectively the ideals generated by the sets of leading monomials of I and $\mathcal{G}$. The reduced Gröbner basis of I is defined in a similar way as in the commutative case. Concerning the relation between Gröbner bases of I and the existence of a PBW K-basis for the quotient algebra $A = K\langle X\rangle/I$, the following result is a generalization of ([7], Proposition 2,14; [9], CH.III, Theorem 1.5).

Proposition 2.2. ([10], Ch.4, Theorem 3.1) *Let $A = K\langle X\rangle/I$ be as above. Suppose that I contains a subset of $\frac{n(n-1)}{2}$ elements*

$$G = \{g_{ji} = X_jX_i - F_{ji} \mid F_{ji} \in K\langle X\rangle,\ 1 \leq i < j \leq n\}$$

such that with respect to some monomial ordering $\prec_X$ on $\mathbb{B}$, $\mathbf{LM}(g_{ji}) = X_jX_i$ holds for all the g_{ji}, where $\mathbf{LM}(g_{ji})$ denotes the leading monomial of g_{ji} with respect to $\prec_X$. The following two statements are equivalent:

(i) *A has the PBW K-basis $\mathscr{B} = \{\overline{X}_1^{\alpha_1}\overline{X}_2^{\alpha_2}\cdots\overline{X}_n^{\alpha_n} \mid \alpha_j \in \mathbb{N}\}$, where each $\overline{X}_i$ denotes the coset of I represented by X_i in $K\langle X\rangle/I$.*

(ii) *Any subset $\mathcal{G}$ of I containing G is a Gröbner basis for I with respect to $\prec_X$.*

□

Remark 1. Obviously, Proposition 2.2 holds true if we use any permutation $\{X_{k_1}, \ldots, X_{k_n}\}$ of $\{X_1, \ldots, X_n\}$ (see an example given in the end of this note). So, in what follows we conventionally use only $\{X_1, \ldots, X_n\}$.

We note that if $G = \{g_{ji} = X_jX_i - F_{ji} \mid F_{ji} \in K\langle X\rangle,\ 1 \leq i < j \leq n\}$ is a Gröbner basis of the ideal I such that $\mathbf{LM}(g_{ji}) = X_jX_i$ for all the g_{ji}, then the *reduced Gröbner basis* of I is of the form

$$\mathcal{G} = \left\{ \begin{array}{rcl} g_{ji} & = & X_jX_i - \sum_q \mu_q^{ji} X_1^{\alpha_{1q}} X_2^{\alpha_{2q}} \cdots X_n^{\alpha_{nq}} \text{ with } \mu_q^{ji} \in K, \\ & & \text{and } \mathbf{LM}(g_{ji}) = X_jX_i,\ 1 \leq i < j \leq n \end{array} \right\}.$$

Bearing in mind Definition 2 and combining this fact, we are now able to present the main result of this note.

Theorem 2.1. *Let $A = K[a_1, \ldots, a_n]$ be a finitely generated algebra over the field K, and let $K\langle X\rangle = K\langle X_1, \ldots, X_n\rangle$ be the free K-algebras*

with the standard K-basis $\mathbb{B} = \{1,\ X_{i_1}\cdots X_{i_s} \mid X_{i_j} \in X,\ s \geq 1\}$. With notation as before, the following two statements are equivalent:

(i) *A is a solvable polynomial algebra in the sense of Definition 2.*

(ii) *$A \cong \overline{A} = K\langle X\rangle/I$ via the K-algebra epimorphism π_1: $K\langle X\rangle \to A$ with $\pi_1(X_i) = a_i$, $1 \leq i \leq n$, $I = Ker\pi_1$, satisfying*

(a) *with respect to some monomial ordering $\prec_X$ on $\mathbb{B}$, the ideal I has a finite Gröbner basis G and the reduced Gröbner basis of I is of the form*

$$\mathcal{G} = \left\{ \begin{array}{l} g_{ji} = X_jX_i - \lambda_{ji}X_iX_j - F_{ji} \text{ with } \lambda_{ji} \in K^*, \\ \quad F_{ji} = \sum_q \mu_q^{ji} X_1^{\alpha_{1q}} X_2^{\alpha_{2q}} \cdots X_n^{\alpha_{nq}}, \mu_q^{ji} \in K, \\ \quad \text{and } \mathbf{LM}(g_{ji}) = X_jX_i,\ 1 \leq i < j \leq n \end{array} \right\},$$

thereby $\mathscr{B} = \{\overline{X}_1^{\alpha_1}\overline{X}_2^{\alpha_2}\cdots\overline{X}_n^{\alpha_n} \mid \alpha_j \in \mathbb{N}\}$ forms a PBW K-basis for $\overline{A}$, where each $\overline{X}_i$ denotes the coset of I represented by X_i in $\overline{A}$; and

(b) *there is a monomial ordering $\prec$ on $\mathscr{B}$ such that*

$$\mathbf{LM}(\overline{F}_{ji}) \prec \overline{X}_i\overline{X}_j \text{ whenever } \overline{F}_{ji} \neq 0,$$

where $\overline{F}_{ji} = \sum_q \mu_q^{ji}\overline{X}_1^{\alpha_{1i}}\overline{X}_2^{\alpha_{2i}}\cdots\overline{X}_n^{\alpha_{ni}}$, $1 \leq i < j \leq n$.

Proof. (i) $\Rightarrow$ (ii) Let $\mathcal{B} = \{a^\alpha = a_1^{\alpha_1}\cdots a_n^{\alpha_n} \mid \alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n\}$ be the PBW K-basis of the solvable polynomial algebra A and $\prec$ a monomial ordering on $\mathcal{B}$. By Definition 2, the generators of A satisfy the relations:

$$a_ja_i = \lambda_{ji}a_ia_j + f_{ji}, \quad 1 \leq i < j \leq n, \tag{$*$}$$

where $\lambda_{ji} \in K^*$ and $f_{ji} = \sum_q \mu_q^{ji} a^{\alpha(q)} \in K\text{-span}\mathcal{B}$ with $\mathbf{LM}(f_{ji}) \prec a_ia_j$. Consider in the free K-algebra $K\langle X\rangle = K\langle X_1, \ldots, X_n\rangle$ the subset

$$\mathcal{G} = \{g_{ji} = X_jX_i - \lambda_{ji}X_iX_j - F_{ji} \mid 1 \leq i < j \leq n\},$$

where if $f_{ji} = \sum_q \mu_q^{ji} a_1^{\alpha_{1q}}\cdots a_n^{\alpha_{nq}}$, then $F_{ji} = \sum_q \mu_q^{ji} X_1^{\alpha_{1q}}\cdots X_n^{\alpha_{nq}}$ for $1 \leq i < j \leq n$. We write $J = \langle\mathcal{G}\rangle$ for the ideal of $K\langle X\rangle$ generated by $\mathcal{G}$ and put $\overline{A} = K\langle X\rangle/J$. Let π_1: $K\langle X\rangle \to A$ be the K-algebra epimorphism with $\pi_1(X_i) = a_i$, $1 \leq i \leq n$, and let π_2: $K\langle X\rangle \to \overline{A}$ be the

canonical algebra epimorphism. It follows from the universal property of the canonical homomorphism that there is an algebra epimorphism φ: $\overline{A} \to A$ defined by $\varphi(\overline{X}_i) = a_i$, $1 \leq i \leq n$, such that the following diagram of algebra homomorphisms is commutative:

$$\begin{array}{ccc} K\langle X\rangle & \xrightarrow{\pi_2} & \overline{A} \\ {\scriptstyle \pi_1}\big\downarrow & \swarrow \varphi & \\ A & & \end{array} \qquad \varphi \circ \pi_2 = \pi_1 \ .$$

On the other hand, by the definition of each g_{ji} we see that every element $\overline{H} \in \overline{A}$ may be written as $\overline{H} = \sum_j \mu_j \overline{X}_1^{\beta_{1j}} \overline{X}_2^{\beta_{2j}} \cdots \overline{X}_n^{\beta_{nj}}$ with $\mu_j \in K$ and $(\beta_{1j}, \ldots, \beta_{nj}) \in \mathbb{N}^n$, where each $\overline{X}_i$ is the coset of J represented by X_i in $\overline{A}$. Noticing the relations presented in $(*)$, it is straightforward to check that the correspondence

$$\begin{array}{rccc} \psi: & A & \longrightarrow & \overline{A} \\ & \displaystyle\sum_i \lambda_i a_1^{\alpha_{1i}} \cdots a_n^{\alpha_{ni}} & \mapsto & \displaystyle\sum_i \lambda_i \overline{X}_1^{\alpha_{1i}} \cdots \overline{X}_n^{\alpha_{ni}} \end{array}$$

is an algebra homomorphism such that $\varphi \circ \psi = 1_A$ and $\psi \circ \varphi = 1_{\overline{A}}$, where 1_A and $1_{\overline{A}}$ denote the identity maps of A and $\overline{A}$ respectively. This shows that $A \cong \overline{A}$, thereby $\mathrm{Ker}\pi_1 = I = J$; moreover, $\mathscr{B} = \{\overline{X}_1^{\alpha_1} \overline{X}_2^{\alpha_2} \cdots \overline{X}_n^{\alpha_n} \mid \alpha_j \in \mathbb{N}\}$ forms a PBW K-basis for $\overline{A}$, and $\prec$ is a monomial ordering on $\mathscr{B}$.

