Anul XLV 2000

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

1

Redacţia: 3400 Cluj-Napoca, str. M. Kogălniceanu nr. 1 Telefon 405300

SUMAR – CONTENTS – SOMMAIRE

RECENZII – REVIEWS – ANALYSES

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

THE RELAXATION OF THE FUNDAMENTAL CONDITIONS OF SCIENTIFIC VISUALIZATION USING EQUIVALENCE CLASSES

DUMITRU RĂDOIU AND ADRIAN ROMAN

Abstract. The paper addresses the issue of scientific data visualization process validation. Three fundamental conditions for scientific visualization are introduced; one of them – the Precision Condition – is discussed in detail. The theory allows a better formal description of the scientific visualization process.

Index terms – scientific visualization, precision classes, and scientific visualization process validation

1. Introduction

Scientific Visualization is a computational process that maps scientific data and its attributes into visual objects [1]. Scientific data can be obtained in many different ways, e.g. by running a simulation or by a DAQ process. Usually, **scientific data objects** are finite representations of complex mathematical objects. We note by O the set of such objects, $o \in O$. During the visualization process, initial data objects, o, are processed through different transformation functions $Mat(o) = o'$, into a new set $o' \in O'$. Objects o' are then mapped $Map(o') = g$ into a set of **virtual geometrical objects** $g \in G$, through a set of **graphical primitives**. Objects g usually are n-dimensional (nD), animated (t) and interactive.

Definition 1 *A group of virtual geometrical objects, logically interconnected, is called a* ***logical visualization*** *of that scene.*

Ideal geometrical objects g, nD, animated (t) and interactive are usually represented $Rep(g) = g'$, $g' \in G'$, on real 2D screens.

Definition 2 *The projection of the logical visualization of a scene on a screen is called a* ***physical visualization*** *of that scene.*

The functions $Rep(g) = g'$ implement classical graphical operations such as composition of the scene, volume generation, isosurface generation, simulation of transparency, reflectivity and lighting conditions, $nD \rightarrow 2D$ projection, clipping,

2000 *Mathematics Subject Classification.* 65D18.

1998 *CR Categories and Descriptors.* I.3.6 [**Computing Methodologies**]: Computer Graphics – *Methodology and Techniques.*

hidden surface removal, shading, animation (t), setting user interactivity (zoom, rotate, translate, pan, etc), etc.

Definition 3. *By **interactivity** we understand the attributes of visual objects (logical and/or physical) whose setting allows nD → 2D projection (zoom, rotate, translate, pan, etc), animation control (t), control of the objects composing the scene and control of the scene as a composite object.*

The scientific visualization process is described by the $Vis(o) = g'$, $Vis(o) = Rep(Map(Mat(o))) = g'$ function. The process is described in figure 1.

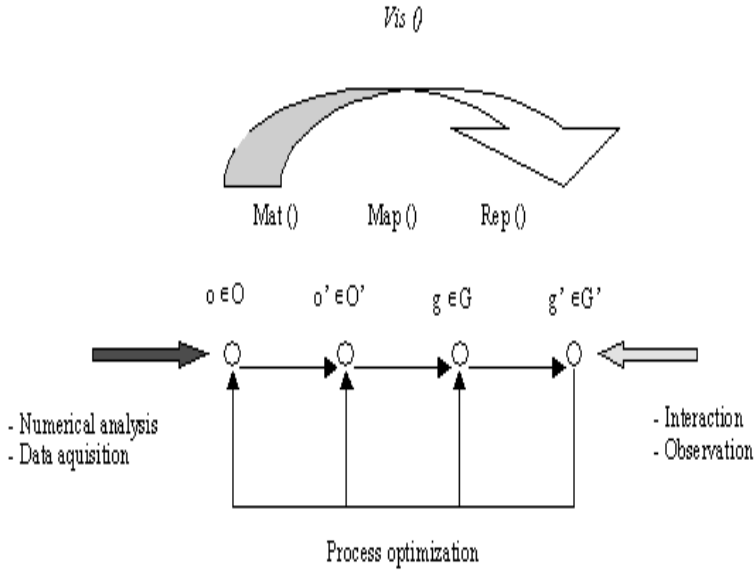


FIGURE 1. Description of the scientific visualisation process

2. Fundamental conditions of scientific visualization

There are many requirements concerning a certain scientific visualization process. We consider three of them to be fundamental. The first one is the **distinctiveness condition**. This condition (although very weak) enables users to distinguish between different data objects based on their display. The condition is necessary as one can imagine many visualization functions that generate images with no use, which reveal none of the data objects characteristics/attributes.

The second condition is the **expressiveness condition**. This condition assures that the attributes of the visual object represent the attributes of the input data set.

The third one is the **precision condition**. This condition insures that the order among data objects is preserved among visual objects.

The distinctiveness condition. *Different input data (different mathematical objects) are represented by different visual objects.*

This condition can be stated:

$$\begin{aligned} o_1 \neq o_2 \quad &\Rightarrow \quad Vis(o_1) \neq Vis(o_2) \Rightarrow Rep(Map(Mat(o_1))) \neq Rep(Map(Mat(o_2))) \\ &\Rightarrow \quad g1' \neq g2', \text{ for any } o1, o2 \in O, g1', g2' \in G' \end{aligned}$$

The interpretation of this condition is that $Vis()$, $Mat()$, $Map()$ and $Rep()$ functions are injective.

The expressiveness condition. *The visual objects express all and only the characteristics of the input data.*

It results that the visualization function should be one to one.

The two conditions are necessary but not sufficient. Another condition is needed to establish an order relation both among data and visual objects. This condition could be seen as a precision relation.

The precision condition. *For any objects $o_1, o_2 \in O$ such that o_1 is "more precise" than o_2 we have that $Vis(o_1)$ is "more precise" than $Vis(o_2)$, with $Vis(o_1)$, $Vis(o_2) \in G'$.*

The precision condition adds something new. If the visualization function is well defined and the input data objects are strictly ordered, the visual objects can be ordered by their "precision".

The first two conditions introduce criteria of validation and control of the visualization process. The visualization function $Vis()$ fulfilling these criteria results in a scientific visualization. The third condition allows further developments by defining mathematical operations on the given ordering.

3. Equivalence classes

We introduce another approach to describe formally the visualization process. There are examples that prove that the above conditions are too "tight". Because we display the visual objects on real screens (i.e. with finite resolution) it is possible that two or more objects o to be mapped into identical visual objects. Therefore a more relaxed approach to formally describe the visualization process of scientific data is necessary. In order to describe the new approach some mathematical concepts are to be presented.

We have already introduced the basic sets denoted by O, O', G and G'. O represents the set of the so called "data objects". O' is the set of the elements obtained from "data objects" through different transformation functions. G represents the set of virtual geometrical objects, nD. Virtual geometrical objects become real geometrical objects (G') by projection/display (e.g. on 2D screens).

The visualization function can be described as the mapping of the set O into G'.

Definition 4. *Let O and G' be two sets and v be a binary relation. We call v a mapping of O in G' if for each element $o \in O$, there is exactly one element $g' \in G'$ that satisfies $\langle o, g' \rangle \in v$.*

The element g' is called the **image** of the element o through v, and o is called the **inverse image** of g' through v. For the mapping v we introduce the notation $v : o \to g'$ and the functional notation $v(o) = g'$. We can write that $v : O \to G'$ to show that $v = Vis()$ is a mapping of O into G'. O is called the domain of v. If the inverse relation is also a mapping, we will denote it by v^{-1}.

From the set theory we know that a *partition* π of a set O is a subset of $P(O)$ (the power set of O) not containing Φ, satisfying the following property: every $o \in O$ is an element of exactly one $A \in \pi$. The elements of a partition are called *blocks*. If π and π' are partitions of O, we will write $\pi \leq \pi'$ if for every block $B \in \pi$ there exists a block $C \in \pi'$ such that $B \subseteq C$.

We use the fundamental theorem of the equivalence relations in order to underline some important aspects:

Theorem 1. [10] (a) Let π be a partition of O and define a binary relation ϵ_p on O by $o_1 \epsilon o_2$ if and only if o_1 and o_2 are in the same block of the partition π. Then ϵ_π is an equivalence relation on O.

(b) If ϵ is an equivalence relation over a set O, then there exists a partition π_ϵ over O such that $o_1, o_2 \in O$ are elements of the same bloc of π_ϵ if and only if $o_1 \epsilon o_2$.

(c) If $\pi \leq \pi'$, then $\epsilon_\pi \leq \epsilon_{\pi'}$. If $\epsilon \leq \epsilon'$, then $\pi_\epsilon \leq \pi'_\epsilon$.

Theorem 1(a) shows that a binary relation is an equivalence relation if it "conserves" the initial partitioning over the given set. Theorem 1(b) states that a partitioning of a set can be obtained starting from a given equivalence relation ϵ. Theorem 1(c) introduces an order relation.

The following remark has to be stated:

Remark 1. *If more than one element o is mapped into the same visual object g', then the set O can be partitioned into non-empty subsets that include all the o elements mapped into the same visual object.*

Remark 1 introduces the idea of equivalence relations as the main tool in order to obtain a more realistic description of the visualization process. A natural equivalence relation ϵ_v can be defined over O. The relation ϵ_v is called the *equivalence relation induced by v* over the set of objects O and it partitions the set O into subsets of objects sharing the same visualization (see theorem 1). We denote by π_v the induced partitioning over O.

The proposed model is based on the concept of equivalence classes.

Definition 5. [9] *The equivalence class of an element $o \in O$, induced by the equivalence relation ϵ, is the subset of those elements from O that are in the relation ϵ with o.*

We denote by $[o]\epsilon$ the equivalence class of $o \in O$, induced by the equivalence relation ϵ. When the equivalence relation is implicit, we use the notation $[o]$.

Further, another theorem is introduced in order for us to be able to formulate the new visualization conditions.

Theorem 2. [10] Any mapping $v : O \to G'$ can de represented as a product of two mappings φ and ϕ, $v = \varphi\phi$, where φ is onto and ϕ is one-to-one; if ϵ is the equivalence relation induced by v, then $\varphi = \varphi_\epsilon : O \to O|\epsilon$ and $\phi : O|\epsilon \to G'$, where $O|\epsilon$ is the set of all equivalence classes induced by ϵ (Figure 2).

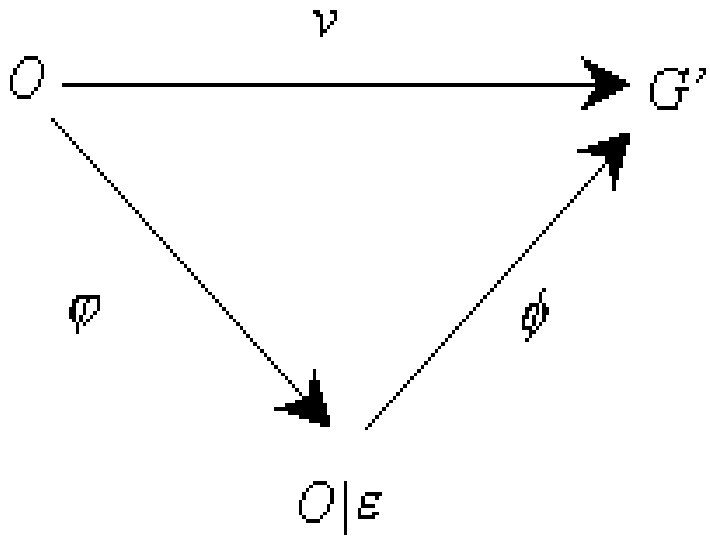


FIGURE 2. Schematic representation for Theorem 2

So, if we have a well-defined equivalence relation over O, then we can consider instead of v a product of two mappings (Figure 3). This approach has the advantage that it reduces the set of objects O to the set of classes $O|\epsilon$. Order relations can be stated over the set of classes.

The presented theory is exemplified below. We consider two data sets having the same format. The equivalence relation ϵ imposes that the attributes of the objects (element by element) have values between:

$$(a_i)_1, (a_i)_2 \in (a_i - \Delta a_i, a_i + \Delta a_i),$$

where $(a_i)_1$ are the attributes of the first object, and $(a_i)_2$ those of the second object. If the resolution of the screen is small enough we observe that, for the same visualization system, the two different data sets will be represented on the screen by the same visual object.

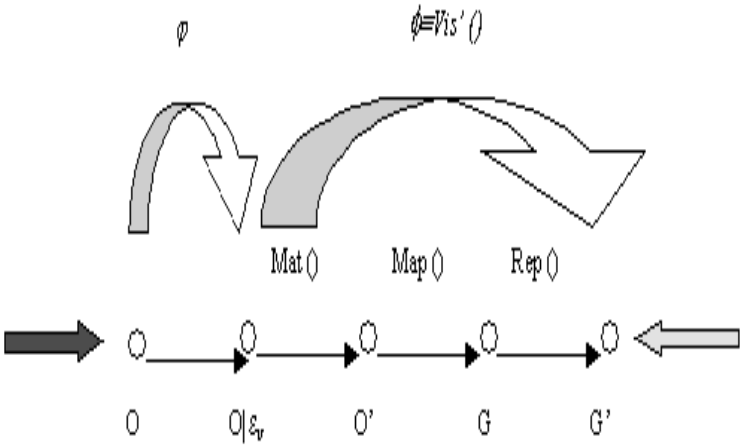


FIGURE 3. Description of the visualization process using equivalence classes

Remark 2. *Assuming that the equivalence relations* ϵ_0, ϵ_1, ..., ϵ_{n-1}, *defined over the same set exist, we conclude that the partitions* π_o, π_1, ..., π_{n-1} *also exist.*

Theorem 1(c) and remark 2 introduce an order relation between the equivalence classes, relation that can be regarded as "precision" relation. For the above example, we consider another equivalence relation ϵ' imposing that the attributes of the objects (element by element) have values between:

$$(a_i{}')_1, (a_i')_2 \in (a_i' - \Delta a_i', a_i' + \Delta a_i'),$$

where $(a_i')_1$ are the attributes of the first object, and $(a_i')_2$ those of the second object, with $\Delta a_i \leq \Delta a_i'$. In this case $\pi \leq \pi'$, where π and π' represent the partitions corresponding to the equivalence relations ϵ and ϵ'. From theorem 1.3 it results that $\epsilon \leq \epsilon'$.

4. The precision relation over the scientific visualization process

An order relation is necessary over the visualization process. We have introduced the "precision relation" as a fundamental condition of the scientific visualization. Now, the equivalence classes allow a further development of the idea. We are especially interested in the O (or $O|\epsilon$), G and G' sets.

Definition 6. *A class of objects, defined by the equivalence relation ϵ (see theorem 1), is "more precise" than another one, defined by the equivalence relation ϵ', if $\epsilon \leq \epsilon'$.*

So, $[o_1]\epsilon \leq [o_2]\epsilon'$ ($\leq$ describes the precision relation) if $\epsilon \leq \epsilon'$.

The set of virtual geometrical objects is denoted by G. A virtual geometrical object g can be regarded as a composition of graphical primitives. We denote by P the set of all types of graphical primitives. Let us denote by $SUM(N, P)$ the sum $\sum_{i=1}^{n} p_i$. Then the virtual geometrical object g can be described as:

$$g = SUM(N, P), \text{ where } p_i \in P, \text{ for a finite N.}$$

Definition 7. *1. A virtual geometrical object $g_1 = SUM(N_1, P)$ is said to be "strictly more precise" than another virtual geometrical object $g_2 = SUM(N_2, P)$ if $N_1 > N_2$.*

2. If $N_1 = N_2$, then a virtual geometrical object $g_1 = Map(Mat(o_1))$, $o_1 \leq [o_1]\epsilon$ is said to be "more precise" than another object $g_2 = Map(Mat(o_2))$, $o_2 \leq [o_2]\epsilon'$ if the class $[o1]\epsilon$ is "more precise" than the class $[o2]\epsilon'$.

Remarks. 1. An object can be represented using several ways (Figure 4). The representation considered "the most (strictly) precise" is the one that uses the highest number of graphical primitives. We call this kind of precision **representation precision**.

2. If the representation uses the same number of graphical primitives then the set G conserves the precision relation over O. The precision induced over G is called **order precision**.

If different numbers of graphical primitives are used, then the representation precision is considered as order relation.

Definition 8. *A visual geometrical object $g_1 \in G$ is said to be "(strictly) more precise" than another visual object $g_2 \in G$ if $g_1 = Rep^{-1}(g_1)$ is "(strictly) more precise" than $g_2 = Rep^{-1}(g_2)$.*

5. The relaxation of fundamental conditions of the scientific visualization

The fundamental conditions of the scientific visualization can be restated:

The distinctiveness condition. *Different equivalence classes are mapped into different visual objects.*

Formally: $[o_1] \neq [o_2] \Rightarrow \phi([o_1]) \neq \phi([o_2]) \Rightarrow g_1' \neq g_2'$
$[o_1], [o_2] \in O|\epsilon, g_1', g_2' \in G'$.

The expressiveness condition. *The visual objects express all the characteristics of input equivalence classes, and only those characteristics.*

Formally: $\forall g' \in G', \exists [o] \in O|\epsilon$ such that $\phi([o]) = g'$.

The distinctiveness condition and the expressiveness condition impose the mapping ϕ to be one-to-one.

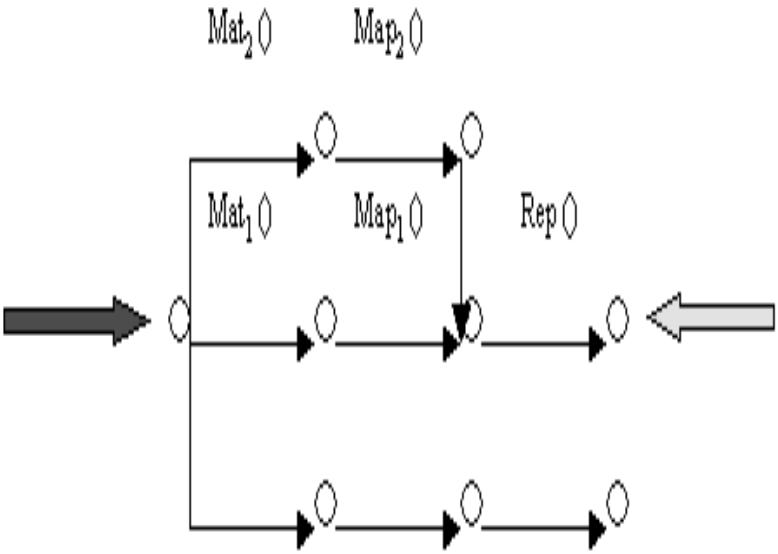


FIGURE 4. Example of visualization pipelines

The precision condition becomes the precision theorem. The equivalence class approach reduces the number of fundamental conditions and in the same time allows the introduction of a well-defined order relation.

Precision Theorem. 1. Let $[o]\epsilon \in O|\epsilon$ be a class of objects and let the ideal geometrical objects $g_1, g_2 \in G$, where g_1' represents the physical visualization of the the $[o]$ class using N_1 graphical primitives, and g_2' represents the physical visualization of the $[o]$ class object using N_2 graphical primitives.

i.: If $N_1 > N_2$ then g_1' is "strictly more precise" than g_2'.
ii.: If $N_1 = N_2$ then g_1' is represented with the same precision as g_2'.
iii.: If $N_1 < N_2$ then g_2' is "strictly more precise" than g_1'.

2. Let $[o1]\epsilon, [o2]\epsilon \in O|\epsilon$ be two classes of objects and let the ideal geometrical objects $g_1', g_2' \in G$, where g_1' represents the physical visualization of the the $[o_1]\epsilon$ class using N_1 graphical primitives, and g_2' represents the physical visualization of the $[o_2]\epsilon$ class object using N_2 graphical primitives. We consider that the class $[o_1]\epsilon$ is "more precise" than $[o_2]\epsilon$.

i.: If $N_1 > N_2$ then g_1' is "strictly more precise" than g_2'.
ii.: If $N_1 = N_2$ then g_1' is "more precise" than g_2'.
iii.: If $N_1 < N_2$ then g_2' is "strictly more precise" than g_1'.

Proof. 1. i. For the objects $g_1 = SUM(N_1, P)$ and $g_2 = SUM(N_2, P)$ we have $N_1 > N_2$. From definition 7.1 it results that g_1 is "strictly more precise" than g_2. From definition 8 we conclude that g_1' is "strictly more precise" than g_2'.

ii. If $N_1 = N_2$ then $g_1 = g_2$ and $g_1' = g_2'$.

iii. The same proof as for i.

2. ii. We assume that a class of objects induced by an equivalence relation ϵ, $[o_1]\epsilon$, is "more precise" than another one, $[o_2]\epsilon$, induced by the equivalence relation ϵ. We have then $[o_1]\epsilon \leq [o_2]\epsilon$.

From the definition of the precision relation for object classes from O, we conclude that $\epsilon \leq \epsilon$ as the result of the relation $[o_1]\epsilon \leq [o_2]\epsilon$. From theorem 1(c) we further conclude that $\pi \leq \pi$, where π and π are partitions of the set O. If π and π are partitions of O, we write $\pi \leq \pi$ if for every block $B \in \pi$ there exists a block $C \in \pi$ such that $B \subseteq C$.

Then, the objects that belong to the equivalence class $[o_1]\epsilon$ are more exact than those from the class $[o_2]\epsilon$ and as a result their representations are more accurate. From definition 7.2 it results that g_1 is "more precise" than g_2.

So, from definition 8, if g_1 is "more precise" than g_2 then $g_1 = Rep(g_1)$ is "more precise" than $g_2 = Rep(g_2)$.

For i. and ii. we use the definition 7.1.

6. Conclusions

This article proves that a more "relaxed" approach of the mathematical description of the process is necessary. The finite screen resolution and the finite accuracy of the system introduce visualization "error": different data sets have sometimes the same display/visualization, i.e. are mapped into the same visual object. The introduced data models allow the definition of different operations between data sets and the definition of a precision relation.

References

[1] Kaufman Arie, Nielson G., Rosenblum L. J., "The Visualization Revolution", IEEE Computer Graphics, July 1993, pp. 16–17

[2] Williams L. Hibbard, Charles R. Dyer, Brian E. Paul, "Towards a Systematic Analysis for Designing Visualizations", Scientific Visualization, IEEE Computer Society, 1997, pp. 229–251

[3] MacKinlay, "Automating the Design of Graphical Presentations of Relational Information", ACM Transactions on Graphics, Vol.5, Nr.2 1986, pp. 110–141

[4] W. Hibbard, C. Dyer, B. Paul, "A lattice Model for Data Display", Proceedings of IEEE Visualization '94, 1994, pp. 310–317

[5] Upson C., Faulhaber, Jr. T., Kamins D., Laidlau D., Schelgel D., Vroom J., Gurwitz R., van Dam A., "The Application Visualization System: A Computational Environment for Scientific Visualization", Computer Graphics and Applications, vol 9, nr.4, 1989

[6] Rădoiu D., Roman A., "Modelarea procesului de vizualizare", in Tehnologii Avansate – Aplicaţii în educaţie, Editura Universităţii Petru Maior, 1999, p. 86–101

[7] Rădoiu D., Roman A., "A Component Based Approach for Scientific Visualization of Experimental Data", Studia Universitatis "Babeş-Bolyai", Series Informatica, XLIV, 1999, no. 2, pp. 50–64
[8] Rădoiu D., "Vizualizarea Ştiintifică a Datelor Experimentale", Editura Universităţii Petru Maior, 2000
[9] V. Cruceanu, "Elemente de algebră liniară şi geometrie", Editura didactică şi pedagogică, 1973, pp. 12
[10] Gratzer G., "Universal Algebra", Springer Verlag, New-York, Berlin, Heidelberg, 1979

Petru Maior University of Tirgu Mures
E-mail address: dradoiu@uttgm.ro

Polytechnic University of Bucharest

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

A COMPRESSION METHOD FOR 3D SCENES

RADU-LUCIAN LUPŞA

Abstract. The article presents a method for compressing the representations of 3D scenes. It is a non-lossy compression, based on a variation of the Huffman algorithm and some ideeas taken from the LZ77 compression method. The method described has been succesfuly implemented and used in a commercial application.

1. Introduction

3D graphics applications need a representation of 3D scenes they work on, for storing or transmitting the information about the objects. A 3D representation must meet the following requirements:

- it must be able to represent the 3D scene with enough accuracy for the application,
- one must be able to convert it fast enough to the representation required by the 3D engine or by other parts of the aplication,
- it must be space-efficient.

There are two basic methods for describing 3D objects:

(1) as geometric bodies
(2) by voxels

As the second method leads to huge memory requirements, we will use the first one.

In order to describe the bounding surface of a 3D body, two elements must be specified:

(1) the shape of that surface
(2) the optical properties of the surface

A body surface can be approximated by an union of elements (or patches), each patch being a (finite) fragment of a plane, of a Bezier surface, or of a B-spline surface.

The relevant optical properties of the patches are:

2000 *Mathematics Subject Classification.* 68T45.

1998 *CR Categories and Descriptors.* I.2.10 [**Computing Methodologies**]: Artificial Intelligence – *Vision and Scene Analysis.*

(1) the color (or the texture of that surface)
(2) the reflection model
(3) the refraction model

The last two properties are omitted from the less elaborated representations, as they can be used for rendering by very costly algorithms only (such as ray-tracing).

If a patch has only one color, we can represent its components; if there is a texture, we represent thar texture as a 2D image (using a normal 2D image representation format) and that image is mapped onto the patch.

Having the patches planar (that is, each patch is a polygon, and the body is therefore a polyhedron) simplifies the computations but the edges are far too visible in the rendering. There are rendering methods for smoothing the rendering; the best known are the Gouraud or Phong methods [1] [7]. Any of those requires an approximation for the (real) normal vector in each of the vertices. However, computing the normals from the patches corners only is peculiar because some of the edges between the patches are to be smoothed, and others are real edges. For that reason, some of the 3D representation formats explicitly represent the normal vector of each patch in each corner.

2. Standard 3D formats

Several 3D formats (for instance, `.obj` (for a front end for OpenGL) and `.3ds` (3D Studio) are constructed the following way

First of all, we have a list of 3D points, a list of 3D vectors, a list of 2D points, and a list of 2D images (the latter being represented using a standard 2D format — for instance, *gif*, *jpeg*, *tiff*, and may even be stored in different files). Next we have the description of the facets (patches). Each facet description contains:

- the vertices list
- the list of the normals in each vertex (in case the face is not planar so the rendering should be smoothed)
- the texturing image
- the 2D points on the texturing image, corresponding to the facet vertices

Each of these pieces of information are in fact the index, in the list at the beginning of the file, of the corresponding 3D point, 3D vector, 2D image, or, respectively, 2D point.

