Anul XLIX 2004

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

2

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

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

MAXIMAL PROCESSOR UTILIZATION IN PARALLEL QUADTREE-BASED FRACTAL IMAGE COMPRESSION ON MIMD ARCHITECTURES

BODÓ ZALÁN-PÉTER

Abstract. Since fractal image compression is computationally very expensive, some researchers tried to parallelize the encoding algorithm. Because this algorithm is applied independently for some image blocks, fractal image compression implicitly encompasses parallelism. The quadtree-based compression proceeds recursively and terminates when the previously fixed threshold remains *unexceeded*, therefore one cannot be able to calculate and store the whole domain pool for classification. This, however, can result some idle processors during the encoding, which is undesirable. In this paper a parallel implementation devoid of classification will be presented, keeping the number of idle processors at minimal.

1. Introduction

Belonging to the class of lossy data compressors, fractal image compression is based on self-similarity in real-world images, where "an image is modeled as the unique fixed point of a contractive operator on the space of images" [6]. Barnsley was who discovered and proved the most important theorem in 1985, namely the *Collage Theorem* [1, pp. 94–95], which serves as a basis for the fractal coding of images. Jacquin wrote down first the famous algorithm of fractal image compression, based on *partitioned iterated function systems* (PIFS). The algorithm partitions the image into smaller independent (range and domain) blocks, where for every range block the best matching domain block is needed to be found. The long search time makes the encoding problematic. Besides high compression ratios fractal image compression provides resolution independent image description, that is the reconstructed image can be zoomed-in without pixelisation.

Received by the editors: August 24, 2004.

2000 *Mathematics Subject Classification.* 65Y05, 94A08.

1998 *CR Categories and Descriptors.* D.1.3. [**Programming Techniques**]: Concurrent Programming – *Parallel programming*; I.4.2. [**Image Processing and Computer Vision**]: Compression – *Approximate methods*.

The main drawback of the proposed quadtree-based fractal image compressions on MIMD (Multiple Instruction stream and Single Data stream) architectures is that increasing the number of processors, after a while the gain from adding a new processor is almost zero. In this paper we present a simple algorithm for MIMD architectures trying to maximize processor utilization during the encoding phase.

The paper is organized in the following way: in Section 2 we present the procedure of fractal image compression with quadtrees. Details about the complexity of fractal encoding of images can be found in Section 3. In Section 4 we present some parallel implementations of fractal coding systems on MIMD architectures. In Section 5 the proposed parallel quadtree-based fractal image compression algorithm is presented in details. The test results and the conclusions can be found in Section 6.

2. Fractal Image Compression with Quadtrees

In fractal image compression the image is modeled as the unique fixed point of the contractive operator

$$W(\cdot) = \bigcup_{n=1}^{N} w_n(\cdot),$$

where w_n, $n = 1, 2, \ldots, N$ are contraction mappings, whose set is called a partitioned iterated function system (PIFS). The well known *copying machine* is an informal denomination of the mathematical structure called iterated function system (IFS). The single difference between IFS and PIFS is that the domains of the member functions of a PIFS are subsets of the plane on which the (P)IFS is defined. These structures simplify the fractal coding of *not really* self-similar sets. The domains of the transformations are called domain blocks (D_i), while the ranges are called range blocks (R_i), and we can write it in the following way:

$$w_i : D_i \to R_i, \quad i = 1, 2, \ldots, N.$$

Let T be an arbitrary grayscale image, that is $T : I^2 \to I$, where $I^2 = \{(x, y) \mid x, y \in [0, 1]\}$ and $I = \{x \mid x \in [0, 1]\}$. However, this space can be extended to arbitrary size, only in theory we work with these domains and ranges for the sake of simplicity. Then we seek such a contractive operator (a set of affine transformations) that

$$T = W(T) = \bigcup_{n=1}^{N} w_i(D_i).$$

To measure the distance between two blocks we need to find a feasible metric. In theory the supremum metric is used, which is easy to work with, but not so advantageous in practice, because it takes only one point from the image, consequently is not relevant for the whole image or image block. In practice the RMS (Root-Mean-Square) metric is used,

$$d_{RMS}(T,T') = \sqrt{\frac{1}{n} \sum_{(x,y)\in I^2} (T(x,y) - T'(x,y))^2}, \quad \forall T, T' \in \tau,$$

where $\tau = \{T : I^2 \to I\}$ denotes the space of digital images. The metric d_{sup} is equivalent with metric d_{RMS}.

To guarantee the z-contractivity of the $w_1, \dots, w_n$ three-dimensional transformations [2, pp. 12–13], in case of grayscale images we can use transformations of the form

$$w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix},$$

where s_i controls the contrast and o_i controls the brightness of the transformation [4, pp. 11, 51]. The encoding algorithm consists of finding for all the range blocks (resulting from the used partition scheme – rectangular, quadtree, horizontal-vertical, Delaunay-triangular, etc.) such a domain block that the distance between them be minimal or at least smaller than a predetermined threshold. Usually the size of the domain blocks is chosen to be greater than that of the range blocks; comparison is realized by subsampling.

If $a_1, a_2, \dots, a_n$ denote the pixel intensities from the set D_i and $b_1, b_2, \dots, b_n$ from the set R_i ($|D_i| = |R_i| = n$, i.e. their cardinal numbers are equal), then we search for s, o so that the following expression be minimal:

$$R = \sum_{i=1}^{n} \left[(s \cdot a_i + o) - b_i\right]^2 .$$

The optimal values of the contrast scaling (s) and brightness (luminance) shift (o) are calculated from the partial derivatives of the above expression. That is,

$$o = \frac{1}{n}\left(\sum_{i=1}^{n} b_i - s\sum_{i=1}^{n} a_i\right), \quad \text{and} \quad s = \frac{n\sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} a_i \sum_{i=1}^{n} b_i}{n\sum_{i=1}^{n} a_i^2 - \left(\sum_{i=1}^{n} a_i\right)^2}.$$

Substituting these expressions into the initial formula we get

$$R = \frac{1}{n}\left[\sum_{i=1}^{n} b_i^2 + s\left(s\sum_{i=1}^{n} a_i^2 - 2\sum_{i=1}^{n} a_i b_i + 2o\sum_{i=1}^{n} a_i\right) + o\left(no - 2\sum_{i=1}^{n} b_i\right)\right].$$

The distance between two blocks is given by $\sqrt{R}$.

The fractal image encoding algorithm can be summarized as follows:

```
Let us determine a partitition for image $T$, made up of
range blocks $R_i$, such that $T = \bigcup R_i$
Fix a $t$ tolerance level
For all $R_i$ do:
    Let us find the domain block $D_i$ for which $d(R_i, D_i)$ is
    minimal or $d(R_i, D_i) < t$
    Store the transformation $w_i$ and the coordinates of $D_i$
End For
```

The quadtree partition is the most popular scheme in the fractal coding literature. It offers an adaptive partition, which gives better approximation than the fixed size. The partition process is not separated from the encoding step. Its name comes from the modeling (Fig. 1). The root of the tree is the whole, *unpartitioned* image. At the first level the image is divided into four equal parts. For all the ranges we scan the image (domain pool) for a domain block (usually twice the range size), which is very close, similar to the current range block. If the distance between the range and the transformed domain block is below a preselected treshold, than we store the domain coordinates and the transformation on the pixel values. If not, we divide the current range block into four equal quadrants, which means adding four childnodes to this range block in the tree representation. Then for each of the ranges the previous process is repeated.

The tree we obtain is called a quadtree, because each node of the tree can have

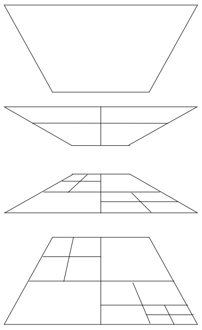


FIGURE 1. Quadtree decomposition.

either four subnodes or not any.

3. About the Complexity of Fractal Coding

Consider an $n \times n$ pixel image being encoded with fixed size partition and full-search (full-search means that no tolerance level is used). Let the size of a range block be $n_r \times n_r$ and the size of a domain block $n_d \times n_d$ (the size of the domain block is often chosen to be twice the range size, that is $n_d = 2 \cdot n_r$). It is easy to calculate that the number of ranges is $\lfloor n/n_r \rfloor^2$ and the number of domains $(n - n_d + 1)^2$. Then the complexity of the encoding procedure will be

$$\begin{gathered} nr\_of\_isometries \cdot \lfloor n/n_r \rfloor^2 \cdot (n - n_d + 1)^2 = \\ nr\_of\_isometries \cdot \mathcal{O}(n^2) \cdot \mathcal{O}(n^2) = \mathcal{O}(n^4) \end{gathered}$$

To illustrate how computationally expensive the encoding step is, let $n = 256$, $n_r = 8$, $n_d = 16$. Then, to encode this image $8 \cdot 1024 \cdot 58\,081 = 475\,799\,552$ comparison steps are needed and, as we have seen, one domain-range comparison is quite computationally expensive.

Ruhl and Hartenstein proved that finding an optimal fractal code is an NP-hard problem. They proved this by reducing the MAXCUT problem – which is NP-hard – to FRACCODE. The proof and the accurate definition of these decision problems can be found in [9]. In this paper they also proved that collage coding is not a ρ-approximating algorithm. That is, for every $\rho \in \mathbf{R}^+$ exists a signal (i.e. image) Γ and a transformation $g \in \pi$ (π is the set of possible fractal codes for the signal Γ) such that

$$\|\Omega_f - \Gamma\|^2 > \rho \cdot \|\Omega_g - \Gamma\|^2,$$

where f is the transformation that gives the best (minimal) collage error, $\|\Gamma - f(\Gamma)\|$; Ω_x is the attractor of the fractal code x.

4. Parallel MIMD Fractal Image Compression

There are two main algorithm classes for fractal image compression on MIMD architectures. The first class includes those algorithms which stores the whole image on each PE (Processing Element), namely when all the PEs have enough memory to have a local copy of the image. Thus each PE has the whole domain pool at its disposal. To each PE a subset of the range blocks is assigned, which can be made statically or dynamically. The second class includes those algorithms which distributes the domain pool among PEs, due either to memory lack or to other reasons. In [5] a more detailed class decomposition can be found.

Jackson and Tinney [7, 8] reported three generic schemes for parallel fractal image coding. In the following subsections these schemes will be presented.

4.1. **Static Load Allocation.**

Static load allocation means that the assignment of the tasks, jobs is made at the beginning and no other modifications can be made afterwards. Due to the static property of this model, only fixed size partition schemes can be used, or those adaptive techniques, where the partition can be realized before, independently of the encoding process.

The most simpler solution is to distribute the range blocks evenly across the PEs. If we have n_p processors, then the work needed to be done by a PE is the $\frac{1}{n_p}$th of the work done in the sequential case. The speedup can be calculated easily here, that is the speedup will be n_p. But the experimental results show that rarely will the algorithm achieve this speedup, because for a range block a matching can be found quickly, while for another range block to find a good matching requires to scan almost the whole domain pool. If we had known something about the *complexity* of the range blocks (a range block is said to be complex if it is difficult to find a match for), then the ranges could have been distributed evenly across the PEs, according to their complexity. That is, a slave processor gets a small number of range blocks to be processed if they are complex, while another PE receives a larger number of simpler range blocks.

M. Chady [3] mentions other two factors which can cause the speedup to fall below the expected value. If we use a classification scheme, then the order of the classes is very important; the common classes should be placed before the uncommon ones, if not, the comparison process to classify a range block will be slowed down. Although if we talk about classification schemes, one cannot realize such a classification which provides equal comparison steps for each range block to classify, except if classification is done in parallel, but this requires as many idle (free) processors as many classes we have.
Chady mentions another problem which could slow down the encoding in case of quadtree partition scheme, but of course quadtree partition cannot be applied within static load allocation, because one can never predict how many range blocks will be finally and where they will be.

For the reasons mentioned above, this solution cannot guarantee uniform workload distribution.

4.2. **Dynamic Load Allocation.**

Dynamic load allocation is often called *load balancing*. Load balancing means distributing the tasks evenly through the processors so that no processing element is overloaded. Load balancing technique is used especially when it is difficult to predict the number of tasks or the complexity of a task (time needed to perform the task).

The following scheme can be used as for fixed size partitions as for adaptive partition schemes like using quadtrees. Jackson defines this method for fixed size partitions in [8]. In dynamic load allocation there is a master or host processor which distributes the tasks among the other (slave) processors (Fig. 2). The master has two queues:

- queue of tasks (range blocks, waiting to be processed)
- queue of slaves (idle processors).

Using fixed size partitions the user can determine the so called package size, i.e. the size of individual assignments allocated by the master by specifying the number of the range blocks per allocation. The master makes the assignments, assigns a task to a slave until there are no more idle slave processors or range blocks to be processed. As a task is assigned to a slave, both the task and the slave is removed from their queues. On each return the master assigns a new task to the PE which returned the result.

Using quadtree partition scheme the differences are very small. The slave needs also to return a value besides the other parameters, which tells the master if a good matching was found or not. If there was a matching found and if the queue of tasks is empty, then the slave is placed back into the queue and the result is stored. If a matching was found and the queue of tasks is not empty, then a new task is assigned to the slave. If there was not found any suitable match, then the master divides the returned range block into four equal subsquares and places back both the range blocks and the slave into the corresponding queues and assigns a task to a slave until one of the queues becomes empty.

4.3. **Dynamic Allocation with Circulating Pipeline Processing.** In this configuration the slave processors communicate in a circulating pipeline fashion (Fig. 2). The domain pool is distributed among the slave processors. Like in the previous scheme the master maintains two queues. The assignments are transmitted from the master PE to idle slave PEs. When a task enters the pipeline enters

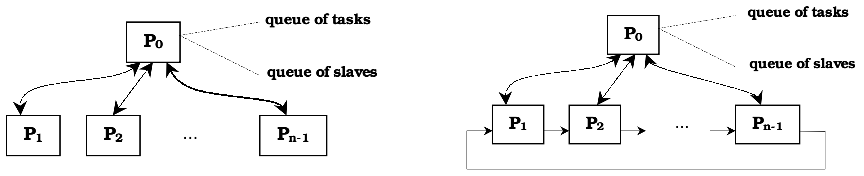


FIGURE 2. The structure of the PEs in dynamic load allocation and circulating pipeline schemes.

with a tag in which the number of the entering PE is stored. A range block will circulate in the pipeline until either a good match is found or all the pipeline nodes have been visited. The tag carrying the number of the entering node is used to check whether the task have visited all the PEs in the pipeline.

This configuration can be used both for fixed size and adaptive partition schemes.

5. Parallel Fractal Image Compression with Maximal Processor Utilization

Using classification schemes in quadtree-based fractal image compression algorithms on MIMD computers may degrade the performance of the encoding. According to Chady [3] the encoding proceeds in phases because the domain pool needs to be calculated for classification, the size of which grows exponentially as we proceed down in the quadtree. Therefore one cannot store the whole domain pool for a quadtree-partitioned image, but for example for one level only, which incidentally gives a good order for storing the parameters. However, there will be a certain period of time when many processors will be idle, since finding a good matching for a more complex range block may require much more computation, while the other processors have to wait for this PE until it terminates searching to proceed to the next quadtree level.
Avoiding classification the utilization of the processors can be increased. In this part we present this algorithm, the main idea being that the assignments are made immediately when new tasks arrive to the master PE. Besides this two recursive algorithms will be given for storing the partition efficiently.

The master uses a temporary file, where the results returned by the slave processes are stored. This is needed, because we want to save some space in the final, compressed file. The master sends the jobs to the slaves, but it is unknown which one will finish sooner, so if we want to save some space with storing no domain coordinates, first we need some space to store the temporary data (instead of using files one can use for example binary trees). When the master divides a range block, always assigns a unique string to the blocks, according to the position of it. For example the string `114` uniquely determines the range block with size $\frac{n}{4} \times \frac{n}{4}$ (if the image is of size $n \times n$) which is the lower-right quadrant of the upper-left quadrant of the upper-left quadrant of the image (see Fig. 3 and 4).

The scheme of the algorithms are presented hereinafter.

Master:

```
Create the queues task and slave
Read the image parameters (size)
```

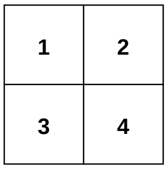


FIGURE 3. The numbering of the quadrants.

```
Put the four initial quadrants into *task*
Put the processor IDs into *slave*
nr_of_ranges := 4
While $task \neq \emptyset$ and $slave \neq \emptyset$ do:
    Get the first task and the first free slave
    Send the task to the slave
End While
While nr_of_ranges$\neq$0 do:
    Block until receive the parameters from a slave
    Put the slave back to *slave*
    If a good matching was found, then:
         nr_of_ranges := nr_of_ranges-1
         Insert the returned parameters into the temporary
         file with unique string tag
    Else
         Divide the range block into four quadrants
         Put them into *task*
         nr_of_ranges := nr_of_ranges+4
    End If
    While $task \neq \emptyset$ and $slave \neq \emptyset$ do:
         Get the first task and the first free slave
         Send the task to the slave
    End While
    If nr_of_ranges = 0, then:
         Send to each process the terminate-message
         Sort the temporary file after the unique string
         tag (this can be made in parallel)
         Write the image/compression parameters to the
         final file
```

```
                Write the partition table to the file
                Copy the needed transformation parameters from the
                temporary file to the final one
           End If
       End While
```

Slaves:

```
       Read the image data
       While TRUE do:
           Block until receive some task
           If the terminate-message was received, then:
                break
           End If
           Search for a matching domain block
           Send to the master process the obtained parameters from
           the search
       End While
```

After the slave processors had been finished their work (they have got the terminate-message from the master), the results are stored in the temporary file or structure. A record contains the range coordinates, the domain coordinates and the parameters of the pixel intensity transformation (s, o). Besides these, all of the records (range blocks) have a unique string, after which the data can be sorted. Then we need a so called *binary partition table*, based on which we will write a recursive function which will draw the decompressed image in the decoder. In this case we don't have to store the coordinates of the range blocks, and thus we can save space.
Suppose that we have the following simple partition represented by strings

$$111 \quad 112 \quad 113 \quad 114 \quad 12 \quad 13 \quad 14 \quad 2 \quad 3 \quad 4.$$

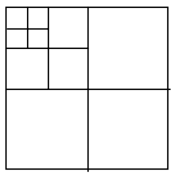


FIGURE 4. Visualization of the example.

Then the corresponding binary partition will be

$$(1(1(0000)000)000).$$

This can be constructed by the following recursive algorithm.

Partition writing algorithm:

```
procedure write_partition(length)
Begin
    For i=1, i<=4 do:
         if |buf.ustring| >length, then:
             write(1)
             write_partition(length+1)
         else write(0)
         if i<4, then read(buf)
    End For
End
```

```
call:  read(buf); write_partition(1)
```

In the decoder (viewer) a drawing function is needed, which visualizes the decoded picture calling the true drawing function.

Decompression algorithm based on the binary quadtree data:

```
procedure draw(x,y,rsize)
Begin
    read(buf)
    If buf=1 then draw(x,y,rsize/2)
    Else paint(x,y,rsize)
    read(buf)
    If buf=1 then draw(x+rsize,y,rsize/2)
    Else paint(x+rsize,y,rsize)
    read(buf)
    If buf=1 then draw(x,y+rsize,rsize/2)
    Else paint(x,y+rsize,rsize)
    read(buf)
    If buf=1 then draw(x+rsize,y+rsize,rsize/2)
    Else paint(x+rsize,y+rsize,rsize)
End
```

```
call:  draw(0,0,squaresize/2)
```

In the above algorithm the `read` function reads a bit from the binary partition into the variable `buf`. The function `paint` realizes the drawing of a range block. The variable `rsize` contains the vertical (horizontal) height (width) of the actual range block. The working of the function is simple: if we read 0 from the partition table we draw the corresponding range block, if the bit we have read was 1, then we call the function recursively with `rsize/2`. The range coordinates we call with depend on where the value 1 was read.

6. Test Results and Conclusions

The application was written in C/C++ under IRIX64 on the SGI Origin 3800 supercomputer (shared memory MIMD architecture with 128 R12000 400 MHz processors) situated at the Johannes Kepler Universität in Linz, Austria.

The tests were performed for two different pictures, which are not so relevant to show here, just to mention that the first one is a real-world image, while the second one is artificial. Although the results obtained are quite similar, we will present them separately. Moreover the test results were rather similar to those obtained using classification.

For measuring execution time and processor utilization we used the `timex` command under the IRIX64 operation system. This command can be parametrized to show the execution time for the "whole command" and among other things to show the execution time and the *hog factor* for each process(or). The hog factor gives the processor utilization – a real number between 0 and 1; it is calculated using the formula (total CPU time)/(elapsed time).

The plots (Fig. 5 and 6) were created using Mathematica. At the x-coordinate $x = 2$ we see the speedup (which is 1) and the processor utilization (which should be 1) of the sequential case, because there always have to be a master processor due to the used configuration.

In this paper we outlined the method of fractal image compression and the adaptive quadtree partition scheme. We discussed and analyzed the three main parallel distribution scheme applied for fractal encoding. The algorithm given in Section 5 avoids classification, but uses temporary data structure for efficient storage in the final, compressed file. We also gave an algorithm for the construction of the binary partition table. The results obtained show almost linear speedup up to a certain number of processors, depending on the complexity and size of the image

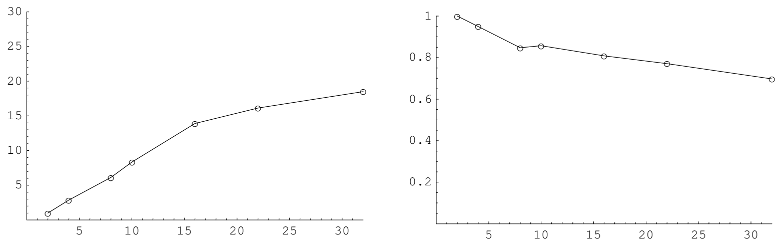


FIGURE 5. The speedup and average processor utilization results for the first image.