We next show that $\mathcal{G}$ forms the reduced Gröbner basis for I as described in (a). To this end, we first show that the monomial ordering $\prec$ on $\mathcal{B}$ induces a monomial ordering $\prec_X$ on the standard K-basis $\mathbb{B}$ of $K\langle X\rangle$. For convenience, as before we use capital letters $U, V, W, S, \ldots$ to denote elements (monomials) in $\mathbb{B}$. We also fix a graded lexicographic ordering $\prec_{grlex}$ on $\mathbb{B}$ (with respect to a fixed positively weighted gradation of $K\langle X\rangle$) such that

$$X_1 \prec_{grlex} X_2 \prec_{grlex} \cdots \prec_{grlex} X_n.$$

Then, for $U, V \in \mathbb{B}$ we define

$$U \prec_X V \text{ if } \begin{cases} \mathbf{LM}(\pi_1(U)) \prec \mathbf{LM}(\pi_1(V)), \\ \text{or} \\ \mathbf{LM}(\pi_1(U)) = \mathbf{LM}(\pi_1(V)) \text{ and } U \prec_{grlex} V. \end{cases}$$

Since A is a domain (Proposition 2.1(i)) and π_1 is an algebra homomorphism with $\pi_1(X_i) = a_i$ for $1 \leq i \leq n$, it follows that $\mathbf{LM}(\pi_1(W)) \neq 0$ for all $W \in \mathbb{B}$. We also note from Definition 2 that if $f, g \in A$ are nonzero elements, then $\mathbf{LM}(fg) = \mathbf{LM}(\mathbf{LM}(f)\mathbf{LM}(g))$. Thus,

if $U, V, W \in \mathbb{B}$ and $U \prec_X V$ subject to $\mathbf{LM}(\pi_1(U)) \prec \mathbf{LM}(\pi_1(V))$, then

$$\begin{aligned} \mathbf{LM}(\pi_1(WU)) &= \mathbf{LM}(\mathbf{LM}(\pi_1(W))\mathbf{LM}(\pi_1(U))) \\ &\prec \mathbf{LM}(\mathbf{LM}(\pi_1(W))\mathbf{LM}(\pi_1(V))) \\ &= \mathbf{LM}(\pi_1(WV)) \end{aligned}$$

implies $WU \prec_X WV$;

if $U, V, W \in \mathbb{B}$ and $U \prec_X V$ subject to $\mathbf{LM}(\pi_1(U)) = \mathbf{LM}(\pi_1(V))$ and $U \prec_{grlex} V$, then

$$\begin{aligned} \mathbf{LM}(\pi_1(WU)) &= \mathbf{LM}(\mathbf{LM}(\pi_1(W))\mathbf{LM}(\pi_1(U))) \\ &= \mathbf{LM}(\mathbf{LM}(\pi_1(W))\mathbf{LM}(\pi_1(V))) \\ &= \mathbf{LM}(\pi_1(WV)), \end{aligned}$$

and $WU \prec_{grlex} WV$ implies $WU \prec_X WV$.

Similarly, if $U \prec_X V$, then $US \prec_X VS$ for all $S \in \mathbb{B}$. Moreover, if $W, U, V, S \in \mathbb{B}$, $W \neq V$, such that $W = UVS$, then $\mathbf{LM}(\pi_1(W)) = \mathbf{LM}(\pi_1(UVS))$ and clearly $V \prec_{grlex} W$, thereby $V \prec_X W$. Since $\prec$ is a well-ordering on $\mathcal{B}$ and $\prec_{grlex}$ is a well-ordering on $\mathbb{B}$, the above argument shows that $\prec_X$ is a monomial ordering on $\mathbb{B}$. With this monomial ordering $\prec_X$ in hand, by the definition of F_{ji} we see that $\mathbf{LM}(F_{ji}) \prec_X X_iX_j$. Furthermore, since $\mathbf{LM}(\pi_1(X_jX_i)) = a_ia_j = \mathbf{LM}(\pi_1(X_iX_j))$ and $X_iX_j \prec_{grlex} X_jX_i$, we see that $X_iX_j \prec_X X_jX_i$. It follows that $\mathbf{LM}(g_{ji}) = X_jX_i$ for $1 \leq i < j \leq n$. Now, by Proposition 2.2 we conclude that $\mathcal{G}$ forms a Gröbner basis for I with respect to $\prec_X$. Finally,

by the definition of $\mathcal{G}$, it is clear that $\mathcal{G}$ is the reduced Gröbner basis of I with respect to $\prec_X$, as desired.

(ii) $\Rightarrow$ (i) Note that (a) + (b) tells us that the generators of $\overline{A}$ satisfy the relations $\overline{X}_j\overline{X}_i = \lambda_{ji}\overline{X}_i\overline{X}_j + \overline{F}_{ji}$, $1 \leq i < j \leq n$, and that if $\overline{F}_{ji} \neq 0$, then $\mathbf{LM}(\overline{F}_{ji}) \prec \overline{X}_i\overline{X}_j$ with respect to the given monomial ordering $\prec$ on $\mathscr{B}$. It follows that $\overline{A}$ and hence A is a solvable polynomial algebra in the sense of Definition 2. □

Remark 2. The monomial ordering $\prec_X$ we defined in the proof of Theorem 2.1 is a modification of the *lexicographic extension* defined in [5]. But our definition of $\prec_X$ involves a graded monomial ordering $\prec_{grlex}$ on the standard K-basis $\mathbb{B}$ of the free K-algebra $K\langle X\rangle = K\langle X_1, \ldots, X_n\rangle$. The reason is that *the monomial ordering $\prec_X$ on $\mathbb{B}$ must be compatible with the usual rule of division*, namely, $W, U, V, S \in \mathbb{B}$, $W \neq V$, and $W = UVS$ implies $V \prec_X W$. While it is clear that if we use any lexicographic ordering $\prec_{lex}$ in the definition of $\prec_X$, then this rule will not work in general.

We end this note by an example illustrating Theorem 2.1, in particular, illustrating that the monomial ordering $\prec_X$ used in the condition (a) and the monomial ordering $\prec$ used in the condition (b) may be mutually independent, namely $\prec$ may not necessarily be the restriction of $\prec_X$ on $\mathscr{B}$, and the choice of $\prec$ is indeed quite flexible.

Example 1. Considering the $\mathbb{N}$-graded structure of the free K-algebra $K\langle X\rangle = K\langle X_1, X_2, X_3\rangle$ by assigning X_1 the degree 2, X_2 the degree 1 and X_3 the degree 4, let I be the ideal of $K\langle X\rangle$ generated by the elements

$$\begin{aligned} g_1 &= X_1X_2 - X_2X_1, \\ g_2 &= X_3X_1 - \lambda X_1X_3 - \mu X_3X_2^2 - f(X_2), \\ g_3 &= X_3X_2 - X_2X_3, \end{aligned}$$

where $\lambda \in K^*$, $\mu \in K$, $f(X_2)$ is a polynomial in X_2 which has degree ≤ 6, or $f(X_2) = 0$. The following properties hold.

(1) If we use the graded lexicographic ordering $X_2 \prec_{grlex} X_1 \prec_{grlex} X_3$ on $K\langle X\rangle$, then the three generators have the leading monomials $\mathbf{LM}(g_1) = X_1X_2$, $\mathbf{LM}(g_2) = X_3X_1$, and $\mathbf{LM}(g_3) = X_3X_2$. It is

straightforward to verify that $\mathcal{G} = \{g_1, g_2, g_3\}$ forms a Gröbner basis for I.

(2) With respect to the fixed $\prec_{grlex}$ in (1), the reduced Gröbner basis $\mathcal{G}'$ of I consists of

$$\begin{aligned} g_1 &= X_1X_2 - X_2X_1, \\ g_2 &= X_3X_1 - \lambda X_1X_3 - \mu X_2^2X_3 - f(X_2), \\ g_3 &= X_3X_2 - X_2X_3, \end{aligned}$$

(3) Writing $A = K[a_1, a_2, a_3]$ for the quotient algebra $K\langle X\rangle/I$, where a_1, a_2 and a_3 denote the cosets $X_1 + I$, $X_2 + I$ and $X_3 + I$ in $K\langle X\rangle/I$ respectively, it follows that A has the PBW basis $\mathcal{B} = \{a^\alpha = a_2^{\alpha_2}a_1^{\alpha_1}a_3^{\alpha_3} \mid \alpha = (\alpha_2, \alpha_1, \alpha_3) \in \mathbb{N}^3\}$. Noticing that $a_2a_1 = a_1a_2$, it is clear that $\mathcal{B}' = \{a^\alpha = a_1^{\alpha_1}a_2^{\alpha_2}a_3^{\alpha_3} \mid \alpha = (\alpha_1, \alpha_2, \alpha_3) \in \mathbb{N}^3\}$ is also a PBW basis for A. Since $a_3a_1 = \lambda a_1a_3 + \mu a_2^2a_3 + f(a_2)$, where $f(a_2) \in K$-span$\{1, a_2, a_2^2, \ldots, a_2^6\}$, we see that A has the monomial ordering $\prec_{lex}$ on $\mathcal{B}'$ such that $a_3 \prec_{lex} a_2 \prec_{lex} a_1$ and $\mathbf{LM}(\mu a_2^2a_3 + f(a_2)) \prec_{lex} a_1a_3$, thereby A is turned into a solvable polynomial algebra with respect to $\prec_{lex}$.

Moreover, one easily checks that if a_1 is assigned the degree 2, a_2 is assigned the degree 1 and a_3 is assigned the degree 4, then A has another monomial ordering on $\mathcal{B}'$, namely the graded lexicographic ordering $\prec_{grlex}$ such that $a_3 \prec_{grlex} a_2 \prec_{grlex} a_1$ and $\mathbf{LM}(\mu a_2^2a_3 + f(a_2)) \prec_{grlex} a_1a_3$, thereby A is turned into a solvable polynomial algebra with respect to $\prec_{grlex}$.

References

[1] J. Apel, W. Lassner. *An extension of Buchberger's algorithm and calculations in enveloping fields of Lie algebras.* J. Symbolic Comput., 6, 1988, pp. 361–370.