3. Compressing the 3D scene

The methods described above still contain a lot of redondancy. Eliminating this redondancy would lead to a better compression.

In the following we will start from a `.obj`-like 3D format and will try to compress it.

The first source of redondancy consists in the fact that a typical application will output the vertices and vectors in approximatively the same order they are used

by the facet descriptions. This sugests us to write in the compressed file the list of differences between the succesive vertex indices of each face, and to compress those differences using the Huffman algorithm [4] [5], with some modifications inspired from other compression techniques.

The first change will be to make an adaptative Huffman algorithm. In the original algorithm, the coding table is computed in a first pass over the input file and written into the compressed file; then the information is encoded using that table.

The modified algorithm will start with a fixed encoding table. As it sees the input data, it computes the frequency table. At some predefined moments (for instance, when the number of already-processed symbols is a power of 2), the encoding table is regenerated based onto the frequency table.

The decoder starts with the fixed encoding table. Relying on it, the decoder can read and decode the first symbols, till the first encoding table recomputing. At that time, the decoder will have exactly the same frequency table as the encoder, and therefore it will generate the same encoding table, so it will be able to continue the decoding process.

The second modification concerns the handling of rarely-used symbols. As we saw earlier in this section, the input symbols for the Huffman compression are the differences between the indices of two succesive points on a facet. These differences, if the indices are 32-bit integers, lay in the interval $-2^{31}+1..2^{31}-1$, but values above a few hundreds are rare. For that matter, statistical data are irrelevant for predicting future occurences of those values. So, we will slightly change the Huffman algorithm the following way: for the Huffman algorithm, we will consider all values outside the interval, let's say, $-127..127$ as being equal. this way, the Huffman part sees 256 distinct symbols, one for each number in the interval $-127..127$ and one for all the other numbers. For the numbers outside the interval $-127..127$ we output the Huffman code of that symbol plus 32 bits representing the actual value.

Sometimes we have a second source of redondancy in the point and vector components. Let's take the sequence of the x coordinates of the points. If there are points grouped in planes orthogonal to the Ox axis, we get repeating values in that sequence. So, instead of coding the actual values, we will code the distance from the last appearance of that value.

4. Conclusions

The method described in the previous section was implemented by the author and is used in a commercial application for sending descriptions of 3D scenes over the Internet. The scenes are output by a CAD-like program and are between 300kB and 5MB in *obj* format. A simple conversion from text to binary reduces the size to one half, and a *zip*-like program reduces it to 1/4..1/5 of the original

size. The compression ratio aquired by the program using the method described above is 1/8..1/10.

References

[1] P. Burger, D. Gilles. *Interactive Computer Graphics.* Addison-Wesley Publishing Company, 1990

[2] F. Preparata, M. Shamos. *Computational Geometry.* Springer-Verlag, 1988

[3] R. Gonzales, R. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1993

[4] Al. Spătaru. *Teoria transmisiunii informaţiei — Information Transmission Theory.* Editura Tehnică, Bucureşti 1965

[5] X. Marsault. *Compression et cryptage de l'information — Information Compression and Encrypting.*

[6] R. Lupşa. *A Method for Compressing Static Images Using Spline Functions.* Proceedings of the "Tiberiu Popoviciu" Itinerant Seminar of Functional Equations, Approximation and Convexity, Cluj-Napoca, May 21–25, 1996

[7] C. van Overveld, B. Wyvill *Phong Normal Interpolation Revised.* ACM Transactions on Graphics, oct. 1997, vol. 16, no. 4

[8] Hee Cheol Yun, Brian Guenter. *Lossless Compression of Computer-Generated Animation Frames* ACM Transactions on Graphics, Oct. 1997, vol. 16, no. 4

[9] M. Lounsbery, T. D. DeRose, J. Warren. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type.* ACM Transactions of Graphics, Jan. 1997, vol. 16, no. 1

Faculty of Mathematics and Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania

E-mail address: rlupsa@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

TRAINING PROBABILISTIC CONTEXT-FREE GRAMMARS AS HIDDEN MARKOV MODELS

ADRIAN DUDA, GABRIELA ŞERBAN, DOINA TĂTAR

Abstract. It is considerred in this moment that the use of mathematical statistics methods in natural language processing represents a leading topic in NLP. Statistical methods have first been applied in the "speech-recognition" area. While Hidden Markov Model (HMM) is unanimously accepted as a mathematical tool in this area, its advantages have been less used in dealing with understanding natural language. In this paper we propose a method for association of a HMM to a context-free grammar (CFG). In this way, learning a CFG with a correct parsing tree will be realized by learning a HMM.

Key words: probabilistic contex-free languages, hidden Markov models, natural language processing.

1. Hidden Markov Model (HMM).

HMM model is a generalization of Markov chains, being possible that more arrows to go out for a given input. As in an HMM we can have more paths coverred for the same input, it implies that $P(w_{1,n})$ (which is the probability to have as input a sequence made up of n words, $w_1\, w_2 \cdots w_n$, shortly written as $w_{1,n}$) is calculated as the sum of the probabilities on all the possible paths. Probability on a given path is calculated by multiplying the probabilities on each segment (arrow) of that path.

Definition

An HMM is a 4-element structure $< s^1, S, W, E >$, where S is a (finite) set of states, $s^1 \in S$ is the initial state, W is a set of input symbols (words), and E is a set of transitions (labelled arrows). We consider the following order of the elements of the sets S, W, E: $S = (s^1, \cdots s^\sigma)$; $W = (w^1, \cdots w^\omega)$;

$E = (e^1, \cdots e^\epsilon)$.

Let us notice the difference between w_i and w^i: the first one means the i-th element (word) of an input sequence, while the second one is the i-th element of the W set. A transition is defined as a 4-element structure: (s^i, s^j, w^k, p) ,

---

2000 *Mathematics Subject Classification.* 68Q42, 65C40.

1998 *CR Categories and Descriptors.* F.4.2 [**Theory of computation**]: Mathematical Logic and Formal languages – *Grammars and other rewriting systems*; G.3 [**Mathematics of Computing Probability and Statistics**]: Markov processes.

representing passing from state s^i to state s^j for input w^k, transition evaluated as having the probability p. As for a given input sequence we have more possible paths, the states that it has been passed through is not deductible from input, but *hidden* (this gives the name of the model we focus on). The sequence of states $s_1, s_2, \cdots, s_{n+1}$ that it has been passed through for an input $w_{1,n}$ is marked by us shortly with $s_{1,n+1}$.

Algorithm to find the highest-probability-path. In the followings, we are using *Viterbi* 's algorithm to find the most probable path. Formally written, we have to find

$$argmax_{s_{1,n+1}} P(w_{1,n}, s_{1,n+1})$$

where $w_{1,n}$ is a sequence of input(entrance) words, and $s_{1,n+1}$ is the set of states that has been passed through. The main idea of the algorithm is calculating the most probable path beginning with the empty input sequence, and processing one word at a time, then the next word that comes in the input sequence. At every step, we calculate the most probable sequence of states which ends up with the state s^i, $i = 1, \cdots, \sigma$, where σ is the total number of states of the Markov model. Formally, we denote: $\sigma_i(t+1)$ *is the most probable sequence of states when it was given as input the sequence of words* $w_{1,t}$ *and the final state being* s^i. The highest-probability-path we are looking for is

$$\sigma_i(n+1) = argmax_{s_{1,t+1}} P(w_{1,t}, s_{1,t}, s_{t+1} = s^i)$$

and has as final state s^i. Dynamic programming principle, that is fundamental for Viterbi's algorithm, allows us to make the following remarque: the highest-probability-path to a state s^i, when it is given as input the sequence $w_{1,t}$, is made up of the maximum-probability-path with the input $w_{1,t-1}$, with the final state (let us note) s^k , which is making the multiplication $P(\sigma_k(t))P(s^k \stackrel{w_t}{\rightarrow} s^i)$ as maximal, and juxtaposing this so-obtained path with the state s^i. Saying the above in another way, $\sigma_i(t+1)$ is calculated like this:

$$\sigma_i(1) = s^i, i = 1 \cdots \sigma \;\; \sigma_i(t+1) = \sigma_j(t) \circ s^i \;\; , j = argmax_{k=1,\sigma}(P(\sigma_k(t))P(s^k \stackrel{w_t}{\rightarrow} s^i)).$$

In the above formula, "$\circ$" represents concatenation.

Algorithm to calculate the probability of an input sequence. We are mentioning here two algorithms to calculate the probability of an input sequence. Let us note by $\alpha_i(t+1)$ the probability that the input sequence $w_{1,t}$ be accepted, and having s^i as the final state. In other words:

$$\alpha_i(t+1) = P(w_{1,t}, s_{t+1} = s^i), t > 0 \qquad (1)$$

Let us notice that having all $\alpha_i(n+1)$ values calculated, the probability $P(w_{1,n})$ is given by:

$$P(w_{1,n}) = \sum_{i=1}^{n} \alpha_i(n+1).$$

Considering that $w_{1,0}$ is the empty word, which has the acceptance probability 1, we have that $\alpha_j(1) = 1., if\ j = 1, and\ it\ is\ 0\ otherwise$, corresponding to the fact that the initial state of every path is s^1. Calculation of $\alpha_j(t)$ is done starting with $\alpha_j(1)$, $\alpha_j(2)$ and going until $\alpha_j(n+1)$, using the recursive relation:

$$\alpha_j(t+1) = \sum_{i=1}^{\sigma} \alpha_i(t) P(s^i \stackrel{w_t}{\rightarrow} s^j).$$

The probabilities $\alpha_i(t)$ are called *forward* probabilities. It is also possible to calculate *backwards* probabilities, $\beta_i(t)$, with the following definition: $\beta_i(t)$ represents the acceptance-probability of input $w_{t,n}$, if the state at step t is s^i. So:

$$\beta_i(t) = P(w_{t,n} \mid s_t = s^i), t > 1.$$

The probability we are looking for will be

$$\beta_1(1) = P(w_{1,n} \mid s_1 = s^1) = P(w_{1,n})$$

Calculation of β function is done starting with values:

$$\beta_i(n+1) = P(\epsilon \mid s_{n+1} = s^i) = 1, i = 1, \cdots, \sigma.$$

For the recursive case, we have:

$$\beta_i(t-1) = P(w_{t-1,n} \mid s_{t-1} = s^i) == \sum_{j=1}^{\sigma} P(s^i \stackrel{w_{t-1}}{\rightarrow} s^j)\beta_j(t)$$

Training Markov models. The training algorithm of a Markov model used in this paper is the Baum-Welch algorithm (or forward-backward). This one, having given a certain *training* input sequence, it ajusts the probabilities of transitions in the HMM, so that the respective sequence have an as big as possible acceptance probability. Application of the algorithm has as prerequisite an HMM structure already having been defined, and only the probabilities of transitions still remaining to be established. The probabilities of transitions are calculated with the formula [2]

$$P(s^i \stackrel{w^k}{\rightarrow} s^j) = \frac{C(s^i \stackrel{w^k}{\rightarrow} s^j)}{\sum_{l=1,m=1}^{\sigma,\omega} C(s^i \stackrel{w^m}{\rightarrow} s^l)} \qquad (2)$$

The C function in the above formula is calculated like this [2]:

$$C(s^i \stackrel{w^k}{\rightarrow} s^j) = \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} \alpha_i(t) P(s^i \stackrel{w^k}{\rightarrow} s^j) \beta_j(t+1) \qquad (3).$$

What can be immediately noticed in this formula is that, for the calculation of $C(s^i \stackrel{w^k}{\rightarrow} s^j)$ we need to know path-probabilities, and so, the probabilities of transitions for the HMM model. Therefore, we start with some 'guessed' probabilities, calculated with the help of the formula (3) the new values of function

$C(s^i \overset{w^k}{\to} s^j)$ and then we adjust the probabilities of transitions using the formula (2). The indicator showing the improvement level of probabilities is the growth of the probability of input sequence $P(w_{1,n})$ compared to the previous estimation. The process of recalculating transition probabilities is finished when these probabilities no more suffers modifications considered as important.

2. Probabilistic context-free grammars.

Definition [3, 2]

A probabilistic context-free grammar (PCFG) is a 5-element structure $< W, N, N^1, R, P >$ where

- $W = \{w^1, \ldots, w^\omega\}$ represents a set of terminal symbols (we also call them words);
- $N = \{N^1, \cdots, N^\nu\}$ represents a set of non-terminal symbols, and N^1 is the initial(start) symbol S;
- R is a set of rules of form $N^i \to \xi^j$, where $\xi^j \in (N \cup W)^*$;
- P is a probability function associating to every rule $N^i \to \xi^j$ a probability $P(N^i \to \xi^j)$ so that the sum of probabilities of the rules having as left-side member (deploying) the same non-terminal is 1.

The probability of a sequence $w_{1,n}$ is equal with to sum of the probabilities of all possible syntactical trees for the analysis of $w_{1,n}$. The probability of a tree is given by the multiplication of the probabilities of the used rules : $P(T) = \prod_{rule\ r\ used\ in\ T} P(r)$.

2.1. **Associating an HMM model to a PCFG grammar.** In the followings, we are providing a way of associating an HMM model to a PCFG grammar, so that each derivation tree corespond to a path in HMM. This allows us to calculate the probability of a sequence as the probability of an accepted sequence by an HMM. In order to describe the algorithm of attaching a HMM to a PCFG, we suppose that the PCFG is in Chomsky-normal-form, that is, all the rules have the form:

$$X \to Y\ Z\ or\ X \to a$$

where X, Y, Z are non-terminals, and a is terminal. There are three possible situations to be discussed:

Case I. A rule has the form $p_X : X \to Y\ Z$ and there exist the rules $p_Y : Y \to a$ and $p_Z : Y \to b$. A derivation tree using these rules looks like that given in the figure 1.

The corresponding path in an HMM is shown in the figure 2.

Case II. A rule has the form $p_X : X \to Y\ Z$, and there exist the rules $p_Y : Y \to U\ V$, $p_U : U \to a$, $p_V : V \to b$, $p_Z : Z \to c$. A derivation tree using these rules looks like in the figure 3.

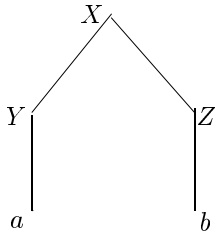


Fig. 1

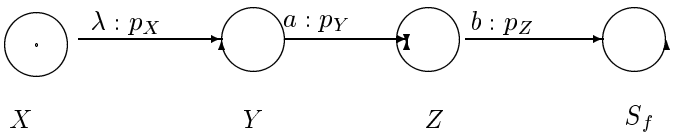


Fig. 2

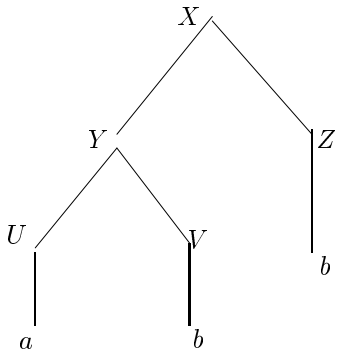


Fig. 3

The corresponding path in an HMM is shown in the figure 4.

Case III. The rule has the form $p_X : X \to Y\,Z$, and there exist the rules $p_Z : Z \to U\,V$, $p_Y : Y \to a$, $p_U : U \to b$, $p_V : V \to c$. A derivation tree using these rules looks like in figure 5.

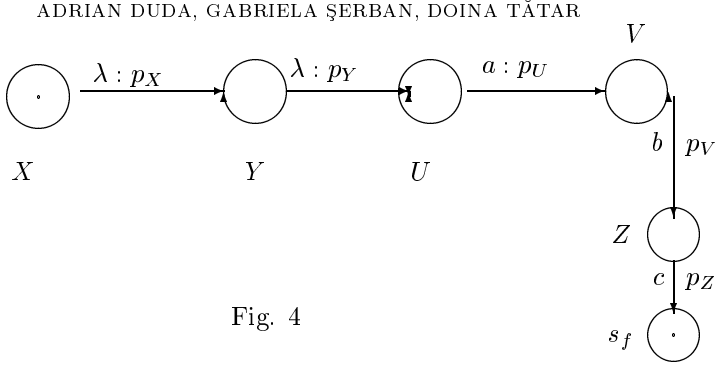


Fig. 4

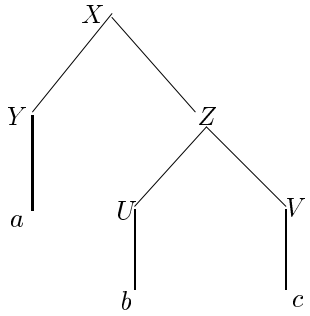


Fig. 5

The corresponding path in an HMM is shown in the figure 6.

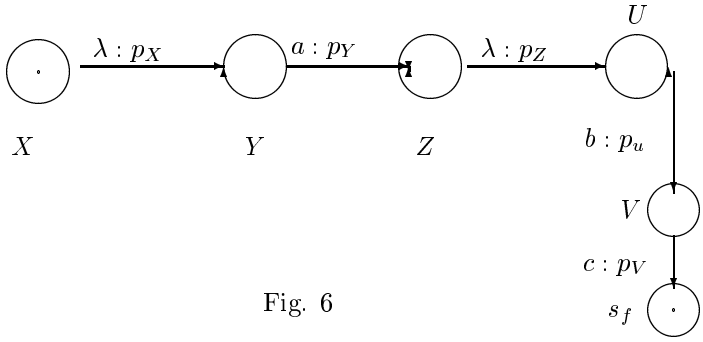


Fig. 6

Definition

A sequence $w_{1,n} \in L(G)$, where G is a PCFG, with the probability $P(w_{1,n})$, if there is a parsing tree T with the root S and the product of rules used for T is $P(w_{1,n})$.

Definition

A sequence $w_{1,n}$ is accepted by a HMM H with the probability $P(w_{1,n})$ if there is a path from S (the start node of HMM) to the final node s_{fin} and the product of probabilities on edges is $P(w_{1,n})$.

Theorem

If G is a probabilistic context-free grammar, H is the HMM associated with G as above and the sequence $w_{1,n} \in L(G)$ with the probability $P(w_{1,n})$, then $w_{1,n}$ is accepted by H.

Proof Let us consider that $w_{1,n} \in G$, where G is in Chomsky normal form. We will prove by induction on the length m of the longest path in the parsing tree of $w_{1,n}$ that $w_{1,n}$ is accepted by H. If the length m is 1, then $w_{1,n} = a$ and in H there is a path (as in Case IV above) from S to s_{fin} labeled by a , of the same probability. We will suppose that the implication is true for each sequence obtained by a parsing tree with the longest path $m-1$ and let as suppose that the sequence $w_{1,n}$ is obtained by a parsing tree T with the longest path m. In this parsing tree the first rule used is of the form $p_S : S \to Y\,Z$. In the tree T Y and Z are roots of parsing tree T_1 and T_2, with the longest path at most $m-1$ and with the frontiers P_1 and P_2. The frontier of T is $w_{1,n}$ so $w_{1,n} = P_1P_2$. Consider that we are in the Case I (the others cases are proved analogously). In this case P_1 is $a = w_1$ and P_2 is $w_{2,n}$. By induction hypothesis P_2 is accepted by a HMM with the start symbol Z. The situation in Case I is as in figure 7.

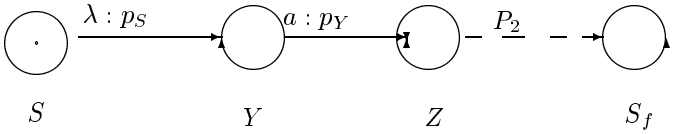


Fig. 7

So, H accepts $a\,P_2 = w_{1,n}$. The probability $P(w_{1,n})$ in T is obtained as the product between p_S , the probability of P_1, and the probability of P_2. In the Case I the probability is: $p_S \times p_Y \times P_2$.

3. Training PCFG - grammars.

The training of the PCFG-grammars is obtained based on the training algorithm for HMM, by passing from a grammar to an HMM, as in the above mentioned theorem. As we have said, Baum-Velch algorithm for training an HMM needs

a given structure to be applied to. Just the same, for a PCFG, it is supposed that the rules have already been defined. Let us consider the phrases used for the training process as *"parenthesis-ed"*, which means, it is defined the way to obtain items from lower-level items (closer to the border of the derivation tree). In order to exemplify, look at the phrase: *" Salespeople sold the dog biscuits"* [2]. Let us now describe the needed steps when training a PCFG, and use the above phrase for better understanding. Parenthesis-ing *(Salespeople (sold (the dog biscuits)))*, generates the folowing rules: $s \to np\,vp$; $np \to noun$; $np \to det\ noun\ noun$; $vp \to verb\,np$. A second possible parenthesis-ing (incorrect) is: *(Salespeople (sold (the dog) biscuits))*. According to this, we have the rules: $s \to np\,vp$; $np \to noun$; $np \to det\ noun$;$vp \to verb\,np\ np$. The overall set of rules we have obtained is shown below, where the sum of probabilities of the rules having the same non-terminal in the left-side member is $s \to np\,vp$: 1.0; $np \to noun$: 0.5; $np \to det\ noun\ noun$: 0.25; $np \to det\ noun$: 0.25; $vp \to verb\,np$. 0.5; $vp \to verb\,np\ np$: 0.5. We are transforming the so-obtained rules to be in Chomsky-normal-form, starting with the rule $np \to noun$: 0.5 and in the next step we are modifying the rules containing more than two non-terminals in their right-side member. After these being done, the final grammar becomes: $s \to np\,vp$: 0.50; $s \to noun\,vp$: 0.50; $np \to det - n\ noun$: 0.50; $np \to det\ noun$: 0.50; $vp \to verb\,np$. 0.20; $vp \to verb\,noun$. 0.20; $vp \to verb - np\ np$: 0.20; $vp \to verb - np\ noun$: 0.20; $vp \to verb - n\ noun$: 0.20; $det - n \to det\ noun$: 1.0; $v - np \to verb\,np$: 1.0; $v - n \to verb\,noun$: 1.0; $noun \to salespeople$: 0.35; $noun \to biscuits$: 0.35; $noun \to dog$: 0.40; $verb \to sold$: 1.0; $det \to the$: 1.0. Let us now consider the first (correct) parenthesis-ing of the phrase: *(Salespeople (sold (the dog biscuits)))*. The derivation-tree of figure 8 corresponds to this situation.

The path in the HMM is given in figure 9.

As far as concerns the second (incorrect) parenthesis-ing, its corresponding derivation-tree looks like in figure 10.

Initially, both of the derivation trees have the same probability (0.00245). After the grammar has been trained with the correctly paranthesised phrase, the correct tree has the probability 0.037037 while the incorrect one has the probability 0.000.

4. The application.

The application is written in Borland Pascal. The application has three parts:

- the first part(A) reads a HMM from a text file
- the second part(B), having as input a HMM and a given entry sequence, finds the probability and also the most probable paths for the entry sequence
- the third part(C), executes the training of the given HMM, for a given entry sequence.

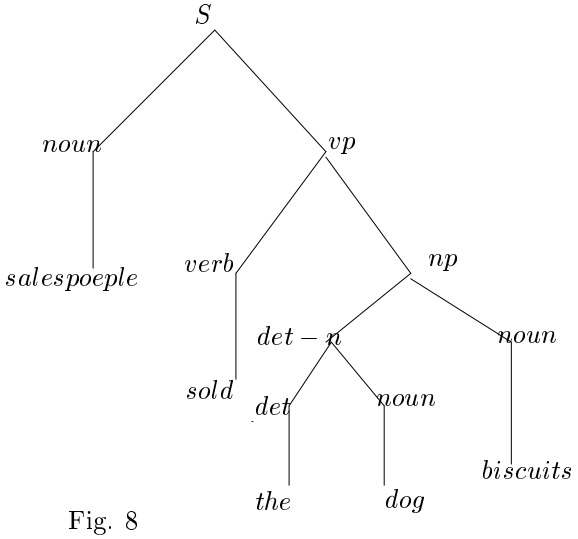


Fig. 8

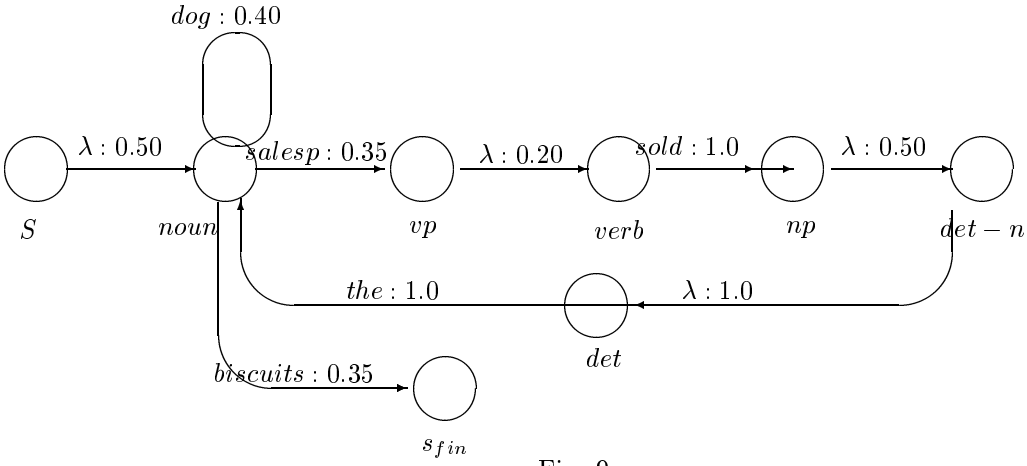


Fig. 9

The algorithms used in the second and the third part of the application are described above.

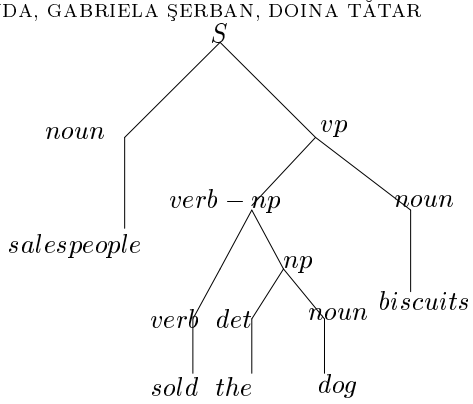


Fig. 10

The application will be sent at request by the second author. In the following we will describe shortly this application.