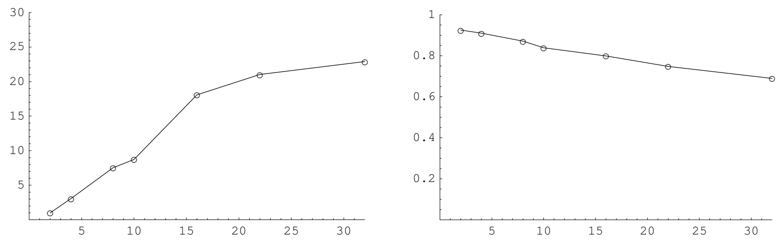


FIGURE 6. The speedup and average processor utilization results for the second image.

being encoded. Although the implemented algorithm performed quite efficient, a better parallelization would be needed to be worth using such an architecture.

REFERENCES

[1] M. F. Barnsley, *Fractals Everywhere*, second ed., Morgan Kaufmann, 1993.

[2] Z.-P. Bodó, *Parallel Fractal Image Compression*, Master Thesis, Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, 2004.

[3] M. Chady, *Application of the Bulk Synchronous Parallel Model in Fractal Image Compression*, School of Computer Science, University of Birmingham, http://citeseer.ist.psu.edu/255267.html.

[4] Y. Fisher (ed.), *Fractal Image Compression - Theory and Application*, Springer-Verlag, New York, 1996.

[5] J. Hämmerle, A. Uhl, *Parallel Algorithms for Fractal Image Coding on MIMD Architectures*, in Proceedings of the First International Conference on Visual Information Systems (Visual'96), Melbourne, February 1996, pp. 182–191.

[6] H. Hartenstein, M. Ruhl, D. Saupe, *Region-Based Fractal Image Compression*, IEEE Trans. on Image Process., Vol. 9, No. 7 (2000), pp. 1171–1184.

[7] D. J. Jackson, G. S. Tinney, *Fractal Image Compression Using a Circulating Pipeline Computation Model*, Technical Report UA-CARL-95-DJJ-01, Computer Architecture Research Laboratory, The University of Alabama, March 1995.

[8] D. J. Jackson, G. S. Tinney, *Performance Analysis of Distributed Implementations of a Fractal Image Compression Algorithm*, Concurrency: Practice and Experience, 8(5) (June 1996), pp. 357–380.

[9] M. Ruhl, H. Hartenstein, *Optimal Fractal Coding is NP-Hard*, in Proceedings DCC'97 Data Compression Conference, J. A. Storer, M. Cohn, eds., IEEE Computer Society Press, March 1997, pp. 261–270.

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

E-mail address: zpbodo@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

WORD SENSE DISAMBIGUATION BY MACHINE LEARNING APPROACH: A SHORT SURVEY

DOINA TĂTAR

Abstract. There is a renewed interest in word sense disambiguation (WSD) as it contributes to various applications in natural language processing. Applications for which WSD is potentially an issue are: Machine Translation, Information Retrieval (IR), QA systems, Dialogue systems,etc. In this paper we survey vector-based methods for WSD in machine learning approache.

1. Introduction

In the last ten years there has been a dramatic shift in computational linguistics to statistical learning methods (or corpus -based methods). This popularity of statistical methods has its origin in the growing availability of big machine-readable corpora and dictionaries. Some concrete publication statistics illustrate the extent of the revolution in NLP: as an example 63.5 % of the papers in ACL'97 proceedings and 47.7% of the papers in the journal Computational Linguistics in 1997 concerned corpus -based methods, compared with 12.8% and 15.4% in 1990. The argument for a statistical learning approach is to be able to interact successfully with uncertain and incomplete linguistic information. On the other hand natural language can provide machine learning with a variety of interesting and challenging problems such as very large feature space or very large training sets.

In this paper we follow a "machine learning" approaches categorization of WSD as: supervised, bootstrapping and unsupervised (sections 2,3,4). A "machine readable dictionary" based approach of WSD is presented in section 5. Some conclusions about Senseval 3 contest, developed in Marts -April 2004, where we participated with a team for Romanian language [15], will be formulated (section6).

Received by the editors: September 2004.

2000 *Mathematics Subject Classification.* 68T50,68Q32.

1998 *CR Categories and Descriptors.* I.2.7 [**Computing Methodologies**]: Artificial Intelligence – *Natural Language Processing*; G.3 [**Mathematics of Computing**]: Statistical Computing .

2. Machine learning approach in WSD

2.1. **The polysemy.** Word sense disambiguation is the task of assigning sense labels to occurrences of an ambiguous word. This problem can be divided into two subproblems [14]: sense discrimination and sense labeling. Word sense discrimination is easier than full disambiguation since we need only determine which occurrences have the same meaning and not what the meaning actually is.

In many applications full disambiguation is needed as for example in the machine translation. In the following we mean by WSD usually both discrimination and labeling of ambiguous words.

WSD has been a research area in NLP for almost the begin of this field due to the phenomenon of *polysemy* that means multiple related meanings with a single word. At least 40 % of semanticaly signifiant words are ambiguous. Also the problem of WSD is AI complete (that means its solution requires a solution to all the general AI problems of representing and reasoning about arbitrary) and it is one of the most important open problems in NLP [6].

2.2. **Meaning and context.** The systems in the supervised learning approach category are trained to learn a classifier that can be used to assign a yet unseen example to one of a fixed number of senses. That means we have a *trained* corpus, where the system learns the classifier and a *test* corpus which the system must annotate. So, supervised learning can be considered as a classification task, while unsupervised learning can be viewed as a clustering task. Word sense disambiguation (for polysemic words) is the process of identifying the correct sense of words in particular contexts. The precise definition of a sense is a matter of considerable debate within the community. However one would expect the words closest to the target word to be of greater semantical importance than the other words in the text. On the other hand, if two words frequently occur in similar context we may assume that they have similar meanings. The context is hence a source of information and is the only means to identify the meaning of a polysemous word.

Context is used in two ways: a) as *bag of words*, without consideration for relationships to the target word in terms of distance, grammatical relations,etc; b) with relational information. The *bag of words* approach works better for nouns than verbs but is less effective than methods that take other relations in consideration. Studies about syntactic relations determined some interesting conclusions: verbs derive more disambiguation information from their objects than from their subjects, adjective derive almost all disambiguation information from the nouns they modify and nouns are best disambiguated by directly adjacent adjectives or nouns [6].

2.3. **Vector Space Model.** In the following we will use the Vector Space Model (VSM)[9]: a context c is represented as a vector $\vec{c}$ of some features. The definition and the numbers of these features depend on the method selected. A common denominator between the methods is that they excavate information using co-occurrence and collocation statistics. The famous dictum “meaning is use” means that to understand the meaning of a word one has to consider its use in the frame of a concrete context. Context size can vary from one word at each side of the focus word to a more “window” or even the complete sentence. The notations used are:

- $s_1, \cdots, s_{Ns}$ the senses for w;
- $c_1, \cdots, c_{Nc}$ the contexts for w;
- $v_1, \cdots, v_{Nf}$ the features selected (or terms).

In generally, a number of most frequently used words are selected for use as features $v_1, \cdots, v_{Nf}$. When these features have a specific position located to the left and/or the right of the target word w they are *collocational* features, when we ignore the exact position of a feature, we call it a *cooccurrence* feature.

As example we can associate to a context c the vector $\vec{c}$:

- $\vec{c} = (w_1, \cdots, w_{Nf})$ where w_i is the number of times the word v_i occurs in context c;
- $\vec{c} = (w_1, \cdots, w_{Nf})$ where w_i is 1 if the word v_i occurs in context c, or 0 otherwise;
- $\vec{c} = (\cdots w_{i-1}, w_{i+1} \cdots,)$ where w_{i-1} (w_{i+1}) is 1 if the word v_i occurs in context c at the left (right) of the word w or 0 otherwise ;
- $\vec{c} = (\cdots w_{i-k}, w_{i-(k-1)}, ..., w_{i-1}, w_{i+1}, ..., w_{i+k} \cdots,)$ where w_{i-j} (w_{i+j}) is 1 if the word v_i occurs in context c at the left (right) of the word w at distance j or 0 otherwise ;
- $\vec{c} = (w_1, \cdots, w_{|W|})$ where w_i is 1 if the word v_i occurs in context c, or 0 otherwise, where v_i is a word from the entire text of $\mid W \mid$ words. In this last example the features are all the words in the contexts.

The similarity between two contexts c_a, c_b (of the same word or different words) is the *normalised cosine* between the vectors $\vec{c_a}$ and $\vec{c_b}$ [7]:

$$cos(\vec{c_a}, \vec{c_b}) = \frac{\sum_{j=1}^{m} w_{a,j} \times w_{b,j}}{\sqrt{\sum_{j=1}^{m} w_{a,j}^2 \times \sum_{j=1}^{m} w_{b,j}^2}}$$

and $sim(\vec{c_a}, \vec{c_b}) = cos(\vec{c_a}, \vec{c_b})$.

In all above examples the number w_i is the weight of th feature v_i. This can be the frequency f_i of the feature v_i (term frequency or tf). On the base of feature relevance principle, the features can be weighted to reflect the distance of the words to the focus word. For example, in a -3 +3 windows the weights for the 6 features could be: 0.25, 0.5, 1, 1, 0.5, 0.25.

Another method to establish the weight w_i is to capture the fashion of distribution of v_i in all the set of contexts by principle: features that are limited to a small number of contexts are useful for discriminating those contexts; features that occur frequently across the entire set of contexts are less useful in this discrimination. In this case one use a new weight for a feature, called "inverse document frequency", denoted by *idf* and defined as below:

Definition

Let us consider that the number of contexts is Nc and the number of contexts in which the feature v_i occurs is n_i. The *inverse document frequency* is :

$$idf_i = \frac{Nc}{n_i} \text{ or } idf_i = log(\frac{Nc}{n_i})$$

Combining the tf with *idf* we obtain $tf.idf$ weighting. In this case: $\vec{c} = (w_1, \cdots, w_{Nf})$, where $w_i = f_i \times idf_i$.

2.3.1. *Second-order co-occurrence.* In [14] the author introduces two types of vectors: word vectors and context vectors. The word vector for a word x is $\vec{x} = (w_1, \cdots, w_{Nf})$ where w_i is the number of times the word v_i co-occurs in the entire corpus. The features v_i can be selected as above. The context vector for a context of an ambiguous word is obtained by summing the vectors of all the vectors of the words in context. Therefore two contexts are similar if the words in these contexts occur with similar words (or, the contextual representation is similar). This is known as *strong contextual hypothesis.* Second order co-occurrence method is more robust than first-order method (as above).

3. Supervised learning of WSD

In such case a system is presented with a training set consisting of a set of input contexts labeled with their appropriate sense (disambiguated corpus). The task is to build a classifier which correctly classifies new cases based on their context of use. The two most known supervised algorithms are Bayesian classification and K-NN classification.

3.1. **Naive Bayes classifier approach of WSD.** This method was been introduced by gale, 1992. In this frame the context of a word w is treated as a bag of words without structure. What we want to find is the best sense s' for an input context c_{new} of an ambiguous word w. This is obtained as:

$$s' = argmax_{s_k} P(s_k \mid c_{new}) = argmax_{s_k} \frac{P(c_{new} \mid s_k) \times P(s_k)}{P(c_{new})} =$$

$$= argmax_{s_k} P(c_{new} \mid s_k) \times P(s_k)$$

The independence assumption (naive Bayes assumption) is:

$$P(c_{new} \mid s_k) = P(\{v_i \mid v_i \in c_{new}\} \mid s_k) = \prod_{v_i \in c_{new}} P(v_i \mid s_k)$$

This assumption (often referred to as a *bag of words* model)has two consequences:

- the structure and order of words in context is ignored;
- the presence of one word in the context doesn't depends on the presence of another.

This is clearly not true, but there is a large number of cases in which the algorithm works well.

Finally, $s' = argmax_{s_k} P(s_k) \times \prod_{v_i \in c_{new}} P(v_i \mid s_k)$.

Thus the supervised algorithm is:

- TRAINING Calculate:

$$P(s_k) = \frac{C(s_k)}{nr.of contexts}; P(v_i \mid s_k) = \frac{C(v_i, s_k)}{C(s_k)}$$

- TEST Calculate for a new context c_{new} the appropriate sense:

$$s' = argmax_{s_k} P(s_k \mid c_{new}) = argmax_{s_k} P(s_k) \times \prod_{v_i \in c_{new}} P(v_i \mid s_k).$$

3.2. **k-NN or memory based learning.** At training time, a k-NN model memorizes all the contexts in the training set by their associated features. Later, when proceeds a new context c_{new}, the classifier first selects k contexts in the training set that are closest to c_{new}, then picks a sense for c_{new}.

This supervised algorithm is:

- TRAINING Calculate $\vec{c}$ for each context c.
- TEST Calculate:

$$A = \{\vec{c} \mid sim(\vec{c_{new}}, \vec{c}) \ is \ maxim, \mid A \mid = k\}$$

that means A is the set of the k nearest neighbors contexts of $\vec{c_{new}}$.

$$Score(c_{new}, s_j) = \sum_{c_i \in A} (sim(\vec{c_{new}}, \vec{c_i}) \times a_{ij})$$

where a_{ij} is 1 if $\vec{c_i}$ has the sense s_j and a_{ij} is 0 otherwise.
Finally, $s' = argmax_j Score(c_{new}, s_j)$.

3.3. **Bootstrapping approach of WSD.** A major problem with supervised approaches is the need for a large sense tagged training set. The bootstrapping methods use a small number of contexts labeled with senses having a high degree of confidence. This could be accomplished by hand tagging with senses the contexts of an ambiguous word w for which the sense of w is clear because some *seed collocations* [19] occur in these contexts.

These labeled contexts are used as seeds to train an initial classifier. This is then used to extract a larger training set from the remaining untagged contexts. Repeating this process the number of training contexts grows and the number of untagged contexts reduces. We will stop when the remaining unannotated corpus is empty or any new context can't be annotated.

The bootstrapping approach is situated between the unsupervised and unsupervised approach of WSD.

For the word *bass* for example, we might begin with $fish$ as a resonable sense for sense $bass^1$ (bass as fish), as presented in WordNet [4] and *play* as a reasonable sense for $bass^2$ (bass as music). A small number of contexts can be labeled with the sense 1 and 2. These labeled contexts are used to extract a larger set of labeled contexts.

In [16] we present an original algorithm which combines this bootstrapping idea with elements of NB algorithms .

4. Unsupervised approach

Unsupervised approach of WSD does not use sense tagged data (training data) at all. Strictly speaking, the task of unsupervised disambiguations is of *sense discrimination* . In this case, vector representations of unlabeled contexts are grouped into clusters, according to a similarity measure. One cluster is considered as representing a sense and a new context c_{new} is classified as having the sense of the cluster to which it is closest according to the similarity measure. An advantage of unsupervised methods in disambiguation is that granularity of sense distinction is an adjustable parameter: a number of 10 clusters induces more fine-grained sense distinction than a number of 2 clusters, for example.

Let us consider that the objects to be clusterized are the vectors of n words, $\{w_1, w_2, \cdots, w_n\}$. A vector

$$\vec{w_i} = (w_i^1, w_i^2, \cdots, w_i^m)$$

is associated with a word w_i as above.

As clustering methods we can use an agglomerative or divisive hierarchical algorithm or a non-hierarchical (flat) clustering algorithm [16, 1]. In the first case each of the unlabeled context is initially assigned to its own cluster. New clusters are then formed in bottom-up fashion by successively fusion of two clusters that are most similar. This process continues until either a specified number of clusters is obtained or some condition about similarity measure between the clusters is accomplished. In generally, a good clustering method is defined as one that maximizes the within cluster similarity and minimizes the between cluster similarity.

- Agglomerative algorithm for hierarchical clustering [9]. The clustering algorithm begins by considering each word in its own cluster and ends when all the words are in the same cluster.
- Non-hierarchical clustering algorithm [9]. A non-hierarchical algorithm starts out with a partition based on randomly selected seeds (one seed per cluster), and then refine this initial partition. The algorithm stops when a measure of cluster quality is accomplished. As such measure we could select: group average similarity (average similarity between members); single link similarity (the similarity of two most similar elements of a cluster); complete-link similarity (the similarity of two least similar elements from a cluster).

  One of the non-hierarchical algorithm is **k-means**; it defines clusters as the mean (the average) of their member.

One other algorithm in unsupervised approach is EM-algorithm. In this case we start with a random computing of parameters $P(v_j \mid s_k)$ and then this parameters are reestimated in an estimation-modification cycle.

5. Dictionary-based disambiguation.

Work in WSD reached a turning point in the 1980s when large-scale lexical resources such as dictionaries, became widely available. The machine readable dictionaries (MRD) have a large development in these days. This section describes disambiguation methods that rely on the definition of senses of a word in dictionaries and thesauri.

5.1. **Lesk's algorithm. Reduced form**

Lesk (1986) starts from the idea that a word's dictionary definition is a good indicator for the senses of this word. He uses the definition in the dictionary directly.

Suppose that for a polysemic word w we have in a dictionary Ns senses $s_1, s_2, \cdots, s_{Ns}$ given an equal number of definitions $D_1, D_2, \cdots, D_{Ns}$. The new context to be disambiguated is c_{new}.

The idea of Lesk's algorithm is :

```
FOR $k = 1, \cdots, Ns$ DO
    $score(s_k) = \mid D_k \cap (\cup_{v_j \in c_{new}} \{v_j\}) \mid$
ENDFOR
Calculate $s' = argmax_k score(s_k)$
```

The score of a sense is number of words that are shared by the sense definition and context.

The method achieved 50-70% correct disambiguation [9].

5.2. **Two claim about senses: one sense per discourse (OSPD), one sense per collocation (OSPC).** In [19] Yarowsky observes that the sense of a target word is highly consistent within any given document or discourse. This is the content of OSPD principle. For example, if a document is about biological life, then each occurrence of the ambiguous word *plant* is more probably linked with the sense of "living being". If the document is about industrial aspects, then *plant* is more probably linked with the sense *factory.* Of course, the definition of discourse is central to the test of OSPD principle.

On the other hand, the sense of a target word is strongly correlated with certain other words in the same phrasal unit, named collocational features. By a collocation we mean usually first /second /third word to the left /right of the target word. In fact, there are words which collocate with the target word w with a high probability. Such a words are considered as strongest in the disambiguation process (OSPC principle). The algorithm proposed by Yarowsky combines both constraints [9].

6. Recent developments.

6.1. **Evaluation of WSD task.** Given the variety in the studies it is very difficult to compare one method with another. Evaluation of WSD programs has excited a great deal of interest. Producing a gold standard corpus annotated corpus is both expansive (many person-months of annotator effort) and hard (different individuals will often assign different senses to the same word-in-context). In

April 1997 a workshop of ACL included first time a session of WSD evaluation [12]. Beginning with 1998 (then in Sussex, England) in each two years take place some WSD evaluation workshops, named SENSEVAL. If at the first edition participated over 20 systems and most research has been in English, in 2004 for the first time was a section for Romanian language where participated 7 teams. For Romanian the manually sense-tagged was worked on a site open at University of North Texas. Almost half the systems used supervised training methods. The evaluation involves comparison of the output of each system using as measures *precision* and *recall.*

The upper bound for accuracy of a WSD system is usually human performance. This is between 97% and 99 % [9]. The lower bound is the performance of the simplest algorithm, baseline, usually the assignment of all contexts to the most frequent sense.

6.2. **Disambiguation and Information Retrieval.** WSD is only an intermediate task in NLP, like POS tagging or parsing. Examples of final applications for which WSD is potentially an issue are: Machine Translation, Information Retrieval (IR), Dialogue systems or improving Parsing. For example, the problem of finding whether a particular sense is connected with an instance of a word is likely the IR task of finding whether a document is relevant to a query. It is established that a good WSD program can improve performance of retrieval by 2%. As IR is used by millions of users, an average of 4 % improvement could be seen as very significant. A test in 1993 compared two term-expansion queries methods for IR: one in which each term was expanded with all related terms and one in which it was only expanded with terms related to the sense used in the query (disambiguated). The conclusion was that disambiguation did not improve the performance of term expansion. In [14] the authors propose a new methods that is beneficial for IR. In this method the features in the definition of vectors are senses, and not words: a feature in a context has a nonzero value if the context contains a word assigned to the sense represented by the feature. This method increased performance by 7,4 % compared to “features equal words” case. The two methods are opposites of each other in the following sense. Term expansion by related terms increases the number of matching documents for a query: if the query contains the word *cosmonaut* and expansion adds *astronaut*, then the number of documents is bigger (the documents containing the word *astronaut* are added). If the word *suit* occurs in the query used in the “legal” sense, then documents that contain *suit*, for example, in the “clothes” sense will not longer be founded. An excellent overview of work in WSD and IR can be found in [6].

References

[1] Avram, Lupşa, D., Şerban, G., Tătar, D.: Hierarhical clustering algorithms for repeating similarity values, Studia Universitatis "Babes-Bolyai", seria Informatica, 2003, nr 2, pp 61 - 72.

[2] Dagan, I., Lee, L., Pereira, F.: Similarity-based models of Word Cooccurences Probabilities, *MLJ*, 34(1-3), 1999.

[3] Daelemans, W.: Machine learning approach in *Syntactic Wordclass Tagging*, Kluwer Academic Publishers, pp 285-304, 1999.

[4] Fellbaum, C. (editor): *WordNet An Electronic Lexical Database*, The MIT Press, 1998.

[5] Gauch, S., Wang, J., Rachakonda, S. M.: A corpus analysis approach for automatic query expansion and its extension to multiple databases, *CIKM'97- Information and Knowledge management.*

[6] Ide, N., Veronis, J.: Introduction to the special issue on WSD: the state of the art, *Computational Linguistics*, 24(1) 1998, pp1-40.

[7] Jurafsky, D., Martin, J.: *Speech and language processing*, Prentice Hall, 2000.

[8] Kilgarriff, A.: *What is WSD good for?*, ITRI Technical Report Series- August, 1997.

[9] Manning, C., Schutze, H. : *Foundation of statistical natural language processing*, MIT, 1999.

[10] Marcus, S. : *Lingvistică matematică*, Ed. Didactică si Pedagogică, Bucureşti, 1966.

[11] Lin, D.:Automatic retrieval and clustering of similar words, *COLING-ACL'98*, Montreal, 1998.

[12] Resnik, P., Yarowsky,D. : Distinguishing Systems and Distinguishing sense: new evaluation methods for WSD , *Natural Language Engineering*, 1 , nr 1, 1998.

[13] Sahlgren, M. : Vector-based semantic analysis: representing word meanings based on random labels, in *The Acquisition and Representation of Word Meaning*,Kluwer Academic Publishers, 2001.