[2] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen polynomideal*, PhD thesis, University of Innsbruck, 1965.

[3] B. Buchberger. *Gröbner bases: An algorithmic method in polynomial ideal theory.* In: Bose, N.K. (ed.) Multidimensional Systems Theory. Reidel Dordrecht, 1985, pp. 184–232.

[4] W. Decker, G.-M. Greuel, G. Pfister, H. Schönemann. SINGULAR 3-1-3 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de(2011).

[5] D. Eisenbud, I. Peeva, B. Sturmfels. *Non-commutative Gröbner bases for commutative algebras.* Proc. Amer. Math. Soc., 126, 1998, pp. 687-691.

[6] A. Galligo. *Some algorithmic questions on ideals of differential operators.* Proc. EUROCAL'85, LNCS 204, 1985, pp. 413–421.

[7] E. L. Green. *Noncommutative Grobner bases and projective resolutions.* In: Michler, Schneider (eds) Proceedings of the Euroconference, Computational Methods for Representations of Groups and Algebras, Essen, 1997. Progress in Mathematics, Vol. 173, Basel, Birkhauser Verlag, 1999, pp. 29–60.

[8] A. Kandri-Rody, V. Weispfenning. *Non-commutative Gröbner bases in algebras of solvable type.* J. Symbolic Comput., 9, 1990, pp. 1–26.

[9] H. Li. *Noncommutative Gröbner Bases and Filtered-Graded Transfer.* Lecture Notes in Mathematics, Vol. 1795, Springer-Verlag, Berlin, 2002.

[10] H. Li. *Gröbner Bases in Ring Theory.* World Scientific Publishing Co., 2011.

[11] H. Li, Y. Wu. *Filtered-graded transfer of Gröbner basis computation in solvable polynomial algebras.* Communications in Algebra, 28(1), 2000, pp. 15–32.

[12] T. Mora. *An introduction to commutative and noncommutative Gröbner Bases.* Theoretic Computer Science, 134, 1994, pp. 131–173.

Huishi Li Received December 17, 2013

Huishi Li
Department of Applied Mathematics,
College of Information Science and Technology
Hainan University, Haikou 570228, China
E–mail: *huishipp@yahoo.com*

Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Dynamic Object Identification with SOM-based neural networks

Aleksey Averkin, Veaceslav Albu, Sergey Ulyanov, Ilya Povidalo

Abstract

In this article a number of neural networks based on self-organizing maps, that can be successfully used for dynamic object identification, is described. Unique SOM-based modular neural networks with vector quantized associative memory and recurrent self-organizing maps as modules are presented. The structured algorithms of learning and operation of such SOM-based neural networks are described in details, also some experimental results and comparison with some other neural networks are given.

Keywords: Neural networks; forecasting; timeseries prediction; dynamic object identification.

1 Introduction

Identification theory solves problems of constructing mathematical models of dynamic systems according to the observations of their behaviour. The object identification step is one of the most important steps while constructing mathematical models of objects or processes. The quality of the model relies on this step and, therefore, the quality of control, that is based on this model, or results of a research with this model also rely on this step.

Dynamic object identification is one of the basic problems which could be solved using neural networks [1] along with different other methods. Object identification is complicated if noises are present in the source data, some of the object parameters change according to unknown laws or the exact number of the object parameters is unknown.

In such cases neural network can be applied for dynamic object identification. There are a lot of different types of neural networks that can be used for dynamic object identification.

Among different neural network architectures applicable for dynamic object identification a class of neural networks based on self-organizing maps (SOM) can be noted. Neural networks of such type will be given special attention in this article due to their wider spread and successful application in solving different kinds of problems [2,3] including problems of forecasting and identification. A number of biomorphic neural networks, architecture of which is the result of studying the structure of the cerebral cortex of mammals, will also be considered.

2 SOM-based neural networks that can be used for dynamic object identification

2.1 Problem definition

Identification of a dynamic object that receives a vector of input parameters $u(t)$ at time t and gives an output vector $y(t)$ can be described as finding the type of a model of this object, that has an output $\hat{y}(t)$ and finding parameters of this model such that minimize error $e = \| y(t) - \hat{y}(t) \|^2$ of this model (see figure 1).

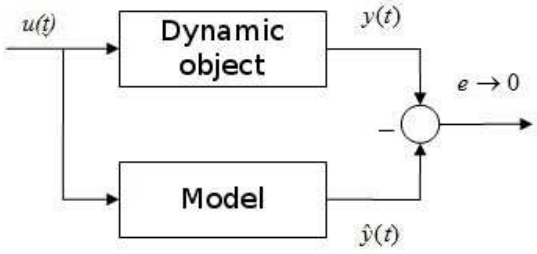


Figure 1. Dynamic object identification scheme

Suppose that there is a sequence of vectors of input parameters $u(t)$, $t\epsilon[0,T]$ and a sequence of output vectors $y(t)$, $t\epsilon[0,T]$, where T

– number of input-output pairs. We will consider the solution of the object identification problem as definition of the type of the function f that will define the model of identified object:

$$\hat{y}(t) = f[y(t-1), ..., y(t-n_y), u(t), u(t-1), ..., u(t-n_u)], \quad (1)$$

where $\hat{y}(t)$ is vector of output parameters of the model. At a single moment of time the input of the model takes current known measured values of the input parameters along with $n_u < T$ previous values and $n_y < T$ previous output parameters of the identified object.

Also we will consider solution of this problem in the following form:

$$\hat{y}(t) = f[\hat{y}(t-1), ..., \hat{y}(t-n_y), u(t), u(t-1), ..., u(t-n_u)], \quad (2)$$

where $\hat{y}(0) = y(0)$, $\hat{y}(t)$ is vector of output parameters of the model, $n_u < T$. This identification scheme has recurrent connections and at a single moment of time the input of the model takes current known measured values of the input parameters of the object along with $n_u < T$ previous values of these parameters and $n_y < T$ previous output parameters of the model.

2.2 Vector quantized temporal associative memory (VQTAM)

VQTAM is a modification of self-organizing maps which can be used for identification of dynamic objects [4,5]. The input vector $u(t)$ of this network is split into two parts: $x^{in}(t)$, $x^{out}(t)$. The first part of the input vector $x^{in}(t)$ contains information about the inputs of the dynamic object and its outputs at previous time steps. The second part of the input vector $x^{out}(t)$ contains information about the expected output of the dynamic object corresponding to the input $x^{in}(t)$. The weights vector is also split into two parts in a similar way [4]. Thus $x(t) = \begin{pmatrix} x^{in}(t) \\ x^{out}(t) \end{pmatrix}$ and $w_i(t) = \begin{pmatrix} w_i^{in}(t) \\ w_i^{out}(t) \end{pmatrix}$, where $w_i(t)$ is weights vector of the i-th neuron, $w_i^{in}(t)$ is the part of the weights vector that contains information on the inputs, and $w_i^{out}(t)$ is the part of the weights vector

that contains information on the outputs. The first part of the input vector contains information on the process inputs and its previous outputs:

$$x^{in}(t) = (y(t-1), ..., y(t-n_y), u(t), u(t-1), ..., u(t-n_u)), \quad (3)$$

where $n_u < T$, $n_y < T$. The second part of the input vector $x^{out} = y(t)$ contains information on the expected output of the process corresponding to the inputs $x^in(t)$.

Each vector in the learning sample consists of a pair of vectors $(y(t), u(t))$ and the sample should contain not less than $max(n_u, n_y)$ vectors. Vectors $y(t)$ are the measured output parameters of the process at time step t, and $u(t)$ are the input parameters of the process at the same time step.

After presenting a subsequent input vector $x(t)$, combined of several vectors from the learning sample, to the network the winner neuron is determined only by the $x^{in}(t)$ part of the vector:

$$i^*(t) = \arg\min_i \left\{ \| \, x^{in}(t) - w_i^{in}(t) \, \| \right\}, \quad (4)$$

where $i^*(t)$ is a number of the winner neuron at time step t.

For weight modification a modified SOM weight modification rule is used:

$$\begin{aligned} \Delta w_i^{in}(t) &= \alpha(t) h(i^*, i, t)[x^{in}(t) - x_i^{in}(t)], \\ \Delta w_i^{out}(t) &= \alpha(t) h(i^*, i, t)[x^{out}(t) - w_i^{out}(t)], \end{aligned} \quad (5)$$

where $0 < \alpha(t) < 1$ is a learning rate of the network, h is neighbourhood function of the i-th and i^*-th neurons. For example a Gaussian function can be chosen as a neighbourhood function $h(i^*, i, t)$:

$$h(i^*, i, t) = \exp\left(-\frac{\| \, r_i(t) - r_{i^*}(t) \, \|^2}{2\sigma^2(t)} \right), \quad (6)$$

where $r_i(t)$ and $r_{i^*}(t)$ are positions on the map of the i-th and i^*-th neurons, $\sigma(t) > 0$ determines the radius of the neighbourhood function at time step t. When the winner neuron i^* is defined, the output of the network is set to $w_{i^*}^{out}(t)$.

On the test sample VQTAM's input takes only $x^{in}(t)$ part of the input, which is used to define the winner neuron, and the output of the network is set to $w_{i^*}^{out}(t)$. Vector $w_{i^*}^{out}(t)$ can be interpreted as the predicted output $\hat{y}(t)$ of the dynamic object at time step t.

Learning algorithm of VQTAM network is similar to regular SOM algorithm:

1. Weights are initialized with random values or values from the training sample.

2. A vector from the sample is presented to the network and the winner neuron is identified according to equation 4.

3. Weights are modified according to the rule 5.

4. Steps 2 and 3 are repeated for each vector from the sample.

5. Steps 2 – 4 are repeated for the sample several times until a specified number of epochs has passed or a desired accuracy of identification has been reached.