The input data are read from a text file, which contains the given HMM, in fact **the number of states, the set of states, the initial state** and **the set of transitions**. We have to specify that the number of entries and the set of the entries is not read from the input file, but is automatically calculated from the set of transitions. We assume that each transition is identified by three components $s1, p, s2$, where s1 and s2 are states, and p is the probability of the transition from s1 to s2 and also, a state is identified by a character (of course, this assumption is not restrictive, if is necessary, a state could be identified by a string).

Constants.

•$MaxNrStari = 25$
- the maximum number of states
•$MaxNrIntrari = 15$
- the maximum number of entries
•$MaxNrDrum = 10$
- the maximum number of paths

Data types.

```
sir=array[1..MaxNrStari] of char
```

- defines the type of the set of states of the HMM (each state is represented as a character)

```
tranzitii=array[1..MaxNrStari,1..MaxNrIntrari,1..MaxNrStari] of real
```

- defines the type of the set of transitions (the structure of a transition was described above)

```
mat=array[1..MaxNrStari,1..MaxNrIntrari] of real
```

- defines the type of a matrix with $MaxNrStari$ lines and $MaxNrIntrari$ columns, for representing the data type of the probabilities $\alpha_i(t)$

```
sirs=array[1..MaxNrDrum] of string;
```

- represents the type corresponding to the array of paths in the HMM (a path is represented as a *string* - an array of characters).

Global variables.

•s - a variable of type sir; represents the set (array) of states
•w - a variable of type sir; represents the set (array) of entries
•p - a variable of type $tranzitii$; represents the set (array) of transitions
•$sigma$- a variable of type $integer$; represents the number of states
•ni - a variable of type $integer$; represents the number of entries
•si - a variable of type $integer$; represents the index of the initial state in the set of states
•$alfa$ - a variable of type mat; represents the matrix containing as elements the probabilities $\alpha_i(t)$ (for a given entry sequence)
•$beta$ - a variable of type mat; represents the matrix containing as elements the probabilities $\beta_i(t)$ (for a given entry sequence)
•y - a variable of type $string$; represents an entry sequence (we assume that the length of this sequence is less or equal than $MaxNrIntrari$)

The algorithm performs the following steps:

Part A

• reads the input data(the HMM) from the text file.

Part B

• reads an entry sequence
• determines for the given entry y, the probabilities $\alpha_i(t)$ and $\beta_i(t)$ (for all $i \in [1..sigma]$ and $t \in [1..length(y)+1]$
• using α and β (calculated at the preceding step), on determine the probability of the entry sequence y
• determines and displays the most probable paths for the entry sequence

Part C

• reads the training entry sequence
• trains the HMM for the entry sequence, using the Baum-Welch algorithm

Subprograms used.

Part A

(P) *procedure citire*($var\ sigma : integer; var\ s, w : sir; var\ p : tranzitii; var\ si : integer; var\ ni : integer$)

- reads the input data (the number of states, the set of states, the set of entries, the set of transitions, the initial state, the number of entries) from a text file

Part B

(F) $function\ apare(x : string; s : sir; ns : integer) : integer$

- determines the index of the string x in the array s having the dimension ns

(F) $function\ alfa\_j\_tplus1(alfa : mat; j, t : integer; y : string) : real$

- calculates $\alpha_j(t+1)$ for the entry sequence y

(F) $function\ beta\_i\_tminus1(alfa, beta : mat; i, t : integer; y : string) : real$

- calculates $\beta_i(t-1)$ for the entry sequence y

(P) $procedure\ calcul\_alfa\_beta(var\ alfa, beta : mat; y : string)$

- using the two above described functions, calculates the probabilities $\alpha_i(t)$ and $\beta_i(t)$ (for all $i \in [1..sigma]$ and $t \in [1..length(y)+1]$

(P) $procedure\ det\_prob\_intrare(y : string; var\ pro : real)$

- calculates the probability p of the entry sequence y, as the sum of the elements from the last column (length(y)+1) of the matrix $alfa$

(P) $procedure\ det\_drum\_cel\_mai\_probabil(y : string; var\ n : integer; var\ z : sirs; var\ max : real)$

- determines the most probable paths for the entry sequence y, each path having the probability max (n represent the number of paths, z represent the array of the most probable paths)

(P) $procedure\ afisare\_drum\_cel\_mai\_probabil(y : string; n : integer; z : sirs; pro : real)$

- displays the most probable paths for the entry sequence y (the paths retained by the above described procedure)

Part C.

(F) $function\ calcul\_c\_i\_k\_j(i, k, j : integer; y : string) : real$

- calculates the value of the numbering function C for the states i, j and the transition y[k] (y is the entry sequence), using the relation (3) given in subsection 1.1; this function uses the values $\alpha_i(t)$ and $\beta_i(t)$ calculated in part B

(P) $procedure\ antrenare\_hmm$

- trains the HMM using a training entry sequence and the Baum-Welch algorithm

Examples.

Part B.

Let us consider the following input file

3 - the number of states
s - the first state
b - the second state
f - the third state
s - the initial state
s 0 s 0.05 - the following lines contain the transitions
s 1 s 0.05
s 0 b 0.9
b 1 s 0.3
b 0 s 0.5
s 1 f 0.1
b 0 f 0.1
b 1 f 0.1

If the entry sequence is *001*, then the results are
• the probability of the entry sequence is *0.0859*
• the probability of the most probable path for the entry sequence is *0.0450*
• the most probable path for the entry sequence is $sbsf$

Part C.

Let us consider the following input file, which codifies the HMM described in section 3.

9
s - the state "S"
a - the state "noun"
b - the state "vp"
c - the state "verb-np"
d - the state "verb"
e - the state "np"
h - the state "det-n"
g - the state "det"
f - the final state "s-fin"
s - the initial state
s l a 0.5 - the transitions
a d a 0.4 - "d" codifies "dog"
a S b 0.35 - "S" codifies "Salespeople"
a b f 0.35 - "b" codifies "biscuits"
b l c 0.2 - "l" codifies "λ"
b l d 0.2
c l d 1.0
d s e 1.0 - "s" codifies "sold"
e l h 0.5

e l g 0.5
h l g 1.0
g t a 1.0 - "t" codifies "the"
a S f 0.35
d s f 1.0
a d f 0.4
s l e 0.5
a d b 0.40

The sequence which codifies the correct sentence *(Salespeople(sold(the dog biscuits)))* (1) is *lSlslltdb*.

The sequence which codifies the incorrect sentence *(Salespeople(sold(the dog) biscuits))* (2) is *lSllsltdb*.

The sequence which codifies the incorrect sentence *(The dog(sold(the dog biscuits)))* (3) is *lltdlslltdb*.

The results obtained for this HMM are the following

• in the given HMM, without training, the sentences 1 and 2 have the same probability *0.00245* and the sentence (3) has the probability *0.0014*

• after training the HMM for the first sentence (1), the probability for the sentence 1 is *0.037037*, the probability for the sentence 2 is *0.00* and the probability for the sequence (3) is *0.00*

References

[1] J.Allen : " Natural language understanding", Benjamin/Cummings Publ. , 2nd ed., 1995.
[2] E. Charniak: "Statistical language learning", MIT Press, 1996.
[3] D. Jurafski, J. H. Martin: "Speech and Language Processing", Prentice Hall, 2000.
[4] S.J.Russell, P.Norvig: "Artificial intelligence.A modern approach", Prentice-Hall International,1995.
[5] D. Tatar: "Unification grammars in natural language processing", in "Recent topics in mathematical and computational linguistics", ed. Academiei, Bucuresti, 2000, pg 289-300.

Faculty of Mathematics and Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania

E-mail address: gabis|dtatar@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

A JAVA-BASED OBJECT-ORIENTED INFRASTRUCTURE FOR HPCC

MARIN IUGA AND BAZIL PÂRV

Abstract. This paper present a Java-based object-oriented infrastructure for an High Performance Computing Center (HPCC). This infrastructure has several functional levels: user- and server-interaction (at client level), and identificating and getting all the relevant information (at communication protocol level). The main functionality of the server is to establish the link between the algorithms requested by the client and their storage environment, by offering additional assistance to clients while browsing through the algorithms collection.

Keywords: High performance computing, Java technologies, client-server architecture, object-oriented infrastructure.

1. The general structure of HPCC

This work is based especially on [2], trying to concretize the abstract specification of a High Performance Computing Center (HPCC) given there. It continues other works on the same topic (see [1] and [3]).

The HPCC application has several functional levels. Figure 1 below presents the way these levels are structured, taking into account the functional needs for data manipulation, and the functional dependencies between them.

As we see in Figure 1, there are five significant functional levels, each level using extensively services exposed by the previous ones. On its turn, each functional level has several sections, each with its specific services.

The first level (starting from top to bottom), **AD user level**, is user interface one; it allows the user to navigate, visualize, or search data contained in the algorithm store. Usually, this level will be an applet running on client machine. This applet will communicate with the data server either using a specific network protocol, or RMI. At this moment, there is no final decision concerning this issue. The main task of this applet is to capture user's needs and to generate queries for the data server. On its turn this server will process these queries by using the

2000 *Mathematics Subject Classification.* 68N19.

1998 *CR Categories and Descriptors.* D.2.2 [**Software**]: Software Engineering – *Design Tools and Techniques.*

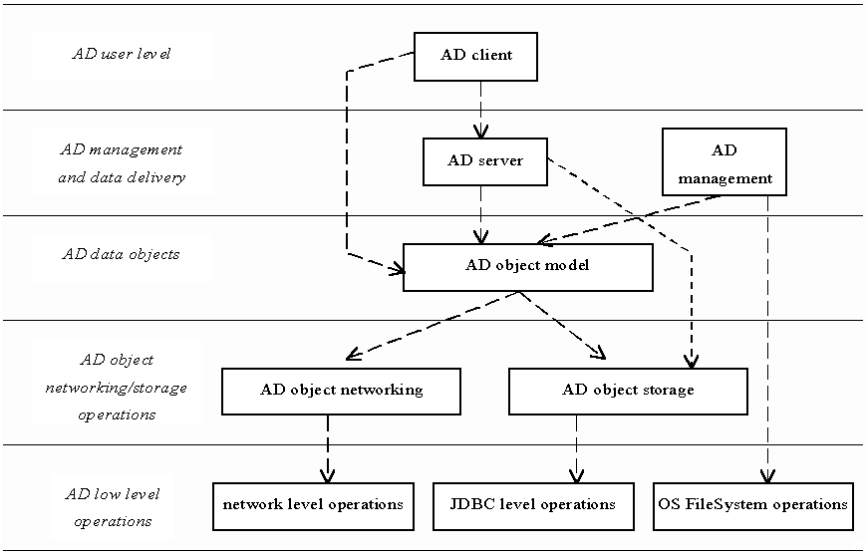


FIGURE 1. HPCC - Functional levels

services offered by the object model. Also, the client level will use the services exposed by the AD data objects level in order to manipulate those objects.

The next level, **AD management and data delivery**, has two different sections: AD server and AD management. The second section, **AD management**, is designed as a separate JAVA application. By using the services exposed by AD data objects level, its functionality covers maintenance of data about algorithms and classes of algorithms.

The first section, **AD server**, deals with data transmission to client applet. If a specific network protocol is used for client communication, AD server needs to be a daemon JAVA application running on server machine. In this case, the protocol for data transmission needs to be defined and implemented. In the second case, which uses RMI for data transmission, there will be a set of interfaces for ensuring communication between client and server. In fact, AD server will be a collection of such interfaces and some additional classes used to implement queries for algorithm store. This second approach for the AD server has the advantage of a simpler implementation, and the drawback of working with JAVA clients only.

The third level, **AD data objects**, is the core part of all applications which constitute HPCC. This level define the structure of objects which manipulate data referring to algorithms and groups of algorithms and implements a series of useful operations on them. These issues are discussed in detail in Section 2 and 3.

The fourth level, **AD object networking/storage operations**, is responsible with storing and transmitting the objects across the network. It has two sections: AD networking and AD object storage. The upper level will use the services of **AD object storage** in order to store/retrieve objects, as we discuss in the third section. **AD networking** exposes services for packing-transmission across the network-unpacking operations.

The basic level of the HPCC application, **AD low level operations**, defines some primitive operations. Functionality of this level has to be fulfilled by using some standard JAVA packages, including JDBC, and the usual functions of the operating system.

Note the pyramidal structure of the application, in which each level is using extensively only the operations exposed by the level below. This structure was designed keeping in mind the functional decomposition of the task and using a stepwise approach for abstractions.

2. Algorithm Store data scheme

Data about algorithms and groups of algorithms are modeled via two objectual counterparts, which use the relational paradigm for assuring their persistence. There are two levels of storing information:

- identification: map each algorithm/group to its specific main folder; this information is kept in two relational tables;
- data: data are stored in standardized structure of different files and folders in the main folder (description, source code Pascal and C++, JAVA applets).

The information contained in **Algorithm Store** is structured in two organizational levels: the algorithm level, **AlgorithmTable**, and the algorithm group level, **GroupTable**. Also, there are some internal tables.

All the tables are managed by the **AD object storage** section from **AD object networking/storage operations level**, which use **JDBC** and **OS FileSystem** operations, located on lowest level.

This section discusses the data scheme, while the next section is discussing the corresponding objects.

2.1. **AlgorithmTable.** Algorithm Store contains data referring to algorithms and groups of algorithms. Each algorithm is characterized by the following attributes: a name, a description file containing its goal, its parameters, and, in the case of functions, the result type. Other attributes include algorithm implementation, using a common programming language (C++, Pascal), and/or the corresponding applet, which is executed on client machine (see Table 1).

As we see in Table 1, the information contained in each line is a kind of directory information. The way this information is used to store all the data referring to an algorithm is as follows:

TABLE 1. The structure of **AlgorithmTable**

Attribute	**Description**
AlgorithmIndex	integer representing the algorithm id in the algorithm table
GroupIndex	integer representing the group id in the GroupTable (id of the class the algorithm belongs)
AlgorithmName	string representing algorithm name
HasDescription	boolean value: True if the algorithm has a description file and False otherwise
HasPascalCode	boolean value: True if the algorithm has a Pascal implementation and False otherwise
HasCPPCode	boolean value: True if the algorithm has a C++ implementation and False otherwise
HasApplet	boolean value: True if the algorithm has a corresponding JAVA applet and False otherwise
KeyWords	string, a list of keywords, separated by commas; the keywords are used in queries

- all the information referring to an algorithm is stored in a folder, called *algorithm main folder* and named `00-AlgorithmIndex`; its sub- folders are discussed below
- `description`: contains the file `description.html`, containing the algorithm description (this subfolder exists only if `HasDescription = True`)
- `pascal`: contains the file `pascal.html` containing the pascal source code (this subfolder exists only if `HasPascalCode = True`)
- `cpp`: contains the file `cpp.html` containing the C++ source code (this subfolder exists only if `HasCPPCode = True`)
- `applet`: contains the start file `applet.html` and all the necessary JAVA files for this applet (this subfolder exists only if `HasApplet = True`).

Note that the fields `HasDescription`, `HasPascalCode`, `HasCPPCode`, and `HasApplet` are redundant in the **AlgorithmTable**, because one can test the existing subfolders in the algorithm main folder. The reason is increasing the speed of the applications which use this table.

2.2. **GroupTable.** Each algorithm belongs to a unique algorithm group (e.g. sorting algorithms, searching algorithms, string pattern-matching algorithms, numerical analysis algorithms and so on). On its turn, a group of algorithms can be divided into subgroups in a tree fashion. The usual attributes for algorithm group are its name and a description file which contains the common features of its algorithms.

The structure of **GroupTable**, which contains data referring to algorithm grouping is detalied in Table 2.

TABLE 2. The structure of **GroupTable**

Attribute	**Description**
GroupIndex	integer representing the group id in the group table. Root group has the index 1
UpperGroupIndex	integer representing the id of the parent group in the group table. For the root group this index equals 0
GroupName	string representing group name
HasDescription	boolean value: True if the group has a description file and False otherwise (usually this flag is True)
KeyWords	string, a list of keywords, separated by commas; keywords are using in queries

All the information referring to a group of algorithms is stored in a folder, called *group main folder* and named `00-GroupIndex`; it contains the file `description.html`, i.e. the group description (this file exists only if `HasDescription = True`).

2.3. **Internal tables.** Algorithm Store also contains several internal tables, designed for a better implementation of its functionality. By using these tables, Algorithm Store server builds several URLs and then sends them to the client applet. On its turn, the client applet displays these URLs in the browser window. Thes internal tables are:

UnusedAlgorithmIds: unused algorithm ids (due to algorithm delete operations)

UnusedGroupIds: unused group ids (due to group delete operations)

GlobalData: contains context information: the root path for the file and folder structure and the prefix used for building URLs.

3. ALGORITHM STORE OBJECT MODEL

Both data and operations concerning algorithms and groups are modeled using objects. Both algorithms and groups are considered objects, which are manipulated by using a specific manager object. The designed classes are:

AlgorithmInfo: models the algorithm object

GroupInfo: models the algorithm group object

ObjectDBManager: models the object manager, which performs load/store operations on objects. Because all objects are stored in a relational database, store and load operations need some specific transformations (i.e. linearization).

The object model also contains some support classes, needed for object propagation across network. These classes are not full implemented. `AlgorithmInfo` and `GroupInfo` classes belong to **AD object model** level, while `ObjectDBManager`

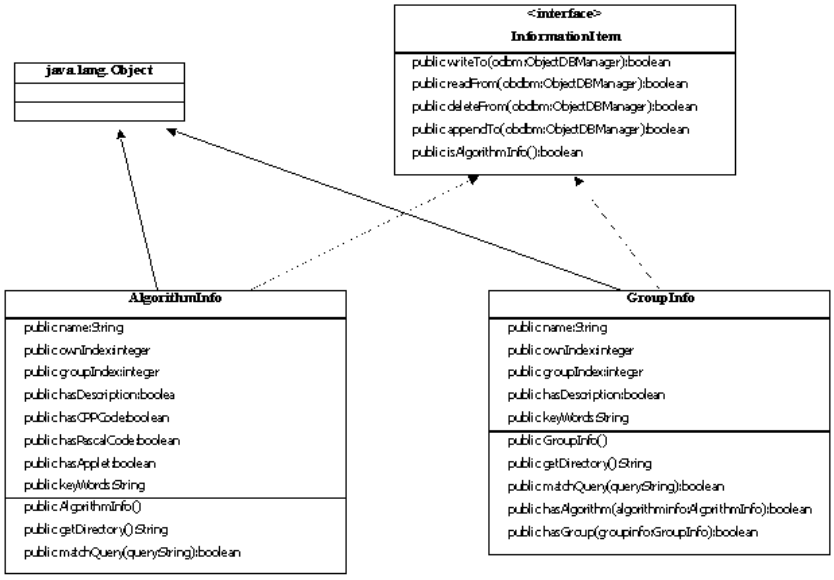


FIGURE 2. Class diagrams – AlgorithmInfo and GroupInfo

is the core of AD object storage section of **AD object networking/storage operations** level.

3.1. `AlgorithmInfo` **and** `GroupInfo` **classes.** Figure 2 presents class diagrams for `AlgorithmInfo` and `GroupInfo`. Both classes are derived from `java.lang.object` and implement the interface `InformationItem`.

Note the 1:1 mapping between their attributes and the structure of corresponding tables (**AlgorithmTable** and **GroupTable**). In order to speed up data manipulation and to decrease memory usage, all attributes are considered `public` (instead of declaring them `private` and using `get/set` methods). The `InformationItem` interface contains usual data manipulation operations: write, read, delete, and append. All these operations use a reference to an `ObjectManager` object. The method `isAlgorithmInfo` is used in dynamic identification of the receptor type.

3.2. `ObjectManager` **class.** `ObjectManager`'s object main task is to make persistent `AlgorithmInfo` and `GroupInfo` objects. The roles of `ObjectManager` support class are:

- to provide the infrastructure for storing/retrievig `AlgorithmInfo` and `GroupInfo` objects in/from a relational database (which contains the tables **AlgorithmTable**, **GroupTable**, **UnusedAlgorithmIds**, **UnusedGroupIds**, and **GlobalData**)
- to help `AlgorithmInfo` and `GroupInfo` objects in managing their own persistence
- to support queries referring to an algorithm or group of algorithms.

Figure 3 presents the diagram for `ObjectDBManager` class.

The `ObjectDBManager` object does not interact directly with the files in the main folders. It is used by the server in order to know if these files exist in the folder structure. **AD Management** component is responsible with creation and updating of these files.

4. How HPCC works

The AD management section of AD management and data delivery level is responsible with creating the standard structure of folders and files. First, the AD server section of the same level, by using AD object model, creates URLs for the root group which are sent to the client. On his behalf, client displays in a tree control the structure of HPCC Algorithm Store. When the user selects a specific algorithm/group, the client applet sends the algorithm/group id to the server, and the server builds the corresponding URLs, which are sent back to the client applet, which displays them in a window.

The user can specify queries by using keywords or algorithm/group names. The parameters are directed to the AD server, which builds the query string and uses ObjectDBManager::doQuery to retrieve the results, considered as a heterogeneous collection of AlgorithmInfo and GroupInfo objects. These results are sent back to the client applet, which displays them in a window.

5. Conclusions and future work

In this moment, the core part of HPCC application is already in place. The remaining components (as client presentation, networking, AD management) will be implemented soon.

References

[1] Avram, D., M. Iurian, B Pârv, *A High Performance Computing Center Based On A Local Network*, in SYNASC 2000, The Second International Workshop on Symbolic and Numeric Algorithms for Scientific Computation, West University, Timisoara, 4-6 Oct. 2000, 87-90.

[2] Pârv, B., *A Component-Based Model for Algorithms*, Babeş- Bolyai Univ., Fac. Math. Comp. Sci. Res. Sem, Seminar on Computer Science, 20 (1998), No. 2, 53-60.

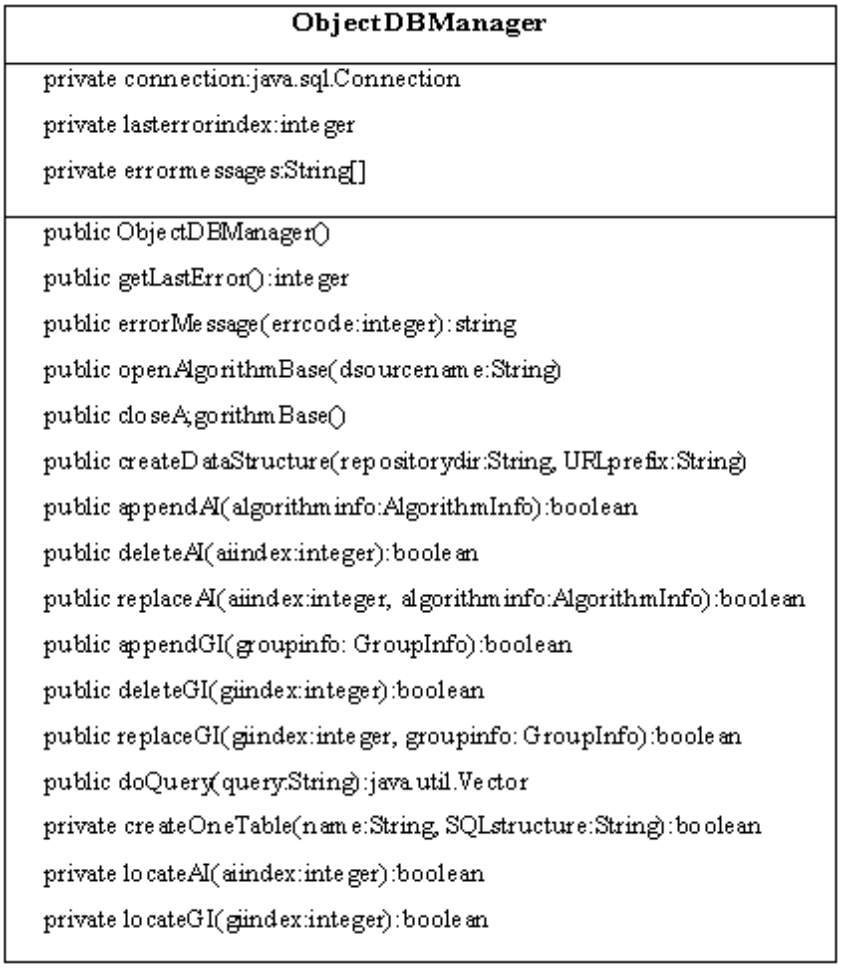


FIGURE 3. Class diagram – ObjectDBManager

[3] Pop D., S. Iurian, M. Iurian, B. Pârv, C. Mihoc, *Objectual Interfaces for Algorithm Databases*, Babeş-Bolyai Univ., Fac. Math. Comp. Sci. Res. Sem, Seminar on Computer Science, 21 (1999), No. 2, 35-42.

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE, "BABEŞ-BOLYAI" UNIVERSITY, CLUJ-NAPOCA, ROMANIA
E-mail address: marin|parv@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

GENETIC CHROMODYNAMICS

D. DUMITRESCU

Abstract. A new evolutionary search and optimization metaheuristics, called *Genetic Chromodynamics* (**GC**), is proposed. The **GC**-based methods use a variable sized solution population and a local interaction principle. Local interactions induce a restricted mating scheme and permit detection of multiple optimal solutions.

The main idea of the **GC** strategy is to force the formation and maintenance of stable sub-populations. Proposed local interaction scheme ensures sub-population stabilization in the early search stages.

Sub-populations co-evolve and eventually converge towards several optimal solutions. The number of individuals in the current population decreases with the generation. Very close individuals are merged. At convergence the number of sub-populations equals the number of optimal solutions. Each final sub-population hopefully contains a single individual representing an optimum point (a solution of the problem).

The GC approach allows as solution representation any data structure compatible with the problem and any set of meaningful variation operators.

GC-based techniques can be used to solve multimodal, static and dynamic, optimization problems.

Keywords: Evolutionary algorithms, Genetic chromodynamics, Multimodal optimization

1. Introduction

Evolutionary computing (EC) deals with adaptive search and optimization techniques that simulate biological evolution and adaptation processes. EC mainly includes Genetic algorithms (GAs), Evolution strategies, Evolutionary programming and Genetic programming. Genetic algorithms represent the most typical instance of EC (see [4]).

Unfortunately standard GAs can not solve all kinds of optimization and search problems, like GA – hard or deceptive problems [8]. While one of the main difficulties arises from the premature local convergence, other difficulties concern the multimodal optimization problems. Standard GAs, as well as the other usual evolutionary procedures, generally fail to detect multiple optimum points.