[14] Schutze, H.: Automatic Word Sense Discrimination, Computational Linguistics, *Computational Linguistics*, 24(1) 1998, pp97-123.

[15] Serban, G., Tatar, D.: UBB system at Senseval3, *Proceedings of Workshop in Word Disambiguation*, ACL 2004, Barcelona , July 2004 , pp 226-229.

[16] Tatar, D., Serban,G.: A new algorithm for WSD, *Studia Univ. "Babes-Bolyai", Informatica*, 2001, nr.2, pp 99-108.

[17] Tatar, D.: Inteligenta artificiala: demonstrare automata de teoreme, prelucrarea limbajului natural, *Editura Albastra, Microinformatica*, 2001.

[18] Widdows, D.: A mathematical model for context and word meaning, *Fourth International Conference on Modeling and using context*, Stanford, California, June 23-25, 2003.

[19] Yarowsky, D.: *Hierarchical Decision Lists for WSD*, Kluwer Acadmic Publishers, 1999.

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca
E-mail address: dtatar@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

HEURISTICS AND LEARNING APPROACHES FOR SOLVING THE TRAVELING SALESMAN PROBLEM

GABRIELA ŞERBAN AND CAMELIA-MIHAELA PINTEA

Abstract. In present, all known algorithms for NP-complete problems are requiring time that is exponential in the problem size. Heuristics are a way to improve time for determining an exact or approximate solution for NP-complete problems.

In this article is introduced and solved a problem based on a generalization of the Traveling Salesman Problem. We compare two classical algorithm results for the application: Branch and Bound and Nearest Neighbor and also two Ant Algorithms: Ant System and Ant Colony System. Being stochastic algorithms, Ant Algorithms have the solutions chosen according to a probability, which depends on the pheromone level, therefore they can be also considered as reinforcement learning techniques.

We also propose a reinforcement Q-learning method for solving the Traveling Salesman Problem.

Keywords: Combinatorial Optimization, Traveling Salesman Problem, Symmetric Graphs, Branch and Bound, Nearest Neighbor, Ant Algorithms, Heuristics, Agents, Reinforcement Learning.

1. Introduction

One of the most important and promising research field in recent years has been Heuristics from Nature. These heuristics, utilizing analogies with natural or social systems are using to derive non-deterministic heuristic methods and obtain very good results in NP-hard combinatorial optimization problems.

A heuristic *is a commonsense rule (or set of rules) intended to increase the probability of solving some problems. It is a general formulation that serves to guide investigation.*

Let us consider the Traveling Salesman Problem. The Traveling Salesman Problem is one of the most studied discrete optimization problems. TSP has many variations and a large number of applications.

Received by the editors: December 10, 2004.

2000 *Mathematics Subject Classification.* 68T05, 68T20, 90C27.

1998 *CR Categories and Descriptors.* I.2.6 [**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *Problem Solving, Control Methods, and Search*; G.1.6 [**Mathematics of Computing**]: Numerical Analysis – *Optimization* .

Today problems are harder and harder to solve, because of the multitude of inputs and the time needed to produce best results. That is why, in such cases we try to find an approximation of the best solution, using a heuristic.

A **construction heuristic** is an algorithm that determines a tour, in the graph associated to the problem, according to some construction rules, but does not try any improvements upon this tour. **Improvement heuristics** are characterized by a certain type of basic move to alter the current tour. We can combine these two types of heuristics in the following way: first we use a constructive heuristic to construct a tour. Then we repeatedly use an improvement heuristic to improve the current tour.

Heuristics in Artificial Intelligence are obtained using a certain amount of repeated trials, employing one or more agents, neurons, particles, chromosomes, ants. Two main features have to be balanced: the degree of exploits and the degree of exploration.

In case of multiple individuals, a cooperation or competition among individuals take place. These agents improve the solution quality at each iteration of the process and are able to adapt to new situations. So, these heuristics are adaptive (to the changes in the environment).

The most used construction heuristic is Nearest Neighbor Heuristic, which has gained in popularity because of the TS problem. The salesman starts at some city and then visits the city nearest to the starting city. From there he visits the nearest city that was not visited so far and so on until all cities will be visited, and the salesman returns to the starting point. It is probably close to the real salesman's approach. But is a poor heuristic, however.

Other heuristics often applied are the **genetic algorithms**, the **neural networks** and the heuristics inspired by social insects: **swarm intelligence**, **ant algorithms**.

In section 2, the subsection Theoretical support, we introduce a problem based on a generalization of Symmetric Traveling Salesman Problem. We will compare the optimal results obtained with two algorithms, in the classical approach, Branch and Bound and Nearest Neighbor and the results are shown in a table.

In the section Ant Algorithms we compare the results of two algorithms Ant System and Ant Colony System, which are applied for the problem mentioned above. The later one is more efficient and faster as we will see from the tables. There are also comparative graphics, that illustrate better the results.

Learning techniques can be considered as a kind of metaheuristics, that is why we consider in section 3 a reinforcement learning approach for solving the TSP problem.

2. A generalization of the Symmetric Traveling Salesman Problem

2.1. **Theoretical support.**

As is shown in [1], these are the definitions used for Generalized Symmetric Travel Salesman Problems.

The Generalized Symmetric Traveling Salesman Problem (GSTSP):

Definition 2.1 *Given a weighted complete digraph K_N and a partition $V_1, V_2, \ldots,$ V_k of vertices, find a minimum weight cycle containing exactly one vertex from each set $V_i, i = 1, \ldots, k$.*

Definition 2.2 *The sets V_i, are called* **clusters** *and a cycle containing exactly one vertex from each cluster is called a* **tour***.*

A common known definition of the Traveling Salesman Problem:

Definition 2.3 *Given N cities, if a salesman starting from his home city his goal is to visit each city exactly once and then return home, find the order of a tour such that the total distance travelled is minimum.*

We introduce the following **problem**:

Given N cities, if a salesman is starting from his home city and his house and want to visit each city exactly once, through exactly one house, and then return home, find the order of a tour such that the total distance travelled is minimum.

In this work, we consider that every city has a given number of houses. The salesman crosses each city once going trough one house from a city. We know the distances between houses from each city. We want to know the shortest way from the starting house and city making a tour of all cities, coming back in the same house and city.

The salesman chooses each time the house that is nearest from an unvisited city in the Nearest Neighbor method, and in the Branch and Bound method he visits all the cities and a random houses from each city, if there is a way between them.

The input files are:

- travel.in including a matrix N x N with 0 and 1, the matrix of distances ex. : (0,1,1,0,1), (1,0,1,1,1), (1,1,0,1,1), (0,1,1,0,1), (1,1,1,1,0);
- travel.txt specifies the city numbers and the numbers of houses in each city ex. (5,1)(1,2)(2,3)(3,1)(4,3)(5,1);
- distance.in specifies the distances between houses of different cities in a $N \times 5$ matrix.

File distance.in

$1,1,2,1,5$	$1,1,5,1,15$	$3,1,5,1,6$	$1,1,2,2,17$
$1,2,5,1,1$	$2,1,4,1,55$	$1,1,2,3,55$	$3,1,4,1,90$
$1,2,2,1,60$	$3,1,4,2,6$	$2,3,4,1,43$	$1,2,2,2,8$
$3,1,4,3,58$	$2,1,4,2,20$	$1,2,2,3,44$	$2,1,5,1,2$
$2,2,4,2,21$	$1,1,3,1,33$	$2,2,5,1,21$	$2,3,4,2,24$
$1,2,3,1,4$	$2,3,5,1,1$	$2,1,4,3,15$	$2,1,3,1,54$
$4,1,5,1,12$	$2,2,4,3,7$	$2,2,3,1,31$	$4,2,5,1,11$
$2,3,4,3,9$	$2,3,3,1,2$	$4,3,5,1,13$	

2.2. **Classical approach.** We use two methods for solving this application, namely Branch and Bound where we randomly choose a house from a city and Nearest Neighbor technique. For Branch and Bound we make 500 runs for each starting house from each city. In some cases Nearest Neighbor failed because there is not a way between cities 1-4.

We denote BB the Branch and Bound method, and NN the Nearest Neighbor method.

The results are presented in Table 1.

City	House	NN	NN Tours	BB	BB Tours
1	1,2		*Failed*	35	(1,1)(5,1)(4,2)(3,1)(2,3)
2	1	33	(2,1)(5,1)(1,2)(3,1)(4,2)	21	(1,2)(5,1)(4,2)(3,1)(2,3)
2	2	56	(2,2)(4,3)(5,1)(1,2)(3,1)	24	(2,1)(5,1)(4,2)(3,1)(1,2)
2	3	36	(2,3)(5,1)(1,2)(3,1)(4,2)	21	(2,3)(3,1)(4,2)(5,1)(1,2)
3	1		*Failed*	15	(3,1)(1,2)(5,1)(2,3)(4,3)
4	1	62	(4,1)(5,1)(1,2)(3,1)(2,3)	20	(4,1)(5,1)(1,2)(3,1)(2,3)
4	2		*Failed*	13	(4,2)(3,1)(1,2)(5,1)(2,3)
4	3	80	(4,3)(2,2)(1,2)(5,1)(3,1)	16	(4,3)(2,3)(3,1)(1,2)(5,1)
5	1	29	(5,1)(1,2)(3,1)(2,3)(4,3)	29	(5,1)(1,2)(3,1)(2,3)(4,3)

Table 1.

Remark 2.1 *From Table 1 we see that Branch and Bound found the best minimum solution and in the last case the solutions are equal, where the computational complexity is low.*

We mention that the time complexity of the NN approach is O(N^2).

Comparative results are also illustrated in Figure 1.

2.3. **Ant Algorithms.** This section shows a heuristic from nature, Ant Algorithm [10]. First, this natural heuristic was introduced in [11], [12].

The Ant Systems have proved to be efficient and robust global optimization methods. Ant Colony Algorithms are very complex hybrid systems. Inspired from the nature, they combine the power of metaheuristic in a highly distributed way. It can be considered as an evolutionary method, where agents are interacting in a cooperative multi-agent system. Ant systems, like Genetic Algorithms, are population based approaches.

At all ant-based algorithms the basic idea is the positive feedback mechanism. A virtual pheromone, used as reinforcement, allows good solutions to be kept in memory. It is also important to avoid good, but not very good solutions from becoming reinforced, which can lead to premature convergence, also called stagnation.

In order to escape local optima traps, a so-called negative feedback is used through pheromone evaporation.

Cooperative behavior is another important concept: ant colony algorithms make use of the simultaneous exploration of different solutions. Ant colonies are multi-agent systems, where the ants have the role of the cooperative agents. The communications between agents is done indirectly using the above-mentioned pheromones, which is also mentioned as the concept of stimergy.

The Ant Colony System Colony was developed to fill the gaps of the AS Algorithm, making more efficient and robust. The aim was to improve the performance of AS, that was able to find good solutions within a reasonable time only for small size problems.

ACS is based on several modifications of AS: a different transition rule, a different pheromone trail update rule, the use of local updates of pheromone trail to favor exploration.

The results for Ant System (AS) and Ant Colony System (ACS) are presented in Table 2.

City	House	AS	AS Tours	ACS	ACS Tours
1	1	57	(1,1)(2,1)(5,1)(4,2)(3,1)	57	(1,1)(2,1)(5,1)(4,2)(3,1)
1	2	38	(1,2)(2,2)(4,3)(5,1)(3,1)	38	(1,2)(2,2)(4,3)(5,1)(3,1)
2	1	91	(2,1)(1,1)(5,1)(4,2)(3,1)	33	(2,1)(5,1)(1,2)(3,1)(4,2)
2	2	56	(2,2)(4,3)(5,1)(1,2)(3,1)	56	(2,2)(4,3)(5,1)(1,2)(3,1)
2	3	29	(2,3)(4,3)(5,1)(1,2)(3,1)	29	(2,3)(4,3)(5,1)(1,2)(3,1)
3	1	29	(3,1)(2,3)(4,3)(5,1)(1,2)	29	(3,1)(2,3)(4,3)(5,1)(1,2)
4	1	184	(4,1)(2,3)(1,2)(5,1)(3,1)	105	(4,1)(5,1)(1,1)(3,1)(2,3)
4	2	33	(4,2)(2,1)(5,1)(1,2)(3,1)	36	(4,2)(2,3)(5,1)(1,2)(3,1)
4	3	80	(4,3)(2,2)(1,2)(5,1)(3,1)	73	(4,3)(2,3)(5,1)(1,2)(3,1)
5	1	75	(5,1)(4,2)(2,1)(1,1)(3,1)	57	(5,1)(2,1)(1,1)(3,1)(4,2)

Table 2.

Remark 2.2 *In half of the cases ACS-Algorithm had better results then AS-Algorithm.*

Comparative results are shown in Figure 2.

We have to mention that adding more agents (ants) to an Ant System leads to better performance.

The complexity of one cycle of the Ant Algorithm is O($N^2 \cdot M$) (N is the number of inputs, M is the number of ants), so it takes very long time to solve the problem with many inputs. Researchers have wondered how many artificial ants (agents) are needed for a given problem and they have found that the number of agents must be approximately equal to the number of inputs [12].

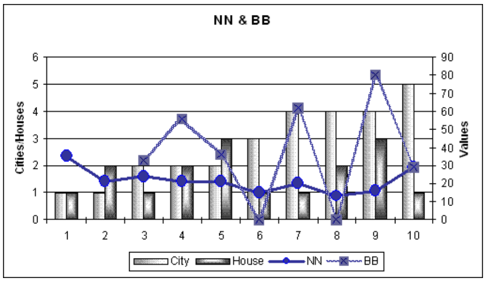


FIGURE 1. NN-BB

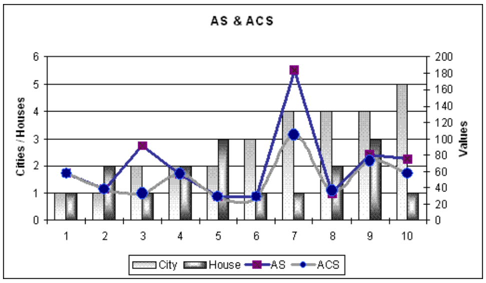


FIGURE 2. AS-ACS

3. Learning approach for solving the TSP problem

In this section we propose a reinforcement learning approach for solving the TSP problem.

Let us consider a finite number of "cities" and their coordinates on a map. We suppose that the cost of travel between a pair of cities is given by the euclidian between them. An agent (the traveling salesman) has to find the cheapest way of visiting all the cities and to return to the starting point. So, the goal of the agent is to learn by reinforcement the optimal policy.

In Reinforcement learning [14], the agent is simply given a goal to achieve and then the agent learns how to achieve that goal by trial-and-error interactions with its environment.

In a reinforcement learning problem, the agent receives a feedback, known as *reward* or *reinforcement*; the reward is received at the end, in a terminal state, or in any other state, where the agent has correct information about what he did well or wrong.

3.1. The SARSA Algorithm.

SARSA [13] is a reinforcement Q-learning algorithm, which combines the advantages of Temporal difference learning and Monte Carlo learning methods.

From the TD methods, the algorithm takes the advantage of learning at each step, instead of waiting the end of an episode. From the Monte Carlo methods, SARSA takes the advantage of going back and using the rewards obtained in each state in order to update the values of the previous action-state pairs and the capacity of functioning without the model of the environment in which the learning takes place [16].

The idea of the SARSA algorithm [13] is to apply the Temporal Difference methods to the state-action pairs, in comparison with the classical methods, where this methods are applied only to the states.

SARSA converges with probability 1 to an optimal policy and action-value function as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the Greedy policy.

The agent learns the optimal policy after some training sequences. A training sequence consists of a number of episodes, during which the agent has the following behavior:

- until a final state is reached (the final state is given by the goal of the agent) or until the number of steps exceed a maximum number, using an action selection mechanism (ϵ-Greedy or SoftMax [13]), the agent updates properly the Q-values (the values of the state-action pairs).

The complete description of the SARSA algorithm is given in [17].

3.2. **Experimental results.** In order to apply the SARSA algorithm described above, we make the following assumptions:

- there are N cities that the agent has to visit;
- from a given city, the agent can visit any other city (the graph is complete);
- the agent starts from the city 1 (the initial state in the RL task);
- the states in the RL task are the cities the agent can visit;
- a city reached by the agent at a given moment is considered to be a final state if the agent has visited all the cities so far;
- the possible actions of the agent in a given state (city) are *N-1* (the other cities that the agent can visit, excepting the current state);
- as an action selection mechanism we use the ϵ-Greedy approach;
- the reward received by the agent in a given state (city) is considered to be the euclidian distance between the current city and the city numbered with 1.

Because of the probabilistic action selection mechanism during the learning process, we repeat the training sequence of the agent several times.

For a simplest evaluation of the results, we choose the example in which there are 4 cities. The coordinates of the cities are given in Table 3.

City	Coordinates on the map
1	(10,10)
2	(10,11)
3	(11,10)
4	(13,11)

Table 3. Coordinates of the cities on the map.

We mention that the optimal tour is 1,2,4,3,1 and the total distance of this tour is **7,24**. The NN approach determines an approximation of the solution, 1,3,2,4,1 with the total distance **8,57**.

The results obtained after applying our learning algorithm are shown in Table 4.

Training sequence	Number of episodes	Tour	Total Distance
1	6	1,4,2,3,1	**8,57**
2	6	1,3,2,4,1	**8,57**
3	6	1,2,4,3,1	**7,24**
4	6	1,2,3,4,1	**7,81**
5	6	1,3,2,4,1	**8,57**
6	6	1,4,3,2,1	**7,81**
7	6	1,4,3,2,1	**7,81**
8	6	1,2,4,3,1	**7,24**

Table 4. Results obtained with the SARSA algorithm.

The results obtained with SARSA are better than the result obtained with the NN approach.

We have applied the algorithm for different values of **N** and for different number of training episodes and we observe that SARSA, in average, gives better results than the NN approach.

We have to mention that the time complexity of the TSP solving with the RL approach during one training sequence is O(N*e), where e is the number of training episodes.

3.3. Further work.

Further work can be done in the following directions:

- how to improve the learning rate;
- to apply other learning approaches for solving TSP (like LRTA* algorithm [18]);
- the generalization of the above approach for multi agent systems (systems in which several agents has to cooperatively learn by reinforcement to solve the TSP problem).

4. Conclusions

Any combinatorial optimization problem has an associated graph, which means that these problems can be solved in terms of graph theory and also provides us with an easy way to view the problem.

An heuristic method reduces the time complexity of the problem solving. As an example, a brute force approach for TSP has $O(2^N)$ complexity (N is the number of cities).

A dynamic programming algorithm also gives the global optimum for this problem (and for many other hard combinatorial optimization problems) but also takes exponential time.

Section 2 is based on GSTSP, where we can compare the cities with clusters and a house with a vertex. A cycle containing exact one house - vertex- from each city- cluster- is a tour. Branch and Bound is a consistent technique that exhaustive explore a searching space, following at each step all the minimum possibilities.

The Nearest Neighbor method we will apply in the case when we want an approximate solution, but obtained in a short time, with a reduced computational complexity. We can improve methods by taking randomly the equal minimal values, at each new execution of the algorithm.

We can extend the problem requiring that the traveller should cross exactly trough a given number of houses in each city, where the given number could be the minimum values of the houses from the cities.

In section 2 the same problem is solved with Ant Algorithms. Ant Algorithms are primarily used for combinatorial optimization, especially for NP-hard tasks. The Traveling Salesman Problem (TSP) is an important application, for which were developed several Ant Colony methods. The basic idea of Ant System is that of simulating the behavior of a set of agents that cooperate to solve an optimization problem by means of simple communications.

As a disadvantage it can be stated the multiple parameters used for these algorithms. It is very hard to set up the optimal values of the parameters, which can be very different from one problem to another.

As a conclusion, we may say that even though classic algorithms give an **exact** solution, heuristics may give very good solution, or even the best one and are very **handy techniques** because they are usually cheap to apply and combine. We should also try to combine heuristics with exact algorithms. Different heuristics can be obtained as follows: using a Greedy algorithm to choose each move, using multiple agents that start from different points in the graph, using clustering techniques to group regions in the search space, using clustering techniques to group agents, using local search techniques to improve solutions.

References

[1] G. Gutin, *Traveling Salesman and Related Problems*,Royal Holloway, Univ.of London, 2003

[2] G. Gutin, A. Punnen, *The Traveling Salesman Problem and its variations*,Kluwer Academic Publishers

[3] M. Dorigo, L.M. Gambardella, *Ant Colonies for the Traveling Salesman Problem*, BioSystems 43, 73-81, 1997

[4] M. Dorigo, L.M. Gambardella, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, IEEE Transactions on Evolutionary Computation 1(1) 53-66, 1997

[5] M. Dorigo, V. Maniezzo, A. Colorni, *The Ant System: Optimization by a Colony of Cooperating Agents*, IEEE Transaction on Systems, Man, and Cybernetics-Part B, 26(1) 29-41, 1996

[6] M. Dorigo, G. Di Caro, *The Ant Colony Optimization Metaheuristic*, in D. Corne, M. Dorigo, F. Glover(Eds.), New Ideas in Optimization, McGraw-Hill, 1999

[7] E. Bonabeau, M. Dorigo, G. Therraulaz, *Swarm intelligence: from natural to artificial systems*, Oxford University Press, 1999

[8] M. Hozefa Botee, E. Bonabeau, *Evolving Ant Colony Optimization*, Advances in Complex Systems 1 (2/3) 149-159, 1998

[9] V. Maniezzo, A. Carbonaro, *Ant Colony Optimization: An Overview*, in Celso C. Ribeiro, Pierre Hansen (Eds.), Essays an Surveys in Metaheuristics, Kluwer Academic Publishers, Boston/ Dordrecht/ London, pp. 21-44, 2002

[10] A. Colorni, F. Dorigo, V. Maniezzo, G. Righini, M. Trubian, *Heuristics from nature for hard combinatorial optimization problems* Published in International Transactions in Operational Research, 3,1, pag.1-21

[11] A. Colorni, F. Dorigo, V. Maniezzo, *Distributed Optimization by Ant Colonies* Proceedings of ECAL-91-European Conference on Artificial Life, Paris, France, F. Varela and P. Bourgine (Eds.), Elsevier Publishing, 1991,pp. 134-142

[12] A. Colorni, F. Dorigo, V. Maniezzo, *An Investigation of Some Properties of an Ant Algorithm* Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92), Brussels, Belgium, R. Manner, B. Manderick(Eds.), Elsevier Publishing, 1992, pp. 509-520.