On a test sample after presenting an input vector containing only the first $x^{in}(t)$ part of the input vector a winner neuron is identified and the output of the network (modeled output of the object) is set equal to the second part of the weight vector of the winner neuron $w_{i^*}^{out}(t)$. The output can be also identified as average between several $w_i^{out}(t)$ of best-matching neurons.

2.3 Recurrent self-organizing map (RSOM)

In RSOM unlike conventional SOMs with recurrent connections, a decaying in time vector of outputs is introduced for each neuron. This vector is used to determine the winner neuron and is used in maps weights modification [6, 7].

The network inputs vector $x(t)$ is represented as follows:

$$x(t) = (y(t-1), ..., y(t-n_y), u(t), u(t-1), ..., u(t-n_u))), \quad (7)$$

where $n_u < T$, $n_y < T$.

Output of each neuron is determined according to the following equation:

$$V_i(t) = \| \nu_i(t) \|, \tag{8}$$

where $\nu_i(t) = (1-\alpha)\nu_i(t-1) + \alpha(x(t) - w_i(t))$, α is the output decay factor ($0 < \alpha \leqslant 1$), $V_i(t)$ is output of the i-th neuron at time step t, $w_i(t)$ is weights vector of the i-th neuron. Further in the article it will be shown, that a neuron with defined like this output is close to definition of a chaotic neuron, and also some benefits of this approach will be described.

After presenting a subsequent input vector to the network a winner neuron is determined as the neuron with minimal output [7]:

$$i^*(t) = \arg\min_i \{V_i(t)\}. \tag{9}$$

To modify the weights a modified conventional SOM rule is used:

$$\Delta w_i(t) = \alpha(n)h(i^*, i, t)\nu_i(t), \tag{10}$$

where $0 < \alpha(t) < 1$ is learning rate, h is neighbourhood function of i-th and i^*-th neurons.

When the learning process is complete, the network is presented again with the learning sample and clusterizes it. Each cluster can be approximated with an individual model, for example a linear function $f_i(t)$ for the i-th cluster. Thus, after presenting the learning sample a linear function is defined for each vector of this sample. These functions can be used to determine the output value at the next time step.

This process can be speeded up by using algorithms for constructing local linear models while training the neural network. Each neuron of the RSOM network is associated with a matrix $A_i(t)$ that contains coefficients of the corresponding linear model:

$$A_i(t) = \left[b_{i,1}(t), ..., b_{i,n_y}(t), a_{i,1}(t), ..., a_{i,n_u}(t)\right]^T, \tag{11}$$

The output value of the network is defined according to the following equation:

$$\hat{y}(t) = \sum_{k=1}^{n_y} b_{i^*,k}(t)u(t-k) + \sum_{l=1}^{n_u} a_{i^*,i}(t)y(t-l) = A_{i^*}^T(t)x(t), \qquad (12)$$

where $A_{i^*}(t)$ is the matrix of coefficients associated with the winner neuron $i^*(t)$. Matrix $A_{i^*}(t)$ is used for linear approximation of model's output.

While constructing the local linear models simultaneously with training of the neural network an additional rule to modify the coefficients of the linear model is needed:

$$A_i(t+1) = A_i(t) + \beta h(i^*, i, t)\Delta A_i(t), \qquad (13)$$

where $0 < \beta < 1$ is learning rate of the model, $\Delta A_i(t)$ is Widrow-Hoff's rule for error correction:

$$\Delta A_i(t) = \left[y(t) - A_i^T(t)x(t)\right] \frac{x(t)}{\parallel x(t) \parallel^2}, \qquad (14)$$

where $y(t)$ is desired output of the model for the $x(t)$ input.

Thus, at each step of the network training a modification of the model coefficients is performed along with modification of the weights of neurons. On the test sample after an input vector is presented to the network a winner neuron $i^*(t)$ is chosen. Then a corresponding coefficients matrix $A_{i^*}(t)$ of the linear model is calculated. Using the determined matrix the output of the model is defined by the equation: $\hat{y}(t) = A_{i^*}^T(t)x(t)$.

So, learning algorithm of RSOM network is similar to SOM training algorithm but with some differences:

1. Weights are initialized with random values or values from the training sample.

2. Parameters of the local models assigned to neurons are initialized with random values.

3. A vector from the sample is presented to the network, outputs of all neurons are calculated according to (8) and the winner neuron is identified by (9).

4. Weights are modified according to the rule (10).

5. Local models parameters are modified according to (13).

6. Steps 3 – 5 are repeated for each vector from the sample.

7. Steps 3 – 6 are repeated for the sample several times until a specified number of epochs has passed or a desired accuracy of identification has been reached.

On a test sample after presenting an input vector, outputs of all neurons are calculated according to (8), the winner neuron is identified and the corresponding local model is chosen and the resulting modelled output is determined by equation (12).

2.4 Modular self-organizing maps

Modular self-organizing maps are presented in Tetsuo Furukava's works [8, 9]. Modular SOM has a structure of an array which consists of functional modules that are actually trainable neural networks (see figure 2), such as multilayer perceptrons (MLP), but not a vector, as in conventional self-organizing maps. In case of MLP-modules modular self-organizing map finds features or correlations in input and output values simultaneously building a map of their similarity. Thus, a modular self-organizing map with MLP modules is a self-organizing map in a function space but not in a vector space [9].

These neural network structures can be considered biomorphic, as their emergence is due to research of the brain structure of mammals [10], and confirmed by a number of further studies [11]. The basis of the idea of the cerebral cortex structure is a model of cellular structure, where each cell is a collection of neurons, a neural column. Columns of neurons are combined in a more complex structures. In this regard it was suggested to model the individual neural columns with neural

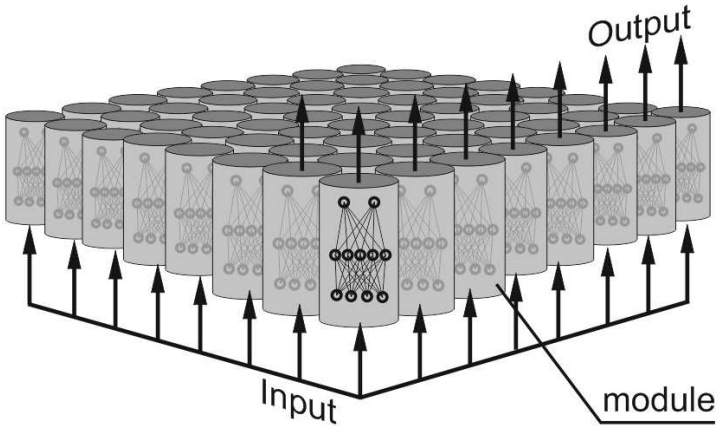


Figure 2. Modular network structure

networks [11]. This idea has formed the basis of the modular neural networks.

In fact, the modular self-organizing map is a common SOM, where neurons are replaced by more complex and autonomous entities such as other neural networks. Such replacement requires a slight modification of the learning algorithm. In the proposed by [9] algorithm at the initial stage the network receives the i-th sample of the input data corresponding to I functions, which will be mapped by the network, and the error is calculated for each network module:

$$E_i^k = \frac{1}{J}\sum_{j=1}^{J} \parallel \hat{y}_{i,j}^k - y_{i,j} \parallel^2, \tag{15}$$

where k is module number, for which the error is calculated, J is number of vectors in the sample, $\hat{y}_j^k$ is output of the k-th module, y_j is expected output of the network on the suggested set of input data. Winner module is calculated as the module that minimizes the error E^k:

$$k_i^* = \arg\min_k E_i^k. \tag{16}$$

As soon as the winner module is defined, the adaptation process

takes place and the weights of the module are being modified according to one of learning algorithms suitable for the networks of this type, after that the weights of the main SOM are being modified. In this process parameters (weights) of each module are considered as the weights of the SOM and are modified according to standard learning algorithms suitable for conventional SOMs.

In this study SOMxVQTAM and SOMxRSOM networks were developed, which are SOMs with modules of VQTAM type and SOMs with modules of RSOM type respectively. Further some application results of such networks will be reviewed.

3 Using SOM-based neural networks for dynamic object identification

For some experiments and comparisons of the algorithms the neural networks of types VQTAM, RSOM, SOMxVQTAM and SOMxRSOM were tested on samples that were used in 2006 – 2007 to identify the winners at neural networks forecasting competition [14]. Results of these competitions were used in this study as there is a detailed description of the place definition method used for all competitors. Also a fair amount of different algorithms took part in this competion and there was a description for the most of those algorithms as well as the learning and testing samples, which allowed a comparative analysis of the neural networks described in this paper with other advanced algorithms. To determine the place in the table, the organizers of the competition [14] suggested to forecast 18 steps for each of the 111 samples. For each of the resulting predictions a symmetric mean absolute percentage error ($SMAPE$) was calculated:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{\parallel y(t) - \hat{y}(t) \parallel}{(y(t) + \hat{y}(t))/2} * 100, \tag{17}$$

where $y(t)$ is the real state of the object at time step t, $\hat{y}(t)$ is the output of the model at time step t, n is the number of vectors in the

testing sample (18 for this competition). Then the place in the table was determined from the average error for all of 111 samples.

Each of the 111 samples had different lengths (from 51 to 126 points in the training samples), those samples represented a monthly measure of several macroeconomic indicators.

Each of the algorithms described in this article were launched with the same parameters for all of 111 samples, that is the parameters for each network were set only once before presenting the set of all 111 samples, but not set individually for each of the 111 samples, which lead to some not very successful forecasts that led to growth of the average error. Despite this fact algorithms could accurately predict the future values, as it is clearly seen from the results table (see Table 1). The $SMAPE$ error of the forecast for most of the 111 samples was lower than 5% (for more than 70% of all samples) but the average error grew due to unsuccessful forecasts with error values up to 60%. The above error can be greatly reduced if the automatic tuning of network parameters for each of the samples would be applied.