2000 *Mathematics Subject Classification.* 68T05.

1998 *CR Categories and Descriptors.* I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *Problem Solving, Control Methods, and Search.*

Several methods have been proposed to solve premature convergence and multimodal optimization problems.

Virus-evolutionary genetic algorithm (VEGA) [10] has been considered to prevent the premature local convergence due to the lack of diversity in the solution population. The VEGA approach is based on the *virus theory of evolution* according to which viruses transport segments of DNA across the species. The VEGA approach implies two populations: a *host population* and a *virus population*. The virus population realizes a horizontal propagation of genetic information in the host population. This propagation is realized by *virus infection*, i.e. by caring solution fragments (substrings) between the individuals in the host population. Therefore the VEGA technique simulates evolution with horizontal propagation and vertical (i.e. usual) inheritance of genetic information.

Multimodal genetic algorithms generally use another biological idea, namely the *niche* concept [1, 2, 6, 7, 8, 9]. Each optimum region in the search space will be considered as a niche. Niching genetic algorithms are able to form and maintain multiple, diverse, optimal solutions. Usually the niche concept is implemented through the use of fitness sharing. The niching process is accomplished by degrading the fitness of an individual according to the presence of nearby individuals.

The *sharing functions* [6] are used to calculate the extent of sharing to be performed between two individuals. For each individual the value of the sharing function is calculated with respect to the individuals in the population. The niche count of an individual is the sum of the corresponding sharing values. The fitness of an individual is divided by its niche count. The obtained updated value is the *shared fitness* of that individual.

The radius s of the estimated niches is considered. The individuals separated by distance greater than s do not degrade each other's fitness.

Sharing tends to spread the population over different optima proportionally to the values of these optima. Unfortunately the niching methods do not guarantee an appropriate selection of all useful solutions, for any situation [8]. The detection of the number of optimal solutions could be a problem, as well.

In this paper we consider a different, non-niching, strategy to prevent premature local convergence and to detect multiple optimal solutions. The proposed approach is called *Genetic Chromodynamics* (**GC**). Let us note that **GC** does not represent a particular evolutionary technique but merely a metaheuristics for solving (multimodal) optimization/search problems.

2. Genetic Chromodynamics principles

Genetic Chromodynamics metaheuristics uses a variable sized population of solutions (chromosomes or individuals) and a local mating scheme. Several solution representations can be considered. For instance solutions may be represented as real- component vectors. Solution representation as binary strings can be also

used. Proposed **GC** strategy allows any data structure suitable for a problem together with any set of meaningful variation/search operators. Moreover the proposed approach is independent of the solution representation.

The main idea of the **GC** strategy is to force the formation and maintenance of sub-populations of solutions. Sub-populations co-evolve and eventually converge towards several (local and global) optimal solutions.

The number of individuals in the population decreases with the generation. Very similar individuals (solutions) are merged. At convergence the number of sub-populations equals the number of optimal (local and global) solutions. In the standard case each final sub-population contains a single individual representing an optimum point (a solution of the problem).

A different color is assigned to each solution in the initial population. In the standard **GC** approach every solution in each generation is selected for recombination or mutation. The recombination mate of a given solution is selected within a determined *mating region*. Before recombination all solutions in a given mating region will receive the color of the best individual within that region.

A (2,1) recombination mechanism is used. The first parent is dominant and the second one is recessive. The unique offspring is labeled as the descendent of its dominant parent. The offspring will inherit its parent color. It is expected that at convergence only different colored solutions will remain in the population.

Two sub-populations will generally have different color sets. The number of colors per region tends to decrease with the time. Hopefully a dominant color will be established in each sub-population.

We may consider that the method encounters two interacting dynamics: a micro and a macro dynamics. The system micro-dynamics is associated with solution modifications. The macro-dynamics is associated with sub-populations formation, modification and stabilization. Macro-dynamics induces a dominant color within each sub-population.

We may consider each of the micro and macro-dynamics as expressing a particular aspect of the global system dynamics.

As **GC** strategy uses a variable-sized solution population, the underlying population dynamics is more complicated than in usual evolutionary algorithms. Therefore the corresponding search process may also be supposed to be more powerful. This feature makes **GC**-based searching methods appealing for solving difficult tasks, like time-dependent, multimodal and multiobjective optimization problems. The macro-dynamics of a variable sized population seems to be adequate to deal with a changing environment. Hence the proposed approach is potential useful for tackling distributed AI applications, like cooperative multi-agents.

3. Subpopulation emergence

GC-based optimization techniques start with a large arbitrary population of solutions. Dimension of the solution population decreases at each generation. There is a highly probability that each new generation will contain some individuals better than the individuals in the previous generation.

Using a local mating scheme the formation and maintenance of solution subpopulation is favored or even forced.

Sub-populations evolve towards compact and well separated solution clusters. Sub- populations within each generation $P(t)$ induce a hard partition (or at least a cover) of the set $P(t)$.

In defining sub-populations we may consider a biological point of view. Recombination of individuals in the same sub-population is highly expected. The probability of mating individuals belonging to the same sub-population is greater than the probability of mating individuals from different sub-populations. Recombining individuals in different sub-populations is not definitively forbidden but usually it is very improbable. Therefore we may say that sub-populations are composed of highly compatible (with respect to recombination) individuals.

GC approach is essentially based on local interactions in a variable-sized population. The role of local mating scheme (local solution interactions) and that of variable sized population may be summarized as follows:

(i) to ensure early sub-population formation and stabilization;
(ii) to avoid massive migration between sub-populations approximating different optimum points (migrations could affect the quality of some already obtained 'pure' or high quality solutions);
(iii) to prevent destruction of some useful (high quality) sub-populations;
(iv) to ensure a high probability of obtaining all useful problem solutions.

Local interaction principle needs a slight modification of the variation search operators.

4. Mating region

Let us consider a distance concept (a metric or a pseudo-metric) δ defined on the solution space Y. Consider an initial population in which each solution has a different color. Let f be the fitness function. As usual $f(c)$ evaluates the quality of the solution c.

As only short range interactions between solutions are allowed, the mate of each solution c has to belong to a neighborhood of c. It is usually convenient to consider this neighborhood as the closed ball $V(c, r)$ of center c and radius r.

We may interpret the parameter r as the *interaction radius* (or *interaction range*) of the individual c. Short range (or local) interactions will ensure an appropriate co-evolution of the sub-populations.

All the individuals within the region $V(c, r)$ receive the color of the best individual in that mating region. The search process starts with a population whose individuals have all different colors.

An adaptation mechanism can be used to control the interaction range r, so as to support sub-population stabilization. Within this adaptation mechanism the interaction radius of each individual could be different. In this way the flexibility of the search process may increase significantly. Each sub-population may have a more independent evolution (more freedom degrees of its behavior). To control the domain interactions we can use a general (problem independent) method or a particular heuristic. Problem dependent approaches seem to be appealing for dealing with some particular situations.

Let us note that the meaning of the mating region $V(c, r)$ is not that of a niche. The resources of this region are not shared between its members, as in the niche approach. It is more suitable to interpret the *mating region* $V(c, r)$ as the *interaction domain* of the individual c.

For some particular problems we may admit migrations (meaning that recombination is permitted) between different interaction domains. Allowing permeable interaction domains may lead to better solutions by increasing population diversity.

5. Termination condition

Various termination conditions for the **GC** search process may be identified. Some stopping conditions may be formulated according to the particular problem considered. Other stop conditions are problem independent. Here we are interested in the second class.

A good, general, problem independent heuristics is to stop the search process if the solution population remains unchanged for a fixed number of generations. This condition represents a natural termination criterion ensuring that the search process continues only how long is necessary.

6. Selection and recombination

Genetic chromodynamics involves two types of selection schemes. Global selection supplies the parent population. Local selection is a mechanism for choosing a mate of a solution in the respective mating region.

6.1. **Global selection.** Within standard version of Genetic chromodynamics approach each solution c in the population $P(t)$ will be considered for recombination. More sophisticated global selection mechanisms may be used. Their efficiency in this context is questionable.

6.2. **Local selection.** According to the proposed local interaction scheme the mate of the solution c will be chosen from the neighborhood (mating region) $V(c,r)$ of c. Local mate selection is done according to the values of the fitness function f.

For selecting the mate of a given solution we may use proportional selection. Let m be a solution in the interaction domain (mating region) $V(c,r)$ of the solution c. The probability that m is selected as the mate of c is denoted by $p(m)$ and is defined as

$$p(m) = \frac{f(m)}{\sum_{a \in V(c,r)} f(a)}.$$

Any other type of selection compatible with the particular considered problem is permitted. Tournament selection is a very powerful mechanism and may be successfully use for mate selection.

6.3. **Recombination.** Let a be the selected partner of c. The ordered pair (c,a) generates by recombination a unique offspring. The first parent is dominant, whereas the second one is recessive.

Let d be the offspring generated by c and a. The offspring d will inherit the color of its (dominant) parent c and will be labeled as the descendent of c only.

The form of the recombination operator will be chosen according to the solution representation and the nature of the problem.

For a real valued solution representation a convex combination of the genes in c and a can be used to obtain the components of d. In the case of convex recombination the i^{th} position of the offspring d has the expression:

$$d_i = qc_i + (1-q)a_i,$$

where q is a real number in the unit interval [0,1].

7. Mutation operator

If the closed ball $V(c,r)$ – the interaction domain of c – is empty then the solution c will be selected for mutation. In this way recombination and mutation are mutually exclusive operators. Mutation may be considered as acting mainly on stray points.

An additive normal perturbation seems to be appropriate for general optimization purposes. By mutation (stray) solutions are usually drawn closer to local optimum points of the objective function. As a side effect solutions are forced towards one of the existing sub-populations.

Various solution components may suffer perturbation with different standard deviation values. In every situation the mutated solution will inherit the color of its parent, as well.

8. Mutation acceptance scheme

Within usual evolutionary algorithms generated mutations are generally unconditionally accepted. Within Genetic chromodynamics based techniques a more sophisticated acceptation mechanism will be considered.

8.1. **General acceptance mechanism.** Standard **GC** approach considers that in each generation every solution is involved in recombination or mutation. Each solution will produce, and possibly be replaced by, an offspring. Whichever is better between a dominant parent and its offspring will be included in the new generation.

According to the proposed mechanism a mutated solution (offspring), which is better than its parent, is unconditionally accepted. This acceptance scheme induces a rapid convergence of the search process.

It seems that no restriction on mutation parameter is needed if the best from parent and offspring survives. This strategy can be useful in the first stages of the search process. In the last stages it may cause a drawback of the search process.

Let us consider a solution representing an optimum point. Its descendant obtained by mutation could belong to a region corresponding to a different optimum point, having a higher fitness. The offspring could surpass its parent fitness. Therefore the offspring will survive and a useful optimum point represented by its parent is lost.

To prevent the extinction of some optimum points – especially in the last search stages - we may admit that a mutated offspring have to belong to the interaction range of its parent. We may fulfill this requirement by choosing an appropriate value of the standard deviation parameter (which ensures a high probability the offspring belongs to the interaction range). This strategy is another facet of the local interactions principle.

According to the particular implementation or to the problem at hand other acceptation mechanisms may be considered.

We may also associate an acceptance probability p to each offspring worse than its parent. A simulated annealing scheme (see [11]) may be used to control the mutated solution acceptance according to the probability value p.

8.2. **Simulated annealing acceptance.** In some situations, it is important to have an additional mechanism for preventing premature local convergence. This task may be accomplished by allowing an offspring that is worse than its parent to be accepted in the new generation. In this regard, an acceptance mechanism analogous to simulated annealing technique (see [11]) may be used.

The *cost* associated with the acceptance (maintenance) of a solution c in the new generation is defined as:

$$C(c) = K - f(c),$$

where the real constant K is chosen such that $C(c) \geq 0$, for each solution c.

Remark. The higher the fitness of a solution, the lower the cost to keep that solution in the next generation.

Let d be an offspring (obtained by recombination or by mutation) which is worse than its parent c, we have:

$$f(d) < f(c).$$

The associated cost variation is:

$$\Delta C = C(d) - C(c).$$

It is easy to see that this cost is positive. The probability p of accepting the offspring d in the new generation is

$$p = e^{-\frac{\Delta C}{kT}},$$

where $k > 0$, and T is a positive parameter signifying system temperature.

The values of the parameter k and T controlling the acceptance probability are chosen depending on the specific problem.

By subsequently lowering the temperature, the acceptance probability decreases over time. In the final search process stages very small acceptation probabilities of worse solutions are needed.

By the proposed acceptance mechanism the solutions will generally get closer to the points corresponding to small cost values (high fitness values). Let us observe that the considered acceptance mechanism does not ensure the system reaches thermodynamic equilibrium at each generation (for each value of the parameter T), like in Metropolis algorithm (see [11]) normally used in simulated annealing. We may suppose the equilibrium will be achieved only at the end of the search process.

The equilibrium corresponds to slow temperature variations. We may consider temperature decreasing according to the schedule:

$$T_g = \frac{T_1}{1 + \ln g},$$

where T_1 is the initial temperature and $g > 1$ is the generation index.

To implement the proposed mechanism a random number R having uniform distribution in [0,1] is generated. If $R < p$ then the offspring (worst than its parent) is accepted in the new generation. Otherwise its parent is accepted.

9. Adapting mutation parameter

An important problem with respect to the proposed evolutionary technique is to choose an appropriate perturbation range for the mutation parameter. A related problem concerns the development of suitable adapting technique for this parameter.

We may consider several adaptation mechanisms for the perturbation standard deviation (representing the perturbation amplitude).

To ensure the *fine tuning* of the search process in its final stages we may allow perturbation amplitude decreasing with time.

Another strategy to control the standard deviation parameter may be realized by a self- adapting process. In this case the standard deviation is included in the solution structure (genotype) and it is adapted by the effect of the variation operators.

10. Interaction- range adaptation

Usually the interaction-range is the same for all the solutions. To control sub-population stabilization we may use a mechanism to adapt the interaction radius depending on the specific problem under consideration. Generally it seems useful the interaction radius be a time decreasing parameter.

A radius control mechanism could also ensure a supplementary tuning of the search process right from the first stages.

A possibility for evolving interaction radius is to consider a symbiosis of the current population $P(t)$ and a secondary population whose individuals represent interaction ranges.

We also may consider each solution has its own interaction radius. This parameter may be included in the genotype and evolved during the search process.

11. Population decreasing and stabilization

Short-range interactions permit early solution clustering in sub-populations. Local interactions also favor sub-population stabilization. As a side effect, after a few generations, some solutions might overlap, or become very close, as two or more sub- populations might evolve towards the same optimum point. To detect the correct number of optima is necessary to have only one solution per optimum. To this end, the population size is subsequently reduced by merging similar (close in terms of distance δ) solutions.

If distance between two solutions is less than an appropriate threshold, then the two solutions are merged. This verification will be done at each insertion of a new solution in the population.

The search process stops if after a (previously fixed) number of generations no significant change occurs in the population. Here a significant change is the acceptation of a new generated offspring.

We obtain the number of optimum points as the number of solutions in the final population. Each solution in the final population gives the position of a global or local optimum point.

Therefore we may consider the **CG** approach as being merely a class of optimisation and search techniques based on the local interaction principle. Any useful heuristic may be incorporated.

12. Local and infra-local optima

By maintaining a diversity of sub-populations the Genetic chromodynamics search methods are expected to avoid the problems due to local premature convergence. The proposed approach seems also to be robust with respect to very close optimum points. Close optima may not represent distinct useful solutions, since they are merely local perturbations (due to noise, for instance) of a certain optimum point. We may call them *infra-local optima*.

For most practical problems infra-local optima are solutions of no interest. Local optima of fractal functions may represent an interesting example of such useless solutions. Infra-local optima represent parasite solutions. Their detection is a time-consuming task. Furthermore parasite solutions can also generate confusion in interpreting the results.

13. Applications

Genetic chromodynamics is intended as a general optimization/search technique. GC-based methods are particularly suitable for solving multimodal and multiobjective optimization problems.

Genetic chromodynamics can also be used to solve mathematical problems that traditionally are not treated by evolutionary approaches. Examples of such problems are: equation solving (algebraic, differential or integral equations), fixed point detection and equation systems solving.

The GC approach may be used to solve real - world optimization problems. Genetic chromodynamics flavor methods can be also applied in various scientific, engineering or business fields involving static or dynamic (process) optimization.

Clustering, data compression and other data mining problems are very suitable for a GC treatment. Genetic chromodynamics clustering based methods can be particularly useful to detect the optimal number of clusters in a data set and the corresponding set of useful prototypes. The method is effective even for a very few number of data points (one data point per class, for instance).

14. Conclusions

An evolutionary metaheuristics is proposed. This metaheuristics is called *Genetic Chromodynamics strategy*. **GC** implementations generate a new class of search/optimization techniques. The **GC** approach uses a variable-sized population and local interactions among solutions. Within the methods in the **GC** family solutions are supposed to have different colors. Population dynamics is accompanied by a color dynamics. Short-range interactions permit early sub-populations

emergence. The considered local interactions also guarantee the sub-populations maintenance and stabilization.

The solution sub-populations evolve towards the local and global optimum points. The final population contains as many solutions as (global and local) optimum points are detected.

Genetic Chromodynamics strategy is intended to prevent local premature convergence and to solve multimodal optimization and search problems. One of the important features of the **GC**-based techniques is their robustness with respect to local perturbations of the optimum points.

Genetic chromodynamics is a flexible method allowing the incorporation of different general or problem-depending heuristics. We have already exemplified this ability by using a version of simulated annealing to control the acceptance mechanism of a new solution. A similar mechanism could be used to control the mutation process. For some particular problems considering elements of tabu search (see [5]) could ameliorate the performance of the GC method.

Therefore we can consider the Genetic chromodynamics approach as being merely a class of optimization and search techniques based on the principle of local interactions and using a variable- sized population. Each particular chromodynamics technique may also incorporate any useful heuristic.

References

[1] Booker, L., Improving the performance of genetic algorithms in classifier system, J.J. Grefenstete (Editor), *Proceedings of the First International Conference on genetic Algorithms*, Lawrence Erlbaum Associates, 1985, pp 80-92.

[2] Davidor, Y., A naturally occurring niche and species phenomenon: the model and the first results, in R.K. Belew , L.B. Booker.(Editors), *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991, pp 257-273.

[3] Dumitrescu, D., Bodrogi, L., A new evolutionary method and its applications in clustering, Babes-Bolyai University Seminar on Computer Science, 2, 1998, pp. 127-134.

[4] Dumitrescu, D., Lazzerini, B., Jain, L.C, Dumitrescu, A., *Evolutionary Computation*, CRC Press, Boca Raton, 2000.

[5] Glover ,F., Laguna,.M., *Tabu Search*, Kluwer Academic, Publishers, Boston, 1997.

[6] Goldberg, D., E., Richardson J., Genetic algorithms with sharing for multimodal function optimization, Proc. 2^{nd} Conference on Genetic Algorithms, 1987, 41- 49.

[7] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, MA, 1989.

[8] Goldberg, D. E., Deb, K., Horn, J., Massive multimodality , deception , and genetic algorithms , in R. Manner , B. Manderick (Editors) , Parallel Problem Solving from Nature, Elsevier,1992, 37-46.

[9] Mahfoud, S.W., *Niching Methods for Genetic Algorithms* Ph.D. Thesis, University of Illinois, 1995.

[10] Shimojima, K., Kubota, N., Fukuda, T., Virus-evolutionary genetic algorithms for fuzzy controller optimization, in F. Herrera, J.L.Verdegay, Eds., Genetic Algorithms.

[11] Van Laarhoven, P.J.M., Aarts, E.H.L., *Simulated Annealing: Theory and Applications*, D. Reidel Publishing.1987.

Faculty of Mathematics and Computer Science, "Babeş-Bolyai" University, RO-3400 Cluj-Napoca, Romania
E-mail address: ddumitr@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

A NEW EVOLUTIONARY APPROACH FOR MULTIOBJECTIVE OPTIMIZATION

D. DUMITRESCU, CRINA GROŞAN, AND MIHAI OLTEAN

Abstract. Several evolutionary algorithms for solving multiobjective optimization problems have been proposed ([2, 5, 6, 7, 8, 9, 10, 12, 13], see also the reviews [1, 11, 14]). All algorithms aim to give a discrete picture of the Pareto optimal set (and of the corresponding Pareto frontier). But Pareto optimal set is usually a continuous region in the search space. It follows that a continuous region is represented by a discrete picture. When continuos decision regions are represented by discrete solutions there is an information loss. In this paper we propose a new evolutionary approach combing a new solution representation, new variation operators and a multimodal optimization technique. In the proposed approach continuous decision regions may be detected. A solution is either a closed interval or a point. The solutions in the final population will give a realistic representation of Pareto optimal set. Each solution in this population corresponds to a decision region of Pareto set. Proposed technique does not use a secondary population of non-dominated already founded.

Keywords: evolutionary algorithms, multiobjective optimization, Pareto optimal set, Pareto frontier, Genetic chromodynamics.

Let $f_1, f_2, \ldots, f_N$ be N objective functions.

$$f_i : \Omega \to \mathbb{R}, \Omega \subset \mathbb{R}.$$

Consider the multiobjective optimization problem:

$$\begin{cases} \text{optimize } f(x) = (f_1(x), \ldots, f_N(x)) \\ \text{subject to } x \in \Omega \end{cases}$$

The key concept in determining solutions of multiobjective problems is that of Pareto optimality.

Definition. (*Pareto dominance*) Consider a maximization problem. Let x, y be two decision vectors (solutions) from Ω. Solution x is said to *dominate* y (also written as $x \phi y$) if and only if the following conditions are fulfilled:

2000 *Mathematics Subject Classification.* 68T05.

1998 *CR Categories and Descriptors.* I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *Problem Solving, Control Methods, and Search.*

(i) $f_i(x) \geq f_i(y)$, $\forall i = 1, 2, \ldots, n$.
(ii) $\exists j \in \{1, 2, \ldots, n\} : f_j(x) > f_j(y)$.

Definition. Let $S \subseteq \Omega$. All solutions which are not dominated by any vector of S are called *nondominated* with respect to S.

Definition. Solutions that are nondominated with respect to the entire search space Ω are called *Pareto optimal* solutions.

Pareto optimal set may consist from decision regions represented as:

(i) a set of points;
(ii) a set of disjoint intervals;
(iii) a set of disjoint intervals and a set of points.

Usual multiobjective optimization algorithms may deal with the first case. The second case is solved in a quite artificial manner. Obtained solutions represent points in a set of non-disjoint intervals. It is problematic to obtain a realistic representation of a union of continuous Pareto optimal regions using such a discrete picture.

When continuous decision regions are modeled by discrete solutions there is an information loss due to fidelity loss between continuous and discrete representations. Any multiobjective optimization problem being computationally solved suffers this fate. Methods for finding Pareto optimal set and Pareto optimal front using discrete solutions are computationally very difficult. Moreover the resulting sets are still only a discrete representation of their continuous counterparts. However the results may be accepted as the 'best possible' at a given computational resolution.

In this paper we propose a new evolutionary approach combing a non-standard solution representation and a multimodal optimization technique. In the proposed approach a solution is either a closed interval or a point. The solutions in the final population will give a more adequate representation of Pareto optimal set.

To evolve population we use a multi-modal optimization metaheuristic called Genetic Chromodynamics ([4]). Each individual from the population is selected for recombination or mutation. A mate for an interval (individual) is another interval that intersects it. If an individual has a mate then they are combined. Otherwise it is mutated. Mutation consists from normal perturbation of interval extremities.

A new variation operator called *splitting operator* is considered. By splitting an interval-solution containing a dominated point is splitted. In this way several Pareto regions existing in the same solution are separated. Performing this operation population size is increased.

Two population decreasing mechanisms are used: *merging* (if an interval is wholly contained in other interval, the first one is remove from the population)

and *vanishing* (very bad intervals are removed from the population). The algorithm stops when the optimal number of solutions is achieved. The evolutionary multiobjective procedure proposed in this paper is called *Continuous Pareto Optimal Set* (CPOS).

1. Solution representation and domination

In this paper we consider solutions are represented as intervals in the search space Ω.

Each interval-solution k is encoded by an interval $[x_k, y_k] \subset \mathbb{R}$. Degenerated intervals are allowed. Within degenerate case $y_k = x_k$ the solution is a point. To deal with this representation a new domination concept needed.

Definition. An interval-solution $[x, y]$ is said to be *interval-nondominated* if and only if all points of that interval $[x, y]$ are nondominated.

Remark. If $x = y$ this concept reduced to the ordinary non-domination notion.

Definition. An interval-solution $[x, y]$ is said to be *total dominated* if and only if each point within $[x, y]$ is dominated (by a point inside or outside the interval).

Remarks.

(i) If no ambiguity arise we will use nondominated (dominated) instead of interval-nondominated (interval-dominated).

(ii) An interval-solution may contain dominated as well as nondominates points.

A common approach of multiobjective optimization is to use a Pareto-ranking mechanism for fitness assignment (see for instance). In our interval-representation this approach is difficult to be used directly due to the infinite member of points to be tested in each interval. For this reason we propose a new approach. The idea is to approximate the concept of *total domination*. In this respect we introduce the notion of *non-domination degree*.

A non-domination concept may be introduced by considering some random points in the solution interval. The number Kxy of random points is proportional to the interval size $|x - y|$. We may define K_{xy} as

$$K_{xy} = F(|x - y|),$$

where F is a linear function.

Let S_{xy} be a set of random numbers within the solution-interval $[x, y]$. The size of the sampling set S_{xy} is equal to K_{xy}:

$$\text{card } S_{xy} = K_{xy}.$$

Definition. Non-domination degree of the interval-solution $[x, y]$ is the number N_{xy} defined as follows:

(1) $x \neq y$ then

$$N_{xy} = \frac{N_1 - N_2}{K_{xk}}.$$

where N_1 (N_2) is the number of non-dominated (dominated) points in the set S_{xy} and $K_{xy} \geq 1$.

(2) $x = y$ then

$$N_{xy} = \begin{cases} 1 & \text{if } x \text{ is non-dominated} \\ 0 & \text{otherwise} \end{cases}$$

Definition. Solution $[x, y]$ is said to be *t-nondominated* if the inequality

$$N_{xy} \geq t$$

holds. In this inequality t is a threshold, $0 \leq t \leq 1$.

2. Fitness assignment

Within our evolutionary multiobjective optimization procedure fitness assignment is realized using non- domination degree.