[13] R. Sutton, A.G. Barto, *Reinforcement learning*, The MIT Press, Cambridge, England, 1998

[14] S.J.Russell, P.Norvig, *Artificial intelligence. A modern approach*, Prentice-Hall International, 1995

[15] M. Harmon, S. Harmon, *Reinforcement Learning - A Tutorial*, Wright State University, www-anw.cs.umass.edu/simmharmon/rltutorial/frames. html, 2000

[16] A. Perez-Uribe, E. Sanchez, *Blackjack as a TestBed for Learning strategies in Neural Networks*, Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'98

[17] G. Serban, *A Reinforcement Learning Intelligent Agent*, Studia Universitatis "Babes-Bolyai", Informatica, XLVI, Number 2, pp.9-18, 2001

[18] G. Weiss, *Multiagent systems - A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, London, 1999

"Babeş-Bolyai" University
E-mail address: gabis@cs.ubbcluj.ro

"George Coşbuc" National College
E-mail address: cami_mihaela@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

TASK MODELING IN SYSTEMS DESIGN

ADRIANA-MIHAELA TARŢA

Abstract. This paper aims to provide a discussion of how model-based approaches and the related tools can be used to obtain usable systems. More than that, in the last years the design of software systems is regarded as a multisciplinary area, where knowledge from psychology, sociology, etnography are interrelated. This paper will show how some techinques specific to these disciplines can be used in user interfaces design.

1. Introduction

People are often using models in their everyday life. Everytime when a more complex problem is encountered, the way of dealing with the complexity is by using models, i.e. identifying the main aspects that should be taken into account and their relations. It is not suprising that in the software design domain, this approach has been adopted and now is a technique often used to represent the various aspects of the problem that must be solved (domain, data, interaction, etc.). The models' goal is to provide a structural description of the relevant information. The purpose of model-based design is to identify high-level models which allow designers to specify and analyse interactive software applications from a more semantic-oriented level rather than starting immediately to address the implementation level. This allows them to concentrate on important aspects without being confused by the implementation details. Models are capturing the semantic meaningful aspects and allow the designers to manage more easily the increasing complexity of interactive systems [Pat04]. The most succesful modeling technique used in software engineering is UML (Unified Modelling Language), which focus on modeling the objects composing a system and then modeling of activities that

Received by the editors: December 10, 2004.

2000 *Mathematics Subject Classification.* 68P99.

1998 *CR Categories and Descriptors.* **D.2.2** [Software]: Software Engineering – *Design Tools and Techniques*; **H.5.2** [Information Systems]: Information Interfaces and Presentation – *User Interfaces* .

manipulates the system. The object-oriented approaches are successful at engineering the software implementation level. One of the basic usability principles says: "focus on the user and their tasks", that's why a new trend have gained attention in the last years, when the comfort of people, the ergonomic aspects of people work are emphasized more and more. This new trend is called task modeling. The task oriented approaches focus also on objects and activities, but the difference from object-oriented approaches is that the task-oriented approaches first identify the tasks that users perform and then the object manipulated.

2. Task Analysis and Modeling

Before discussing the task analysis and modeling techniques, we have to understand the meaning of "task" concept. A task is an activity that should be performed in order to reach a goal. A goal is a desired modification of state or an inquiry to obtain information on the current state of an object (system)[GM02].

Task models' goal is to identify useful abstractions highlighting the main aspects that should be considered when designing interactive systems. The main advantage of task models is that they represent the logical activities that an application must support. In order to better understand the user, his work and his expectations, there are two types of task models: descriptive and prescriptive. The descriptive task models describe the way in which the task is performed currently (even in an manual way, or supported by another system). The prescriptive task model describes how the task should be supported by a new developed system [Pri03].

The descriptive task models are developed by psychologists, etnographers and domain experts, using data aquisition techniques like: interviews, video recording or direct observation. After data acquisition the information are gathered, analyzed, and a descriptive task model is generated.

The prescriptive task model is build on the descriptive task model, but it is restructured, in order to include the modification introduced by the use of new technology (some redundant tasks will be eliminated, some user tasks will change in application tasks or interaction tasks, when the user and the system perform a task in a collaborative manner). The prescriptive task model is conceived by the designer teams and human factor experts, in order to assure a high level of usability.

A task model in the design of an interactive system describes a set of activities that users intend to perform while interacting with the system. There are two types of task models: *the system task model* that provide information about how the designed system requires tasks to be performed, and *user task model* which is how users expect to perform their activities. It is desirable that these two models be very similar, otherwise some usability problems will be present.

Task models are essential in the design of interactive systems because they represent the logical activities that should support users in reaching their goals. From informal representations only (like mock-ups or scenarios) the designers won't have the necessary information to support the design decisions. Task models represent the intersection between user interface design and more systematic approaches by providing designers with means of representing and manipulating an abstraction of activities that should be performed to reach user's goals.

Task models are built after task analysis is performed. Task analysis aims to identify the relevant tasks and how activities are performed currently. Task models describe the semantic and temporal relations between the identified tasks.

3. Task Modeling Techinques and Tools

Task analysis and modeling is an old method of describing and analyzing the structure of work and was first used in psychology. The first important method of task analysis was HTA (Hierarchical Task Analysis), developed in 1960 [DFAB93]. The method represents the structure of task in a hierachical decomposition. The representation of task structure had two forms: a textual description, where indentation and numbering of task/subtasks were used, and a graphical representation, using trees. The order of task performance was given by plans which describe the performance order using the tasks/subtasks numbers.

This approach have been used lately in other task analysis techniques like GOMS (Goals, Operators, Methods, and Selection Rules), where the tasks and it's goals were represented hierarchically, using a textual description.

Even in the last years, the hierarchical decomposition of tasks is considered the best suited method to describe task models. Because of the interdisciplinary effort for the design of interactive systems, the graphical representation was adopted. Task analysis methods like GTA (Groupware Task Analysis)[vdVvW00] and CTT (ConcurTaskTrees)[GM02] use a tree representation of task in their associated tools. GTA is a method that starts from the basic idea that nowadays people are performing their work in a collaborative way, so the work analysis is made for groups of people. People who are performing a common set of tasks form *roles*, and the system or the people is considered to be an *agent*. In task performance, agents manipulate *objects*. The performance of a task is triggered by *events* or by another task. In GTA the temporal relation between tasks can't be precisely specified. The method doesn't provide any formal mean of specifying such relations, providing support only for the description of sequential tasks. These are the main concepts used to describe the task world and the temporal relations between tasks [vW01].

While the number of task analysis method is signifiant, there are only few tools that support the use of these methods and that generates task models. Only two tools are freely available on the Internet: EUTERPE, the task analysis and modeling tool that supports the use of GTA technique [vW01], and CTTE (ConcurTaskTrees Environment)[GM02] corresponding to CTT (ConcurTaskTrees) analysis and modeling method. EUTERPE provides functionality for describing the task models in a graphical manner and to store all the objects, responsible roles, triggered tasks and triggering events for each task. The models can be verified using a validation tool which verifies a set of constraints that must hold for every correct model (e.g. tasks that are never triggered, roles which aren't responsible for any task). EUTERPE also generates the work flow based on the triggering events or tasks, but the temporal description is quite ambiguous, because of the lack of formal methods for representing the time relations between tasks.

The problem of unambiguous specification of time relations is well solved in CTT (ConcurTaskTrees) that has an associated tool called CTTE (ConcurTaskTrees Environment). CTT use also a hierarchical representation of tasks using task trees, but the relations between tasks is described using LOTOS operators: choice, order independency, concurrent, concurrent with info exchange, CTT disabling, suspend/resume, enabling, enabling with info exchange [GM02]. In CTT descriptions are also considered the agents, roles and objects related to tasks. CTTE provides means of describing single user task models and also cooperative task models. The tool provides also a simulator that helps designers understanding the logical order of tasks' performance, because when having task models for complex systems is difficult to verify all the possible paths in task execution.

From our experience in using the above mentioned tools the conclusion is that each of them has its advantages and disadvantages. EUTERPE supports the phase of task analysis because of the various information about task's world which can be stored by filling the task templates forms and associating documentation fragments (video fragments, audio fragments, sketches). From a GTA approach, we can say that EUTERPE supports very well the building and the validation of task model 1 (the descriptive model of the task). On the other hand, CTTE supports the building, verification and simulation of task model 2 (the prescriptive one). The use of temporal operators is an essential feature and opens a road to the specification of presentations and interactions of the modeled system.

4. Task Modeling in the design of interactive systems

In the design of interactive systems the task analysis can be used with different goals:

- requirement analysis - when through a task analysis designers identify requirements that should be satisfied in order to obtain an useful system;
- design of interactive applications - the information from task models is used to better identify the interaction techniques and the presentations of the application (in this case the modeling technique should provide temporal information about the logical order of tasks);
- usability evaluation - the system task model and the user task model are compared in order to get information about the matching between these models; also, having a structured task model some techniques like KLM (Keystroke Level Model) can be applied to get information about the time needed to perform a task - this kind of approximation may be used also to compare different task models addressing the same problem.

Task models are built for many different situations: usually, a task model is built when designing a new application with the goal of obtaining precise information about: the order of task performance, objects from the domain manipulated in the task performance process, agents and roles responsible for the task performance, events triggering task performance, preconditions and postconditions for task performance. Also, a task model can be built for an existing application, in this case the goal is to understand the underlying design, analyse its limitations, and solutions to overcome them. Task model can address the problem of designing an entire application, or just a part of it.

4.1. **User Interface Design Based on Task Analysis.** Having stored in a repository all the information acquired in the task analysis phase, having the structured representation of activities implied by a task performance in a task model and having the relations between activities, objects manipulated, agents or roles responsible for each task, using some additional information about the type of tasks (editing, selection of one/multiple choices, responding to alerts, etc.) and some descriptions about the objects manipulated (numerical values, texts, etc), we have the possibility of automatic determination of the main presentations of the interactive system (the kind of widgets needed in a presentation); also, from the task model we know the navigational path in the interactive system. The information obtained might support the process of building an abstract specification of the user interface (using specification languages like XML, XIML (eXTended Interface Markup Language) or UIML (User Interface Markup Language)) which can be then rendered in a concrete user interface. Task models play a key role when the goal is to develop multiple user interfaces for different devices. In this situation, the task models capture all the relevant requirements at the task level and using some tools the abstract specifications, and then the concrete specification for different platforms can be generated. The structure of task model provides guidance for the navigational structure of the concrete user interface and for the grouping of the object manipulated in the task performance [LM03].

5. Problems with Task Modeling Techniques and Tools

Although the use of task modeling approaches even in real case study have proved its efficiency, there are some aspects criticised in the domain literature [vW01, Pat04].

One problem related to the development of task models in software design process is the extra time needed. Indeed, the process of analyzing people work and of representing the most important aspects in a structured model need a time for data acquisition and analysis, but now there are tools that help designers in building such task models or functionality of modeling tools (e.g. EUTERPE can display multimedia files registered in the working environment that helps designers in building task models, the CCT task models can be built from an informal specification (scenarios) using the El-TaskModels tool which helps designers identifying roles, objects, tasks from informal descriptions [GM02]. Also, if the goal is to develop a complex application, it is hard to manage complexity without creating models that provides the semantic and temporal information. The graphical representation of models takes advantage of the readability for different kind of specialists involved in the design process.

Another criticism of task modeling technique is that they aren't applicable for creative tasks, because in that special case there is no structured task model. Task models support only functionality where users want to achieve a goal and the activities needed can be modeled in a structured fashion.

Task models are often used for the design of user interfaces and interaction styles. Task models lead to verb-noun interaction style which is not preffered by some users who like first select an object and then the actions on that objects. We have to mention that these are two different ways of achieving goals, and task models can be used for implementing both interaction styles, even the verb-noun style is more intuitively extracted from models.

6. Conclusions

We have presented in this article a different approach that can be used in the design of usable software systems. This approach focuses on user and user's tasks and from the analysis of this information a task model is built that supports the further design of interaction and presentations of the designed system. We have presented also some available tools that support the task analysis and design. Because this approach is still at the beginning the number of such tools is reduced, but from the experience of using them we know now what are the requirements for such a tool: the presence of a graphical editor for task trees, support for multiple representations (formal and informal), consistency between representation, support for handling different media types in the documentation process of building models (audio files, video fragments), support for models' verification and support for simulation. Most of these requirements are met by the above mentioned presented tools, but there is an essential aspect that isn't covered by any of them. While building a task model, there are several persons implied in this activity, each of them bringing it's own contribution to the task model. There isn't any tool that offers support for collaborative building of tasks that implies the management of versions and changes. We consider that this is an important aspect for future research and development of task analysis and modeling tools. Using the above mentioned tools we have found the difficulties in using the tak analysis methods and the corresponding task analysis tools. Our future work will focus on developing a new task analysis and modeling tool based on the GTA approach. Some improvements will be made on the specification of temporal relations between tasks following the ConcurTaskTrees approach. The tool should have the functionality of automatic generation of an abstract user interface specification from the task models.

References

[DFAB93] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-Computer Interaction.* Prentice Hall, 1993.

[GM02] Carmen Santoro Giulio Mori, Fabio Paternò. Ctte: Support for developing and analyzing task models for interactive system design. *IEEE Transactions on Software Engineering*, Vol. 28(9), Sepember 2002.

[LM03] C. Santoro L. Marucci, F. Paternò. *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, chapter Supporting Interactions with Multiple Platforms Through User and Task Models, pages 217–238. Wiley and Sons, 2003.

[Pat04] Fabio Paternò. Model-based tools for pervasive usability. Technical report, University of Pisa (Italy), 2004.

[Pri03] Costin Pribeanu. *Introducere în interacţiunea om-calculator*, volume 1 of *Interacţiunea om-calculator*. Matrix Rom, 2003.

[vdVvW00] Gerrit van der Veer and Martijn van Welie. Task based groupware design: putting theory into practice. In *DIS '00: Proceedings of the conference on Designing interactive systems*, pages 326–337. ACM Press, 2000.

[vW01] Martijn van Welie. *Task-based User Interface Design*. PhD thesis, Vrije Universiteit, 2001.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, 1, M. Kogalniceanu Street, RO-400084 Cluj-Napoca, Romania, EUROPE

E-mail address: `adriana@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

PARALLEL MUTATION BASED GENETIC CHROMODYNAMICS

ANCA GOG AND D. DUMITRESCU

Abstract. Genetic Chromodynamics is a strategy for preventing premature convergence and detecting multiple optimal solutions. A new technique of applying genetic operators is proposed. The Parallel Mutation Based Genetic Chromodynamics (PMGC) improves the local search from the standard approach and combines it with the global search, by using an appropriate mutation strategy.

Keywords: Evolutionary Computation, Genetic Operators, Genetic Chromodynamics

1. Introduction

Genetic Chromodynamics is an evolutionary technique for multimodal optimization. The main idea of Genetic Chromodynamics is to force the formation and maintenance of stable sub-populations. This aim is achieved by using a local interaction scheme ensuring sub-population stabilization in the early search stages. One of the Genetic Chromodynamics principles is the stepping stone search mechanism, every solution being involved in a search process by means of recombination or mutation operators.

A new technique of applying genetic operators is proposed. This method preserves the benefits of local interaction scheme from the standard approach of Genetic Chromodynamics and improves it by means of recombination followed by a small rate mutation, and is also making the exploration of the solutions space by means of a large step mutation. These two operators are simultaneously applied, and the two obtained offspring compete for survival only if they belong to the same optimal region and after they compete with the parents. If they belong to regions of different optimal points, both of them will be kept in the next generation. The enlargement of the population is not a problem, because of the merging operator that is fusing similar individuals. This way the population size will be reduced.

Received by the editors: November 2004.

2000 *Mathematics Subject Classification.* 62E99, 68T99.

1998 *CR Categories and Descriptors.* I.2.8 [**Artificial Intelligence**]: Problem solving, Control methods, and Search – *Heuristic methods*; G.3 [**Mathematics of Computing**]: Probability and Statistics – *Distribution functions* .

For the small rate mutation, the Gaussian strategy is applied. The Cauchy mutation strategy is applied for the large step mutation. The experimental results show an improvement of the existing technique, by applying the proposed method on several benchmark multimodal functions. The proposed model is called *Parallel Mutation Based Genetic Chromodynamics* (PMGC).

The paper is organized as follows: Section 2 presents an overview of the Standard Genetic Chromodynamics (SGC) approach. The new technique of applying genetic operators is presented in Section 3. Section 4 describes the two mutation strategies used in the new approach. Experimental results prove the efficiency of the proposed model in Section 5. There are also conclusions presented in Section 6.

2. Genetic Chromodynamics

Many evolutionary techniques for solving multimodal optimization problems have been proposed. Genetic Chromodynamics is a non-niching strategy that maintains population diversity and detects multiple optima. The main principles of Genetic Chromodynamics are:

1. population size is variable;

2. sub-population structure is not predefined, but emergent;

3. each individual within the current population is considered to be a stepping-stone for the search process;

4. a new operator for merging very close individuals is considered;

5. at convergence, the number of sub-populations represents the number of optimal solutions.

In the standard Genetic Chromodynamics approach, every solution is involved either in recombination or mutation. The best between the dominant parent and the offspring created by recombination or mutation will be kept in the next generation. In this approach, the offspring obtained by mutation will be unconditionally accepted if it is better than its parent. Thus, this strategy can be useful in the first generations of the search process, but in the later stages could cause some optima extinction. To prevent this situation, the mutation will always create offspring belonging to the interaction range, by choosing an appropriate value of the standard deviation parameter [4].

3. Parallel Mutation for Genetic Chromodynamics

We may consider a new recombination – mutation scheme, depicted in Figure 1. This scheme may be viewed as a new composed search operator. Recombination followed by mutation can be considered as a unique search operator. It will be called RM. We also consider another mutation operator, called M, which acts independently. In the proposed model, the difference between the two mutation operators is made. M1 has a small rate of mutation, and M2 has a higher step.

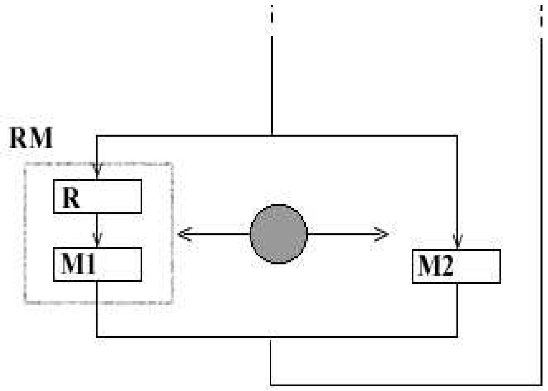


FIGURE 1. PMGC model

The proposed search model applies the two operators simultaneously and is called Parallel Mutation Based Genetic Chromodynamics (PMGC). Both genetic operators will create offspring that will compete for survival first with their parents. After this, the best of them will survive if the two obtained offspring belong to the region of the same local optimum. If they belong to regions of different local optima, both of them will survive. This approach will not affect the local interaction scheme and will improve the exploration of the solutions space.

The role of the very small rate mutation (operator M1) is to avoid the interference between recombination and high mutation. If the offspring obtained after recombination is a good solution for the problem, we do not want to loose this descendent by applying a high mutation rate [1]. The proposed strategy is the Gaussian one, which ensures a small mutation rate.

Larger mutation step (operator M2) ensures the exploration of the solutions' space. Also, when recombination cannot be made anymore, this mutation will attend faster the optimal points. Using a Cauchy mutation strategy ensures the larger step (see Section 4).

The reason why both descendents survive if they belong to different regions of optimal points is to avoid the extinction of useful potential optima. This situation can interfere if the offspring obtained by M has a higher fitness that the fitness obtained by RM and it belongs to a region corresponding to a different optimum point. This way, a useful optimum point represented by the RM offspring is lost.

In order to keep all useful solutions, distance between the offspring is taken into account. If the two possible solutions belong to different regions (the second does not belong to the interaction range of the first one) then both of them will be kept.

4. Gaussian and Cauchy mutation strategies

Gaussian mutation strategy accomplishes the request of a small mutation rate that follows the recombination. Cauchy strategy is used for the mutation that acts simultaneously to it. The two mutation strategies are described in the following paragraphs:

4.1. **Gauss mutation strategy.** Let us consider the following chromosome:

$$\{x_1, x_2, \dots, x_m\}$$

If the element x_k is selected for mutation, $k = 1, \dots, m$, the result will be:

$$\{x_1, \dots, x'_k, \dots, x_m\}$$

Gaussian mutation has two parameters: a mean value and a standard deviation. In this mutation approach, the following relation transforms the element x_k into x'_k:

$$x'_k = x_k + \eta N_k(0,1), \tag{1}$$

where the correction step η is the standard deviation for Gaussian mutations. $N_k(0,1)$ denotes a normally distributed one-dimensional random number with mean 0 and standard deviation 1. This random number is generated anew for each value of k, $k = 1, \dots, m$ [3].

4.2. **Cauchy mutation strategy.** Cauchy mutation can perform longer jumps with high probability. The search step size is much larger than the search step of the Gaussian mutation. The shape of Cauchy density function is similar to that of the Gaussian density function but approaches the axis so slowly that an expectation does not exist [7]. As a result, the variance of the Cauchy distribution is infinite. Figure 2 shows the difference between Cauchy and Gaussian density functions.
The one-dimensional Cauchy density function centered at the origin is defined by:

$$f_t(x) = t/\pi(t^2 + x^2), -\infty < x < \infty,$$

where $t > 0$ is a scale parameter. The corresponding Cauchy distribution function is given by:

$$F_t(x) = 1/2 + arctan(x/t)/\pi.$$

In this mutatin strategy, the following relation transforms the element x_k into x'_k:

$$x'_k = x_k + \eta\delta_k, \tag{2}$$

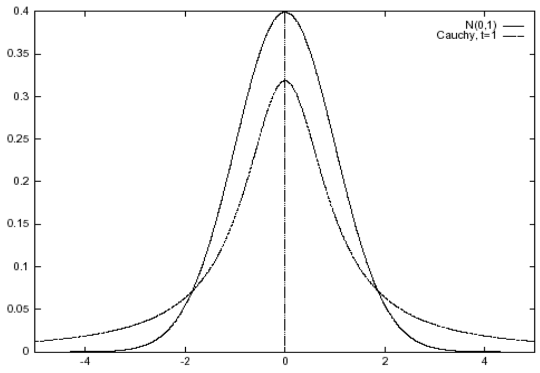


FIGURE 2. Comparison between Cauchy and Gaussian density functions

where δ_k is a Cauchy random variable. The method used to generate Cauchy random numbers is based on the inverse transformation, i.e. the inverse distribution function, and is defined by:

$$\delta_k = t tan[\pi(U(0,1) - 1/2)],$$

where U(0,1) denotes the unit rectangular variate [5].