Table 1: Results table for different forecasting algorithms

Num.	Algorithm name	SMAPE
1	Stat. Contender - Wildi	14,84%
2	Stat. Benchmark - Theta Method (Nikolopoulos)	14,89%
3	Illies, Jager, Kosuchinas, Rincon, Sakenas, Vaskevcius	15,18%
4	Stat. Benchmark - ForecastPro (Stellwagen)	15,44%
5	CI Benchmark - Theta AI (Nikolopoulos)	15,66%
6	Stat. Benchmark - Autobox (Reilly)	15,95%
7	Adeodato, Vasconcelos, Arnaud, Chunha, Monteiro	16,17%
8	Flores, Anaya, Ramirez, Morales	16,31%
9	Chen, Yao	16,55%
10	D'yakonov	16,57%
11	Kamel, Atiya, Gayar, El-Shishiny	16,92%

Continuation of Table 1

Num.	Algorithm name	SMAPE
12	Abou-Nasr	17,54%
13	Theodosiou, Swamy	17,55%
–	**VQTAM**	17,61%
–	**SOMxVQTAM**	17,70%
14	CI Benchmark - Naive MLP (Crone)	17,84%
–	**RSOM**	17,94%
15	de Vos	18,24%
16	Yan	18,58%
17	CI Benchmark - Naive SVR (Crone, Pietsch)	18,60%
18	C49	18,72%
19	Perfilieva, Novak, Pavliska, Dvorak, Stepnicka	18,81%
20	Kurogi, Koyama, Tanaka, Sanuki	19,00%
21	Stat. Contender - Beadle	19,14%
22	Stat. Contender - Lewicke	19,17%
23	Sorjamaa, Lendasse	19,60%
24	Isa	20,00%
25	C28	20,54%
26	Duclos-Gosselin	20,85%
–	**SOMxRSOM**	21,64%
27	Stat. Benchmark - Naive	22,69%
28	Papadaki, Amaxopolous	22,70%
29	Stat. Benchmark - Hazarika	23,72%
30	C17	24,09%
31	Stat. Contender - Njimi, Melard	24,90%
32	Pucheta, Patino, Kuchen	25,13%
33	Corzo, Hong	27,53%

3.1 Example of learning on one of the samples

For example, take one of the 111 samples and see the results of the identification done by all four types of the described neural networks. The original sample is shown in Figure 3, the test 18 points of the

sample are separated from the learning sample with a vertical dashed line.

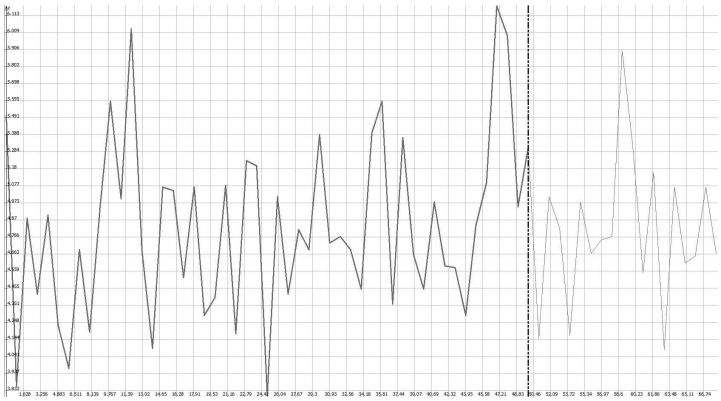


Figure 3. The original sample divided into learning and test samples with a vertical dashed line

In Figure 4 the results of testing VQTAM network on the last 18 points of the sample are overlain on the original sample. The SMAPE of 7.54% has been observed.

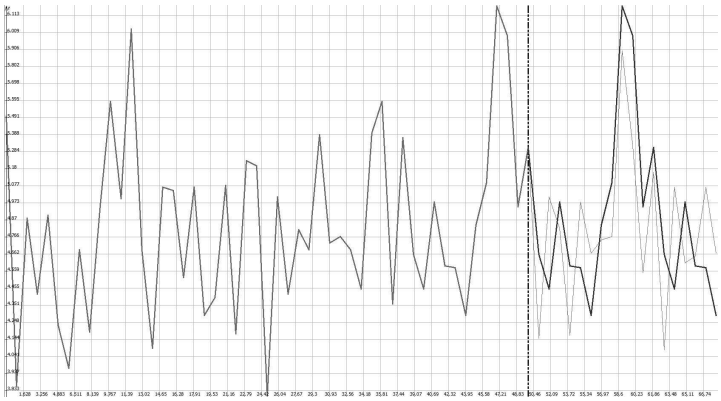


Figure 4. Results of testing VQTAM network, SMAPE 7.54%

In Figure 5 the results of testing RSOM network are overlain on

the original sample. The SMAPE of 7.79% has been observed.

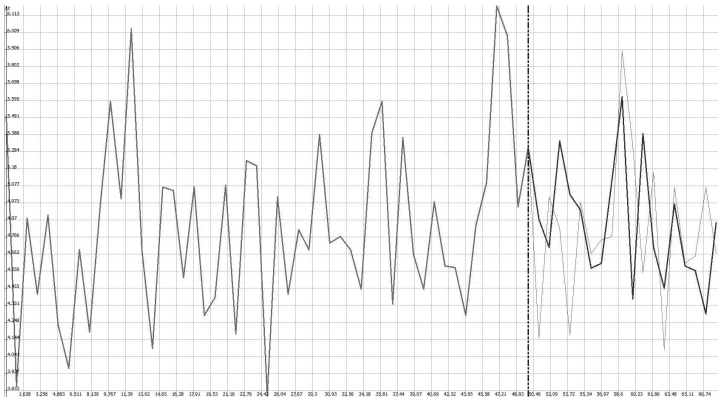


Figure 5. Results of testing RSOM network, SMAPE 7.79%

In Figure 6 the results of testing SOMxVQTAM modular network are presented. The SMAPE of 6.12% has been reached.

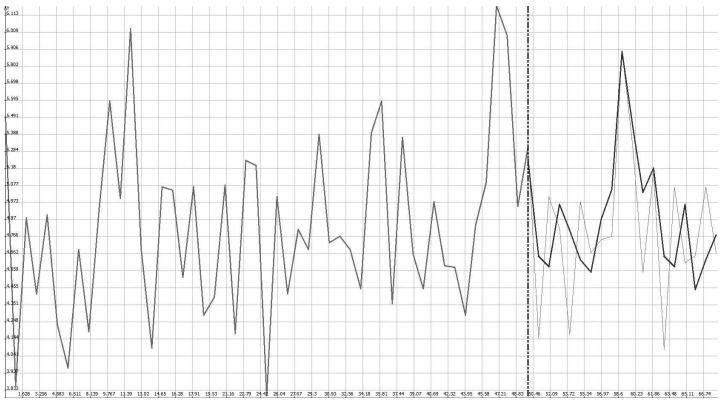


Figure 6. Results of testing SOMxVQTAM modular network, SMAPE 6.12%

In Figure 7 the results of testing SOMxRSOM modular network are overlain on the original sample. The SMAPE of 9.20% has been observed.

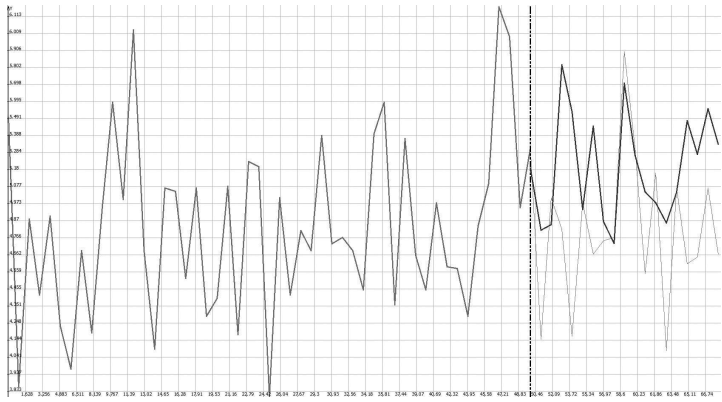


Figure 7. Results of testing SOMxRSOM modular network, SMAPE 9.20%

This and other tests that were concluded lead to a suggestion that modular modification gives a significant increase if accuracy in case of VQTAM modules, but modular networks are more sensitive to changes of learning parameters. The higher SMAPE in case of the SOMxRSOM modular network compared to RSOM network can be explaned by the fact that RSOM itself contains local models and during the learning process of the SOMxRSOM network those local models are treated as weight vectors for the whole SOMxRSOM network, but the local models are constructed for different parts of the input data and the local model corresponding to one of the neurons of one of the RSOMs are most likely to be constructed for the different part of the input data compared to the local model contained in another RSOM for the neuron on the same position.

4 Conclusion

In this article several SOM-based neural networks, that can be successfully used for dynamic object identification, have been described. Also, this study proves the necessity of investigating the possibilities of modular neural networks of higher complexity for identification of

dynamic objects and study of the ability of such networks to identify patterns in time series. Also it is necessary to develop an algorithm to automatically select learning parameters of presented in this paper neural networks in order to reduce prediction error.

References

[1] S. Haykin. *Neural Networks – A Comprehensive Foundation, 2nd Edition.* Prentice Hall International, Inc. (1998).

[2] N. Efremova, N. Asakura, T. Inui. *Natural object recognition with the view-invariant neural network.* The Proceedings of the 5th International Conference of Cognitive Science (CoSci), (2012).