Let $[x, y]$ be a solution. Fitness of the solution $[x, y]$ is denoted eval($[x, y]$) and

$$\text{eval}([x, y]) = N_{xy}.$$

Remark. Proposed fitness assignment scheme may supply different fitness values for several sampling sets S_{xy}. This is not a major drawback. As a matter of fact, we may consider the statistical character of fitness assignment process as an advantage. It may results in an increasing flexibility of the corresponding search procedure.

3. Population model and search operators within CPOS procedure

For preserving all useful solutions in the population CPOS procedure use a multi?modal optimization technique. Our experiments emphasize that Genetic chromodynamics meta?heuristic proposed in [4] outperforms other standard methods like niching, restricted mating or island models. Genetic chromodynamics uses a variable?sized population and a local mating scheme.

The method allows a natural termination condition. Each solution in the last population supplies a Pareto optimal region contributing to the picture of Pareto optimal set.

Most of the multiobjective optimization techniques based on Pareto ranking use a second population that stores nondominated individuals. Members of second population P_{second} may be used to guide the search process. As dimension of secondary population may dramatically increase several mechanisms to reduce

P_{second} size have been proposed. In [13] and [14] a population decreasing technique based on a clustering procedure is considered. We may observe that preserving only one individual from each cluster implies a loss of information. Present approach does not use a secondary population. This makes CPOS procedure more robust and less costly. It does not imply a loss of information about Pareto optimal set during the search process.

3.1. **Selection for recombination.** Only most fit individuals are allowed to recombine. According to our elitist scheme only 1?nondominated individuals are recombined. Each 1-nondominated solution is considered for recombination. For each parent a restricted mating scheme is used to find the other parent. Let $[x, y]$ be an 1?nondominated solution. If the solution is the degenerate interval $x = y$ then its mate is selected from the closed ball $V(x, R)$, where R is the mating range. R represents a parameter of the procedure.

The mate of the one non-degenerate interval $[x, y]$ is selected from all one non-degenerate solutions $[u, v]$ such that they are not disjoint and do not include each other. This means that the following conditions are have to be fulfilled for recombining interval solutions $[x, y]$ and $[u, v]$:

(i) $[x, y] \cap [u, v] \neq \emptyset$;
(ii) $[x, y] \cap [u, v] \neq [x, y]$;
(iii) $[x, y] \cap [u, v] \neq [u, v]$.

The individuals that can be selected as mates of $[x, y]$ represent the breeder set of $[x, y]$. From the breeder set the second parent is selected using a certain procedure like a tournament or proportional selection schemes.

3.2. **Recombination operator.** Recombining the individual $[x, y]$ and its selected mate it results a unique offspring. The first parent $[x, y]$ will be replaced by this offspring.

If the parents are nondegenerated solutions the offspring is the union of the parent intervals.

For degenerated case the offspring may be, for instance, the convex combination of its parents.

According to the proposed recombination operator the mate of a (non) degenerated solution has to be (non) degenerated too.

3.3. **Mutation operator.** An individual $[x, y]$ is mutated if and only if no mate can be selected for it. This happens when the breeder set of $[x, y]$ is empty.

3.3.1. *Mutating an interval.* There are several ways of realizing mutation. These possibilities are:

a) mutate the left extremity of the interval;

b) mutate the right extremity of the interval;
c) mutate the both extremities of the interval.

Remarks.

(i) For extremities perturbation we use an additive normal perturbation with standard deviation σ, where σ is a parameter of the method.
(ii) Degenerated?solutions mutation is included in the general scheme.
(iii) Mutation type (a, b or c) is randomly chosen.

3.3.2. *Degenerated interval-solutions.* By mutation an interval can be reduced to a point. This may happen in the following situations:

(i) mutation of the right (left) interval extremity is less (greater) than the left (right) interval extremity;
(ii) if by mutation the interval extremities coincide (with respect to a given computational resolution).

3.4. **Splitting operator.** For segregation two disjoint Pareto regions that are represented by the same interval-solution we introduce a new type of variation operator called *splitting operator*.

Splitting operator is applied to an interval-solution and produces two offspring. This operator increases population size.

Applying recombination or mutation to all individuals in the current population $P(t)$ a new intermediary population $P^1(t)$ is obtained. Splitting operator is applied to the intermediate population $P^1(t)$.

To apply splitting operator an interval-solution $[x, y]$ is randomly chosen from $P^1(t)$. A cut-point p, $x < p < y$, is randomly chosen. If p is a dominated point then $[x, y]$ may include disjoint Pareto regions. For separating these regions we apply the splitting operator.

The offspring resulted by splitting the solution $[x, y]$ are $[x, p]$ and $[p, y]$. We may thus write

$$\text{split}[x, y] = \{[x, p], [p, y]\}.$$

Splitting operator is not applied if the randomly generated point p is nondominated.

4. Population dynamics within CPOS algorithm

To detect the correct number of Pareto optimal regions it is necessary to have only one solution per Pareto optimal region. Using Genetic Chromodynamics technique population size decreases during the search process such that eventually equals the number of optimal solutions.

Several population decreasing mechanisms may be used. In our implementation we consider two complementary schemes. Two new operators implement the

considered population decreasing mechanisms. The proposed operators are called merging and vanishing. They act only on nondegenerated solutions.

(i) **Merging operator.** If an 1-nondominated solution T_1 is completely included in another 1- nondominated solution T_2, then the solution are merged. The solution T_1 is discarded.

(ii) **Vanishing operator.** If a solution is (-1) nondominated then the solution is discarded. This operation is very useful because performing split mutation the number of bad solutions may grow considerably.

These verifications needed by the operators are done when a new solution is included in the population.

5. Stop condition

Genetic chromodynamics deals with a very natural termination condition. According to this stop condition the chromosome population remains unchanged for a fixed number of generations (given by the parameter `MaxIteration` in our algorithm) then the search process stops.

6. CPOS algorithm

Continuos Pareto optimal set (CPOS) algorithm proposed in this paper may be outlined as below:

CPOS Algorithm
begin
Population initialization:
generate randomly a interval population ($P(0)$);
$t = 0$;
Evolving intervals:
repeat
for each individual c **in** $P(t)$
if `Has_Mate`(c) {c has a possible mate}
then
select b – a mate for c; {select mate using proportional selection}
Perform recombination:
$z =$`Recombination`(b, c);
else ***Perform mutation of individual*** c**:**
$z =$`Mutate`(c);
endif
add z to intermediate population $P'(t)$;
endfor
Apply merging operator on individuals in intermediate population $P'(t)$***:***

$P''(t) =$`merge`$(P'(t))$;
Apply vanishing operator on individual on $P''(t)$***:***
$P'''(t) =$`vanish`$(P''(t))$;
$P(t+1) = P'''(t)$; {next generation}
$t = t + 1$;
until `MaxIterations` is reached
end.

Remark. Algorithm stops if there is no population modification for a number of `MaxIterations` successive iterations.

7. Numerical experiments

Several numerical experiments using CPOS algorithm have been performed. For all examples the detected solutions gave correct representations of Pareto set with an acceptable accuracy degree. Some particular examples are given below.

Example 1. Consider the functions $f_1, f_2 : [-4, 6] \to \mathbb{R}$ defined as

$$f_1(x) = x^2,$$

$$f_2(x) = (x-2)^2.$$

Consider the multiobjective optimization problem:

$$\begin{cases} \text{minimize } f_1(x), f_2(x) \\ \text{subject to} x \in [-4, 6] \end{cases}$$

Pareto optimal set for this multiobjective problem is the interval [0, 2].

The initial population is depicted in Figure 1. For a better view the chromosomes are drawn one above another.

For the value

$$\sigma = 0.1$$

of the standard deviation parameter solutions obtained after 10 generations are depicted in Figure 2.

The population obtained after 24 generations is depicted in Figure 3.

The final population, obtained after 40 generations, is depicted in Figure 4.

Final population obtained after 40 generations contains only one individual. This individual is:

$$s = [0.01, 1.98],$$

and represent a continuous Pareto optimal solution.

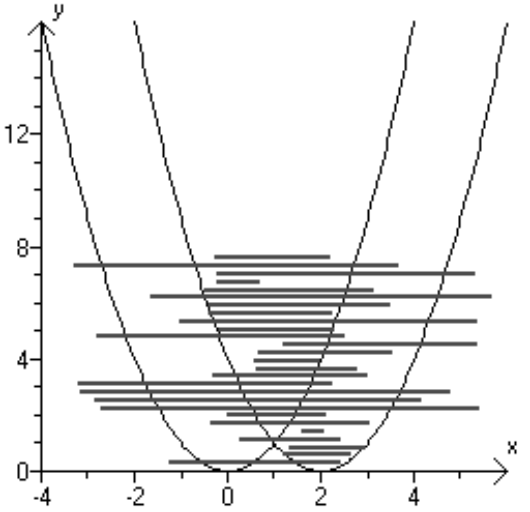


FIGURE 1. Initial population

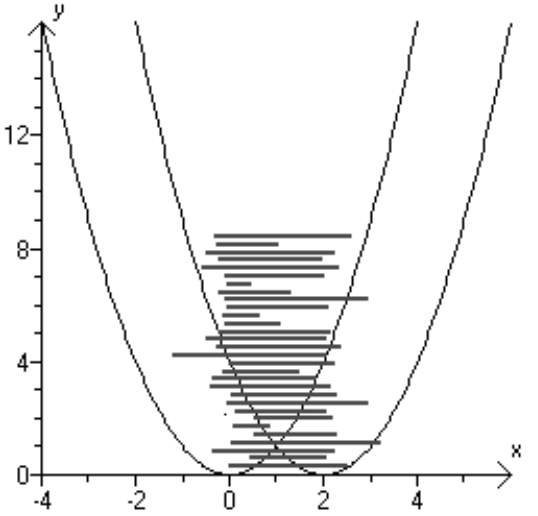


FIGURE 2. Population after 10 generations

The obtained solution accuracy may be increased, if necessary, by decreasing the parameter standard deviation of normal perturbation. Of course the number of iterations needed for convergence increases this case.

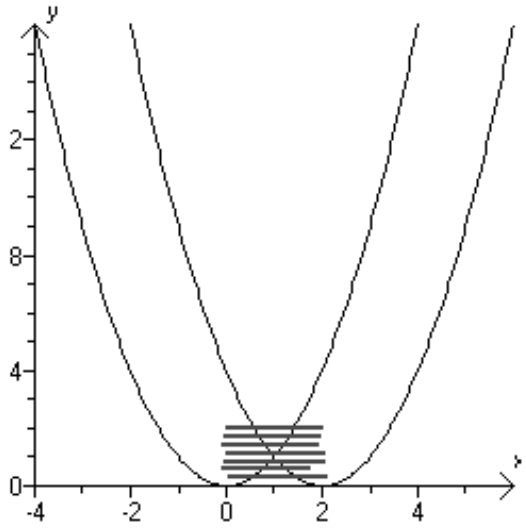


FIGURE 3. The population obtained after 24 generations

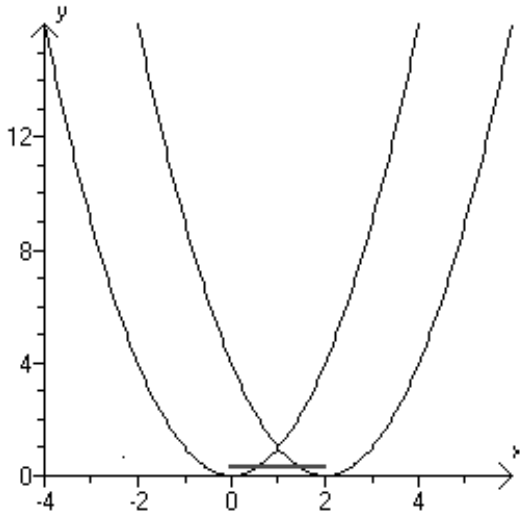


FIGURE 4. Final population obtained after 40 generations

For example, if we consider the value

$$\sigma = 0.01,$$

the solution

$$s = [0.004, 1.997],$$

is obtained after 60 iterations.

Example 2. Consider the functions $f_1, f_2 : [-10, 13] \to \mathbb{R}$ defined as

$$f_1(x) = \sin(x),$$

$$f_2(x) = \sin(x + 0.7).$$

and the multiobjective optimization problem:

$$\begin{cases} \text{minimize } f_1(x), f_2(x) \\ \text{subject to} x \in [-10, 13] \end{cases}$$

The initial population is depicted in Figure 5.

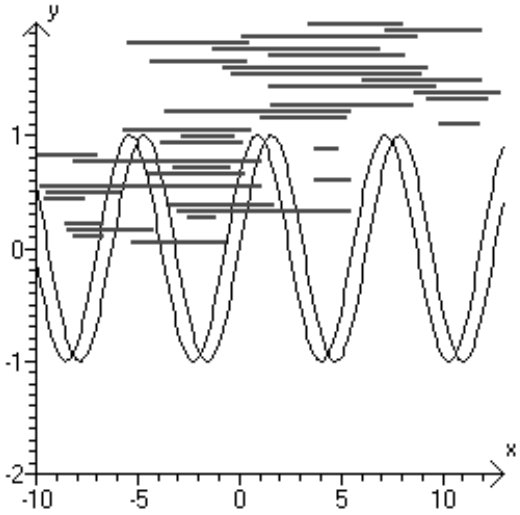


FIGURE 5. Initial population

For the value

$$\sigma = 0.1$$

solutions obtained after 5 generations are depicted in Figure 6.

We may observe four distinct, well-separated, subpopulations are already segregated after 5 generations. Therefore useful subpopulations are stabilized very

early. Let us remark that, for the sake of clarity, segments in the same class are separately represented. In reality they partially overlap.

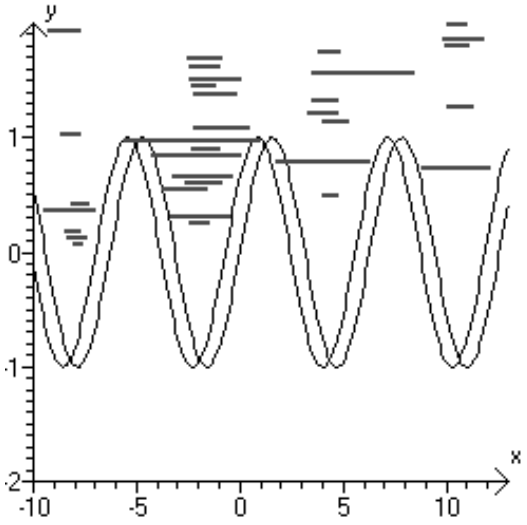


FIGURE 6. The population after 5 generations

The population after 10 generations is depicted in Figure 7. Subpopulations are well individualized and nested.

The final population, obtained after 120 generations, is depicted in Figure 8.

Solutions in the final population are:

$$s_1 = [-8.47, -7.86],$$
$$s_2 = [-2.26, -1.56],$$
$$s_3 = [4.01, 4.69],$$
$$s_4 = [10.29, 10.99].$$

Example 3. Consider the functions $f_1, f_2 : [-9, 9] \to \mathbb{R}$ defined as

$$f_1(x) = x^2,$$
$$f_2(x) = 9 - \sqrt{81 - x^2}.$$

and the multiobjective optimization problem:

$$\begin{cases} \text{minimize } f_1(x), f_2(x) \\ \text{subject to} x \in [-9, 9] \end{cases}$$

The initial population is depicted in Figure 9.

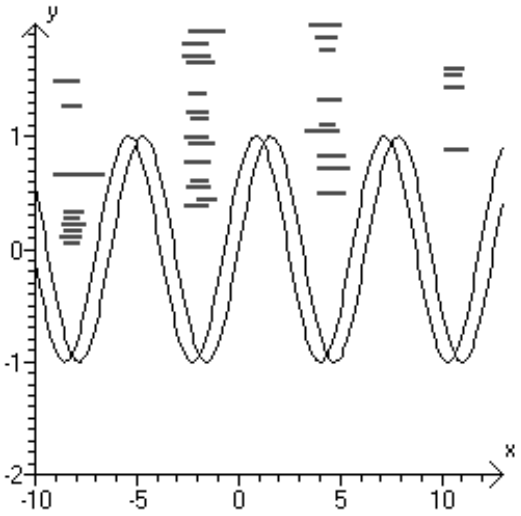


FIGURE 7. Population after 10 generations

Consider the standard deviation parameter value

$$\sigma = 0.1,$$

In this case population obtained after 3 generations is depicted in Figure 10.

It is very interesting to observe that very early population stabilizes to a single individual. This individual will be improved at subsequent iterations.

The population after 7 generations is depicted in Figure 11.

The final population, obtained after 120 generations, is depicted in Figure 12.

Final population obtained at convergence after 120 generations contains only one individual represented as degenerated interval (i.e. a point)

$$s = -0.001.$$

Therefore detected Pareto optimal set consists from a single point:

$$P_{\text{detect}} = \{-0.001\}.$$

We may remark that detected Pareto set represents a good estimation of the correct Pareto optimal set

$$P_c = \{0\}.$$

Accuracy of this estimation can be easy improved by using smaller values of the parameter ? (standard deviation). In this case a larger number of generations are needed for convergence.

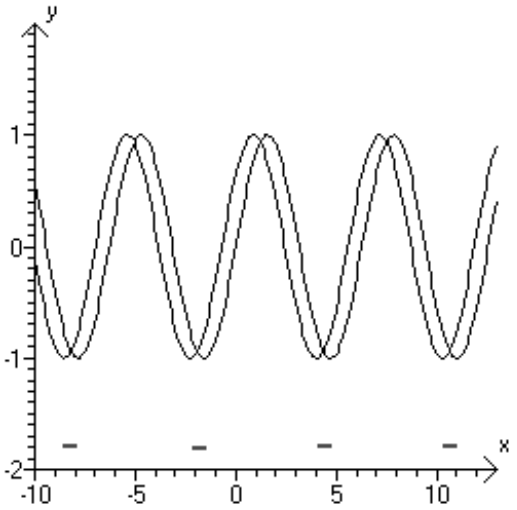


FIGURE 8. Four solutions within the final population (obtained after 120 generations)

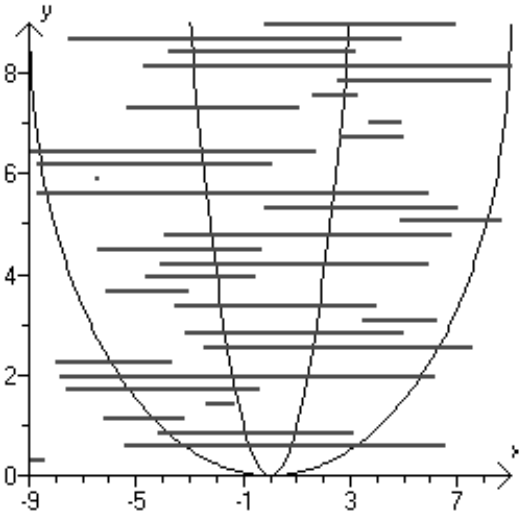


FIGURE 9. Initial population

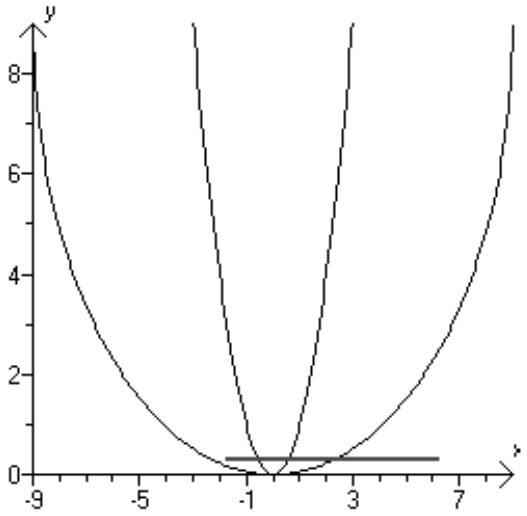


FIGURE 10. Population after 3 generations

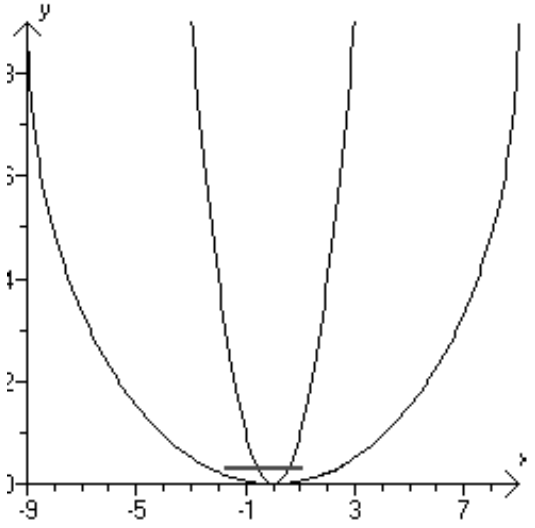


FIGURE 11. Population after 7 generations

For instance, if we put

$$\sigma = 0.01,$$

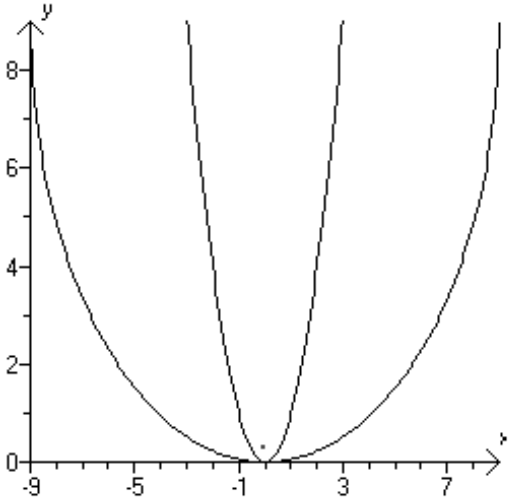


FIGURE 12. Final population obtained after 120 generations

the obtained solution is

$$s = 0.0008.$$

8. Concluding remarks and further researches

A new evolutionary technique for solving multiobjective optimization problems involving one variable functions is proposed. A new solution representation is used. Standard search (variation) operators are modified accordingly. Three new search operators are introduced. The proposed evolutionary multiobjective optimization technique does not use a secondary population of non-dominated solutions.

Proposed multiobjective optimization method uses a new evolutionary metaheuristic called Genetic chromodynamics for maintaining multiple optimal solutions on the calculated Pareto set during the search process.

All known multiobjective optimization techniques supply a discrete picture of Pareto optimal solutions and of Pareto frontier. But Pareto optimal set is usually non-discrete. Finding Pareto optimal set and Pareto optimal frontiers using a discrete representation is not a very easy computationally task (see [11]).

CPOS technique supplies directly a continuous picture of Pareto optimal set and of Pareto frontier. This makes our approach very appealing for solving problems where very accurate solutions detection is needed.

Another advantage is that CPOS technique has a natural termination condition derived from the nature of evolutionary method used for preserving population diversity.

Experimental results suggest that CPOS algorithm supplies correct solutions in a very few iterations.

Further research will concentrate on the possibilities to extend the proposed technique to deal with multidimensional domains.

Another direction is to exploit the solution representation as intervals for solving inequality systems and other problems for which this representation is natural.

References

[1] Coello, C. A. C, A comprehensive survey of evolutionary- based multiobjective optimization techniques, *Knowledge and Information Systems*, 1(3), 1999, 269-308.

[2] Deb, K., Multiobjective evolutionary algorithms: problem difficulties and construction of test problems, *Evolutionary Computation*, 7, 1999, 205-230.

[3] Dumitrescu, D, Lazzerini, B, Jain, L., C., Dumitrescu, A., *Evolutionary Computation*, CRC Press, Boca Raton, 2000.

[4] Dumitrescu, D., Evolutionary Chromodynamic, Studia Univ. Babeş-Bolyai, Ser. Informatica, 2000.

[5] Fonseca, C.M., Fleming, P.J., An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation*, 3, 1995, 1-16.

[6] Goldberg, D.E., *Evolutionary Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, 1989.

[7] Horn, J., Nafpliotis, N., Multiobjective optimization using niched Pareto evolutionary algorithms, *IlliGAL Report 93005*, Illionois Evolutionary Algorithms Laboratory, University of Illinois, Urbana Champaingn.

[8] Horn, J., Nafpliotis, N., Goldberg D. E. A niche Pareto evolutionary algorithm for multiobjective optimization, *Proc. 1st IEEE Conf. Evolutionary Computation*, Piscataway, vol 1, 1994, 82-87.

[9] Schaffer, J.D., Multiple objective optimization with vector evaluated evolutionary algorithms, *Evolutionary Algorithms and Their Applications*, J.J. Grefenstette (Ed.), Erlbaum, Hillsdale, NJ, 1985, 93-100.

[10] Srinivas, N., Deb, K., Multiobjective function optimization using nondominated sorting evolutionary algorithms, *Evolutionary Computing*, 2, 1994, 221-248.

[11] Veldhuizen, D.A.V., Multiobjective Evolutionary Algorithms: Classification, Analyses and New Innovations, Ph.D Thesis, 1999, Graduated School of Engineering of the Air Force Institute of Technology, Air University.

[12] Veldhuizen, D.A.V., Lamont, G.B., Multiobjective evolutionary algorithms: analyzing the state-of-the-art, Evolutionary Computation, 8, 2000, 125-147.

[13] Zitzler, E., Thiele, L, Multiobjective evolutionary algorithms: A comparative study and the strength Pareto approach, IEEE Trans on Evolutionary Computation, 3 (1999), 257-271.

[14] Zitzler, E., Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Doctoral Dissertation, 1999, Swiss Federal Institut of Technology Zurich,Tik-Schriftenreihe nr. 30.

Faculty of Mathematics and Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania

E-mail address: ddumitr|cgrosan|moltean@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

PHRASE GENERATION IN LEXICAL FUNCTIONAL GRAMMARS AND UNIFICATION GRAMMARS

DOINA TĂTAR, DANA AVRAM

Abstract. In this paper we compare the process of deriving a phrase structure in a lexical functional grammars with the process of obtaining feature structure for the symbol S of an unification grammar. If the $c-structure(D, C, e)$ generates the feature structure F, then F is the feature structure obtained as $MGSat(\psi)$, where ψ is a conjunction of a set of descriptions from $Desc$.

1. Lexical Functional Grammar-*LFG*

LFG is a lexical theory, this means that the lexicon contains a lot of information about lexical entries. *LFG* grammars present two separate levels of syntactic representation: *c-structure*, about constituent structures (in much the same way as derivation trees in CFG grammars) and *f-structure* , which is used to hold information about functional relations, encoded using equations between feature structures (see the next section). We will introduce here the design of the grammar rules and the lexicon, as well as the process applied to derive a phrase.