In the continuous case, the uniform distribution is also called the *rectangular distribution* because of the shape of its probability density function. The *standard uniform distribution* is the continuous uniform distribution with the values of a and b set to 0 and 1 respectively, so that the random variable can take values only between 0 and 1.

Generally t is taken to be 1. δ_k is generated anew for each value of k. Correction step η may have the same value as in correction rule (1).

5. EXPERIMENTAL RESULTS

Multimodal functions having local optima are often regarded as being difficult to optimize. The effectiveness of the method is demonstrated on a number of eight multimodal test functions [2], [6]. One-dimensional functions have been chosen for implementation, but the method can be easily extended to n-dimensional functions.

The following eight benchmark functions are used for testing the proposed model:

$$f_1(x) = [0.002 + \sum_{i=1}^{25} 1/(i + (x - a_i)^6)]^{-1}, -1000 \leq x \leq 1000$$
$$f_2(x) = 0.00025(x - 100)^2 - cos(x - 100) + 1, -600 \leq x \leq 600$$

$f_3(x) = -\sum_{i=1}^{5} c_i[exp(-(x-a_i)^2/\pi)cos(\pi(x-a_i)^2)], 0 \le x \le 10$
$f_4(x) = 20 - 20exp(-0.2x + e - exp(cos(2x\pi)/n)), 1 \le x \le 30$
$f_5(x) = 10 + (x^2 - 10cos(2x\pi)), 1 \le x \le 5$
$f_6(x) = 418.9828872724339 - xsin(\sqrt{|x|}), -500 \le x \le 300$
$f_7(x) = lnxsin(e^x) + sin(3x), 0.1 \le x \le 4$
$f_8(x) = exp(-2ln2((x-0.1)/0.8)^2)sin(5x\pi)^2, 0 \le x \le 3$

Remarks:
(i) Coefficients a_i, $i = 1, \ldots, 25$ in function f_1 (Shekel's Foxholes function), are components of the vector a=(-32 -16 0 16 32 -32 ... 0 16 32).
(ii) Coefficients c_i, $i = 1, \ldots, 5$ in function f_3 (Langerman's function), are components of the vector c=(0.806 0.517 1.5 0.908 0.965), and coefficients a_i, $i = 1, \ldots, 5$ are components of the vector a=(9.681 9.4 8.025 2.196 8.074).

Example:
Let us consider the function defined as:

$$f_3(x) = -\sum_{i=1}^{5} c_i[exp(-(x-a_i)^2/\pi)cos(\pi(x-a_i)^2)],$$
$$0 \le x \le 10.$$

Algorithm parameters are given in Table 1.

TABLE 1. PMGC parameters for test function f3.

Initial population size	**300**
Interaction (mating) radius	**0.5**
Mutation step size	**0.001**
Merging threshold	**0.1**
Number of epochs before stopping	**10**
Remarks	**The function has one global optimum and seven local optima.**

Results obtained are given in Table 2.

Initial population is depicted in Figure 3. Populations obtained at several intermediate stages (epochs) are depicted in Figures 4 and 5. Final population is depicted in Figure 6. Final population contains only problem optima. All optima are correctly detected.

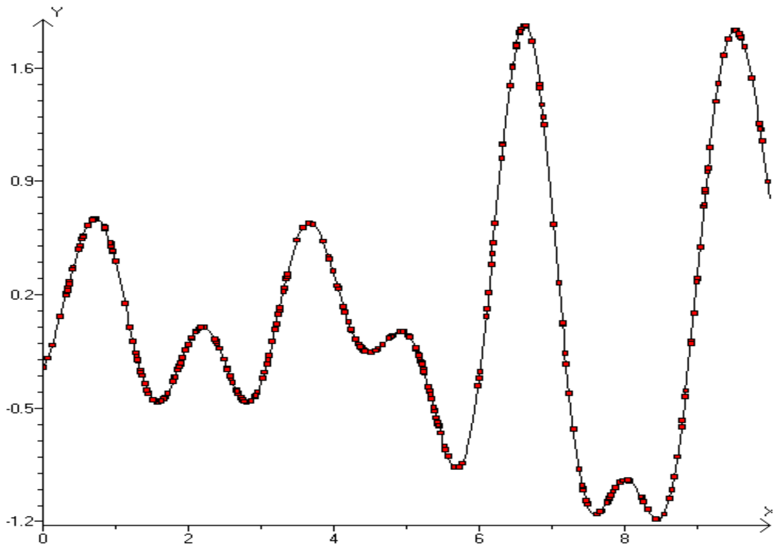


FIGURE 3. Initial 300 members population.

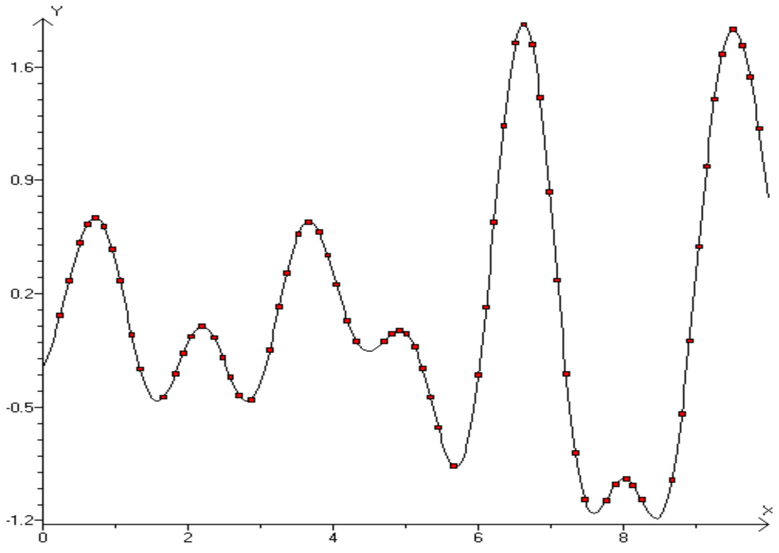


FIGURE 4. PMGC population after 2 epochs. Population has 68 members.

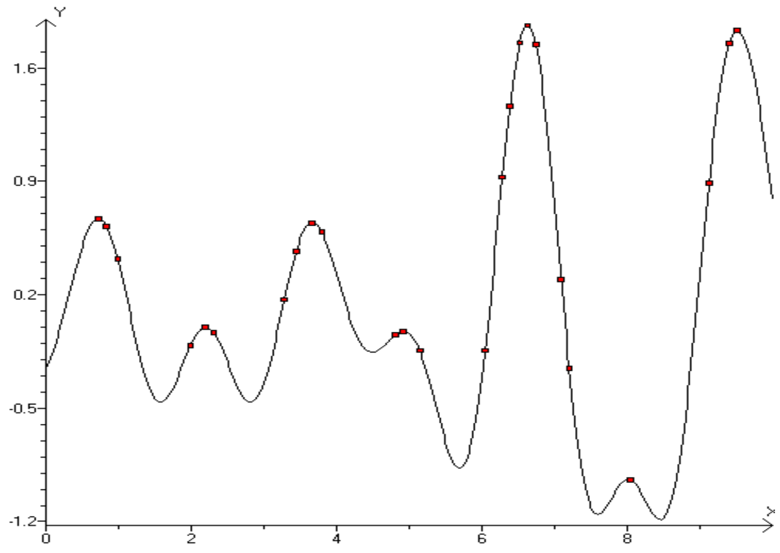


Figure 5. PMGC population after 5 epochs. Population has 25 members.

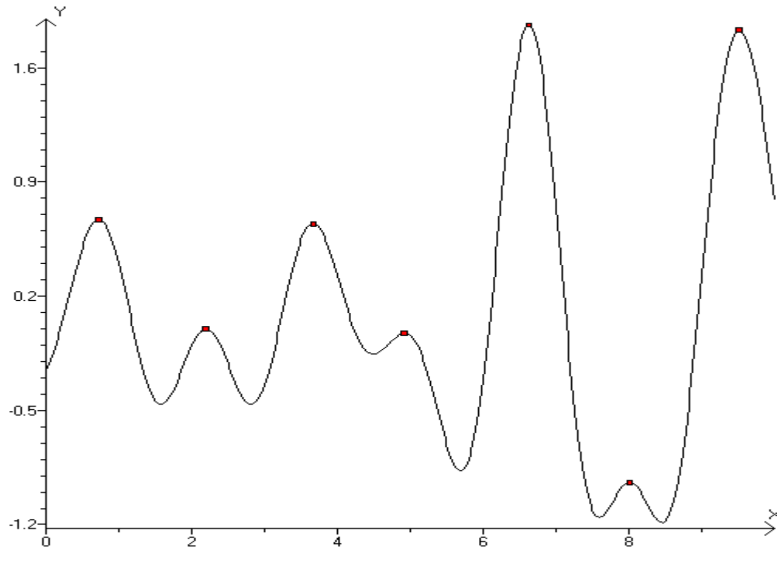


Figure 6. PMGC final population obtained after 35 epochs. Population has 7 members representing optimal solutions. All optimal solutions are correctly detected in this run.

Table 2. PMGC results obtained for test function f3 with parameters in Table 1.

Number of detected optima	**7**
Number of epochs needed for convergence	**35**
Number of recombinations involved	**232**
Number of mutations involved	**242**

Table 3 summarizes the final results obtained after 100 runs of SGC and PMGC for all considered benchmark functions. It can be seen that PMGC performs better than SGC consistently for these functions. The proposed PMGC model outperforms SGC as regards the number of epochs needed to find the local optimal points. The results regard the average number of generations needed to locate the optimal points in both SGC and PMGC approaches after 100 runs of both algorithms. Also, the best value obtained in both approaches can be seen in Table 3.

Table 3. Results obtained after 100 runs of PMGC and SGC; the average and the best number of epochs needed to find the optimal points.

Test function	**Average PMGC**	**Average SGC**	**Best PMGC**	**Best SGC**
f_1	**57**	**83**	**39**	**64**
f_2	**53**	**66**	**32**	**35**
f_3	**71**	**102**	**31**	**60**
f_4	**55**	**99**	**35**	**50**
f_5	**52**	**107**	**34**	**51**
f_6	**53**	**95**	**37**	**52**
f_7	**69**	**792**	**52**	**685**
f_8	**52**	**53**	**35**	**38**

Regarding the number of optimal points detected, PMGC and SGC have similar results. For the considered test functions, in both approaches, in 98% of the cases all the optimal points have been detected.

6. Conclusions

A new model of applying genetic operators consistently improves the results obtained by the standard approach of Genetic Chromodynamics. The model is based on the parallel action of two genetic operators: recombination followed by small rate mutation and high rate mutation. The obtained offspring will compete with the parents for survival. The two chromosomes with the higher quality will survive if they belong to regions of different optimal points. The Gaussian mutation strategy is proposed for the mutation that follows recombination and the Cauchy strategy is proposed for the mutation that acts parallel to recombination.

References

[1] Bäck, T., Fogel, D.B., Michalewicz, Z. (Editors), *Handbook of Evolutionary Computation*, Institute of Physics Publishing, Bristol and Oxford University Press, New York, 1997.

[2] Digalakis, J.G., Margaritas, K. G., *An experimental study of benchmarking functions for genetic algorithms*, Intern. J. Computer Math., 2002, Vol. 79(4), pp. 403–416.

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

[4] D. Dumitrescu, A. Gog, *A new evolutionary technique for multimodal optimization.* International Conference on Computer and Comunications (ICCC) Oradea, 2004, p. 119-123.

[5] Maulik, U., Bandyopadhyay, S., Pakhira, M., *Clustering Using Annealing Evolution: Application to Pixel Classifcation of Satellite Images*, The 3^{rd} Indian Conference on Computer Vision, Graphics and Image Processing: ICVGIP-2002 Proceedings, December 2002.

[6] Naujoks, B., Bäck, T., Willmes, L., *Test Case Computation Results* INGENET Project Report D 5.21 (ICD), January 2000.

[7] Yao, X., Liu, Y., Lin, G. *Evolutionary Programming Made Faster*, IEEE Transactions on Evolutionary Computation, 3(2), 1999, p.82-102.

Babes-Bolyai University of Cluj-Napoca, Faculty of Mathematics and Computer Science, Computer Science Department

E-mail address: {anca,ddumitr}@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

SPEAKER INDEPENDENT PHONEME CLASSIFICATION IN CONTINUOUS SPEECH

MARGIT ANTAL

Abstract. This paper examines statistical models for phoneme classification. We compare the performance of our phoneme classification system using Gaussian mixture (GMM) phoneme models with systems using hidden Markov phoneme models (HMM). Measurements show that our model's performance is comparable with HMM models in context independent phoneme classification.

Key words: Phoneme classification, Gaussian mixture models, Continuous speech recognition, Unsupervised learning

1. Introduction

In order to build a continuous speech recognition system it is necessary to model sub-word units. It is impossible to model all the words even in a reduced size vocabulary. Word models will be formed from the concatenation of sub-word units. These units will be the phonemes. The phonemes are a set of base-forms for representing the sounds in a word. Replacing one phoneme in a word with another is usually enough to turn that word into a different word (or no word).

There are two general methods used for evaluating phoneme recognition systems, classification and recognition. In classification the segmentation is given and the goal is to find the most likely label for each segment of speech, given its beginning and end time. In the more general problem of recognition, on the other hand, both the labels and the segmentation are unknown.

This paper investigates the problem of phoneme classification. All the measurements were done on the DARPA TIMIT speech corpus which is a manually segmented speech database. The state-of-the-art technology in speech recognition is Hidden Markov Models [5]. Commercial speech recognition systems usually use HMM models for phonemes with continuous densities. The common model is a left-to-right HMM with three states (see Fig.1). The reason for using three states

Received by the editors: November 2004.

2000 *Mathematics Subject Classification.* 68T10, 65C40.

1998 *CR Categories and Descriptors.* I.5.1 [**Computing Methodologies**]: Pattern Recognition – *Models*; I.5.4 [**Computing Methodologies**]: Pattern Recognition – *Applications* .

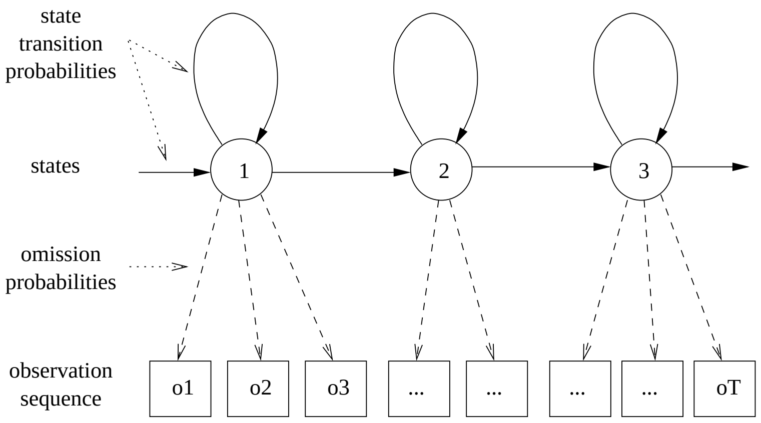


FIGURE 1. Three-state left-to-right phoneme model

is that the first part and last part of a phoneme are usually different from the middle due to co-articulation. This means that every state is responsible to model one third of a phoneme. Several authors [1] noticed that the transition probabilities have a negligible impact on the recognition accuracy and are often ignored. That is why in our system we used GMM for modeling phonemes. A GMM is a HMM with a single state, so there are no state transition probabilities, only observation emitting probabilities. Measurements show that our system's classification accuracy is close to systems using HMM for context-independent phoneme models and with Maximum-likelihood estimation of the parameters.

The structure of this paper is as follows. First we provide a short review of the phoneme classification problem and present the GMM modeling technique. Then we describe the acoustic features which were used. The final part of the paper discusses aspects of the experiments and the obtained results.

2. Phoneme Classification

2.1. **Phoneme Modeling.** Phoneme classification in continuous speech is a special pattern classification problem. It is easier than the phoneme recognition, because in classification phoneme boundaries are given. There are several approaches to pattern classification. Roughly we can divide these approaches into generative and discriminative modeling.

The HMM-GMM approach belongs to the generative models. Recently these models were extended and new discriminative training algorithms were proposed [6]. In the HMM-GMM approach each phoneme in the speech signal is given as a series of observation vectors $O = o_1,\ o_2, \ldots, o_T$, and each one has one model for each phoneme c. These models return a class-conditional likelihood $P(O|c)$.

The models are composed of states, and for each state we model the probability that a given observation belongs to this state. Time warping is handled by state transition probabilities, that is the probability that a certain state follows the given state. Supposing that the observations are independent (which is true only if we perform feature extraction in a very special way), the final probability for a sequence of observations can be computed using the forward algorithm [5].

HMM has several different forms like the discrete observation HMM and the continuous observation HMM. The discrete observation HMM is restricted to the production of a finite set of discrete observations. On the other hand in continuous observation HMM the observations are continuous and vector-valued. The usual way is to use a mixture of weighted Gaussian probability density function characterizing the distribution of observations within each state. The probability density function is given as

$$p(o_j) = \sum_{i=1}^{k} P_i \mathcal{N}(o_j, M_i, \Sigma_i) \tag{1}$$

where $\mathcal{N}(\ . \ , M_i, \Sigma_i)$ denotes the multidimensional normal distribution with mean M_i and covariance matrix Σ_i, k is the number of mixtures, and P_i are positive weighting factors which sum to 1. The D-dimensional normal density function has the form

$$\mathcal{N}(o_j, M_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^D det(\Sigma_i)}} e^{-\frac{1}{2}(o_j - M_i)^T \Sigma_i^{-1} (o_j - M_i)} \tag{2}$$

While HMMs are used for computing the class conditional likelihoods, discriminative models try to model the surfaces that separate the classes. For discriminative models one can use Artificial Neural Networks [7] (ANN) or a relatively new technology called Support Vector Machines [8]. A special ANN which was successfully applied for phoneme recognition is the Self-Organizing Map (SOM). This was introduced by Kohonen [9] for Finnish phoneme models and good results were also obtained for English vowels [10].

2.2. **Gaussian Mixture Models.** Our classification system was built for generative models. Instead of three states HMM we used only one state HMM which is a GMM. Our aim was to achieve the same recognition accuracy as for the classical three states HMM. We performed context independent classification (context dependent models perform better) on the DARPA TIMIT database and obtained slightly better classification accuracy than those using HMM on the same database.

Gaussian Mixture Models can be viewed as a special type of clustering. Clustering is an unsupervised classification and in the contributed papers appears as normal decomposition. Decomposition of a distribution into a finite number of normal distributions has been studied extensively. The parameters of normal distributions can be estimated using the method of moments or maximum likelihood estimation. The maximum likelihood (ML) method is more reliable especially for high-dimensional cases. Having high dimensional feature vectors as data, we chose this one. Let us assume that $p(X)$ consists of k normal distributions as

$$p(X) = \sum_{i=1}^{k} P_i \cdot p_i(X) \tag{3}$$

where $p_i(X)$ is a normal distribution ($\mathcal{N}(X, M_i, \Sigma_i)$) with the expected vector M_i and covariance matrix Σ_i. Our problem simplifies to estimation of P_i, M_i, Σ_i $(i = 1 \ldots k)$ from the n available samples X_1, X_2, …, X_n, drawn from $p(X)$. Our goal is to maximize $\prod_{j=1}^{n} p(X_j)$ with respect to P_i, M_i,and Σ_i under the constraint $\sum_{i=1}^{k} P_i = 1$. This optimization problem must be solved iteratively. One solution to this problem can be found by applying the maximum likelihood estimation technique (ML). Taking the logarithm of $\prod_{j=1}^{n} p(X_j)$, the criterion to be maximized becomes $\sum_{j=1}^{n} \ln p(X_j)$. Using the Lagrange multiplier's method the criterion to be maximized is

$$J = \sum_{j=1}^{n} \ln p(X_j) - \mu \left(\sum_{i=1}^{k} P_i - 1 \right), \tag{4}$$

where μ is a Lagrange multiplier. Computing the derivatives with respect to P_i, M_i and Σ_i and making them zeros we obtain the following formulas.

$$P_i = \frac{1}{n} \sum_{j=1}^{n} q_i(X_j) \tag{5}$$

$$M_i = \frac{1}{N_i} \sum_{j=1}^{n} q_i(X_j) X_j \tag{6}$$

$$\Sigma_i = \frac{1}{N_i} \sum_{j=1}^{n} q_i(X_j)(X_j - M_i)(X_j - M_i)^T, \tag{7}$$

where

$$q_i(X) = \frac{P_i \, p_i(X)}{\sum_{j=1}^{k} P_j \, p_j(X)} \tag{8}$$

is the a posteriori probability of X belonging to class i and satisfies $\sum_{i=1}^{k} q_i(X) = 1$. N_i is the number of samples belonging to class i.

The parameter estimation process can be described as follows.

Step 1. Choose an initial classification, $\Omega(0)$, and calculate P_i, M_i and Σ_i $(i = 1, \ldots, k)$.

Step 2. Having calculated $P_i^{(l)}$, $M_i^{(l)}$, and $\Sigma_i^{(l)}$, compute $P_i^{(l+1)}$, $M_i^{(l+1)}$, and $\Sigma_i^{(l+1)}$ by 5, 6 and 7 . The new $q_i^{(l+1)}(X)$ can be calculated as

$$q_i^{(l+1)}(X_j) = \frac{P_i^{(l)} \cdot p_i^{(l)}(X_j)}{\sum_{s=1}^{k} P_s^{(l)} p_s^{(l)}(X_j)} \tag{9}$$

Step 3. When $q_i^{(l+1)}(X_j) = q_i^{(l)}(X_j)$ for all $i = 1, \dots k$ and $j = 1, \dots n$, then stop. Otherwise, increase l by 1 and go to step (2)

In order to reduce the computation time we can force the covariance matrices to be diagonal. This assumption is justified since we can extract statistically uncorrelated features from speech. A diagonal covariance matrix allows expressing (2) as:

$$N(o^{(j)}, M^{(i)}, \Sigma^{(i)}) = \frac{1}{\sqrt{(2\pi)^D \prod_{k=1}^{D} (\sigma_k^{(i)})^2}} e^{-\frac{1}{2}\sum_{k=1}^{D} \frac{(o_k^{(j)} - M_k^{(i)})^2}{(\sigma_k^{(i)})^2}}$$

where $M^{(i)} = (M_1^{(i)},\ M_2^{(i)},\ \dots, M_D^{(i)})$, $\sigma^{(i)} = (\sigma_1^{(i)},\ \sigma_2^{(i)},\ \dots, \sigma_D^{(i)})$ and $o^{(j)} = (o_1^{(j)},\ o_2^{(j)}, \dots, o_D^{(j)})$.