[3] A. Trofimov, I. Povidalo, S. Chernetsov. *Usage of the self-learning neural networks for the blood glucose level of patients with diabetes mellitus type 1 identification.* Science and education. vol. 5 (2010), (In Russian). http://technomag.edu.ru/doc/142908.html

[4] T. Koskela. *Neural network methods in analyzing and modelling time varying processes.* Espoo. (2003), pp. 1–72.

[5] Luis Gustavo M. Souza, Guilherme A. Barreto. *Multiple Local ARX Modeling for System Identification Using the Self-Organizing Map.* Proceedings, European Symposium on Artificial Neural Networks – Computational Intelligence and Machine Learning. Bruges (Belgium). (2010).

[6] M. Varsta, J. Heikkonen. *A recurrent Self-Organizing Map for temporal sequence processing.* Springer. (1997), pp. 421–426.

[7] A. Lotfi, J. Garibaldi. *In Applications and Science in Soft Computing, Advances in Soft Computing Series.* Springer. (2003), pp. 3–8.

[8] K. Tokunaga, T. Furukawa. *SOM of SOMs.* Neural Networks. vol. 22 (2009), pp. 463–478.

[9] K. Tokunaga, T. Furukawa. *Modular network SOM.* Neural Networks. vol. 22 (2009), pp. 82–90.

[10] N. K. Logothetis, J. Pauls, T. Poggiot. *Shape representation in the inferior temporal cortex of monkeys.* Current Biology. vol. 5 (1995), pp. 552–563.

[11] T. Vetter, A. Hurlbert, T. Poggio. *View-based Models of 3D Object Recognition: Invariance to Imaging Transformations.* Cerebral Cortex. vol. 3 (1995), pp. 261–269.

[12] K. Aihara, G. Matsumoto, M. Ichikawa. *An alternating periodic-chaotic sequence observed in neural oscillators.* Phys. Lett. A. vol. 111(5) (1985), pp. 251–255.

[13] K. Aihara, T. Takabe, M. Toyoda. *Chaotic neural networks.* Phys. Lett. A. vol. 144(6/7) (1990), pp. 333–340.

[14] *Artificial Neural Network & Computational Intelligence Forecasting Competition.* (2007). http://www.neural-forecasting-competition.com/NN3/results.htm

Aleksey Averkin, Veaceslav Albu,
Sergey Ulyanov, Ilya Povidalo

Received December 2, 2013

Aleksey Averkin
Institution of Russian Academy of Sciences Dorodnicyn Computing Centre of RAS
Vavilov st. 40, 119333 Moscow, Russia
E–mail: averkin2003@inbox.ru

Veaceslav Albu
Institute of Mathematics and Computer Science
Academiei 5, Kishinev, MD 2028 Moldova
E–mail: vaalbu@googlemail.com

Sergey Ulyanov
International University of Nature, Society and Man "Dubna"
Universitetskaya st. 19, 141980 Dubna, Moscow region, Russia
E–mail: ulyanovsv@mail.ru

Ilya Povidalo
International University of Nature, Society and Man "Dubna"
Universitetskaya st. 19, 141980 Dubna, Moscow region, Russia
E–mail: ipovidalo@gmail.com

Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Evaluating the impact of software metrics on defects prediction. Part 2

Arwa Abu Asad, Izzat Alsmadi

Abstract

Software metrics are used as indicators of the quality of the developed software. Metrics can be collected from any software part such as: code, design, or requirements. In this paper, we evaluated several examples of design coupling metrics. Analysis and experiments follow hereinafter to demonstrate the use and value of those metrics. This is the second part for a paper we published in Computer Science Journal of Moldova (CSJM), V.21, N.2(62), 2013 [19]. We proposed and evaluated several design and code coupling metrics. In this part, we collected source code from Scarab open source project. This open source is selected due to the availability of bug reports. We used bug reports for further analysis and association where bugs are used to form a class for classification and prediction purposes. Metrics are collected and analyzed automatically through the developed tool. Statistical and data mining methods are then used to generalize some findings related to the collected metrics. In addition classification and prediction algorithms are used to correlate collected metrics with high level quality attributes such as maintainability and defects prediction.

Keywords: Design metrics, Object-Oriented Designs, Coupling metrics, software faults.

1 Introduction

Design activity should consider the dependency between classes so that changing in one class should not be propagated to several classes. Quality attributes such as: dependency and modularity can be measured or

evaluated through coupling metrics. Coupling describes how classes are related and dependent on each other. Coupling is considered as one of the fundamental design metrics that aims to minimize coupling among different modules facilitating understanding, maintaining, reusability, modularity and testing tasks of the software and design. In addition, they provide information to the designers regarding the capability of their design to change or to be reused. Low coupling results in components' self-containment, which in turn increases class understandability in isolation. Furthermore, it improves maintainability and increases potentials for reuse. On the other side, high coupling between two classes or components makes it more difficult to understand one of them in separation from the other. Thus, ripple changes are increased due to high dependency among classes. Myers et al., (1974) [18] defined six coupling metrics between pairs of modules. Those are: content, common, external, control, stamp, and data coupling. Eder et al. (1994) [7], Hitz et al. (1995) [10], and Briand et al. (1997) [3] defined OO coupling frameworks. These three frameworks were unified by Briand et al. (1999) to a more formalized framework.

A software product can hardly be free from errors. Maintenance and testing stages involve looking for bugs and fixing them. However, the allocation of potential regions where errors are or can be located plays important role in budget and effort reduction in software projects. In this study, we have used fault proneness as a quality predictor. On the other hand, we have investigated several coupling metrics as possible predictors for fault proneness.

In this paper several design coupling metrics proposed by Briand et al (1999) [4] are assessed using a large set of open source code projects. One of the major design goals is to minimize coupling. Therefore, it is important to be able to develop tools for coupling assessment before or after code development.

2 Coupling Metrics

Coupling is a measure of interconnection among modules. One of major goals in software design is that classes should be loosely coupled.

Therefore, Simple connection between modules will produce more understandable and maintainable software. Loosely coupled software will be less subjective to ripple effect in case of code changes. Myers et al. (1974) [18] defined six procedural coupling types, which are explained in Table 1.

Table 1. Procedural coupling levels

Coupling type	Description
DATA COUPLING	Data coupling occurs when passing pure simple data between two modules by parameters using a simple elementary piece of argument list and every item in the list is used.
STAMP COUPLING	Stamp coupling occurs between modules when passing composite data through parameters using a data structure containing fields, which may or may not be used.
CONTROL COUPLING	Control coupling occurs between modules when data are passed that influence the internal logic of a module (e.g. flags)
COMMON COUPLING	Common coupling occurs when modules communicate sharing global data areas so common coupling is also known as global coupling.
CONTENT COUPLING	Content coupling occurs between two modules if one modifies the internals of other module. In practice, only assembler language allows content coupling. Most object-oriented programming languages do not allow implementing content coupling.

These metrics were realized into object oriented design resulting in many metrics. This research is interested in investigating coupling metrics unified by Briand et al. (1999) [4] framework. Here is a brief preview of coupling metrics that will be measured in this research:

Coupling between object (CBO) (Chidamber & Kemerer, 1994 [6])

Two classes are coupled when methods declared in one class use methods or instance variables defined by the other class. Multiple accesses to the same class are counted as one access. Only method calls and variable references are counted as the original definition: "two classes are coupled when methods of one class use methods or instance variables defined by the other class".

Definition 1

CBO = number of classes to which a class is coupled

Response for a Class (RFC) (Chidamber & Kemerer, 1994 [6]) The response set of a class is a set of methods that can potentially be executed in response to a message received by an object of that class It counts only the first level of calls outside of the class. RFC is simply the number of methods in the set. It regards all methods and properties declared. Calls to properties: Set and Get are all counted separately.

Definition 2

RFC = M + R (First-step measure) where M = number of methods in the class, R = number of remote methods directly called by methods of the class

Message passing coupling (MPC) (Li & Henry, 1993) [13]

The message passing is the number of call statements defined in a class or the number of messages between objects in local method of a class. Thus, this includes only counting method invocations to other classes, and these classes with inheritance relationship have been excluded from counting.

Data Abstraction Coupling (DAC) (Li & Henry, 1993) [13]

Data Abstraction Coupling is the total number of other class types in attribute declarations. It is also referred to as aggregation coupling. DAC does not count primitive types, system types, and inherited types from the base classes. DAC has two variants, which are:

Definition 3

DAC: The number of attributes in a class that have another class as their type (count the repetition of class type)
DAC1: The number of different classes that are used as types of attributes in a class

Information-flow-based coupling (ICP) (Lee et al., 1995) [11]

Information-flow-based coupling is the number of implemented method m of one class plus the number of polymorphically invoked methods of other classes, weighted by the number of parameters of the invoked method.

Definition 4

$$ICP^c(m) = \sum_{m' \in PIM(m) - (M_{NEW}(c) \cup M_{OVR}(c))} (1 + |Par(m')|) \cdot NPI(m, m')$$

Where:
$M_{OVR}(c)$: is the set of overriding methods of class c.
$M_{NEW}(c)$: is the set of non-inherited, non-overriding methods of class c.
$Par(m)$: is the set of parameters of method m.
$PIM(m)$: is the set of Polymorphically Invoked Methods of m.
$NPI(m, m')$: is the Number of Polymorphic Invocations of m' by m.

Briand et al. (1999) [4] redefined ICP as the number of method invocations in one class weighted by the number of parameters of the methods invoked by class methods (the weight is number of parameters +1 but for empirical consideration they refined it to number of parameters only).