Definition

A *LFG* grammar over a set *Feats* of attributes and a set *Types* of types is a 5-uple (N,T,P,L,S) where:

- N is a finite set of symbols, called nonterminals;
- T is a finite set of symbols called terminals;
- P is a finite set of production rules

$$A_0 \to A_1, \cdots, A_n$$
$$E_1, \cdots, E_n.$$

where $n \geq 1$, $A_1, \cdots, A_n \in N$ and E_i, $1 \leq i \leq n$, is a finite set of equations of the forms:

$$\uparrow|\downarrow \phi = \uparrow|\downarrow \phi'$$

2000 *Mathematics Subject Classification.* 68Q42.

1998 *CR Categories and Descriptors.* F.4.2 [**Theory of computation**]: Mathematical Logic and Formal languages – *Grammars and other rewriting systems.*

$$\uparrow|\downarrow \phi'' = v$$

with $\phi, \phi' \in Feats^*, \phi'' \in Feats^+$ and $v \in Types$;

- L is a finite set of lexicon rules

$$A \rightarrow t$$

$$E$$

where $A \in N$, $t \in T \cup \varepsilon$ and E is a finite set of equations of the form

$$\uparrow|\downarrow \phi = v$$

with $\phi \in Feats^+$ and $v \in Types$;

- $S \in N$ is the start symbol.

As an example let us consider the rule:

$$S \rightarrow NP \qquad VP$$
$$\uparrow subj = \downarrow \qquad \uparrow = \downarrow$$

The equations (or functional schemes) are interpreted as referring to the feature structures (section 2) associated, in the following way: the meta-variable $\uparrow$ refers to the f-structure that is associated with the head of the rule, $\downarrow$ refers to the f-structure associated with the daughter to which the equation is attached.

The $c-structure$ based on a LFG grammar G is a tree, in much the same way as derivation trees in a CFG grammar, but the nodes are annotated not only with elements from $N \cup T$ but also with sets of equations E. More exactly:

Definition

A tree domain D is a set $D \subseteq N^*$, (where N is the set of natural numbers, and N^* is the Kleene closure of N) such that if $x \in D$ then all prefixes of x are also in D. The out degree $d(x)$ of an element x in tree domain D is the cardinality of the set $\{i \mid xi \in D, i \in N\}$. Let us denote by $term(D)$ the set $\{x \mid x \in D, d(x) = 0\}$.

We can now define a *c-structure* based on a LFG grammar :

Definition[2]

A constituent structure (*c-structure*) based on a LFG grammar $G = (N, T, P, L, S)$ is a triple (D, C, e) where

- D is a finite tree domain;
- C is a function $C : D \longrightarrow N \cup T \cup \{\varepsilon\}$;
- e is a function $e : D \setminus \{\varepsilon\} \longrightarrow \Gamma$ where Γ is the set of all equation sets in P and L, such that $C(x) \in T \cup \{\varepsilon\}$ if $x \in term(D)$, $C(\varepsilon) = S$ and for all $x \in (D - term(D))$, if $d(x) = n$ then

$$C(x) \rightarrow C(x_1) \cdots C(x_n)$$

$$e(x_1) \cdots e(x_n)$$

is a production or lexical rule in G.

Definition

A terminal string for a *c-structure* is the string $C(x_1)\cdots C(x_n)$, with $x_1,\cdots,x_n \in term(D)$ and $x_i \leq_{lex} x_{i+1}$ for $i = 1,\cdots,n-1$.

The existence of a *c-structure* is a necessary but not sufficient condition as terminal string belongs to the $L(G)$. Nodes of the *c-structure* are associated with feature structures (denoted by f_i), and the equations induce some equations between f_i as unknowns. The minimal solution of this set of equations (if a solution exists) represents a feature structure F.

Definition

The *c-structure*(D, C, e) generates the feature structure F if F is the minimal solution of the set of equations e. We denote this by

$$F \models' \bigcup_{x\in D} e(x).$$

In the next section we will present unification grammars and will illustrate the connection between unification grammars and LFG grammars.

2. Unification Based Phrase Structure Grammars.

The unification grammars are phrase structure grammars in which non-terminal and terminals symbols are replaced by feature structures. Intuitively, a feature structure (FS) is a description of some linguistic object, specifying some or all of the information that is asserted to be true of it [3, 5]. We will present shortly two definitions of (untyped) feature structures.

Definition:

A feature structure over a signature *Types* and *Feats* is a labeled rooted directed graph represented by the tuple:

$$F =< Q, \bar{q}, \theta, \delta >$$

where :

- Q is the finite set of nodes of the graph;
- $\bar{q} \in Q$ is the root node;
- $\theta : Q \longrightarrow$ **Type** is a *partial* node typing function;
- δ : **Feat** $\times Q \longrightarrow Q$ is a partial value function, which associates with a node i the nodes $i_1,\cdots,i_n$ if $\delta(FEAT_1, i) = i_1,\cdots,\delta(FEAT_n, i) = i_n$.

In the rewriting relations two notions about FS's are important: subsumption relation and unification operation.

Definition

A feature structure F *subsumes* another feature structure G or $F \sqsubseteq G$ iff:

• if a feature $f \in$ **Feat** is defined in F then f is also defined in G and its value in F *subsumes* the value in G;

• if the values of two paths are shared in F , then they are also shared in G.

Thus, $F \sqsubseteq G$ if G contains more information than F or F is *more general* than G.

The notion of subsumption can be used to define the notion of unification, the main information combining operation in unification based grammars. Unification conjoins the information in two feature structures into a single result if they are consistent and detects an inconsistency otherwise.

Definition

The result of the unification of two FS's F and F' is an other FS (if it exists), denoted $F \sqcup F'$ which is *the most general* FS (in the sense of relation $\sqsubseteq$) subsumed by both input FS's.

Thus, $F \sqcup F'$ is the l. u. b of F and F' , if it exists, on the ordering relation $\sqsubseteq$.

The FS's can be described, as an other modality, by a logical expression, which is denoted "description". The big advantage of this kind of representing FS's is the linearity of displaying.

Definition [1] The set of descriptions over the set **Types** of types and **Feats** of features is the least set, $Desc$, such that:

$\sigma \in Desc$, if $\sigma \in$ **Types**

$\pi : \phi \in Desc$ if π is a path, $\phi \in Desc$

$\pi_1 \doteq \pi_2 \in Desc$, if π_1 and π_2 are paths

$\phi \wedge \psi, \phi \vee \psi \in Desc$, if $\phi, \psi \in Desc$

The priority among the operations is:

$$\doteq \mid : \mid \ \wedge \mid \ \vee \mid$$

A satisfaction relation between FS's and the set $Desc$ is defined as:

Definition The relation $\models$ is the least relation such that:

$F \models \sigma$ if $\sigma \in$ **Types**, $\sigma \sqsubseteq \theta(\overline{q})$

$F \models \pi : \phi$ if $F@\pi$ is defined and $F@\pi \models \phi$

$F \models \pi_1 \doteq \pi_2$ if $\delta(\overline{q}, \pi_1) = \delta(\overline{q}, \pi_2)$

$F \models \phi \wedge \psi$ if $F \models \phi$ and $F \models \psi$

$F \models \phi \vee \psi$ if $FF \models \phi$ or $F \models \psi$.

The following theorem establishes the duality between a (non-disjunctive) description and the most general FS which satisfies this description:

Theorem ([1]**)**. There is a partial function (algorithm)

$$MGSat : Non - Disj - Desc \to \mathcal{TFS}$$

such that for each ϕ and F

$$F \models \phi \ \ iff \ \ MGSat(\phi) \sqsubseteq F.$$

($MGSat(\phi)$ is constructed as most general total well typed FS which satisfies ϕ.)

Remark: The algorithm considers recursively the cases of descriptions: σ, $\pi : \phi$, $\pi_1 = \pi_2$, $\phi \wedge \psi$ and construct (learn) $MGSat(\phi)$. The most important case is:

$$MGSat(\phi \wedge \psi) = MGSat(\phi) \sqcup MGSat(\psi).$$

The UBPSG's are phrase structure grammars in which non-terminal or category symbols are replaced by FS's in rewriting rules, the lexical entries are terminals, and an inheritance hierarchy $< \mathbf{Types}, \sqsubseteq >$ is associated.

UBPSG's was introduced by Shieber (1988) [5] , Gazdar and Mellich (1989) [4].

Definition. (UBPSG) For an inheritance hierarchy $< \mathbf{Types}, \sqsubseteq >$ with an appropriateness specification, a set **Feats** of features, a set Lex of terminals (lexical entries), a UBPSG is a set of rewriting rules:

$$E_0 \rightarrow E_1 \ldots E_n,$$

where each E_i is either a feature structure or a terminal (and in this case $n = 1$).

The interpretation of such a rule is: the category E_0 can consist of an expression of category E_1, followed by the category E_2, etc.

Alternatively, the rewriting rule can be given as:

$$D_0 \rightarrow D_1 \ldots D_n$$

where D_i are descriptions, such that

$$E_i = D_i, \text{ if } D_i \text{ is a terminal}, E_i = (total\ well{-}typed\ )MGSat(D_i), \text{ if } D_i \in Desc.$$

Remarks:

If the $c\text{-}structure(D, C, e)$ generates the feature structure F, then F is the feature structure obtained as $MGSat(\psi)$, where ψ is obtained as conjunction of the set of $Desc$ as follows:

- If an equation refers to a single unknown (with the form: $f_i\pi = v$, f_i being an unknown, π being a path from $Feats^*$, $v \in Types$), then $\pi : v \in Desc$;
- If two equations are as $f_i\pi = v$ and $f_i\pi' = v'$ then $\pi : v \wedge \pi' v' \in Desc$;
- If an equation is of the form $f_i = f_j$, and $f_i \models \phi_i$ and $f_j \models \phi_j$, then $\phi_i \wedge \phi_j \in Desc$.

These remarks can be summarized in the following:

Theorem

If $F \models' \bigcup_{x \in D} e(x)$ then $F \models \psi$, where $\psi = \bigwedge_{\phi \in Desc} \phi$, and ϕ are the descriptions obtained as above.

3. Example

The lexical rules of this example from [3] are:

$$N \longrightarrow' Raluca'$$

$$\uparrow pred =' Raluca', \uparrow pers =' 3', \uparrow nr =' sing'$$

$$N \longrightarrow' marea'$$

$$\uparrow pred =' marea', \uparrow pers =' 3', \uparrow nr =' sing'$$

$$V \longrightarrow' priveste'$$

$$\uparrow pred =' priveste', \uparrow pers =' 3', \uparrow nr =' sing'$$

The nonlexical rules let be:

$$S \rightarrow NP \quad VP$$

$$\uparrow subj = \downarrow, \uparrow = \downarrow$$

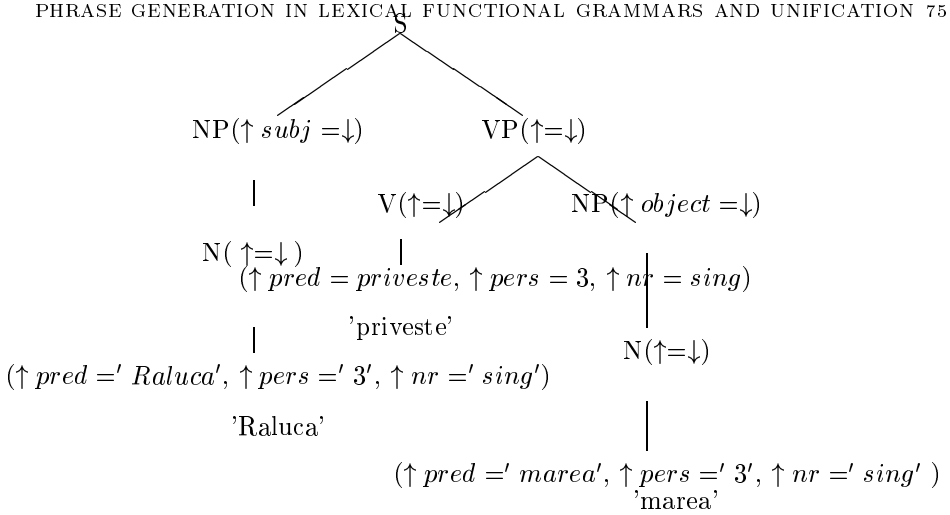
$$VP \rightarrow V \quad NP$$

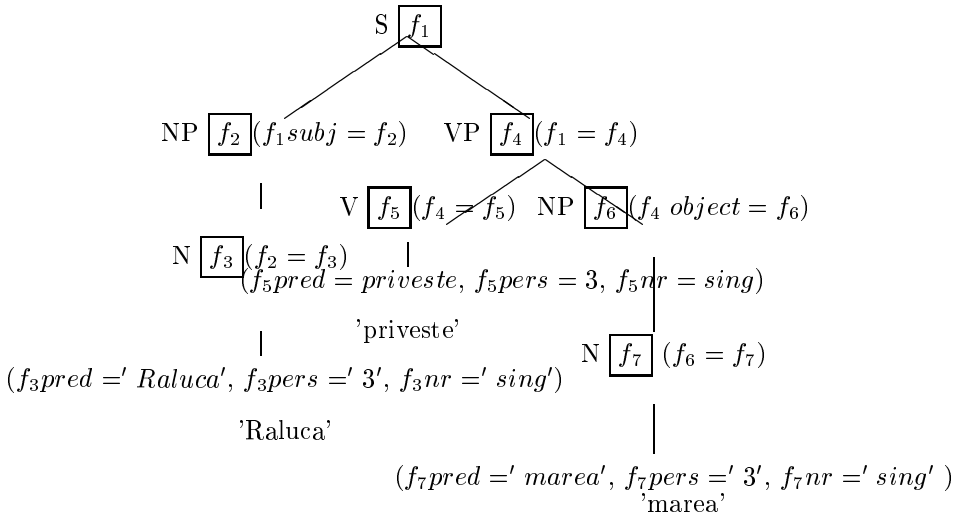
$$\uparrow = \downarrow, \uparrow obj = \downarrow$$

$$NP \rightarrow N$$

$$\uparrow = \downarrow$$

We will construct the $c - structure$ based on the above LFG grammar, than we will proceed to decorate the $c - structure$ by names of feature structures f_i and will apply the equation between them. The decorated $c - structure$ with the instantiated equations attached to its nodes for the above example is also presented as bellow.





We will proceed in the following to obtain the (minimal) solution of the set of equation (or to determining the unsolvability of it).

The steps of this procedure are:

1. Solving the set of equations referring to a single unknown (with the form: $f_i\pi = v$, f_i being an unknown, π being a path from $Feats^*$, $v \in Types$).

2. Interpreting equal unknowns with different values as results of an unification ($f_i\pi v$ and $f_i\pi' v'$ induce the feature structure $\boxed{f_i}\begin{bmatrix}\pi\ \text{v}\\ \pi'\ \text{v'}\end{bmatrix}$).

3. Removing the unknowns which are not used effectively by their equals (if $f_i = f_j$ and f_i is not defined, one use f_j).

4. Solving the equations with two feature structure names (if $f_i = a\ \ f_j$, then the feature structure $\boxed{f_i}\begin{bmatrix}\text{a} & \boxed{f_j}\begin{bmatrix}\ \end{bmatrix}\end{bmatrix}$ is obtained).

5. Solving the equations of the form $f_i = f_j$, where both feature structures f_i and f_j are defined, by unification of the values of f_i and f_j and denoting the result as: $\boxed{f_i}\boxed{f_j}\begin{bmatrix}....\end{bmatrix}$

6. As $\boxed{f_1}$ is associated with S , the feature structure for f_1 (if exists), is the feature structure of the entire *correct* phrase.

For the above example, the set of equations is:

$f_1 subj = f_2$
$f_1 = f_4$
$f_2 = f_3$
$f_3 pred =' Raluca'$
$f_3 pers = 3$
$f_3 nr = sing$
$f_4 = f_5$
$f_4 object = f_6$
$f_5 pred =' priveste'$
$f_5 pers = 3rd$
$f_5 nr = sing$
$f_6 = f_7$
$f_7 pred =' marea'$
$f_7 pers = 3rd$
$f_7 nr = sing$

By execution of the above calculus 1-4 steps we obtain the following feature structures:

$\boxed{f_1}\begin{bmatrix}\text{subj:} & \boxed{f_2}\end{bmatrix}$

$\boxed{f_3}\begin{bmatrix}\text{pred:} & \text{'Raluca'}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}$

$\boxed{f_4}\begin{bmatrix}\text{object:} & \boxed{f_6}\end{bmatrix}$

$$\boxed{f_5}\begin{bmatrix}\text{pred:} & \text{'priveste'}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}$$

$$\boxed{f_7}\begin{bmatrix}\text{pred:} & \text{'marea'}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}$$

From equations $f_1 = f_4$, $f_2 = f_3$, $f_4 = f_5$, $f_6 = f_7$, we obtain the following feature structures:

$$\boxed{f_1}\begin{bmatrix}\text{subj:} & \boxed{f_2}\,\boxed{f_3} & \begin{bmatrix}\text{pred:} & \text{'Raluca'}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}\end{bmatrix}$$

$$\boxed{f_4}\begin{bmatrix}\text{object:} & \boxed{f_6}\,\boxed{f_7} & \begin{bmatrix}\text{pred:} & \text{'marea'}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}\end{bmatrix}$$

$$\boxed{f_5}\begin{bmatrix}\text{pred:} & \text{'priveste '}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}$$

For the equations $f_1 = f_4$, $f_4 = f_5$, we apply the step 5 as above and we obtain:

$$\boxed{f_1}\,\boxed{f_4}\,\boxed{f_5}\begin{bmatrix}\text{pred: 'priveste'} & & \\ \text{subj:} & \boxed{f_2}\,\boxed{f_3} & \begin{bmatrix}\text{pred:} & \text{'Raluca'}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}\\ \text{object:} & \boxed{f_6}\,\boxed{f_7} & \begin{bmatrix}\text{pred:} & \text{'marea'}\\ \text{nr:} & \text{sing}\\ \text{pers:} & 3\end{bmatrix}\end{bmatrix}$$

The same feature structure can be obtained from descriptions as at the end of section 2.

4. Conclusions.

In this paper we replace the construction of a feature structure, given as the most general satisfier of a conjunction of descriptions, by obtaining the solution of a set of lexical rules equations. The bases of this replacing are the remarks expressed by the theorem at end of section 2.

References

[1] B.Carpenter: "The logic of typed feature structures",Cambridge University Press, 1992.
[2] T. Burheim: "Indexed languages and unification grammars" , Univ. Bergen, Norway, Internal Raport.
[3] N.Francez, S. Wintner: " Feature structure based linguistic formalisms", draft 1998, http.
[4] G. Gazdar, C. Mellish: " NLP in Prolog. An introduction to CL", Addison Wesley, 1989.
[5] M.Shieber: "Introduction to unification-based approaches to grammars", CSLI, 1986.
[6] D. Tatar, M. Lupea: "Indexed grammars and unification grammars", Studia Univ. "Babeş-Bolyai", Informatica, 1998, nr 1 , pp 39-46.

Department of Computer Science, Faculty of Mathematics and Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania
E-mail address: dtatar|davram@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

SEMANTIC ANALYSIS IN DIALOGUE INTERFACES

ADRIAN ONEŢ, DOINA TĂTAR

Abstract. One crucial issue for the NL interfaces is the use of an "intermediate meaning representation formalism" which will support the semantic and pragmatic reasoning processes of the system. The paper presents a syntactic-semantic analyzer based on the approach of lambda-calculus, realised by the first author, as a kind of syntax-driven, context independent and inference free approach. The first level of this application contains the semantic engine (written in SWI-Prolog); the second one contains an interface with the user (written in Delphi); the extra level is for the graphical representation of the parse tree (written in Visual Prolog).

1. Dialogue interfaces

A fundamental goal of artificial intelligence is the manipulation of natural languages (NL's) using the tools of computing science. The mains challenges raised by NL processing arise at many levels: conceptual model, semantic theories, parsing theories, user modeling. The NL phenomenon has some important characteristics that must be considered when one implement an NLP system [15] :

- Lack of an explicit definition;
- Presence of incomplete and ill structured sentences, without preventing the understanding;
- Influence of the context;
- Ambiguities .

These few characteristics show that NLP requires techniques different from the traditional techniques. Several scientific disciplines have made natural language an object of study: artificial intelligence, linguistics, philosophy, logic, psychology. All these attempt to answer at the question of " automatic NL understanding". The most used criterion now is the reasoning process operating on some internal representation of the meaning of the NL input.

The first major success for natural language processing (NLP) was in the area of database access. One first such interfaces was Fernando Pereira's CHAT system

2000 *Mathematics Subject Classification.* 68U35.

1998 *CR Categories and Descriptors.* I.2.1 [**Computing Methodologies**]: Artificial Intelligence – *Appplications and Expert Systems.*

(1983) about a geographical database. Over the last decade, some commercial systems have built up large grammars and lexicons to handle a wide variety of inputs."The main challenge for current systems is to follow the context of on interaction" ([10]).

One crucial issue for the NL interfaces is the use of an "intermediate meaning representation formalism" which will support the semantic and pragmatic reasoning processes of the system. Such of representation is called "intermediate logical form" and it is the principal point through which results coming from the field of logic can be used in a NL processing (NLP) system .

The semantics of the phrases expressed in a natural language has two aspects: semantics and pragmatics. Semantics refer to those aspects of the meaning that are not influenced by the context, and the pragmatics is concerned with the context and the intention of the speaker. Almost every approach for the semantic interpretation of a phrase is made with the principle of compositionality :*the meaning of a phrase is a function of the meanings of its parts* .

The dialogue-based application include [1]:

- question-answering systems, where NL is used to query a database;
- automated customer service;
- tutoring systems;
- spoken language control of a machine;
- general cooperative problem-solving systems.

A dialog interface does have to process sequences of sentences exchanged between a user and an application system. Each of these sentences has to be precisely understood. The discourse domain of one interface is usually restricted, and thus easier to model from a semantic point of view. From a historical perspective, can be distinguished three generations of NL interfaces [14]:

- The "direct translation systems", performing a direct translation of the NL input into an output string, suitable for the purposes of the application. The parser of such a system does not make use of a general meaning representation formalism. These systems are not portable and is difficult to implement in them the semantic inferences.

- The second generation of NL interfaces separates the understanding process into two steps: in a first step an analyzer will process the NL input and produce a representation of its meaning in an intermediate meaning representation formalism, usually an intermediate logical form (ILF). In a second step, an interpreter will study this representation and will find out related actions, accordingly with the application. Both analysis and interpretation are based on an explicit model

of the discourse domain, as a knowledge base defining the ideas referred, providing semantic and pragmatic information and performing the logical inferences necessary for understanding.

• The third generation of NL interfaces includes, besides the model of discourse domain, an explicit model of user with "static" information, such as the level of competence possessed by a specific user, and "dinamic" information expressing the knowledge and beliefs of the user and the evolution of these knowledge and beliefs within the dialogue. This kind of information can be used to improve the resolution of ambiguities, the processing of incomplete sentences and the generation of cooperative responses.

The study of intermediate meaning representation (IMR) formalism has been the subject of large disputes. The question was of deciding whether IMR should be "logical" or not (based on frames, semantic networks, conceptual dependencies, etc) [13]. Is it largely accepted that an IMR formalism must combine different kinds of elements, all of which are necessary for the interpretation process [15]:

• Logical structure;

• Conceptual content: the variables and constants of the logical notation appear as instances of a class system that provides a conceptual model of the discourse domain. This class structure can be organized hierarchically as a lattice and forms the skeleton of the knowledge base used in NL interface;

• Speech act indication representing the expected impact that the speaker tries to have on his inter locutor by uttering a proposition, depending on the nature of this utterance: request, order, information, etc. This expected impact can be modeled in terms of "wants", "knowledge" and " beliefs" of the inter locutor. The primitives expressing this levels can be logically axiomatized and support a reasoning process improving the behavior of an NL interface;

• Pragmatic annotations about determination of logical quantifiers.

The phase of interpretation of an ILF , after his production by the parser, is accomplished in some well defined steps [15]. These steps includes a set of processes as: resolution of anaphoric references, resolution of scoping ambiguities and other types of ambiguities which could not be solved in the parsing phase. Also, NL interface that process more than one isolated sentence needs a dialogue manager and the possibility to control interpretation, for example detecting wrong presupposition.

2. Semantic analysis by lambda-calculus

Semantic analysis (SA) is the process whereby semantic representations are composed and associated with a linguistic input. The sources of knowledge that are used are: the meanings of words, the meanings associated with the grammatical

structure and the knowledge about the context in which the *discourse* occurs (*semantics of the discourse*).

One approach of SA is by lambda-calculus and it is a kind of syntax-driven SA, context independent and inference free. Such approach is sufficient to produce useful results. Others two approaches are semantic grammars and information extraction [6]. The lambda-calculus SA is based on the principle of compositionality which assert that the meaning of a sentence can be composed from the meanings of its parts. The input of a semantic analyzer is an output of a syntactic analyzer , that means a parse tree or a feature structure, etc. We will assume that it is a parse tree.

In lambda-calculus approach of SA every context free grammar rule is augmented by a semantic rule which specify how to compute the meaning representation of a construction from the meanings of its constituent parts [6]. An augmented rule is :

$$A \longrightarrow \alpha_1\alpha_2\cdots\alpha_n\{A.sem = f(\alpha_j.sem\cdots\alpha_k.sem)\}, 1 \leq j \leq k \leq n$$

The denotation $A.sem = f(\alpha_j.sem, \cdots, \alpha_k.sem)$ means that the semantics of A, $A.sem$, will be obtained as a function f on the $\alpha_j.sem, \cdots, \alpha_k.sem$.

Let us consider an example generated by a small subset of rules from ATIS grammar [6]: *Continental serves meat.*

The small subset of ATIS rules is:

$$S \longrightarrow NP\ VP$$

$$VP \longrightarrow Verb\ NP$$

$$NP \longrightarrow ProperNoun$$

$$NP \longrightarrow MassNoun$$

$$Verb \longrightarrow serves$$

$$ProperNoun \longrightarrow Continental$$

$$MassNoun \longrightarrow meat$$

The augmented rules are:

$$NP \longrightarrow ProperNoun\ \{NP.sem = ProperNoun.sem\}$$

$$NP \longrightarrow MassNoun\ \{NP.sem = MassNoun.sem\}$$

$$ProperNoun \longrightarrow Continental\ \{ProperNoun.sem = Continental\}$$

$$MassNoun \longrightarrow meat\ \{MassNoun.sem = meat\}$$

These rules assert that the semantics of NP's are the same as the semantics of their individual components. In general will be the case that for non-branching

grammar rules, the semantics associated with the child will be copied unchanged to the parent.