3. Experimental results

3.1. **The DARPA TIMIT speech corpus.** We used the TIMIT corpus for all experiments. This corpus was designed for training and testing continuous speech recognition systems. The database contains 6300 sentences, 10 sentences uttered by each of 630 speakers from 8 major dialect regions of the United States. The data were recorded at a sample rate of 16 KHz and a resolution of 16 bits. This corpus was manually segmented so the phonetic boundaries are given. The phoneme set is divided into the following 6 categories: vowels, stops, affricates, fricatives, nasals, semivowels (and glides). Other symbols are used for silence and closure intervals of stop consonants and affricates. There are 61 symbols used for labeling phonetic segments but most research papers present results on a reduced 39 symbol set. Table 1 presents the reduced 39 TIMIT symbol set.

3.2. **Features, training and testing.** We used for feature extraction 16 ms frame with 8 ms frame shift. We chose 16 ms for frame length because the phonemes belonging to the Stop category are usually very short. For example we obtained 17.93 ms average length for the *b* phoneme. This length was computed considering 915 occurrences of the phoneme in the speech corpus. From every frame we computed 12 mel frequency cepstrum coefficients[2, 3] and the energy of the frame. These coefficients do not incorporate any information about the way the signal changes over longer periods. However, it is well known that such information is essential in identifying sounds. In order to incorporate such

Table 1. TIMIT symbols

Category	Group	Category	Group	Category	Group
Vowel	ah, ax, axh	Vowel	ih, ix	Fricative	z
Vowel	iy	Semivowel	el, l	Fricative	f
Vowel	ih, ix	Semivowel	r	Fricative	th
Vowel	eh	Semivowel	w	Fricative	v
Vowel	ey	Semivowel	y	Fricative	dh
Vowel	ae	Semivowel	hh, hv	Stop	b
Vowel	aa, ao	Nasal	m, em	Stop	d
Vowel	aw	Nasal	n, en, nx	Stop	g
Vowel	ay	Nasal	ng, eng	Stop	p
Vowel	oy	Affricate	jh	Stop	t
Vowel	ow	Affricate	ch	Stop	k
Vowel	uw, ux	Fricative	s	Stop	dx
Vowel	axr, er	Fricative	sh, sz	Closure	epi,q,bcl, dcl,gcl, kcl,pcl,tcl,pau,h#

information we computed the first and second order derivatives which resulted in 39 features for a frame [4].

The 6300 utterances from TIMIT corpus were split into 4620 training utterances and 1680 testing ones. We performed three types of training. In the first type we used only 200 occurrences of every phoneme from the training set, in the second type we selected 400 occurrences and in the third one we used 1000 occurrences per phoneme. From these data we trained our phoneme models. For testing we used only the first 200 utterances from the standard 1680 set.

For every experiment we used the reduced 39-TIMIT phoneme set.

3.3. **Baseline GMM system.** In this experiment we fixed all parameters of the system except the number of mixtures. We used only diagonal covariance matrices in order to speed up the parameter estimations. Table 2 summarizes the overall classification accuracy obtained for different number of mixtures and for different numbers of phoneme occurrences used in training the models.

The best result for every type of training is in bold type. From this table we can conclude that for a fixed number of training data always exist an optimal model. If we use 200 occurrences per phonemes, the optimal model is formed by 32 normal densities. When we increased this number, there were not enough training data for accurate estimation of model parameters.

TABLE 2. Classification accuracies vs. number of Gaussians

Nr. of Gaussians	200 occ./phone	400 occ./phone	1000 occ./phone
1	41.56%	42.41%	44.68%
2	45.27%	47.77%	47.73%
4	48.54%	50.37%	51.20%
8	49.27%	52.34%	52.49%
16	51.98%	55.99%	55.68%
32	**53.55%**	57.10%	58.54%
64	52.68%	**57.77%**	58.89%
128		57.28%	60.16%
256			**60.43%**

TABLE 3. HMM

Paper	Frame size/shift	Features	Mixtures/state	Accuracy
[6]	32ms/10ms	MFCC-13+*Delta*	5	58.97%
[11]	25.6ms/unspec.	CC-12+*Delta*	8	57.10%
[12]	20ms/10ms	MFCC-18+*Delta*	unspec.	63.00%

3.4. **Baseline HMM systems.** Comparing our results with results obtained by other researchers is not an easy task, although the experiments are conducted on the same speech corpus. This difficulty is due to the incomplete presentation of the parameters of the experiments. In this section we try to summarize results obtained by others in the task of phoneme classification using HMM technology for phoneme modeling. We present results obtained by context independent phoneme models with measurements performed on TIMIT corpus. All the cited papers used 3 state hidden Markov phoneme models. Table 3 presents the results obtained by other research papers on the same task. It must be noted that [12] used full covariance matrices and the others (including our paper) diagonal covariance matrices for normal densities modeling.

More results can be found on the phoneme recognition topic, especially reported on the context dependent phoneme recognition task.

3.5. **Frames selection.** Several authors select only a few frames from the middle of each phoneme and use only these frames for training the models [9, 10].

Table 4. Classification accuracies vs. number of frames

Nr. of frames	Classification accuracy
3	53.07%
5	56.06%
7	56.70%
all	57.10%

Sometimes they use such a frame selection for simplifications, but it can be proved experimentally that the middle part of phonemes contain the real phoneme specific features. Our experiment was performed using as models 32 Gaussian mixtures and 200 occurrences per phonemes in training. Instead of using all frames belonging to the phoneme segment we used only a fixed number of frames and computed the recognition accuracy for the overall system. We ran this experiment for the following number of frames: 3, 5, 7 and all frames belonging to the phoneme. Table 4 summarizes the results.

3.6. **Intra-category classification.** The best classification accuracy for 400 occurrences per phoneme in training was obtained for 64 Gaussian models. We calculated the classification accuracies for the six phoneme categories. Using these models we computed the intra-category classification accuracy. Obviously the intra-category classification is higher than the all phonemes classification (see Table 5) because classification accuracy decreases with increasing the number of classes. One way to deal with this problem is to divide the entire phoneme set into phoneme categories by a category classifier and then to recognize phonemes in each category by a phoneme classifier. These categories contain all phonemes from Table 1 except silence, because this is the only category having only one phoneme group.

We computed the confusion matrix for 64 Gaussian models with 1000 occurrences per phoneme in training. The best classified phoneme was the vowel *oy* and the worst *uh*. Both are vowels. The most common errors are between symbols belonging to the same category (see Table 6).

4. Conclusions and Future Work

The main conclusions of this paper are as follows. For a fixed number of training data always exist an optimal model. This was demonstrated experimentally. Our measurements show that using GMM as phoneme models one can reach slightly better classification accuracy than using three-states hidden Markov models. Experiments show that the middle part of the phoneme contains the most phoneme

TABLE 5. Intra category and all phonemes classification accuracies

Phoneme category	All phonemes	Intra category
Vowels	56.05%	60.70%
Nasals	48.63%	61.26%
Fricatives	68.80%	77.64%
Semivowels	59.91%	84.47%
Stops	51.19%	60.78%
Affricates	57.35%	72.80%

TABLE 6. The ten most common errors

Hand Label	Recognizer Label	Percentage of all errors
n, en, nx	ng, eng	2.7%
Closures	dx	2.6%
ih, ixx	ah,ax,ax-h	1.9%
b	g	1.9%
ih, ix	iy	1.6%
d	g	1.6%
ih, ix	ey	1.4%
r	axr, er	1.3%
Closures	dh	1.3%
Closures	g	1.3%

specific information. The confusion matrix demonstrates the viability of category classification scheme, because the most common errors are between symbols belonging to the same category.

In the future we would like to implement the category classification scheme and after that to transform our system into a phoneme recognition system. Another aim is to test our system using context dependent phoneme models and to improve the parameter estimation using other methods than the ML (Maximum Likelihood).

5. Acknowledgements

We are grateful to *Artificial Intelligence Research Group of the Hungarian University of Szeged*[1] for the possibility to use their speech database for these experiments.

References

[1] M. K. Omar, M. Hasegawna-Johnson, S. Levinson, "Gaussian Mixture Models of Phonetic Boundaries for Speech Recognition", Automatic Speech Recognition and Understanding Workshop, 2001.
[2] L.R. Rabiner, B.H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, Englewood Cliffs, 1993.
[3] J.R. Deller, Jr. J. H.L. Hansen, J. G. Proakis, "Discrete-Time Processing of Speech Signals", John Wiley&Sons, 2000.
[4] X. Huang, A. Acero, H.-W. Hon, "Spoken Language Processing", Prentice Hall, 2001.
[5] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition." Proceedings of the IEEE, vol. 37, no. 2, pp. 257-86, February, 1989.
[6] R. Chengalvarayan, L. Deng, "Speech Trajectory Discrimination Using the Minimum Classification Error Learning", IEEE Transactions on Speech and audio Processing, Vol. 6, No. 6, pp. 505-515, Nov. 1998.
[7] C. M. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press Inc., New-York, 1996.
[8] V. N. Vapnik, "Statistical Learning Theory", John Wiley & Sons Inc., 1998.
[9] T. Kohonen, "Self-Organizing Map", 3^rd edition, Springer, Berlin, 2001.
[10] N. Arous, N. Ellouze, "Cooperative supervised and unsupervised learning algorithm for phoneme recognition in continuous speech and speaker-independent context", Neurocomputing 51, pp. 225-235, 2003.
[11] B. Logan, P. Moreno, "Factorial HMMs for acoustic modeling", ICASSP, Vol. 2, pp. 813-816, 1998.
[12] R. Merwe, "Variations on Statistical Phoneme Recognition - a hybrid approach", master thesis, 1997.

Sapientia – Hungarian University of Transylvania, Faculty of Technological and Human Sciences, 540053 Târgu-Mureş, Romania
E-mail address: `manyi@ms.sapientia.ro`

[1]*Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged,* H-6720 Szeged, Aradi vertanuk tere 1., Hungary, http://www.inf.u-szeged.hu/speech

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

PERFORMANCE ANALYSIS MODEL FOR GOAL DRIVEN MEASUREMENTS IN SOFTWARE DEVELOPMENT PROCESS

D. RADOIU AND A. VAJDA

Abstract. The paper proposes a Performance Analysis Model (PAM) as base for analysis of goal driven measurements within software development process.

The model is developed on common view of the process for both acquirer and supplier enabling parties both to identify problem areas and to improve the overall process.

Keywords: Performance Analysis Model, Process Measurements, Outsourcing Process Management, Application Development, Improvement.

1. Introduction

The research was organized as a practical project at Infopulse[1] and a research project at Petru Maior University[2] of Tirgu Mures.

The paper firstly reviews the Acquirer and Supplier roles and responsibilities in a sourcing process, points on process areas which are within the control of both parties, proposes a set of relevant goal driven measurements, and a performance analysis model to interpret them.

Issues in sourcing are complex and multidimensional yet both Supplier and Acquirer have similar fundamental objectives such as:

- on time/schedule
- at cost/on budget
- with all required functionalities
- without defects/of required quality (no rework after delivery)

Received by the editors: December 2004.

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.9 [**Software**]: Software Engineering – *Management.*

[1]www.infopulse.ro

[2]www.upm.ro

We start from the assumption that the goal (for the proposed set of measurements) is enabling both parties to identify problem areas in the sourcing process and to improve the overall process. "The balanced set of measurements helps prevent dysfunctional behavior by monitoring the group's performance in several complementary aspects of their work that lead to project success."[5].

As the Acquirer and the Supplier have control and best insight on different process areas, the interpretation of measurements with regard to the above mentioned goals is inherently different. Therefore agreement over the Sourcing Process Model (SPM), roles, responsibilities, measurements and their interpretation to secure success, is required.

This can be done by developing a Performance Analysis Model (PAM) agreed by both parties.

The fundamental idea of this paper is that the agreed set of goal driven measurements interpreted through a commonly agreed performance analysis model enables:

a. Identification of problem areas and quick implementation of appropriate action
b. Process quality improvement
c. Better planning estimates based on historical data
d. Common view (for both Acquirer and Supplier) of the process

2. Sourcing Process Model

The Sourcing Process Model (SPM) is depicted in Figure1 [1]

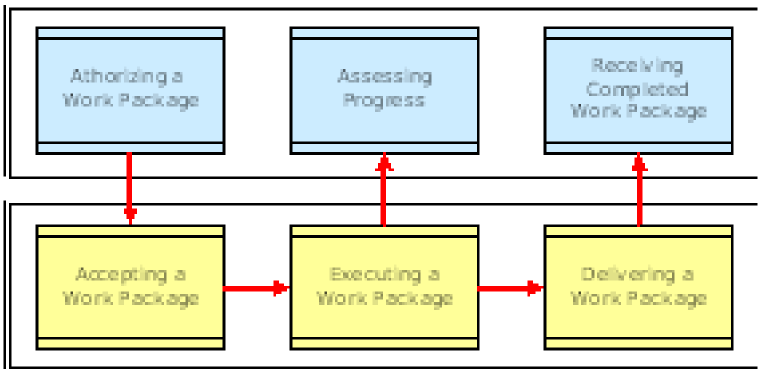

Athorizing a Work Package
Assessing Progress
Receiving Completed Work Package
Accepting a Work Package
Executing a Work Package
Delivering a Work Package

Figure 1. Sourcing Process Model (SPM)

Different terms are used interchangeably in available literature to describe the roles and the responsibilities of the parties involved in the sourcing process:

- Acquirer/Partner to describe the party authorizing a work package (acquisition) and
- Supplier/Vendor/Developer/Contractor for the party accepting the work package (contracting)

We start by identifying process areas and which party exercises control over it. The overall process consists of six major areas:

- Authorizing a work package (Acquirer)
  - Activities: requirements management, defining the work package and stating acceptance criteria for executed work
- Accepting a work package (Supplier)
  - Activities: clarifying work package definition, acceptance criteria, assessing risk
- Executing a work package (Supplier)
  - Activities: work package planning and execution, reporting progress (status). Activities are strongly influenced by directions and corrections, change requests, from Supplier and communication quality.
- Assessing progress (Acquirer)
  - Activities: monitoring progress, directions and corrections
- Delivering executed work package (Supplier)
  - Activities: check executed work package against acceptance criteria (usually changed according to post-award change requests due to poor requirements management) and timely delivery
- Accepting executed work package (Acquirer)
  - Activities: receiving executed work package and checking requirements satisfaction

Acquirer has full control over:

- Work package (WP) definition
- Work package (WPS) stability
- Requirements management (area that usually is poorly measured/managed problems being transferred to the Supplier)
- Project monitoring (status) and oversight (directions and corrections)
- Requirements satisfaction (by checking deliverables against acceptance criteria)

Supplier has full control over:

- Internal development processes
- Deliverables quality (e.g. executed work package, intermediate deliverables)

Supplier & Acquirer share control over:

- Communication effectiveness
- Relationship management
- Contract execution
- Communication channels
- Risk management

3. Goal Driven Measurements and Performance Analysis Model

Measurements goal is to insure successful software development by early identification of problem areas and implementation of corrective actions. Performance Analysis model (PAM, Figure 2) suggests both a set of relevant measurements [1] and a consistent way to interpret them.

3.1. **Authorizing and Accepting a WP measurements.**

The sourcing process is initiated by the Acquirer which authorizes a WP and proposes it to the Supplier. A WP could be described in various ways, e.g. WP size, based on Functional Points (FP) and WP Accuracy (WPA), a quantitative estimation of how well the assignment is described through analysis documents, design documents, acceptance criteria.

Communication (e.g. questions and answers, QA) plays a very important role in clarifying WP accuracy related issues. QA could be quantified simply by their number. But regardless of their number it is important that questions are answered in a reasonable time (quantified by average response time, ART) and answers are specific and complete (quantified by communication quality, CQ).

Based on agreed estimating models (e.g. historical data) the parties can quantify measurements either quantitatively (numerical values) or qualitatively (e.g. color code levels). For instance, using the largely accepted color code, communication quality could be quantified as good ("Green"), average ("Amber") or poor ("Red"). Similarly

WPA could be also described using the same color code or quantitatively by:

$$\mathrm{WPA} = 1 - [\mathrm{QA}/\mathrm{FP}]$$

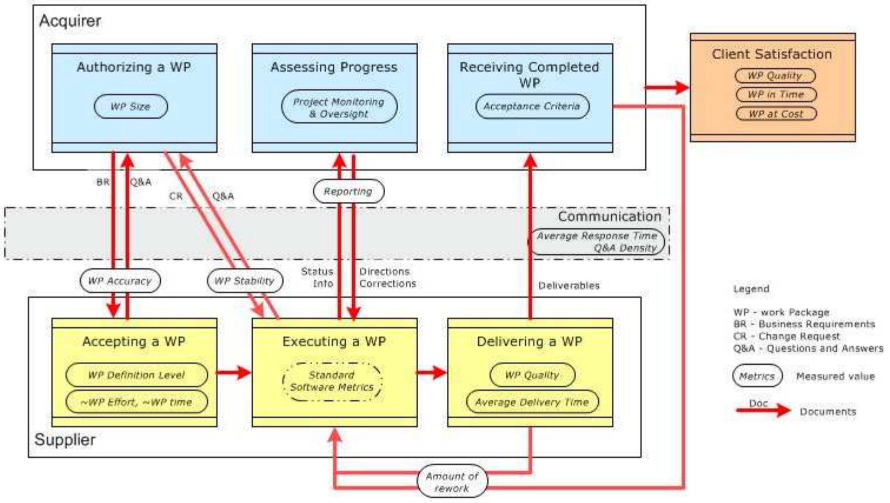


FIGURE 2. Performance Analysis Model (PAM)

3.2. **Executing a WP and Assessing Progress Measurements.** This phase is initiated by the Supplier by accepting a WP and acknowledging the acceptance criteria for the executed WP. Let us call the specific WP accuracy at the moment of awarding as WP definition level.

Although the development process starts by executing the WP at the definition level, frequently the initial request is changed via formal change requests (CR). If accepted by the Supplier, CR also modifies acceptance criteria. Consider a first approximation where the WP stability is also a measurement of acceptance criteria stability.

WP Stability (WPS) is a quantitative estimation of change requests (CR) per functional point (FP) after the WP has awarded by Acquirer and accepted by Supplier:

$$\mathrm{WPS} = 1 - [\mathrm{CR}/\mathrm{FP}]$$

Based on statistics, WPS associated risk could be color coded as depicted in Figure 3.

Ideal stability means no change requests are made after the work package has been accepted by the Supplier (WPS = 1).

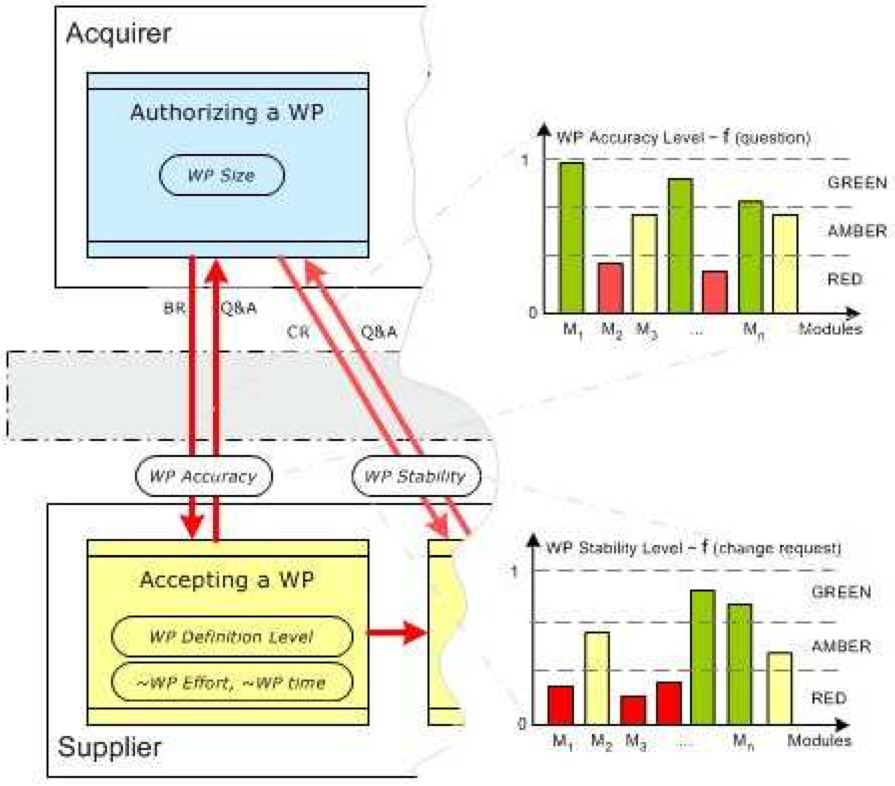


FIGURE 3. WP Stability influenced by change requests

3.3. **Delivering and Receiving a WP Measurements.** As in the previous stages, the focus is on co-operation issues (i.e. between the Supplier and the Acquirer) therefore measurements will not focus on WP quality, amount of rework, etc.

Average delivery time, defined as:

$$\text{ADT} = \text{delivery time}/\text{FP}$$

Could be used to assess delays introduced by different interference factors like change requests, communication quality, corrections and directions.