Definition 5

$$\frac{Total\ Number\ of\ Method\ Invocations}{Total\ Number\ of\ Method\ Invocations + \sum_{i=0}^{all\ methods} parmOf(m_i)}$$

ICP has two variations, which are:

- H-ICP of just inheritance invocation are considered invocation to method of ancestors classes only
- NIH-ICP non inheritance invocation

ICP is the sum of IH-ICP and NIH-ICP

Definition 6

$$ICP = IH - ICP + NIH - ICP$$

Coupling factor (COF) (Abreu et al., 1995)

COF is defined as: the ratio of the maximum possible number of couplings in the system to the actual number of couplings not imputable to inheritance.

Definition 7

Coupling Factor

$$COF = \frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC - 2 \times \sum_{i=1}^{TC} DC(C_i)}$$

$$is\_client(C_c, C_s) = \begin{vmatrix} 1 & iff & C_c \Rightarrow C_s \wedge C_c \neq C_s \\ & & \wedge\neg(C_c \rightarrow C_s) \\ 0 & otherwise & \end{vmatrix}$$

- $is\_client(x, y) = 1$ iff a dependency exists between the client and the server classes. 0 otherwise;
- $(TC2 - TC)$ is the total number of possible dependencies.

Briand suite [3-5]

Briand et al. suggested suite of coupling metrics. The abbreviations used in Briand measurements:

A: Coupling to ancestor classes.

D: Coupling to Descendents.

IC: import coupling, the measure counts for a class C all interactions where C is using another class;

EC – export coupling, counts interactions where class D is the used class.

CA – Class-attribute interaction

CM – Class-method interaction

MM – Method-method interaction

The definitions for CA, CM, and MM are as follows:

Class-attribute interaction (CA) (Briand et al., 1997 [3])

Class-attribute interaction occurs if a class contains an attribute of type another class. It is the number of class-attribute interactions from one class C to another class D.

Definition 8

$$CA(c,d) = \left|a|a \in A_I(c') \wedge T(a) = d\right|.$$

A: is the set of attributes in class c.

$T(a)$: is the type of attribute where the attribute type will be a class.

Class-method interaction (CM) (Briand et al., 1997 [3])

Class-method interaction occurs if a newly defined method of one class has a parameter of type another class.

Definition 9

$$CM(c,d) = \sum_{m \in M_{NEW}(c)} |a|a \in Par(m) \wedge T(a) = d|.$$

$T(a)$: is the type of attribute where the attribute type will be a class.

$Par(m)$: is the set of parameters of method m.

Method-method interaction (MM) (Briand et al., 1997 [3])
Method-method interaction occurs if a method implemented at class c statically invokes a method of class d (newly defined or overriding), or receives a pointer to such a method. The number of method-method interactions from class c to class d.

Definition 10

$$MM(c,d) = \sum_{m \in M_I(c)} \sum_{m' \in M_{NEW}(d) \cup M_{OVR}(d)} (NSI(m,m') + PP(m,m')).$$

$NSI(m,m')$: is the number of static invocations of method m' by m.

M: is the set of methods in a class.

$M_{OVR}(c)$: is the set of overriding methods of class c.

$M_{NEW}(c)$: is the set of non-inherited, non-overriding methods of class c.

Afferent and Efferent Coupling (Martin 2002) [14]
The OO Design package or system metrics of Martin (Afferent and Efferent Coupling), based on fan-in and fan-out metrics, are largely used in the industry and commonly referenced in the academy. Lots of static analysis tools are able to extract them. They can help in understanding how defects are likely to ripple among different classes or components. They measure how much a particular class, method, component etc. is coupled with other components as either calling them or being called by them. The remainder of this paper is organized as follows: Section 3 provides an overview of related work. Section 4 presents goals and approaches. Section 5 presents an experimental section and paper is concluded in Section 6.

3 Related Work

In this section, we will list some examples of related papers of evaluating software design or coupling metrics.

Hitz and Montazeri (1995) [10] distinguished between coupling among objects (CLO) and coupling among classes (CLC). CLC is principally important in maintenance or change dependencies within program. On the other hand, OLC is relevant for runtime-oriented activities like testing and debugging. In addition, they argued that various levels of coupling depend on three attributes: Stability, Access Type and Scope of Access, and each combination of these attributes would have different strength of coupling. Coupling measures the strength of connection between two classes as pairs. Although, coupling was defined as characteristics of pairs of classes, but as a metric, it is the total number of couples that one class has with other classes. Therefore, implicitly all coupling connections are supposed to be of equal strength among a class with others in the system level (Norman et al., 1992). Therefore, in the proposed system coupling was computed using previous definition.

Briand et al. (1999) [4] refined existing framework defined by (Eder et al., 1994 [7]; Hitz and Montazeri, 1995 [10]; and Briand et al. 1997 [3]) into comprehensive and formalized framework for coupling measurements. The framework for coupling consists of six criteria, which are:

1. Type of connection
2. Direction of connection: Fan-in or Fan-out.
3. Granularity of the measure: Domain of the measure and how to count coupling connections.
4. Stability of server: whether a class stable or changing frequently.
5. Direct or indirect coupling.
6. Inheritance: Inheritance-based or non inheritance-based coupling.

Briand et al. (1999) [4] also distinguished two directions of coupling, which are import and export. Export occurs when a class is used as server class in the interaction. On the other hand, import occurs when

a class is used as client class in the interaction. But Lee (2007) [12] specified that Dependency between classes could be in one direction or two directions. A high fan-out represents a class coupling to other classes. High fan-in represents a good level of reuse.

Many object oriented metrics were proposed in the previous studies to predict software quality attributes. One of these attributes is fault proneness. Michael English et al. (2009) [9] summarized the results from 23 papers that evaluated software metrics and their correlation with software fault prone modules. Their evaluation showed that CK and LOC metrics are the most used and evaluated metrics. Emam et al. (2001) [8] used the CK metrics in addition to Briand's coupling metrics to predict faults on a commercial Java system [3-5]. Basili et al. (1996) studied the correlation of fault-proneness with CK metrics for eight student projects. The results showed that WMC, CBO, DIT, NOC and RFC have a significant correlation with faults whereas LCOM didn't have same significant correlation with faults. Tang et al. (1999) [16] found that higher WMC and RFC were found to be associated with fault-proneness on three real time systems. Menzies et al. (2007) [15], on the other hand, evaluated fault proneness for C and Java projects.

There are various types of methods to predict faulty classes such as statistical methods, machine learning methods, etc. However, the trend recently is moved from traditional statistical methods to machine learning methods. Zhou et al. (2006) [17] used logistic regression and machine learning methods to show how OO metrics and fault proneness are correlated. The results showed that WMC, CBO, and SLOC were found to be strong predictors across all severity levels based on the public domain NASA datasets.

4 Goals and Approaches

As described earlier, this is the second part of one paper. In the first part that was published in CSJM, V.21, N.2(62), 2013 [19], we described in details the coupling metrics that we want to investigate. We described them with examples and also described the developed tool to automatically collect those coupling metrics. In this part, we assembled

a case study of a large number of source code.

We will define the terminologies and the abbreviations that are used in later sections. Table 2 shows the terms and the definitions used in this paper as well as the abbreviation of terms.

5 Results and Analysis

In this section, we will present the results from the collection of metrics and their statistical and correlation analysis with bug reports for the same source code. The first section includes descriptive statistics for the collected coupling metrics.

5.1 Descriptive Statistics for Coupling measures

Table 3 presents the descriptive statistics for the coupling measures collected from Scarab project. The columns: Min, Max, Mean, and Std. Deviation are the: minimum value, maximum value, mean value and standard deviation, respectively. $N > 5$ is NO if the count of non-zero values is less than six.

The following observations can be made from Table 3:

- The measures that are counting various relationships through inheritance (i.e. ACAIC, DCAEC, ACMIC, and ACMIC) have all relatively zero values for all columns which means that no extensive use of inheritance (at the attribute and method use, override and reuse) is observed in the Scarab project.
- The largest maximum value is for RFC, which also has the largest mean. This may be explained by the fact that RFC counts method invocations plus the number of methods. MPC has the next maximum mean value as it counts the sent statement.
- All measures with significant variance (i.e. six or more non-zero values – $N > 5$) were subjected to further analysis. Therefore, ACAIC, DCAEC, ACMIC, and DCMEC are removed from further analysis.

Table 2. Terminology Abbreviation

Term	Definition	Abbreviation
Defined methods	Methods declared within class C	M_{def}
Inherited methods	Methods declared within parent class and inherited and (not overridden) within child class C	M_{inh}
Polymorphic method	Methods defined within an interface	M_{poly}
New methods	Methods declared within class C that do not override inherited ones	M_{new}
Overriding methods	Methods declared within class C that override (redefine) inherited ones	M_{over}
Invoked methods	Methods that can be invoked in association with class C	M_{inv}
External calls	Invocations to a method defined in other classes	$Call_{ex}$
Internal calls	Invocations to a method in the same class	$Call_{int}$
Polymorphic calls	Invocations to a method defined in an interface	$Call_{poly}$
Inherited calls	Invocations to a method defined in parent class through object of child class	$Call_{inh}$
Defined attributes	Attributes declared within class C	$Attr_{def}$
Overriding attributes	Attributes declared within class C that overrides (redefines) inherited ones	$Attr_{over}$
Inherited attributes	Attributes inherited (and not overridden) in class C	$Attr_{inh}$
Used attributes	Attributes that can be manipulated in association with class C (those external public attribute from other classes used in method of class C)	$Attr_{use}$
Instance attributes	Attributes of type class	$Attr_{inst}$
Instance parameters	Parameters of type class	arg_{ins}

Table 3. Descriptive statistics for collected metrics

Descriptive Statistics					
Metrics	Min	Max	Mean	Std. Deviation	N>5
CBO	0	53	5.17	8.13	
RFC	0	1098	33.42	85.49	
MPC	0	928	24.56	71.79	
DAC	0	12	.36	1.11	
DAC1	0	11	.33	.99	
ICP	.0	1.6	.46	.39	
ACAIC	0	0	.00	.00	NO
DCAEC	0	0	.00	.00	NO
ACMIC	0	0	.00	.00	NO
DCMEC	0	0	.00	.00	NO
AMMIC	0	32	.21	1.81	
DMMEC	0	44	.21	2.37	
NOC	0	31	.33	2.42	
NDC	0	37	.39	2.87	
NAC	0	4	.39	.70	

5.2 A defect prediction model

In this section, we built a defect prediction model to validate the correlation between coupling metrics and reported bugs. On the other hand, we also measured which coupling metrics have more significant effect in comparison with the rest of collected metrics.