To come up with the semantics for VP's, we will use a notational extension to first order predicate calculus (FOPC) , lambda-calculus, (Church , 1940) that provides the kind of formal parameter that we need.

The $\lambda-$expression

$$\lambda x P(x)$$

must be understand as a formula (with $P(x)$ a formula from FOPC), where the free variable x is bound to the specific terms in FOPC. The process of bounding of x with a specific term in FOPC is a $\lambda-$ reduction and is illustrate by the equality:

$$\lambda x P(x)(A) = P(A)$$

The variables denoted by λ can be in a arbitrary number and their order is the same with the order of their binding to the terms.

With λ notation the augmented rule for $Verb$ is:

$$Verb \longrightarrow serves \ \{Verb.sem = \lambda x \lambda y \exists e IS - A(e, Serving)$$

$$\wedge Server(e, y) \wedge Served(e, x)\}$$

and for VP is:

$$VP \longrightarrow Verb \ NP \ \{VP.sem = Verb.sem(NP.sem)\}$$

The calculus for $VP.sem = Verb.sem(NP.sem)$ is :

$$\lambda x \lambda y \exists e IS - A(e, Serving) \wedge Server(e, y) \wedge Served(e, x)(NP.sem) =$$

$$\lambda y \exists e IS - A(e, Serving) \wedge Server(e, y) \wedge Served(e, Meat).$$

So, $VP.sem = \lambda y \exists e IS - A(e, Serving) \wedge Server(e, y) \wedge Served(e, Meat)$.

With λ notation the augmented rule for S is:

$$S \longrightarrow NP \ VP\{S.sem = VP.sem(NP.sem)\}$$

The calculus for $S.sem$ is:

$$S.sem = VP.sem(NP.sem) = \lambda y \exists e IS - A(e, Serving) \wedge$$

$$\wedge Server(e, y) \wedge Served(e, Meat)(NP.sem)$$

$$= \lambda y \exists e IS - A(e, Serving) \wedge Server(e, y) \wedge Served(e, Meat)(Continental)$$

$$= \exists e IS - A(e, Serving) \wedge Server(e, Continental) \wedge Served(e, Meat).$$

In the applications is used another new notation that facilitates the compositional creation of the desired semantics: *complex-term*. Formally, a complex-term is an expression with the following three-part structure: $\langle Quantifier \, Variable \, Body \rangle$ The formulas which use complex-terms usually refereed as *quasi-logical forms*.

To convert a quasi-logical form in a FOPC formula we will use the following schema of rewriting any predicate having a complex-term argument:

$$P(\langle Quantifier\ Variable\ Body\rangle) \wedge U$$

$$\rightarrow Quantifier\ Variable\ (Body\ \ Connective\ \ P(Variabila) \wedge U).$$

where $Connective$ is $\wedge$ for $\exists$ and $\longrightarrow$ for $\forall$.

Let us consider the sentence: *A restaurant serves meat.*

The needed augmented rules are:

$$Det \longrightarrow a\ \ \{Det.sem = \exists\}$$

$$Nominal \longrightarrow Noun\ \ \{Nominal.sem = \lambda x IS - A(x, Noun.sem)\}$$

$$Noun \longrightarrow restaurant\ \ \{Noun.sem = restaurant\}$$

$$NP \longrightarrow Det\ Nominal\{NP.sem = \langle Det.sem\ x\ Nominal.sem(x)\rangle\}.$$

The bottom-up calculus is:

$$Nominal.sem = \lambda x IS - A(x, Noun.sem) = \lambda x IS - A(x, Restaurant)$$

$$S.sem = VP.sem(NP.sem) = (Verb.sem(NP.sem))(NP.sem) =$$

Using $VP.sem$ as above we obtain:

$$(\lambda y)(\exists e)(IS - A(e, Serving) \wedge Server(e, y) \wedge Served(e, Meat))(NP.sem)) =$$

$$(\lambda y)(\exists e)(IS - A(e, Serving) \wedge Server(e, y) \wedge Served(e, Meat))$$

$$(\langle Det.sem\ z\ (\lambda x)IS - A(x, Restaurant)(z)\rangle)$$

$$(\exists e)(IS - A(e, Serving) \wedge Server(e, \langle Det.sem\ z\ IS - A(z, Restaurant)\rangle)\wedge$$

$$\wedge Served(e, Meat))$$

$$\exists e(IS - A(e, Serving) \wedge (\exists z)(IS - A(z, Restaurant) \wedge Server(e, z))\wedge$$

$$\wedge Served(e, Meat))$$

$$(\exists e)(\exists z)(IS - A(e, Serving) \wedge IS - A(z, Restaurant) \wedge Server(e, z)\wedge$$

$$\wedge Served(e, Meat)).$$

Let us observe that a sentence as: *Every restaurant has a menu* has two semantic representation, one which corresponds to the common-sense interpretation (*every restaurant has its own menu*), but also the interpretation which state that *there is one menu that all restaurants share.*

The two interpretation are obtained processing the two complex-term in the following formula in a different order:

$$(\exists e)(IS - A(e, Having) \wedge Haver(e, \langle IS - A(x, Restaurant)\rangle)$$

$$\wedge Had(e, \langle(\exists y)IS - A(y, Menu)\rangle)$$

If the first complex-term is processed first, then the obtained formula is:

$$(\exists e)(\forall x)(IS-A(e,Having) \wedge IS-A(x,Restaurant) \longrightarrow Haver(e,x)$$

$$\wedge(\exists y)(IS-A(y,Menu) \wedge Had(e,y)))$$

If the second complex-term is processed first, then the different formula is:

$$(\exists e)(\exists y)(IS-A(e,Having) \wedge IS-A(y,Menu)$$

$$\wedge Had(e,y) \wedge (\forall x)(IS-A(x,Restaurant) \longrightarrow Haver(e,x)).$$

The same results will be obtained for the example in the next section.

3. Context independent sentences mapping in logical form. The syntactic-semantic analyzer

Since the very beginning of computer science the natural language represented an important preoccupation for the specialists. The applications in this domain want to resolve two essential issues: the voice recognition (if the user speaks) and text processing (its meaning).

We provide in this paper an application which begins with the semantic representation idea of the context independent sentences in the natural language like expressions in extended first order predicate calculus. First of all we must specify what we mean by the extended first order predicate calculus. Starting with the FOPC we provide a new set of quantifiers, among the existential and universal ones, necessary for the representation of the quantitative sentences semantic. By using this quantifiers we will represent a quantitative sentence semantic like *Most people laugh* as

$$\exists_N X.(people(X) \wedge laugh(X) \wedge most(N)),$$

where $\exists N$ belongs to the new set of quantifiers.

This FOPC extension will be noted by FOPC/QS (first order predicate calculus for quantitative sentences). For further details see [9].

Back to our application, this will have as entry a natural language sentence introduced from the standard input from which it will result the FOPC/QS of this sentence and a graphical representation of its parse tree. It is very difficult to compare the natural language functionality and the computer systems operation. Problems appear when we deal with semantic ambiguities resolved by the human mind through context and convention. We have tried to eliminate part of these ambiguities introduced by the domain of quantifiers and of operators by the underspecified method. Thus for *Every boy loves a dog* the semantic representations will be like in figure 1:

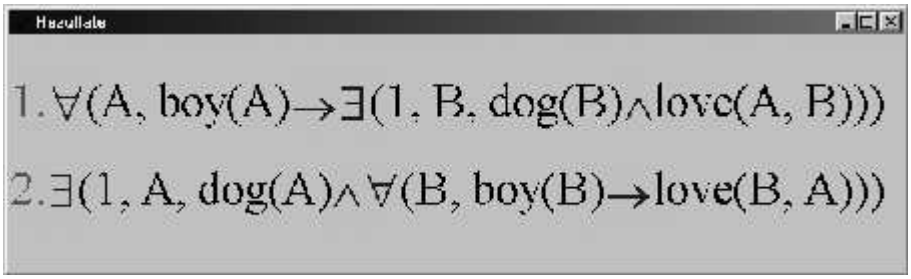


Figure 1

The ambiguities given by the multiple sense of the words will be considered in a future upgrade of the application, which could use the semantic network representation of the Lexis. We must specify that the sentences recognized by the application have to be introduced by an existent grammar. In other words, the user can not modify in any way the existent grammatical rules, but the application could be improved by allowing the user to construct the grammar he needs. This application allows the Lexis entries actualization by an interactive interface. The user could test, after resolving the problems which permit the grammar modifications too, the application in every natural language which describes that grammar. Thus, for every natural language will exists a file which contains its grammar, a file with its lexical entries and also a file which will contain the mapping of every atom structures of its sentences into the semantic representation. For every given sentence the application also presents the advantage of the parse tree graphical representation. Such an example is given as follows: *Every boy loves a dog.* (See figure 2)

We must also say that in the present the application doesn't resolve yet totally the parse of the sentence, more precisely, the gender, person and number agreement. This situation could be improved by modifying the grammatical rules by adding new arguments which represent these agreements. One advantage is that the application can help to design new applications, such as the natural language for querying knowledge bases, natural language conversation. For example, we can create an algorithm which will map every natural language sentence in an equivalent SQL statement in the first type applications. Concerning the structure of this application, it is built on two levels, plus an extra level for the parse tree representation. The first level contains the semantic engine (written in SWI-Prolog); the second one contains an interface with the user (written in Delphi); the extra level is for the graphical representation of the parse tree (written in Visual Prolog). The communication among these levels is done by the use of the Windows

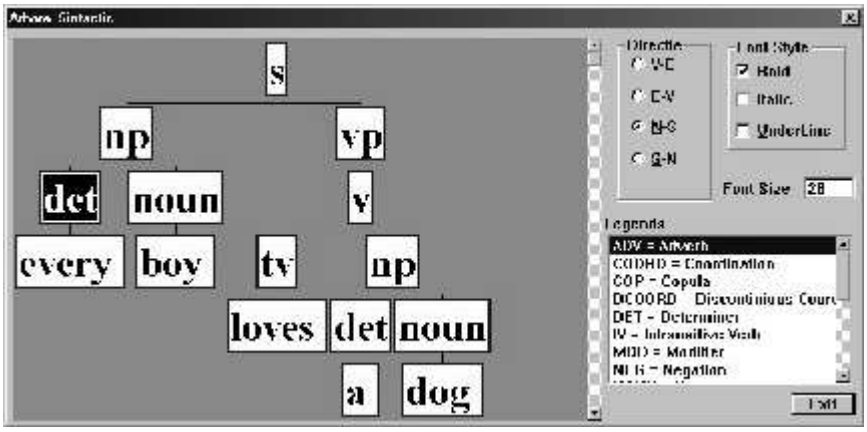


Figure 2

Operating systems specific DDE (dynamic data exchange), we can also use for these communication more evolved techniques such as COM/DCOM.

By its specific, our application construction is based on more programming languages mixture; it also succeeds in tacking advantages on these programming languages characteristics. We believe that this technique can be the starting point for resolving some natural language semantic problems.

References

[1] J.Allen : " Natural language understanding", Benjamin/Cummings Publ. , 2nd ed., 1995.

[2] C.Beardon, D.Lumsden, G.Holmes: " NL and computational linguistics", Ellis Howood Series, 1991.

[3] P.Flach: "Simply logical.Intelligent reasoning by example", John Wiley and Sons, 1994.

[4] A. Flycht-Eriksson: "A domain knowledge manager for dialogue systems", Proceedings of ECAI2000, pp 431-435.

[5] A.Gaal , G.Lapalme, P.Saint-Dizier: "Prolog for NLP", 1991.

[6] D.Jurafsky, J.H.Martin: "Speech and language processing", Prentice Hall, 2000.

[7] R.Kasper, P.Davis, C.Roberts: "An integrated approach to reference and pressuposition resolution", The 37th Annual Meeting of ACL, june 1999.

[8] I.A. Leţia and all : " Multi-agent systems", Casa cărţii de ştiinta, Cluj, 1999.

[9] A. Onet: " Semantic representation of the quantitative natural language sentences" to appear in Studia Universitatis "Babes-Bolyai", seria Informatica.

[10] S.J. Russell, P.Norvig: "Artificial intelligence.A modern approach",

[11] I. A. Sag, T. Wasow: "Syntactic Theory:A formal introduction " 1997, http://ling.ohio-state.edu/HPSG

[12] D. Tătar: "Unification Grammars in Natural Language Processing" , in "Recent topics in mathematical and computational linguistic", ed. C. Martin-Vide, G. Paun, Editura Academiei, 2000, pg 289-300.

[13] A. Thayse (editor): "From standard logic to logic programming", John Wiley and Sons, 1988.

[14] A. Thayse (editor): "From modal logic to deductive databases", John Wiley and Sons, 1989.

[15] A. Thayse (editor): "From natural language processing to logic for expert systems", John Wiley and Sons, 1990.

Department of Computer Science, Faculty of Mathematics and Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania

E-mail address: adrian|dtatar@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

NEW INTERACTION MECHANISMS BETWEEN JAVA DISTRIBUTED OBJECTS

FLORIAN MIRCEA BOIAN AND CORINA FERDEAN

Abstract. This article proposes some solutions to very common problems and requirements concerning the interaction between Java objects spread across several machines. Thus, an object should be able to access another remote object without knowing where that object resides. This location transparency induces also migration transparency, allowing objects to be found and accessed by their clients, even if they are changing their location.

Another extension, of the standard interacting protocols for the collaboration between distributed objects, could be the definition of patterns used to match the remote objects, which have certain attributes or which implement the services specified in the pattern.

1. Introduction

Basically, every distributed system implies two or more active entities (processes, threads, running objects) performing computations, in different address spaces, potentially on different hosts. Also, these active execution entities should be able to communicate.

For a basic communication mechanism, the Java programming language supports the *sockets*, which are flexible and sufficient for general communication. However, sockets require the client and server to define applications-level protocols to encode and decode messages for exchange, and the design of such protocols is cumbersome and sometimes error-prone. Besides, even if these protocols allows the communication between programs written in different languages and on heterogeneous platforms, they are not flexible and neither extensible.

Another distance communication mechanism, as an alternative to sockets, is *Remote Procedure Call* (RPC), which abstracts the communication interface to the level of a procedure call. Instead of working directly with sockets, the programmer has the illusion of calling a local procedure, when in fact the arguments of the call are packaged up and send to the remote target of the call. RPC systems encode

2000 *Mathematics Subject Classification.* 68M14.

1998 *CR Categories and Descriptors.* C.2.4 [**Computer Systems Organizations**]: Computer-Communication Networks – *Distributed Systems.*

arguments and return values using an external standard data representation, such as XDR.

However, the RPC mechanism is not suitable for the distributed object systems, where communication between program-level objects residing in different address spaces is needed. In order to match the semantics of object invocation, distributed object systems require *remote method invocation* or RMI. In such systems, a local surrogate (stub) object manages the invocation on a remote object.

2. Java RMI mechanism and JRMP protocol

Java RMI (Remote Method Invocation) offers a distributed object model for the Java Platform. Thus, the Java RMI system assumes the homogeneous environment of the Java virtual machine (JVM), and it uses the standard Java object model, extending it into a distributed context.

RMI is unique in that it is a language-centric model that takes advantage of a common network type system. In other words, RMI extends the Java object model beyond a single virtual machine address space.

The underlying communication protocol used in Java RMI mechanism is JRMP. This protocol allows the object methods to be invoked between different Virtual Machines across a network, and actual objects can be passed as arguments and return values during method invocation. The JRMP protocol uses object serialization to convert object graphs to byte-streams for transport. Any Java object type can be passed during invocation, including primitive types, core classes, user-defined classes, and JavaBeans. Java RMI could be described as a natural progression of procedural RPC (Remote Procedure Call), adapted to an object-oriented paradigm for the Java platform environment.

In the following we'll describe shortly how a typical object interaction works in Java RMI.

Any object whose methods are available to be invoked by another Java object must publish these methods by implementing an interface, which extends the `java.rmi.Remote` interface.

To make a remote object accessible to other virtual machines, a program typically registers it with the RMI registry. The program supplies to the registry the string name of the remote object as well as the remote object itself.

A client program, in fact a Java object, which wants to access a remote object, must supply the remote object's string name to the registry that is on the same machine as the remote object.

The string name accepted by the RMI registry has the syntax "`rmi://hostname:port/remoteObjectName`", where `hostname` and `port` identify the machine and port, respectively, on which the RMI registry is running and `remoteObjectName` is the string name of the remote object. `hostname`, `port`, and the prefix, "`rmi:`"

are optional. If `hostname` is not specified, then it defaults to the local host. If `port` is not specified, then it defaults to 1099. If `remoteObjectName` is not specified, then the object being named is the RMI registry itself.

The registry returns to the caller a reference, called *stub*, to the remote object.

As it turns out, the communication between Java objects is structured in a layer hierarchy, depicted in Figure 1.

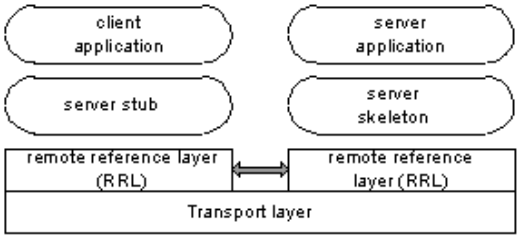


FIGURE 1. RMI Architecture

When the object's methods are invoked remotely, its arguments are *marshalled* and sent from the local virtual machine to the remote one, where the arguments are *unmarshalled* and used. When the method terminates, the results are marshalled from the remote machine and sent to the caller's virtual machine. An important observation which worths mentioning here is that the remote objects (that implement the Remote interface) are passed by reference, and the others objects by value (also, they must implement the `Serializable` interface). Another observation is that passing by value a Java object in a different Java environment is equivalent with a primitive form of object migration, where the "mobile agent" (the object passed by value) is static, and can be called when and if its destination environment decides).

3. PROVIDING JAVA RMI WITH SUPPORT FOR LOCATION TRANSPARENCY AND FAULT-TOLERANCE

A natural question, which arises in a Java RMI communication context, is how it would be possible for a client object to access a remote object, without having to know a priori the server object location. This feature of location independency becomes a fundamental requirement if it is assumed that Java server objects could change their location, migrating between different hosts.

3.1. **Using JNDI and LDAP.** The first solution proposed for providing location transparency is based on complementary technologies like JNDI (Java Naming Directory Interface) and LDAP (Light Directory Access Protocol).

Our discussion begins with a brief description of these technologies, followed by the presentation of the support they provide for accessing Java objects transparently.

JNDI (Java Naming Directory Interface). The Java Naming and Directory Interface (JNDI) is an application programming interface (API) that provides *naming* and *directory* functionality to applications written using the Java programming language [4, 5]. This API is defined to be independent of any specific directory service implementation, allowing a variety of directories to be accessed in a common way.

The JNDI architecture consists of an API and a service provider interface (SPI).

The primary goal for Java applications, that use the JNDI API, is to access a variety of naming and directory services. The services can be plugged in transparently, by using SPI [11]. This interface allows the developers of different naming/directory service providers to hook up their implementations so that the corresponding services are accessible from applications that use JNDI [4, 5].

These implementations include those for the *Initial Context* and for its descendent contexts that can be plugged in dynamically to the JNDI architecture to be used by the JNDI application clients.

JNDI is included in the Java 2 SDK, v1.3 and later releases. It is also available as a Java Standard Extension for use with the JDK1.1 and the Java 2 SDK, v1.2.

As it turns out, in order to use the JNDI, besides the JNDI classes, also, one or more service providers should be available. The Java 2 SDK, v1.3 includes three service providers for the following naming/directory services:

- Lightweight Directory Access Protocol (LDAP);
- Common Object Request Broker Architecture (CORBA) Common Object Services (COS) name service;
- Java Remote Method Invocation (RMI) Registry.

In this survey, we use LDAP as a directory service that provides a repository for the Java distributed shared objects.

LDAP. LDAP was originally developed as a front end to X.500, the OSI directory service. X.500 defines the Directory Access Protocol (DAP) for clients to use when contacting directory servers. DAP is a heavyweight protocol that runs over a full OSI stack and requires a significant amount of computing resources to run. LDAP runs directly over TCP and provides most of the functionality of DAP at a much lower cost [6].

LDAP directory service is based on a client-server model. One or more LDAP servers contain the data making up the LDAP directory tree. An LDAP client connects to an LDAP server and asks it a question. The server responds with the answer, or with a pointer to where the client can get more information (typically, another LDAP server). No matter which LDAP server a client connects to, it sees the same view of the directory; a name, presented to one LDAP server, references the same entry it would at another LDAP server. This is an important feature of a global directory service, like LDAP [7, 8].

In LDAP, directory entries are arranged in a hierarchical tree-like structure that reflects political, geographic and/or organizational boundaries. Entries representing countries appear at the top of the tree. Below them, there are entries representing states or national organizations. Below them, might be entries representing people, organizational units, printers, documents, or any other entities someone needs to define.

In addition, LDAP allows the control and the configuration of which attributes are required and allowed in an entry, through the use of a special attribute called `objectclass`. The values of the `objectclass` attribute determine the schema rules the entry must obey.

Using LDAP and JNDI to extend Java distributed computing. In the Java distributed computing context, LDAP provides a centrally administered and possibly replicated service for use by Java applications spread across the network. For example, an application server might use the directory for registering objects that represent the services that it manages so that a client can later search the directory to locate those services as needed.

The JNDI provides an object-oriented view of the directory, thereby allowing Java objects to be added to and retrieved from the directory without requiring the client to manage data representation or location execution issues.

There are different ways in which Java applications can use the directory to store and locate objects. Thus, an application might store (a copy of) the object itself, a reference to an object, or attributes that describe the object.

In general terms, a Java object's serialized form contains the object's state and an object's reference is a compact representation of addressing information that can be used to contact the object. An object's attributes are properties that are used to describe the object; attributes might include addressing and/or state information.

Which of these three ways to use depends on the application/system that is being built and how it needs to interoperate with other applications and systems that will share the objects stored in the directory. Another factor is the support provided by the service provider and the underlying directory service.

Transparent Java remote objects communication. In this survey, we will show how Sun's LDAP service provider supports the binding of `java.rmi.Remote` objects into directories. When `java.rmi.Remote` objects and/or RMI registries are bound into an LDAP enterprise-wide shared namespace, RMI clients can look up `java.rmi.Remote` objects without knowing on which machine the objects are running [1, 9, 10].

Instead of storing the entire serialized state of an object, which could be too large, it is preferable to store, into directories, a reference to that object. For that purpose, JNDI offers the `javax.naming.Reference` class. This class makes it possible to record address information about objects not directly bound to the directory service. The reference to an object contains the following information [7]:

- The class name of the referenced object;
- A vector of `javax.naming.RefAddr` objects that represents the addresses, identifying the connections to the object;
- The name and location of the object factory to use during object reconstruction.

`javax.naming.RefAddr` is an abstract class containing information needed to contact the object (e.g., via a location in memory, a lookup on another machine, etc.) or to recreate it with the same state. This class defines an association between content and type. The content (an object) stores information required to rebuild the object and the type (a string) identifies the purpose of the content.

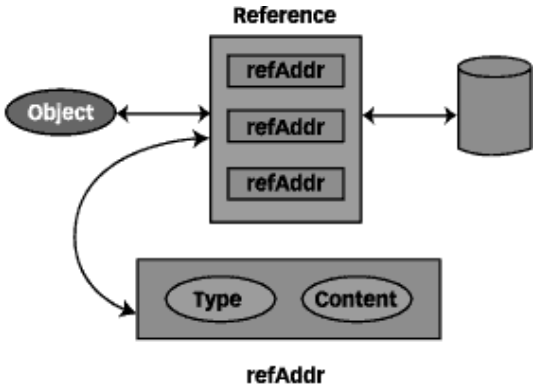


FIGURE 2. The relation between a Reference, RefAddr, Type, and Content

RefAddr also overrides the methods `java.lang.Object.equals(Object obj)` and `java.lang.Object.hashcode()` to ensure that two references are equal if the content and type are equal. `RefAddr` has two specific subclasses, namely, `javax.naming.StringRefAddr` and `javax.naming.BinaryRefAddr`, which store strings, and respectively arrays of bytes. For example, a string reference address could be an IP, URL, hostname, etc.

4. Example

In the following, we'll give a simple example of storing references to Java remote objects in a LDAP directory. We mention that it is also possible to store copies of objects as streams of bytes, but this alternative requires much more space, and it isn't flexible, as it is not possible to change an object implementation once it was bound in the directory service. Using references to objects provide this flexibility, and besides it saves a lot of space in the directory tree.