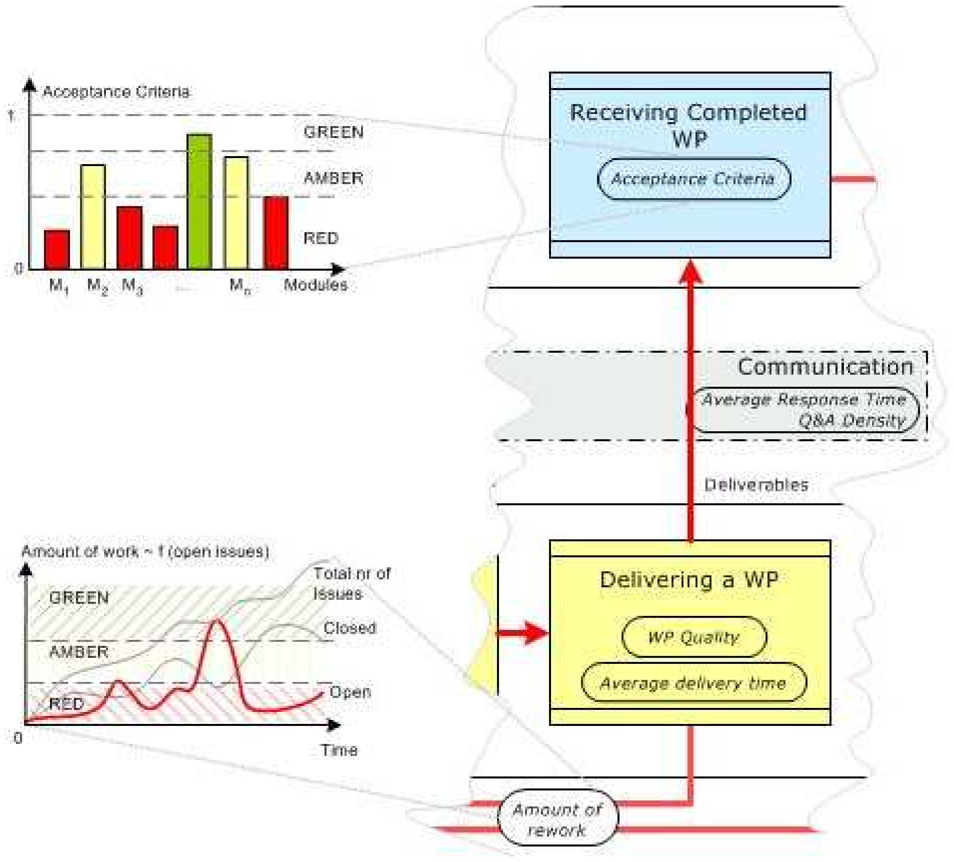


FIGURE 4. Delivering and receiving an executed WP

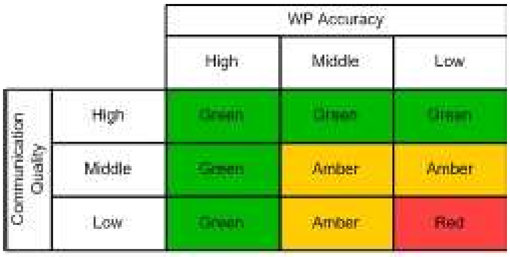
4. INTERPRETING MEASUREMENTS

Table 1 contains measurements to be used with our performance analysis model (Measurements have no meaning apart from their context).

Let us use them to quickly asses risks associated with accepting a work package. We will use qualitative measurements for WP accuracy and communication.

Table 2 reveals that a poorly described WP and a poor communication relationship certainly leads to failure (highest risk level, Red). Any combination of Middle-Low WPA and communication quality requires immediate corrective actions (medium risk level, Amber). A high level of WPA could lead to success

Measurements	Acronym	Value	Tracked Process/Project Issues
WP size	WP	FP	Product complexity, growth
WP accuracy	WPA	WPA = 1 − [QA/FP]	Requirements gathering quality, client expectations management, design quality
WP stability	WPS	WPS = 1 − [CR/FP]	Sourcing process efficiency, productivity, rework, schedule delays
Communication quality	CQ	Green, Amber, Red	Efficiency, productivity
Questions and answers	QA	number	Analysis and design quality
Average response time	ART	$ART = \frac{\sum \text{response time}}{QA}$	Efficiency, productivity
WP definition level	WPD	WP at the moment of starting execution	Product size, complexity, required effort, schedule, acceptance criteria
Directions and corrections	DC	Green, Amber, Red	Responsiveness and effectiveness
Average Delivery Time	ADT	ADT = delivery time/FP	Productivity



		WP Accuracy		
		High	Middle	Low
Communication Quality	High	Green	Green	Green
	Middle	Green	Amber	Amber
	Low	Green	Amber	Red

Table 2. Risk level for accepting a WP

(assuming a high stability of the requirements) almost regardless of the communication quality between parties (low risk level, Green).

		WP Accuracy and Stability		
		High	Middle	Low
Communication Quality&Directions	High	Green	Green	Green
	Middle	Green	Amber	Amber
	Low	Green	Amber	Red

Table 3. Executing a WP Risk Level

Again, as the purpose of this PAM is to identify problem areas in a sourcing process, the focus is not on the software development process but on the co-operation between parties.

Executing a WP is influenced by WP accuracy and requirements stability. Requirements stability is an area usually not very well managed by Acquirer which means customer management issues are transferred to Supplier. Risks could be lowered though by a high quality communication and effective directions and corrections (Table 3).

Risk level is high (Red) when we have to deal with a poor described WP when execution is interfered by a relatively high number of change requests, poorly supported by directions via a poor quality communication. Yet, this table also reveals that very good communication and quality directions and corrections support even a poor WP accuracy interfered by CR.

WP delivery risk level is similarly evaluated in Table 4.

		WP Execution quality		
		High	Middle	Low
Acceptance criteria stability	High	Green	Green	Green
	Middle	Green	Amber	Amber
	Low	Green	Amber	Red

Table 4. Delivery and acceptance risk level

5. Conclusions and further work

PMA was imagined to support sourcing process analysis for small teams and small to medium size projects. Although PMA is independent of the supplier's team, increased team size brings more complexity and therefore requires more attention.

Quantitative measurements bring more insight into the overall process. For instance, based on historical data one can infer that a certain CR average score indicates that up to a certain percentage of the requirements were not known when the WP was awarded.

We must also observe that PMA does not take into consideration factors outside control of Acquirer and Supplier (e.g. end user).

References

[1] Radoiu D., Vajda A., "Process-Oriented Metrics for Application Development Outsourcing", internal report

[2] Jeannine Siviy, Goethert W., Ferguson R.,Trading Places: Measurement and Analysis in the Eyes of the Acquirer and the Supplier, SEPG 2004, March 2004

[3] McGarry J., Card D., Jones C., Layman B., Elizabeth Clark, Dean J., Hall F., Practical Software and Systems Measurement A Foundation for Objective Project Management, Guidebook, Addison Wesley Professional, October 2001, http://www.psmsc.com/PSMBook.asp

[4] Florac, Park, and Carleton, Practical Software Measurement: Measuring for Process Management and Improvement, CMU/SEI-97-HB-003, 1997

[5] Paulk, Weber, Garcia, Chrissis, Bush, Key Practices of the Capability Maturity Model, Version 1.1., CMU/SEI-93-TR-25, ESC-TR-93-178, 1993

Department of Mathematics and Computer Science, Petru Maior University, Târgu Mureş, Romania

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

A NOTE ON THE COMPLEXITY OF THE GENERALIZED MINIMUM SPANNING TREE PROBLEM

PETRICĂ CLAUDIU POP AND CORINA POP SITAR

Abstract. We consider the Generalized Minimum Spanning Tree Problem denoted by GMST. It is known that the GMST problem is $\mathcal{NP}$-*hard*. We present a stronger result regarding the complexity of the problem, namely, the GMST problem even on trees is $\mathcal{NP}$-*hard*. As well as we present three cases when the GMST problem is solvable in polynomial time.
Keywords: combinatorial optimization, complexity theory, $\mathcal{NP}$-*hard*, generalized minimum spanning tree problem, dynamic programming.

1. Introduction

We are concerned with the generalized version of the minimum spanning tree problem (MST) called the generalized minimum spanning tree problem (GMST). Given an undirected graph whose nodes are partitioned into a number of subsets (clusters), the GMST problem is then to find a minimum-cost tree which includes *exactly* one node from each cluster. Therefore, the MST is a special case of the GMST problem where each cluster consists of exactly one node.

The model fits various problems of determining the location of regional service centers (e.g. public facilities, branches, distribution centers) which should be connected by building links (e.g. highways, communication links). For example, when a company tries to establish marketing centers, one for each market segment, and construct a communication network which interconnects the established centers, the company faces a GMST problem. For another example, when designing metropolitan area networks [7] and regional area networks [15], we are to interconnect a number of local area networks. For this model, we must select a node in each local network as a hub (or a gateway) and connect the hub nodes via transmission links such as optical fibers. Then, such a network design problem reduces to a GMST problem.

The GMST problem has been introduced by Myung, Lee and Tcha in [10]. Since then it appeared several papers studying different aspects of the GMST problems: Feremans, Labbé and Laporte in [5] and Pop in [11] present several integer

Received by the editors: October 5, 2004.
2000 *Mathematics Subject Classification.* 90C05,90C25, 90C34.

formulations of the GMST problem, in [5] Feremans *et al.* study the polytope associated with the GMST problem, in [12] Pop presents approximation results of the problem, etc. The GMST problem was solved to optimality for graphs with up to 200 nodes by Feremans [4] using a branch-and-cut algorithm and by Pop [13] for graphs with up to 240 nodes using a so-called rooting procedure.

A variant of the GMST problem is the problem of finding a minimum cost tree including *at least* one vertex from each cluster. This problem was introduced by Dror *et al.* in [3]. These authors provide also five heuristics including a genetic algorithm [8]. In the present paper we confine ourselves to the problem of choosing exactly one vertex per cluster.

2. Definition of the problem

We begin this section with the definition of the well-known minimum spanning tree problem.

Let $G = (V, E)$ be a connected graph. A spanning tree is a graph where all the nodes in the graph are connected in some way with the requirement that there are no cycles in the graph. If each edge has a weight or cost connected to it, denoting how much you have to pay in order to use the edge, the total sum of the costs of the edges can vary from one edge to another, in the same graph. The one of these possibilities which has the lowest sum is called the minimum spanning tree.

The minimum spanning tree problem can be solved by a polynomial time algorithm, for instance the algorithms of Kruskal [9] or Prim [14]. However as we will show the GMST problem is $\mathcal{NP}$-*hard* [1].

The GMST problem is defined on an undirected graph $G = (V, E)$ with nodes partitioned into m clusters. Let $|V| = n$ and $K = \{1, 2, \ldots, m\}$ be the index set of the node sets (clusters). Then, $V = V_1 \cup V_2 \cup \ldots \cup V_m$ and $V_l \cap V_k = \emptyset$ for all $l, k \in K$ such that $l \neq k$. We assume that edges are defined only between nodes belonging to different clusters and each edge $e = \{i, j\} \in E$ has a nonnegative cost denoted by c_{ij} or by $c(i, j)$.

The GMST problem is the problem of finding a minimum-cost tree spanning a subset of nodes which includes exactly one node from each cluster. We will call a tree containing one node from each cluster a generalized spanning tree.

3. Complexity of the GMST problem

Garey and Johnson [6] have shown that for certain combinatorial optimization problems, the simple structure of trees can offer algorithmic advantages for efficiently solving them. Indeed, a number of problems that are $\mathcal{NP}$-*complete*, when are formulated on a general graph, become polynomially solvable when the graph is a tree. Unfortunately, this is not the case for the GMST problem. We will show that on trees the GMST problem is $\mathcal{NP}$-*hard*.

Let us consider the case when the GMST problem is defined on trees, i.e. the graph $G = (V, E)$ is a tree.

To show that the GMST problem on trees is $\mathcal{NP}$*-hard* we introduce the so-called *set cover problem* which is known to be $\mathcal{NP}$*-complete* (see [6]).

Given a finite set $X = \{x_1, ..., x_a\}$, a collection of subsets, $S_1, ..., S_b \subseteq X$ and an integer $k < |X|$, the *set cover problem* is to determine whether there exists a subset $Y \subseteq X$ such that $|Y| \leq k$ and

$$S_c \cap Y \neq \emptyset, \ \forall \, c \text{ with } 1 \leq c \leq b.$$

We call such a set Y a *set cover* for X.

Theorem 1. *The Generalized Minimum Spanning Tree problem on trees is* $\mathcal{NP}$*-hard.*

Proof: In order to prove that the GMST problem on trees is $\mathcal{NP}$*-hard* it is enough to show that there exists an $\mathcal{NP}$*-complete* problem that can be polynomially reduced to GMST problem.

We consider the set cover problem for a given finite set $X = \{x_1, ..., x_a\}$, a collection of subsets of X, $S_1, ..., S_b \subseteq X$ and an integer $k < |X|$.

We show that we can construct a graph $G = (V, E)$ having a tree structure such that there exists a set cover $Y \subseteq X$, $|Y| \leq k$ if and only if there exists a generalized spanning tree in G of cost at most k.

The constructed graph G contains the following $m = a + b + 1$ clusters $V_1, ..., V_m$:

- V_1 consists of a single node denoted by r
- $V_2, ..., V_{a+1}$ node sets (corresponding to $x_1, x_2, ..., x_a \in X$) each of which has two nodes: one 'expensive' (see the construction of the edges) say $\overline{x_i}$ and one 'non-expensive' say $\hat{x_i}$, for $i = 2, ..., a$, and
- b node sets, $V_{a+2}, ..., V_m$ with $V_\nu = S_{\nu-(a+1)}$, for $\nu = a + 2, ..., m$.

Edges in G are constructed as follows:

(i) Each 'expensive node', say $\overline{x_t}$ of V_t for all $t = 2, ..., a + 1$, is connected with r by an edge of cost 1 and each 'non-expensive' node, say $\hat{x_t}$ of V_t for all $t = 2, ..., a + 1$, is connected with r by an edge of cost 0.

(ii) Choose any node $j \in V_t$ for any $t \in \{a + 2, ..., m\}$. Since $V_t \subset X$, then j coincides with a node in X, say $j = x_l$. We construct an edge between j and (the expensive node) $\overline{x_l} \in V_l$ with $l \in \{2, ..., a\}$. The cost of the edges constructed in this way is 0.

By construction the graph $G = (V, E)$ has a tree structure.

Suppose now that there exists a generalized spanning tree in G of cost at most k then by choosing

$$Y := \{x_l \in X \mid \text{the expensive vertex } \overline{x_l} \in V_{l+1} \text{ corresponding to } x_l \text{ is a vertex of the generalized spanning tree in } G\}$$

we see that Y is a set cover of X.

On the other hand, if there exists a set cover $Y \subseteq X$, $|Y| \leq k$ then according to the construction of G there exists a generalized spanning tree in G of cost at most k. ■

The following theorem due originally to Myung *et al.* [10] is an easy consequence of Theorem 1.

Theorem 2. *The Generalized Minimum Spanning Tree problem is $\mathcal{NP}$-hard.*

Remark 3. To show that the GMST problem is *$\mathcal{NP}$-hard*, Myung, Lee and Tcha [10] used the so-called *node cover problem* which is known that is *$\mathcal{NP}$-complete* (see [6]) and showed that it can be polynomially reduced to GMST problem. Recall that given a graph $G = (V, E)$ and an integer $k < |V|$, the *node cover problem* is to determine whether a graph has a set C of at most k nodes such that all the edges of G are adjacent to at least one node of C. We call such a set C a *node cover* of G.

4. Polynomially solvable cases of the GMST problem

As we have seen the GMST problem is *$\mathcal{NP}$-hard.* In this section we present some cases when the GMST problem can be solved in polynomial time.

A special case in which the GMST problem can be solved in polynomial time is the following:

Remark 4. If $|V_k| = 1$, for all $k = 1, ..., m$ then the GMST problem trivially reduces to the classical Minimum Spanning Tree problem which can be solved in polynomial time, by using for instance the algorithm of Kruskal or the algorithm of Prim.

Another case in which the GMST problem can be solved in polynomial time is given in the following proposition:

Proposition 5. *If the number of clusters m is fixed then the GMST problem can be solved in polynomial time (in the number of nodes n).*

Proof: We present a polynomial time procedure based on dynamic programming which solves the GMST problem in this case.

Let G' be the graph obtained from G after replacing all nodes of a cluster V_i with a supernode representing V_i, that we will call the *global graph.* For convenience, we identify V_i with the supernode representing it. We assume that G' with vertex set $\{V_1, ..., V_m\}$ is complete.

Given a global spanning tree of G', which we shall refer to as the *global spanning tree*, we use dynamic programming in order to find the best (w.r.t. cost minimization) generalized spanning tree.

Fix an arbitrary cluster V_{root} as the root of the global spanning tree and orient all the edges away from vertices of V_{root} according to the global spanning tree. A directed edge $\langle V_k, V_l \rangle$ resulting from the orientation of edges of the global spanning tree defines naturally an orientation $\langle i, j \rangle$ of an edge $(i, j) \in E$ where $i \in V_k$ and $j \in V_l$. Let v be a vertex of cluster V_k for some $1 \leq k \leq m$. All such nodes v are potential candidates to be incident to an edge of the global spanning tree.

The "subtree" rooted at a vertex v, $v \in V_k$ with $k \leq m$, denoted by $T(v)$ includes all the vertices reachable from v under the above orientation of the edges of G, based on the orientation of the edges of the global spanning tree of G'. The *children* of v denoted by $C(v)$ are all those vertices u with a directed edge (v, u). Leaves of the tree have no children.

Let $W(T(v))$ denote the minimum weight of a generalized "subtree" rooted at v. We want to compute:

$$\min_{r \in V_{root}} W(T(r)).$$

We give now the dynamic programming recursion to solve the subproblem $W(T(v))$. The initialization is:

$$W(T(v)) = 0 \text{ if } v \in V_k \text{ and } V_k \text{ is a leaf of the global spanning tree.}$$

The recursion for $v \in V$ an interior vertex is then as follows:

$$W(T(v)) = \sum_{l, C(v) \cap V_l \neq \emptyset} \min_{u \in V_l} \{c(v, u) + W(T(u))\},$$

where by $c(v, u)$ we denoted the cost of the edge (v, u).

For computing $W(T(v))$, i.e. find the optimal solution of the subproblem $W(T(v))$, we need to look at all the vertices from the clusters V_l such that $C(v) \cap V_l \neq \emptyset$. Therefore for fixed v we have to check at most n vertices. So the overall complexity of this dynamic programming algorithm is $O(n^2)$, where $n = |V|$.

Notice that the above procedure leads to an $O(m^{m-2} n^2)$ time exact algorithm for GMST problem, obtained by trying all the global spanning trees, i.e. the possible trees spanning the clusters, where m^{m-2} represents the number of distinct spanning trees of a completely connected undirected graph of m vertices given by Cayley's formula [2]. Therefore when the number of clusters m is fixed the above procedure leads to a polynomial time algorithm for solving the GMST problem. ■

The last case when the GMST problem can be solved in polynomial time is given in the following proposition:

Proposition 6. *Consider the GMST problem on trees. If the number of leaves is bounded then the problem can be solved in polynomial time.*

Proof: Because the number of leaves of the graph $G = (V, E)$ (having a tree structure) is bounded, the number of possible generalized spanning trees, i.e. trees containing exactly one node from each cluster is finite. Therefore the GMST problem in this case is solvable in polynomial time. ■

References

[1] G. Ausiello, Crescenzi, P., Gambosi, G., Kann,V., Marchetti-Spaccamela, A., Protasi, M., *Complexity and Approximation Combinatorial optimization problems and their approximability properties*, Springer Verlag, 1999.

[2] A. Cayley , *On the mathematical theory of isomers*, Philosophical Magazine, 67, 1874, 444.

[3] M. Dror, Haouari, M., Chaouachi, J., *Generalized Spanning Trees*, European Journal of Operational Research, 120, 2000, 583-592.

[4] C. Feremans, *Generalized Spanning Trees and extensions*, Ph.D. thesis, Universite Libré de Bruxelles, Belgium, 2002.

[5] C. Feremans, Labbé, M., Laporte, G., A *Comparative Analysis of Several Formulations for the Generalized Minimum Spanning Tree Problem*, Networks 39(1), 29-34, 2002.

[6] M.R. Garey, M.R., Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, California, 1979.

[7] M. Gerla, Frata, L., *Tree structured fiber optics MAN's*, IEEE J.Select. Areas Comm. SAC-6, 1988, 934-943.

[8] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[9] J.H. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical Society 7, 48-50, 1956.

[10] Y.S. Myung, Lee, C.H., Tcha, D.w., *On the Generalized Minimum Spanning Tree Problem. Networks*, Vol 26, 1995 231-241.

[11] P.C. Pop, *The Generalized Minimum Spanning Tree Problem*, Ph.D. thesis, University of Twente, The Netherlands, 2002.

[12] P.C. Pop, Kern, W., Still, G., *An Approximation Algorithm for the Generalized Minimum Spanning Tree Problem with bounded cluster size*, EIDMA 2001 Symposium, Oostende, Belgium, 25-26 October, 2001.

[13] P.C. Pop, Kern, W., Still, G., *A New Relaxation method for the Generalized Minimum Spanning Tree Problem*, to appear in European Journal of Operational Research.

[14] R.C. Prim, *Shortest connection networks and some generalizations*, Bell Systems Technical Journal, 36, 1389-1401, 1957.

[15] J.J. Prisco, *Fiber optic regional area networks in New York and Dallas*, IEEE J. Select. Areas Comm. SAC-4, 1986, 750-757.

Department of Mathematics and Computer Science, Faculty of Sciences, North University of Baia Mare, Romania

E-mail address: pop_petrica@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

INFEASIBLE PRIMAL-DUAL ALGORITHM FOR MINIMIZING CONVEX QUADRATIC PROBLEMS

H. ROUMILI, A. KERAGHEL, AND A. YASSINE

Abstract. Problems with a convex quadratic objective function and linear constraints are important in their own right, and they also arise as sub problems in Methods for general constrained optimization, such as sequential quadratic Programming and augmented Lagrangian methods.

In this paper, we propose and implement an infeasible primal-dual algorithm in order to minimize a convex quadratic function subject to bounded and linear equality constraints.

Preliminary experimentations are particularly encouraging.

Key words : feasible interior points methods, convex quadratic Programming, infeasible interior points methods.

1. Introduction

Interior point's methods are recognized to be efficient for solving many optimization problems. However, finding a strictly feasible initial point (phase1) is difficult.