Figure 1 shows the results from Gain Ratio (GR) feature selection method applied on the collected dataset. It can be seen that inheritance metrics have no significant role in bug prediction. CBO, RFC, MPC and ICP are the most significant metrics in defects prediction. In particular, CBO has the most significant effect.

J48graft data mining method is then used on the collected dataset after applying feature selection. Table 4 shows prediction performance

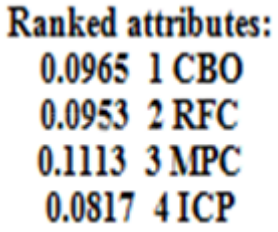


Figure 1. Gain Ratio feature evaluator

metrics: recall, precision, and accuracy. The Table 4 shows the recall, precision and accuracy for faulty (i.e. true case), and for not faulty classes (i.e. false case). The Table 4 shows also the recall, precision and accuracy for overall average of both. *10 cross validation* method is used as the selection for testing and training.

Table 4. Scarab performance metrics with J48graft and 10 cross validation

	Precision	Recall	Accuracy
Not faulty	0.77	0.69	0.69
Faulty	0.61	0.70	0.70
Weighted Avg.	0.70	0.70	0.70

Figure 2 shows the classification tree after applying gain ratio using J48graft.

5.3 Study Limitation

Similar to most experiments conducted in this field one of the limitations is the dependency of the results in one source code project.

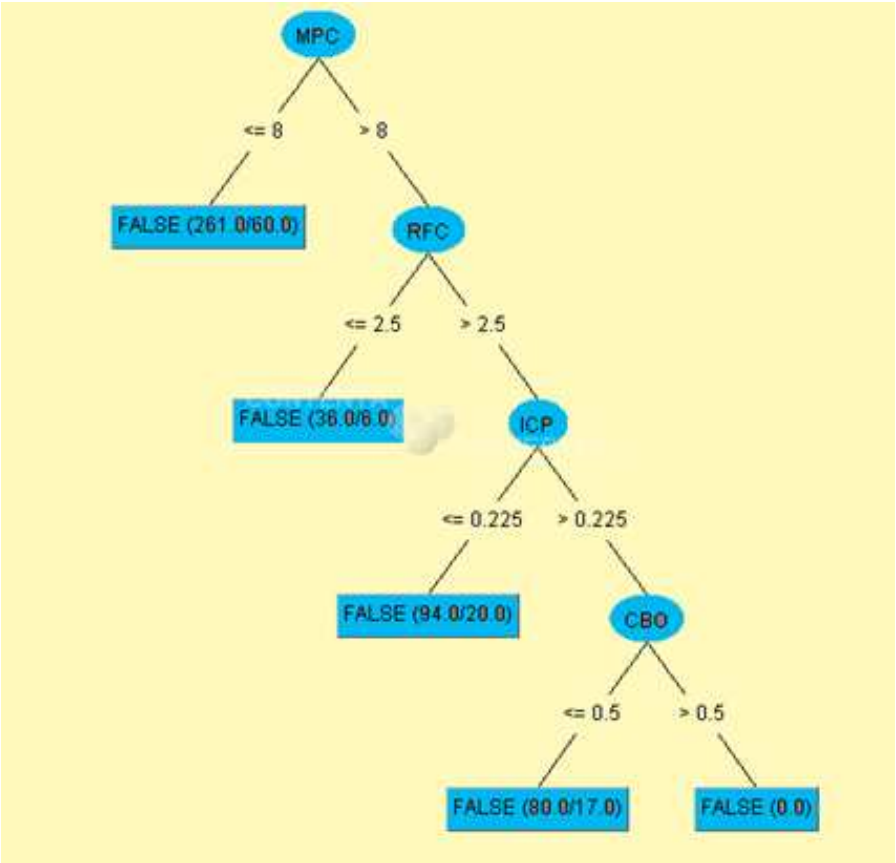


Figure 2. Metrics-bugs-dependent classification tree

This may have a risk of having possibly biased results that may not be generalized to all software source codes especially those that may not share the project same domain, environment, etc.

In addition, the dataset used in this research may not be perfect or complete. Bugs reporting and judgments are also human subjective. In addition, bugs were not classified or categorized based on their seriousness or importance. Those are examples of some of the possible limitations on generalizing the results in this study on other software products.

6 Summary and Conclusion

In this paper, we presented several metrics for evaluating the quality of software design. Those metrics focus on evaluating design coupling quality where software components are expected to be moderately coupled. Experiments and analysis were conducted to demonstrate the value of the proposed and evaluated metrics. Scarab open source is used in the case study due to the availability of bug reports related to the usage of this software. We used such bug reports for classification and prediction where a bug class is formed with values of true or false based on whether the subject class contains bugs or not (based on bug reports). Some coupling metrics such as: CBO, RFC, MPC and ICP showed significant correlation with bugs. This means that those coupling metrics can be used for bugs prediction. Monitoring those metrics early in the software project can help software project managements monitor quality aspects especially those related to bugs.

References

[1] F. Abreu, M. Goul, R. Esteves. *Toward the design quality evaluation of object-oriented software systems*, Proc. Fifth Int'l Conf. Software Quality, Austin, Texas, (1995), pp.44–57.

[2] V. Basili, L. Briand, W. Melo. *A Validation of Object Oriented Design Metrics as Quality Indicators*, IEEE Transactions on Software Engineering, 22, (1996), pp. 751–761.

[3] L. Briand, P. Devanbu, W. Melo. *An investigation into coupling measures for C++*, *Proc. 19th Int'l Conf. Software Eng, ICSE*, (1997), pp. 412–421.

[4] L. Briand, J.W. Daly, J.K. Wust. *A unified framework for coupling measurment in objmct-oriented systems*, IEEE Tranractions on Software Engineering, 25 (1) (1999), pp. 91–121.

[5] L.C. Briand, J.K. Wust, J.W. Daly, D.V. Porter *Exploring the relationships between design measures and software quality in object oriented systems'*, Journal of systems and Software, 51(3), (2000), pp. 254–273.

[6] S. Chidamber, C. Kemerer. *A metrics suite for object oriented design*, IEEE Trans Journal. Software Eng, 20 (6), (1994), pp. 476–493.

[7] J. Eder, C. Kappel, M. Schrefl. *Coupling and Cohesion in Object-Oriented Systems*. Technical Report, Univ. of Klagenlurt, available at ftp://ftp.ifs.uni-inz.ac.at/pub/publications/1993/0293.ps.gz, (1994).

[8] K.E. Emam, W. Melo, J.C. Machado. *The prediction of faulty classes using object-oriented design metrics*, Journal of Systems and Software, 56(1), (2001), pp. 63–75.

[9] M. English, Ch. Exton, I. Rigon, B. Cleary. *Fault Detection and Prediction in an Open-Source Software Project*, In Proceedings of Promise, (2009).

[10] M. Hitz, B. Montazeri. *Measuring coupling and cohesion in object-oriented systems*, Proc. ESEC '95 fifth European Software Eng. Conf., Barcelona, Spain, 1 (4), (1995), pp. 2–10.

[11] Y. Lee, B.S. Liang, S.F. Wu, F.J. Wang. *Measuring the coupling and cohesion of an object-oriented program based on information flow*, Proc. Int'l Conf. Software Quality, Maribor, Slovenia 12, (1995), pp. 81–90.

[12] Y. Lee. *Automated source code measurement environment for software quality*, Doctor Thesis, (2007) Auburn, Alabama, USA.

[13] W. Li, S. Henry. *Object-Oriented metrics that predict maintainability*, J. of Systems and Software, 23 (2), (1993), pp. 111–122.

[14] R.C. Martin. *Agile Software Development, Principles, Patterns, and Practices*, Prentice Hall (2002).

[15] T. Menzies, J.Gneenwald, A. Frank. *Data mining static code attributes to learn defect predictors*, IEEE Trans. on Soft. Eng., 33(1), (2007), pp. 2–13.

[16] M.H. Tang, M.H. Kao, M.H. Chen. *An empirical study on object-oriented metrics*, In Sixth International Software Metrics Symposium, (1999), pp. 242–249.

[17] Y. Zhou, H.H. Leung. *Empirical Analysis of Object-Oriented Design Metrics for Predicting High and low Severity Faults*, IEEE Transactions on Software Engineering, 32(10), (2006), pp. 771–789.

[18] W. Stevens, G. Myers, L.L. Constantine. *Structured design*, IBM Systems Journal, 13 (2), (1974), pp. 115–139.

[19] A.A. Asad, I. Alsmadi. *Design and code coupling assessment based on defects prediction. Part 1*, Computer Science Journal of Moldova (CSJM), V.21, N.2(62), (2013), pp. 204–224.

Arwa Abu Asad, Izzat Alsmadi

Received October 17, 2012

Arwa Abu Asad
Institution: Yarmouk University
Address: CIS department
E–mail: *arwa_abuasad@yahoo.com*

Izzat Alsmadi
Institution : Yarmouk University
Address : CIS department
E–mail: *ialsmadi@yu.edu.jo*