Our example is constructed conforming to the following steps, performed on different machines:

(1) We define on the machine `cronos.cc.ubbcluj.ro` a Java shared object `HelloImpl`, which implements a `Remote` interface called `Hello`. We register this object with the `rmiregistry` name service, on the same machine.

```
import java.rmi.*;
public interface Hello extends Remote {
    public String sayHello() throws RemoteException;
    }
```

Program 1. Hello.java

```
import java.rmi.*;
import java.rmi.server.*;
public class HelloImpl extends UnicastRemoteObject
implements Hello {
    public HelloImpl() throws RemoteException {
    }

    public String sayHello() throws RemoteException {
        return ("Hello, the time is " + new java.util.Date());
    }
}
```

Program 2. HelloImpl.java

```
import java.rmi.*;
public class ServHello {
```

```
public static void main(String args[]) {
    try {
        System.setSecurityManager(new RMISecurityManager());

        // create a registry if one is not running already.
        try {
            java.rmi.registry.LocateRegistry.createRegistry(1099);
        } catch (java.rmi.server.ExportException ee) {
            // registry already exists, we'll just use it.
        } catch (RemoteException re) {
            System.err.println(re.getMessage());
            re.printStackTrace();
        }
        Naming.rebind("rmi://cronos.cc.ubbcluj.ro/hello",
                new. HelloImpl());
    } catch(Exception e) {
        System.out.println("Error: "+e.getMessage());
        e.printStackTrace();
    }
}
}
```

Program 3. ServHello.java

(2) On another machine, `hermes.cc.ubbcluj.ro`, we create a reference of type `StringRefAddr` to the `HelloImpl` object, which contains an RMI URL of the form `rmi://cronos.cc.ubbcluj.ro/RemoteObjectName` and it is bound to a name into a LDAP directory. We also define a value for the `javacodebase` attribute, which will be used by the service provider to find the stub class for the remote object.

```
import java.util.Hashtable;
import javax.naming.*;
import javax.naming.directory.*;
import java.rmi.*;
public class HelloServ {
public static void main(String argv[]) {
    String rmiurl = "rmi://cronos.cc.ubbcluj.ro/hello";

    // Set up environment for creating the initial context
    Hashtable env = new Hashtable();
    env.put(Context.INITIAL_CONTEXT_FACTORY,
            "com.sun.jndi.ldap.LdapCtxFactory");
```

```
env.put(Context.PROVIDER_URL,
        "ldap://rave.scs.ubbcluj.ro:389/cn=CORI,
        o=CCUBB,c=RO");

try {
    // Create the initial context
    DirContext ctx = new InitialDirContext(env);

// Create the reference containing the (future) location of the object
        Reference ref = new Reference("Hello",
                new StringRefAddr("URL", rmiurl));

        BasicAttributes battr = new BasicAttributes(
                "javaCodebase",
                "http://www.cs.ubbcluj.ro/~cori/t/");

        // Bind the object to the directory
        ctx.rebind("cn=RefHello", ref);

        ctx.close();
    } catch (NamingException e) {
        System.out.println("Operation failed: " + e);
    }
```

Program 4. HelloServ.java

(3) We finally invoke the remote object from a client resident on machine `nessie.cs.ubbcluj.ro`. As we proposed from the beginning, the client makes the remote invocation without knowing the server object address, which allows the latter to change its location, without affecting the potential clients.

```
import java.util.Hashtable;
import javax.naming.*;
import javax.naming.directory.*;
import java.rmi.*;
public class HelloCl {
public static void main(String argv[]) {
    String rmiurl = "rmi://cronos.cc.ubbcluj.ro/hello";

    // Set up environment for creating the initial context
    Hashtable env = new Hashtable();
    env.put(Context.INITIAL_CONTEXT_FACTORY,
```

```
                "com.sun.jndi.ldap.LdapCtxFactory");
        env.put(Context.PROVIDER_URL,
                "ldap://rave.scs.ubbcluj.ro:389/cn=CORI,
                o=CCUBB,c=RO");

        try {
            DirContext ctx = new InitialDirContext(env);

            // lookup the object
            Hello h = (Hello)ctx.lookup("cn=RemoteHello");
            System.out.println(h.sayHello());

            ctx.close();
        } catch (NamingException e) {
            System.out.println("Operation failed: " + e);
        } catch (RemoteException e1) {
            System.out.println("Operation failed: " + e1);
        }
    }
    }
```

Program 5. HelloCl.java

The method that we presented uses the information stored in the directory. This information, represented by the `Reference` object, is actually a pointer to the information stored in another naming service (the RMI registry), which in turn, contains the reference to the `java.rmi.Remote` object.

Even if, in the simple example presented above, this level of indirection seems to be overheading, besides location transparency, it has important applications like providing fault-tolerance to distributed Java objects.

We use fault-tolerance to refer to the situation when a server object isn't available anymore (it was stopped or its host crashed), its services are being provided by other identical backup server objects. This is the traditional method of providing fault-tolerance by replication of the services that require high availability. In our case, a fault-tolerant Java server object is registered with different `rmiregistries`, and the corresponding rmi object's identifying URLs are stored as a `Reference` in a LDAP directory.

A client invocation uses one of the available server objects (in fact, the first available server in the stored addresses references order), without being aware of the duplication. The management of the duplicated objects is done totally transparent for the potential clients, and is completed by replication service provided by LDAP (for example `slapd` – Stand-alone LDAP Daemon – can be configured to

provide replicated service for a database with the help of `slurpd`, the standalone LDAP update replication daemon) [12].

5. Conclusions

In the article we presented the standard RMI mechanism available on the Java platforms, and some possible extensions to its basic features.

Within the Java language domain, Java RMI offers powerful new features for remote object distribution. Besides the powerful objects interaction facilities this mechanism provides, it can be extended with features that respects new constraints and requirements like location and migration transparency of the server objects. Also, the basic distributed systems requirement of fault-tolerance can be successfully integrated into the Java RMI mechanism.

References

[1] **Directory Examples**
http://java.sun.com/products/jndi/tutorial/getStarted/examples/directory.html
http://java.sun.com/products/jndi/tutorial/objects/storing/src/RemoteObj.java

[2] **Filterfresh: Hot Replication of Java RMI Server Objects**
http://www.usenix.org/publications/library/proceedings/coots98/full_papers/baratloo/baratloo_html/baratloo.html

[3] **Java IDL**
http://sophia.dtp.fmph.uniba.sk/javastuff/tutorial/idl/summary/

[4] **JNDI API**
http://sunsite.ccu.edu.tw/java/jdk1.3/api/javax/naming/InitialContext.html# ENVIRONMENT
http://java.sun.com/products/jndi/

[5] **JNDI Tutorial**
http://java.sun.com/products/jndi/tutorial/

[6] **LDAP: A Next Generation Directory Protocol**
http://www.intranetjournal.com/foundation/ldap.shtml

[7] **LDAP and JNDI: Together forever**
http://www.javaworld.com/javaworld/jw-03-2000/jw-0324-ldap_p.html

[8] **RFC LDAP**
http://www.ietf.org/rfc/rfc2713.txt

[9] **RMI and Java Distributed Computing**
http://java.sun.com/features/1997/nov/rmi.html

[10] **RMI Registry Service Provider JNDI**
http://sunsite.ccu.edu.tw/java/jdk1.3/guide/jndi/jndi-rmi.html#USAGE

[11] **SLAPD Daemon**
http://www.umich.edu/ dirsvcs/ldap/doc/guides/slapd/1.html#RTFToC1

[12] **SPI**
http://java.sun.com/j2se/1.3/docs/guide/jndi/spec/spi/jndispi.fm.html#1005286
Service Provider Package
http://java.sun.com/products/jndi/tutorial/getStarted/overview/provider.html

Department of Computer Science, Faculty of Mathematics and Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania
E-mail address: florin|cori@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

ON THE USING OF ARTIFICIAL NEURAL NETWORKS IN AUTOMATIC METALOGRAPHIC ANALYSIS

IOAN ILEANĂ AND REMUS JOLDE

Abstract. This paper presents several considerations and preliminary results in implementing an automatic metallographic analysis system using artificial neural networks. The optical microscope images of special prepared samples of metals and alloys may be classified by a neural network trained with standards. We present some of the results and problems we encountered in our work. Our contribution mainly consist in analysis system design, images preprocessing and network training.

Keywords: *metallographic analysis, pattern recognition, artificial neural network, preprocessing.*

1. Introduction

One of the important investigation methods used by the physical metallurgy is optical metallography, which also concerns micrographic analysis using the optical microscope (magnifying rate up to 2000:1). The images obtained by microscope give direct indications on the chemical and structural composition, also indirectly informing on the physical and mechanical properties of the metallic alloys. One can as well get data on the structural changes occurred under the influence of various mechanical processing previously applied to the alloy.

When considering pure metals or monophasic alloys, micrographic analysis allows observing the size and the orientation of the crystalline grains, the particularities of the dendritic structure, even the repartition of the dislocations. As for polyphasic alloys, which present more complex structural aspects, one can determine the nature, quantity, shape, size and repartition of the various phases in the structure.

Microscopic analysis is an important information source. Its efficiency is partly influenced by the place where the samples are collected and the collecting manner, as well as the skills and experience of the specialist performing the analysis. Figure 1 presents images of samples taken from different materials.

It is to be noticed that the information is "coded" in graphical patterns-images (using gray tones or colors) that have to be interpreted by the person that does

2000 *Mathematics Subject Classification.* 68T10.

1998 *CR Categories and Descriptors.* I.5.1 [**Computing Methodologies**]: Pattern Recogniton – *Models.*

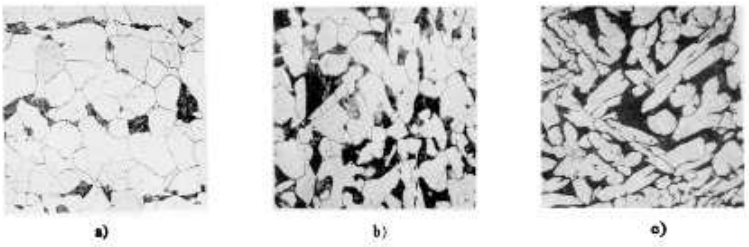


FIGURE 1. Metallic surfaces viewed through optical microscope: a) steel with 0.08–0.15% C, rolled at warm; b) steel with 0.16–0.25% C normalized at 880degC; c) bronze with biphasic cast aluminium. Source: [8].

the analysis. This operation is difficult, demanding a lot of time and experience. Therefore a very useful improvement would consist in the automation of these analysis by creating a system that is able to classify and recognize, possibly in real time, in the images obtained by microscope structures, flaws, previous processing.

2. Automatic Metallographic Analysis System

Our team, in collaboration with the industrial partner "SC SATURN SA"Alba Iulia, has started a project concerning the implementation of an automatic system for metallographic analysis (fig. 2), where the recognition and classification functions are performed by a neural network.

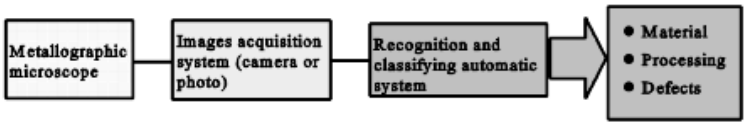


FIGURE 2. Automatic metallographic analysis system

During the current stage of the project, our attention has been focused on the interpretation and classification of the material samples images.

The interpretation of micrographic images is part of the larger area of pattern classifying and recognition. As it is shown by the example in figure 1, identifying rather simple patterns can require the interpretation of mega-dimensional databases, with complicated structure and unknown topological relations. In general there aren't known possible transformations that could simplify this structure and a multilevel hierarchy system of feature extraction becomes necessary.

Another general issue in model based pattern recognition consists in correct input image identifying, even when the image is a geometrically transformed version

of the model. The invariant recognition can be achieved using, instead of the initial pattern, the result of a mathematical transformation, that necessarily assures a certain invariance (Fourier transform, Mellin transform etc.). Unfortunately this mathematical pattern preprocessing implies a great computing effort in electronic (hardware and software) implementations. Optical and optoelectronic systems can bypass this drawback due to the parallel computing.

In our case, for metallographic optical analysis, we can assume that the prototype (standard) images and those to be recognized and interpreted, will have the same scale factor, so that the system must be only translation and rotation invariant.

We intend to use for image interpretation a software simulated artificial neural network (ANN), therefore we have evaluated several ANN categories and several preprocessing techniques, in order to find an acceptable solution. The following section present some preliminary results of our work.

3. Neural Network Model

In our work we used two kinds of artificial neural ntworks: a recurrent network and then a feed forward neural network, trained with backpropagation method. The processing unit (artificial neuron) used in the two cases is displayed in figure 3.

In this figure $x_1, x_2, \ldots, x_n$ are neuron inputs, $w_1, w_2, \ldots, w_n$ are the interconnection weights, θ is the neuron threshold, $f()$ is activation function and y is neuron output.

We note: $x = [x_1, x_2, \ldots, xn]^T$ the input vector, $w = [w_1, w_2, \ldots, w_n]^T$ synaptic weights vector,

$$net = \sum_i w_i x_i = w^T x \tag{1}$$

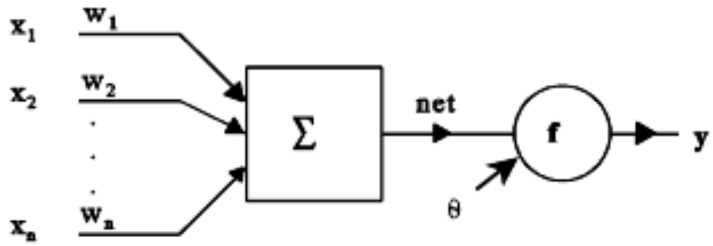


Figure 3. The processing unit used

Then the neuro output may be writen:

$$y = f(net - \theta) = f(w^T x - \theta) \tag{2}$$

A) For the recurrent neural network, the model is presented in figure 4. Let's consider the single-layer neural network built from totally connected neurons, whose states are given by $x_i \in -1, 1$, $i = 1, 2, \ldots n$, (fig.4).

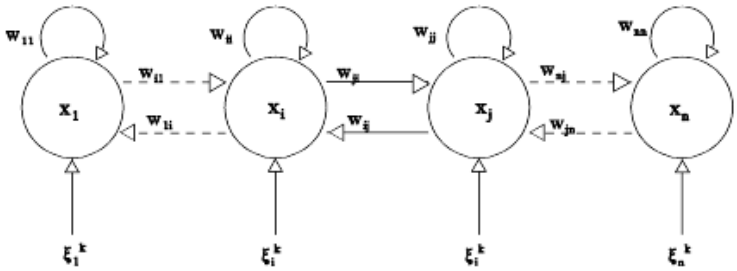


FIGURE 4. The recurrent network model

We denote: $W = [w_{ij} : 1 \leq i, j \leq n]$ the weights matrix, $\theta = [\theta_1, \ldots, \theta_n]^T \in R^n$ the thresholds vector, $x(t) = [x_1(t), \ldots, x_n(t)]^T \in -1, 1^n$ the network state vector.

The evolution in time of the network is described by the following dynamic equation:

$$x_I(t+1) = sgn\left[\sum_{j=1}^{n} w_{ij} x_j(t) - \theta_i\right], i = 1, 2, \ldots, n \tag{3}$$

with the convention:

$$\sum_{j=1}^{n} w_{ij} x_j(t) - \theta_i = 0, x_i(t+1) = x_i(t) \tag{4}$$

where:

$$sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{5}$$

Notes:

(1) We may consider networks where the neurons' state is not bipolar: -1,1, but binary: 0,1. A relation between the two representations can be easily found.

(2) In many situations we may give up the neural network threshold z_i and we'll do this whenever it doesn't affect the results.

For the autoassociative memory described in this paper, the weight matrix W will be built as follows: given a set of n-dimensional prototype vectors $X = [\xi^1, \xi^2, \ldots, \xi^p]$, we establish the synaptic matrix W and the threshold vector θ, so that the prototype vectors become stable points for the associative memory, i.e.:

$$\xi^i = sgn(W\xi^i - \theta), i = 1, 2, \ldots, p \tag{6}$$

where the sgn function is applied to each component of the argument.

Several classical rules for determining the weights matrix proved successful in time:

- the 'Hebb' rule
- the projection rule
- the delta projection rule (the gradient method)

B) In the second approach we used a feed forward network with three layers, trained with back propagation method. The number of neurons in the first layer is determined by the dimension of the input image. The number of neurons in the output layer depends on the number of classes in which the input images must be classified. In the hidden layer we tried several configuration and the final network used the best structure. For the neurons in hidden and output layer we used as activation function the sigmoid function.

4. Preliminary Results

Because of our industrial partner's interest in the metallographic analysis of cast iron (its field of production) samples, we've studied the synthesis of an ANN that could allow the recognition and classification of real samples reported to standards. Some standards used for these experiments are shown in figures 5 and 6.

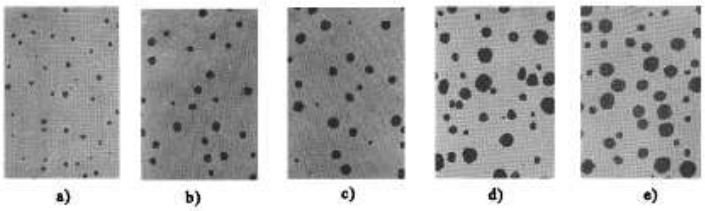


Figure 5. Standard structures of cast iro with nodule graphite: a) below 3%; b) 3–5%; c) 5–8%; d) 8–12%; e) over 12%. Source: [8]

Using samples taken from these standard images, we investigated the training methods for various types of ANN in order to perform micrographic images classification. The images used as prototypes have been preprocessed as to enhance their specific features (fig. 7).

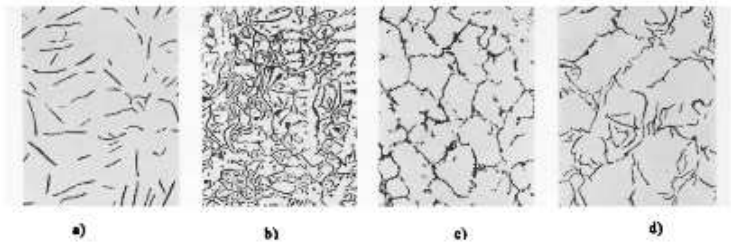


FIGURE 6. Stabdards for gray cast iron with lamellar graphite: a) isolated separations; b) agglomerations with low isolation degree; c) punctiform graphite net; d) lamellar graphite net. Source: [8]

FIGURE 7. Some preprocessed prototypes

A. One first tested ANN category was a recurrent network used to implement an associative memory. We used as prototypes 32×32 pixels images randomly selected from the standard images. Rotation and translation invariance has been obtained by storing several images of the same prototype, randomly transformed [6]. The associative memory thus built has been verified with a great number of test images. The statistical results were very good in what noise contamination is concerned (up to 50% noise contamination). As for geometrical transforms invariance, the results were rather unsatisfactory; the correct recognition rate would be from 40% up to no more than 80%, depending on the prototype image.

B. A second simulation category consisted in the setup of a feed-forward ANN, trained with the same input data used in the previous approach. We investigated several feed-forward topologies, with 2 and 3 layers. Within the limits of available input data, the 3 layers structure provided acceptable results. We faced some difficulties when using 32×32 pixels images, therefore we had to work with 16×16 pixels images.

C. In order to obtain rotation and translation invariance, we also tried to use invariant moments, as presented in [1]. The difficulties we encountered in this approach are connected to the large computation volume and to the necessity

for these descriptors to be different enough as to separate the different standard classes. For the 5 standards classes in fig. 5 and the 4 standard classes in fig. 6, the above mentioned descriptors are shown in fig. 8 and 9, respectively. One may notice a rather insignificant difference, which leads to difficulties and errors in data interpretation. We currently work on finding more efficient preprocessing, that could lead to stronger discrimination among invariant descriptors of different classes.

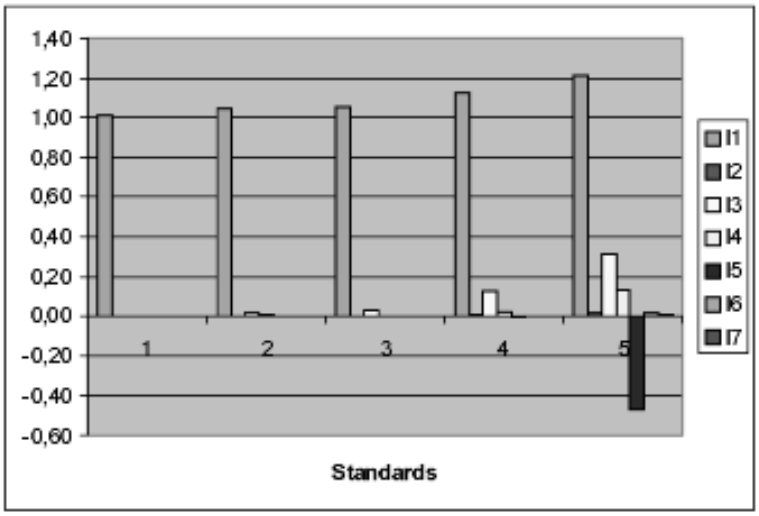


FIGURE 8. Moment invariants for images in fig. 3

5. CONCLUSIONS

The implementation of an automatic system for optical metallographic images analysis is an important objective for the laboratories where such tasks are performed. Moreover, such a system once implemented, it could be used in flaw analysis and even in biological tissue analysis.

This paper has presented some preliminary results obtained by our team in using ANN to perform the recognition and classification of optical micrographic images of material samples, as reported to standards.

The main difficulties we had to overcome were the following:

- The necessity of using relatively large images (over 32×32 pixels) in order to extract significant features out of the sample structure; consequently troubles in training and simulating the ANN were connected to the required memory space, as well as to the computation speed.

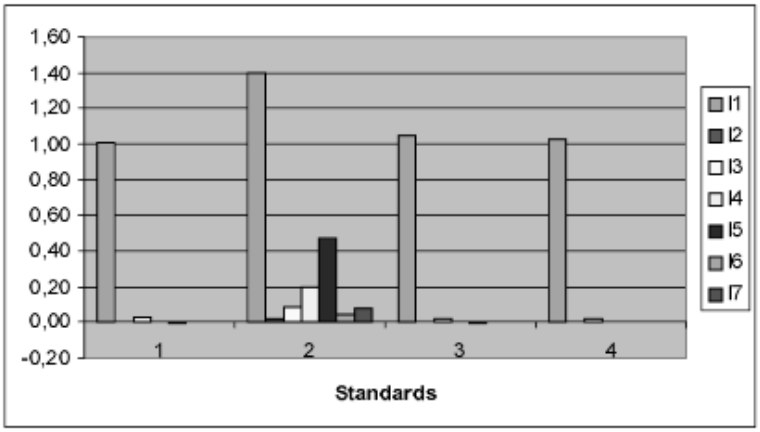


FIGURE 9. Moment invariants for images in fig. 4

- The necessity of recognition immunity, regarding the noise contamination of the images, and also their various geometrical transforms.

We investigated several methods to build a system that would accomplish these requirements and we may conclude that ANN do offer a realist perspective, if solving the above mentioned difficulties. The solutions we currently have in view partly refer to using a faster and more powerful computer for network training and simulation, and partly consist in using more efficient preprocessing methods for the input images.

REFERENCES

[1] Alastair, Mc. Aulay: *Optical computer architectures: the application of optical concepts to next generation computers*, John Wiley & Sons, Inc, 1991.
[2] Cojoc Dănuţ-Adrian: *Aplicaţii ale corelaţiei optice în recunoaşterea formelor*, Teză de doctorat, Universitatea "Politehnica", Bucureşti.
[3] Dumitrescu D., Hariton Costin: *Reţele neuronale. Teorie şi aplicaţii*, Ed. Teora, 1996.
[4] Jianchang Mao, Anil K. Jain: "Artificial neural Network for Features Extraction and Multivariant Data Projection", *IEEE Transactions on Neural Network*, vol. 6, Nr. 2, march 1995.
[5] Ileană Ioan, Iancu Ovidiu Corneliu: "Optoelectronic associative neural network for some graphical patterns recognition", *Proceedings of SPIE*, SIOEL '99, Volume 4068, p.733–739.
[6] Ileană Ioan, Iancu Ovidiu Corneliu, Joldeş Remus: "Recunoaşterea invariantă la translaţia, rotaţia sau scalarea formelor", *Annales Universitatis Apulensis, Series Economica*, Tom 1, 2000, p. 175–185.
[7] Nedevschi Sergiu: *Prelucrarea imaginii şi recunoaşterea formelor*, Editura Albastră, Cluj-Napoca, 1998.
[8] Rădulescu M., Drăgan N., Hubert H., Opriş C.: *Atlas Metalografic*, Ed. Tehnică, 1971.

"1 DECEMBRIE 1918" UNIVERSITY OF ALBA IULIA

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

ALAIN DARTE, YVES ROBERT AND FREDERIC VIVIEN, "SCHEDULING AND AUTOMATIC PARALLELIZATION", BIRKHAUSER BOSTON, 2000, ISBN 0-8176-4149-1

ALEXANDRU VANCEA

This book offers a detailed and self-contained presentation for studying loop transformations, the detection of parallel loops, and how to use them to detect parallelism in a specific program. It provides careful explanation and exposition for all parallel-loop algorithms that have been designed recently in a framework of scheduling algorithms on cyclic graphs, primarily task graph scheduling and loop nest scheduling perspectives.

Program restructuring techniques are important optimization methods used in parallelizing compilers. The focus is on loop transformations, because there is where a program spends most of its execution time. The authors, well known in the parallelizing compilers community, have original contributions regarding loop restructuring based on unimodular transformations and general affine transformations.

Scheduling and Automatic Parallelization offers an explanation of how to incorporate these transformations in algorithms, which transformations to apply, and how to optimize them. It provides a full study of loop transformations, and algorithms for parallel loop detection in a scheduling perspective, making the link with cyclic scheduling and systems of uniform recurrence equations.

One of the main contributions of the book is building a unifying theory of loop nest scheduling. This theory is developed based upon the previous work of Karp, Miller, Winograd and Lamport and it relies on sophisticated mathematical tools : unimodular transformations, parametric integer linear programming, Hermite decomposition, Smith decomposition etc.

The book is an essential reference for the latest developments in automatic parallelization methods used for scheduling, compilers, and program transformations. It is intended for graduate and postgraduate students interested in automatic parallelization techniques, researchers interested in scheduling, compilers and program transformations. Software engineering and computer engineering professionals will

find it a very good resource and reference. It is also suitable for self-study purposes by practitioners.

Department of Computer Science, Faculty of Mathematics and Computer Science, "Babe-ş-Bolyai" University, Cluj-Napoca, Romania
E-mail address: vancea@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 1, 2000

WALLIS, W.D., "A BEGINNER'S GUIDE TO GRAPH THEORY", BIRKHUSER, BOSTON-BASEL-BERLIN, 2000, ISBN 0-8176-4176-9, 230PP.

TEODOR TOADERE

This is a very good course on graph Theory for students in mathematics, computer science, engineers and psychologists. The author has taught "graph theory courses at the University of Newcastle and Southern Illinois University over the past 30 years".

It is a lucid book and has attained a balance between the theoretical and practical approaches.

The book is divided into 13 chapters and two appendices: Hint & Answers and Solutions.

Each chapter presents theoretical notions, examples and exercises.

The first four chapters introduce the reader in graph theory (it is presented the concepts of graphs, walks, paths, cycles, cuts, connectivity and tree).

The fifth chapter deals with the application of several vector spaces concepts graphs theory.

The next four chapters presents: factorizations, graph coloring, planarity and Ramsey theory.

Chapter 10 introduces directed graphs. The two following chapters are devoted to two important application areas: critical paths and network flows. The last chapter is dedicated for computational considerations.

The book has 109 references cited in text.

We think that this is a very good book, which can be useful to any person who wants to introduce himself in the graph theory or to deepen its study.

Department of Computer Science, Faculty of Mathematics and Computer Science, "Babe-ş-Bolyai" University, Cluj-Napoca, Romania

E-mail address: toadere@cs.ubbcluj.ro