In theory, we can overcome this difficulty by introducing some artificial variables and by transforming the problem in a new one into a space of superior dimension. This transformation requires using parameters in general (unknown) to big values as in the approach of "Big M" in linear programming. Inconveniences of this approach are known:

(1) One does not know if the size of these large parameters values risk to destabilize the algorithm.
(2) The reformulation can impair the structure of the original problem because we add lines and columns. Because of these failings, this approach is not very welcomed, or is carefully used. Considerable research efforts are dedicated to initializing interior point's methods.

Received by the editors: October 15, 2004.

2000 *Mathematics Subject Classification.* 65K05, 90C20, 90C51.

Several approaches are proposed in which the variant phase1 and phase2 and where no large parameter is used. The artificial variables are introduced through lines and supplementary columns. The relative process to this approach spread between 1986 and 1991. All these works main objective consisted in elaborating algorithms that do not necessarily start inside the feasible domain (of the original problem) along with theoretical properties, as the polynomial complexity. Researchers did not globally aim at the numeric aspect. So, in these methods, the initial point is not necessary but transforming the problem is unavoidable.

Efforts of research are oriented towards numeric performances. About this, a set of practical variants is proposed with the comparative numeric tests from 1989 to 1992. All these algorithms do not require feasible initial point transformation but start from any positive point and tempt to achieve feasibility and optimality in a simultaneous manner.

These algorithms are called infeasible interior point's methods; the most part of these algorithms is that of Newton type. This last class is the object of our study. We concentrated our efforts on methods favoured by a rich theory and also by a lot of numeric subtleties. Indeed, the latter starts phase 2 directly.

Regarding this, several researchers like Zhang, and all authors of principalsŠ relative development of these methods regarding the linear programming, think that these algorithms are more effective. These subjects seem to be very logical. On one hand, phase 1 is eliminated and on the other hand, phase 2 iteration does not defer too much from the feasible case. The preliminary study that we did stimulates of the numeric behaviour of the convex quadratic programming development.

2. General presentation of the convex quadratic program

A convex quadratic program with constraints means optimization problem, in which the objective is a convex quadratic form. The constraints are linear.

Without loss of generalities, we can write it as follwos:

$$(QP)\begin{cases} min\ c^t x + \frac{1}{2}x^t Q x \\ Ax = b \\ x \succeq 0 \end{cases}$$

where Q is a $n \times n$ matrix assumed to be positive semidefinite, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and A is a $m \times n$ matrix of full rank.

The dual of (QP) is:

$$(QD)\begin{cases} max\ b^t y - \frac{1}{2}x^t Q x \\ A^t y + z - Qx = c \\ z \geq 0, y \in \mathbb{R}^m \end{cases}$$

where $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$.

We impose the following assumptions:

- $(\mathbf{H1})$: $S_{int} = \{x \in \mathbb{R}^n \ / \ Ax = b, x \rangle 0\} \neq \Phi$ interior feasible solutions of (P) is non-empty;
- $(\mathbf{H2})$:$T_{int} = \{(y, z) \in \mathbb{R}^m \times \mathbb{R}^n \ / \ A^t y + z - Qx = c, z \rangle 0\} \neq \Phi$ interior feasible solutions of (D) is non-empty.

These assumptions are often used to develop the interior pointŠs algorithms.

2.1. **Principle.** Most of the new interior pointŠs methods are motivated by the the logarithmic barrier function technique of Frisch (1955) to problem (QP) application.

Indeed, to the problem (QP), one associates the problem gate non linear next one:

$$(QP_\mu) \begin{cases} min\ c^t x + \frac{1}{2} x^t Q x - \mu \sum\limits_{i=1}^{n} ln\ x_i = f_\mu(x) \\ Ax = b \\ x \rangle 0 \end{cases}$$

ă The principle of these methods is to solve the system of Karuch-Kuhn-Tucker (KKT) partner to the problem (QP_μ) by the method of damped Newton, while leaving from any positive point which is not necessarily feasible. The resolution of (QP_μ) is equivalent at that of (QP) with that if $x^*(\mu)$ is an optimal solution of (QP_μ) then $x^* = \lim\limits_{\mu \to 0} x^*(\mu)$ is an optimal solution of (QP). To achieve feasibility and optimality we introduce a merit function defined by:

$$\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x}^t \mathbf{z} + \mathbf{r}(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

where $r(x, y, z) = \|Ax - b\| + \|-Qx + A^t y + z - c\|$.

It is clear that r measure feasibility and $x^t z$ (duality gap) control the optimality. The idea is to make the value of this function towards zero during iterations.

2.2. **Resolution of** (P_μ)**.** x is an optimal solution of (P_μ) if an only if there is $y \in \mathbb{R}^m$ such that:

$$(\mathbf{1}) \begin{cases} c - \mu X^{-1} e - A^t y + Qx = 0 \\ Ax = b \\ x > 0 \end{cases}$$

where: $X^{-1} = diag(1/x_i)$. We apply the method of damped Newton to solve the system of nonlinear equations $(\mathbf{1})$ from an infeasible starting point (which is not necessarily feasible) $(x, y, z) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$, $(x, z) \rangle 0$ and $\mu = x^t z / n \rangle 0$, we gets the following system:

$$\begin{cases} X\Delta z + Z\Delta x = -XZe + \sigma\mu e \qquad 0 < \sigma < 1, \qquad Z = diag(z_i) \\ A\Delta x \ = b - Ax \\ -Q\Delta x + A^t\Delta y + \Delta z = c - A^t y + Qx - z \end{cases}$$

where the solution is: $(\Delta x, \Delta y, \Delta z)$, the new iterate is then: $(\hat{x}, \hat{y}, \hat{z}) = (x, y, z) + \alpha(\Delta x, \Delta y, \Delta z)$

With $\alpha \rangle 0$ is the displacement step chosen such a way that $(\hat{x}, \hat{z}) \rangle 0$ and ϕ^k decreases. If the test of stop is not satisfied one replaces μ by $\mu_1 (\mu_1 \prec \mu)$ and reiterate.

Our infeasible interior point algorithm is described as follows:

```
BASIC ALGORITHM
Beginning :
  Initialisation:
    Start with $(x,\ z) \rangle 0$, $y \in \mathbb{R}^m$ (arbitrary) and calculate $\phi$. Either $\varepsilon \ \rangle 0$ a parameter of precision.
K = 0
    when $\phi \rangle \varepsilon$ do:
   step 0:
    Calculate $\mu = (1/n)\,(x)^t\, z$ and choose $\sigma \in (0,1)$
   step 1:
    Solve the following linear system:
      $$\begin{bmatrix} Z & 0 & X \\ A & 0 & 0 \\ -Q & A^t & I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} \mu\sigma e - XZe \\ b - Ax \\ c - A^t y + Qx - z \end{bmatrix}$$
   step 2:
    finds a step of displacement $\alpha \rangle 0$ such as:
    $x = x + \alpha\Delta x \rangle 0$, $z = z + \alpha\Delta z \rangle 0$ and $\phi$ decreasing.
   step 3:
    $y = y + \alpha\Delta y$
K = k + 1
End.
```

2.3. **Convergence of the Algorithm.** The convergence of the algorithm is studied in [19, 20, 21] for linear and complementarity programming, we extend these results for quadratic convex programming. Under hypotheses (**H1**) and (**H2**), the convergence of the algorithm is based on the following lemma:

Lemma 1. *Let $\{(x^k, y^k, z^k)\}$ be the sequence of iterates generated by the algorithm then we have:*

1) $A(x^k + \alpha^k \Delta x^k) - b = (1 - \alpha^k)(Ax^k - b) = v^{k+1}(Ax^0 - b)$

2) $A^t(y^k+\alpha^k\Delta y^k)-Q(x^k+\alpha^k\Delta x^k)+(z^k+\alpha^k\Delta z)-c = v^{k+1}(A^ty0+z0-Qx0-c)$

3) $(\mathbf{x}^k + \alpha^k\Delta x^k)^t(z^k + \alpha^k\Delta z^k) = (x^k)^tz^k(1-\alpha^k+\alpha^k\sigma^k)+(\alpha^k)^2(\Delta x^k)^t\Delta z^k$, where $v^{k+1} = (1-\alpha^k)v^k = (1-\alpha^j) \succeq 0, v^0 = 1$.

Proposition 2. *The sequence* $\left\{\phi^k\right\}$ *generated by the algorithm satisfies:*

$$\phi^{k+1} = \left(1-\delta^k\right)\phi^k$$

Where $\delta^k = \delta\left(\alpha^k\right) = \frac{\left[\alpha^k\left(1-\sigma^k\right)\left(x^k\right)^t z^k+\alpha^k v^k r0-\left(\alpha^k\right)^2\left(\Delta x^k\right)^t\Delta z^k\right]}{\left[\left(x^k\right)^t z^k+v^k r0\right]}$.

Corollary 3. *It is easy to prove that the sequence* $\left\{\phi^k\right\}$ *converge linearly if* $0 \rangle \alpha^k \leq 1$ *to which case we have* $0 \rangle \delta^k \rangle 1$ *and if* δ^k *offers toward* 1 *the convergence becomes super linear.*

Proposition 4. *Let us suppose the initial point is given by* $\left(x^0,y^0,z^0\right) = \zeta\left(e,0,e\right)$ *(* $\zeta \rangle 0$*) then: the algorithm converges on at most* $O\left(n^2\left|log\left(\varepsilon\right)\right|\right)$ *iterations (* ε *a parameter of precision).*

2.4. **Determination of the displacement step.** The displacement step choice is based on the decreased monotonous of the merit function and on the strict positivity of (x,z). To get the global convergence, two supplementary hypotheses are necessary to know:

C1) $h\left(\alpha\right) = \left[min(X\left(\alpha\right)z\left(\alpha\right)) - \gamma\left(x\left(\alpha\right)\right)^t z\left(\alpha\right)/n\right] \succeq 0 \quad \alpha \in (0,1]$

ă **C2**) $g\left(\alpha\right) = \left[\left(x\left(\alpha\right)\right)^t z\left(\alpha\right) - v\left(\alpha\right)\left(x^0\right)^t z^0\right] \succeq 0 \quad \alpha \in (0,1]$

ă where $0 \rangle \gamma \rangle 1$ satisfied $\gamma \leq \min\left(X^0z^0\right)/(\left(x^0\right)^t z^0/n)$ et $X\left(\alpha\right) = \text{diag}\left(x^{k+1}\right)$, $x\left(\alpha\right) = x^{k+1}$, $z\left(\alpha\right) = z^{k+1}$, $v\left(\alpha\right) = v^{k+1}$ ă The **C1** condition is essential for interior pointŠs methods. Its role is to prevent iterates to approach prematurely the border (before the optimality), while the **C2** condition gives the priority to the feasibility on complementarity (the feasibility is achieved at the latest at the same time than complementarity:
$(x^k)^tz^k/\left(x^0\right)^t z^0 \succeq ((r^k/r^0) = v^k)$.

Let us determine then α^k while taking account the two previous conditions and the maximization of $\delta\left(\alpha\right)$ in $(0,1]$ that is to say:

$$\alpha^k = \text{arg max}\left\{\delta\left(\alpha\right) : h\left(\beta\right) \succeq 0, g\left(\beta\right) \succeq 0 \text{ for any } \beta \leq \alpha\right\} \qquad (1)$$

The solution of (1) is given below by the lemma:

Lemma 5. *If the* **C1** *condition is verified to every iteration,then the problem (1.3) admits a unique solution:*

$$\alpha^k = \begin{cases} min\ \left(1, \alpha_1^k, \alpha_2^k\right) & si\ \left(\Delta x^k\right)^t \Delta z^k \leq 0 \\ min\ \left(1, \alpha_1^k, \alpha_3^k\right) & si\ \left(\Delta x^k\right)^t \Delta z^k \ \rangle\ 0 \end{cases}$$

where:

$$\alpha_1^k = min\ \{\alpha\ \rangle\ 0 : h(\alpha) = 0\}$$

ă $\alpha_2^k = \begin{cases} 1 & si\ \left(\Delta x^k\right)^t \Delta z^k = 0 \\ min\ \{\alpha\ \rangle\ 0 : g(\alpha) = 0\} & si\ \left(\Delta x^k\right)^t \Delta z^k\ \langle\ 0 \end{cases}$ ă *and*

$$\alpha_3^k = \left[(1-\sigma)\left(x^k\right)^t z^k + v^k r^0\right] / 2\ \left(\Delta x^k\right)^t \Delta z^k. \text{ [18]}$$

Remark 1. *Let us note that the previous choice of step constitutes a condition sufficient only for the convergence, which gives us a certain liberty in practice. Indeed, a less expensive choice is possible, it is about choosing α so that $(x, z)\ \rangle\ 0$ (strictly positive) while taking account of the decrease of the merit function.*

ă *In the implementation our suitable choice of the largest step size is given by :* ă $\alpha_x = \beta\alpha_x^{'}$ *and* $\alpha_z = \beta\alpha_z^{'}$ $(0\langle\beta\langle 1)$ ă *where* ă

$$\alpha_x^{'} = \begin{cases} min\left(-x_i/\Delta x_i\right) & si\ \Delta x_i\ \langle\ 0 \\ 1 & si\ \Delta x_i \succeq 0 \end{cases} \text{ ă } \alpha_z^{'} =$$

$$\begin{cases} min\left(-z_i/\Delta z_i\right) & si\ \Delta z_i\ \langle\ 0 \\ 1 & si\ \Delta z_i \succeq 0 \end{cases}$$

ă Whose new iterate is: $x = x + \alpha_x \Delta x$, $z = z + \alpha_z \Delta z$ and $y = y + \alpha_z \Delta x$.

All time, it is important to signal that performances of interior points methods feasible or no, depend greatly on the choice of the displacement step.

2.5. **Calculation of the displacement direction.**

In the algorithms, the cost of an iteration is dominated by the calculation of the displacement direction $\Delta w = (\Delta x, \Delta y, \Delta z)$ that is by solving the following of the linear system:

$$\begin{bmatrix} Z & 0 & X \\ A & 0 & 0 \\ -Q & A^t & I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} \mu\sigma e - XZe \\ b - Ax \\ c - A^t y + Qx - z \end{bmatrix} \qquad (2)$$

Several procedures can be used to solve (2) like Gauss elimination. The obtained results are valid but limited with small dimensions. Moreover, the strategy is not optimal. The manipulated matrix is of dimension $(2 \times n + m, 2 \times n + m)$. To calculate the displacement direction, we introduce a more economic alternative which consists in reducing the implemented matrix size. For that, with simple calculations, one obtains the system of equations according to:

$$\begin{cases} A(QX+Z)^{-1}XA^t\Delta y = b - Ax + A(QX+Z)^{-1} \\ \qquad [Xze - \mu\sigma e - X(c - A^t y + Qx - z)] \\ \Delta x = (QX+Z)^{-1}[XA^t\,\Delta y + \mu\sigma e - Xze - X(c - A^t y + Qx - z)] \\ \Delta z = (X)^{-1}(\mu\sigma e - Xze - Z\Delta x) \end{cases}$$

Only matrix $A(QX+Z)^{-1}XA^t$of size $(m \times m)$ will be necessary for to resolve the system in question.

The matrix: $A(QX+Z)^{-1}XA^t$ is positive semi definite and:

i) $A(QX+Z)^{-1}XA^t = \left(A(QX+Z)^{-1}XA^t\right)^t$

ii) $x \neq 0, \langle\ \left(AA(QX+Z)^{-1}XA^t\right)\ \left(A(QX+Z)^{-1}XA^t\right)^t x, x\ \rangle =$
$\langle\ \left(A(QX+Z)^{-1}XA^t\right)x, A(QX+Z)^{-1}XA^t x\ \rangle =$
$\left\|\left(A(QX+Z)^{-1}XA^t\right)x\right\|^2\rangle\ 0$

Therefore, the Cholesky factorization (which is a particular case of the Gauss method) is frequently used in solving the system.

2.6. **Numerical experiments.** This paragraph is dedicated to preliminarily numerical results presentation in order to test our algorithm, implemented on a Pentium II and in TURBO-PASCAL programming.

Examples are stated under the following canonical form:

$$(QP)\begin{cases} min\ c^t x + \frac{1}{2}x^t Qx \\ Ax = b \\ x \succeq 0 \end{cases}$$

The dual of (QP) is:

$$(QD)\begin{cases} max\ b^t y - \frac{1}{2}x^t Qx \\ A^t y + z - Qx = c \\ z \geq 0, y \in \mathbb{R}^m \end{cases}$$

ă **Example 1:** ă

$$\begin{cases} \underset{(x,t)}{min}\ f(x,t) = 6.5x + 0.5x^2 - t_1 - 2t_2 - 3t_3 - 2t_4 - t_5 \\ Az \leq\ b \\ z = (x,t)^t \\ x \succeq 0,\ t_1 \succeq 0, t_2 \succeq 0 \\ 0 \leq\ t_i\ \leq 1 \quad i = 3,4 \\ 0 \leq\ t_5 \leq\ 2 \end{cases}$$

$b = (26, -11, 24, 12, 3)$

$$A = \begin{pmatrix} 1 & 2 & 8 & 1 & 3 & 5 \\ -8 & -4 & -2 & 2 & 4 & -1 \\ 2 & 0.5 & 0.2 & -3 & -1 & -4 \\ 0.2 & 2 & 0.1 & -4 & 2 & 2 \\ -0.1 & -0.5 & 2 & 5 & -5 & 3 \end{pmatrix}$$

ă The optimal solution of (QP) is:

$z^* = (x^*, t^*)^t = (0, 7.987342, 0.253165, 2, 2, 0)^t$

The optimal solution of (QD) is:

$y^* = (-0.246835, 0, 0, -0.253165, 0)^t$

objective function: -18.493671 ă **Example 2:**

$$\begin{cases} \underset{x}{min}\ f(x) = 0.5 \sum_{i=1}^{6} \beta_i (x_i - \alpha_i)^2 \\ Ax \preceq b \\ x \succeq 0 \end{cases}$$

$b = (-5, 2, -1, -3, 5)^t$ ă

$$A = \begin{pmatrix} -3 & 7 & 0 & -5 & 1 & 1 \\ 7 & 0 & -5 & 1 & 1 & 0 \\ 0 & -5 & 1 & 1 & 0 & 2 \\ -5 & 1 & 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & 2 & -1 & -1 \end{pmatrix}$$

ă **case 1**: $\beta_i = 1$, $\alpha_i = 2$ for $i : 1, ..., 6$ ă The optimal solution of primal problem is: ă $x^* = (1.010453, 0.749129, 1.303136, 1.442509, 0, 0)^t$ ă The optimal solution of dual problem is: ă $y^* = (-0.661232, -0.646117, -1.217533, -0.709915, 0)^t$ ă Objective function: 2.680600 ă **case 2**: $\beta_i = 1$, $\alpha_i = 2$ for $i : 1, ..., 6$ ă The optimal solution of primal problem is:

$x^* = (1.715685, 0.965687, 2.397060, 1.431373, 0.544120, 0)^t$

The optimal solution of dual problem is:

$y^* = (-7.862744, -3.183211, -13.518993, -2, -11.590071)^t$

Objective function: -11.467500

case 3: $\beta_i = 1$, $\alpha_i = 2$ for $i : 1, ..., 6$

The optimal solution for primal problem is:

$x^* = (1.715685, 0.965687, 2.397060, 1.431373, 0.544120, 0)^t$

The optimal solution for dual problem is:

$y^* = (-15.725487, -6.366421, -27.037984, -4, -23.180141)^t$

Objective function: -22.935000

case 4: $\beta_i = 2$, $\alpha_i = 0$ for $i : 1, ..., 6$

The optimal solution of primal problem is:

$x^* = (2.05, 1.30, 3.40, 2.10, 2.55, 0)^t$

The optimal solution for dual problem is:
$y^* = (-13, -4.80, -22.40, -2.60, -19.40)^t$
Objective function: -11.400000

Example 3:

$$\begin{cases} \min\limits_{x} f(x) = \sum\limits_{i=1}^{9} x_i x_{i+1} + \sum\limits_{i=1}^{8} x_i x_{i+2} + x_1 x_9 + x_1 x_{10} + x_2 x_{10} + x_1 x_5 + x_4 x_7 \\ \sum\limits_{i=1}^{10} x_i = 1 \\ x \succeq 0 \end{cases}$$

ă The optimal solution of primal problem is:
$x^* = (0, 0.249335, 0.25, 0, 0.000665, 0.017431, 0, 0.25, 0.232568, 0)^t$
The optimal solution of dual problem is:
$y^* = 0.25$
Objective function: 0.125010

3. Conclusion

In this paper, we presented an implementing method for convex quadratic programming which stimulates greatly the development of the numeric behaviour of infeasible methods for problems of optimization. One can conclude that these methods constitute a valid solution as to the algorithm initialization problem. This one deserves some supplementary efforts essentially when choosing the step displacement. This, until now, is the object of numerous researches aiming to reduce the iteration cost and, by the same time, improve the numeric behaviour distinctly. This one deals not only with the linear and convex quadratic programming but is extended to non linear programming.

References

[1] I. ADLER, R.D.C. MONTEIRO: “Interior path following primal-dual algorithms, Part II: Convex quadratic programming”, Mathematical Programming 44 (1989) 43-66.

[2] I. ADLER, R.D.C. MONTEIRO: “Interior path following primal-dual algorithms, Part I: Linear programming”, Mathematical Programming 44 (1989) 27-41.

[3] S. BAZARA, H.D. SHERALI, C.M. SHETTY: “Nonlinear programming, theorie and algorithms”, Second édition (1993).

[4] J.F. BONNANS,J.C. GILBERT, C. LEMERECHAL, C. SAGASTIZABAL “Méthodes numérique d’optimisation(§4)”, INRIA, 78153 Lechesnay France (1995).

[5] A. COULIBALY: “Méthodes de points intérieurs en programmation linéaire”, Thèse de l’université Blaise Pascal (1994).

[6] J.C. CULIOLI: “Introduction à l’optimisation”, Edition Marketing (1994).

[7] A.V. FICCO, G.P. MCCOMICH: “Nonlinear programming sequential unconstrained minimization technique(1968) ”, J.Wiley, New York.

[8] C. GANZAGA: “Path-following methods for linear progmming ”. SIAM Review Vol 34.NO.2 (1992).

[9] Z. KEBBICHE: “Mise en oeuvre d’une méthode de trajectoire centrale pour les problèmes complémentaires linéaires monotones”, Thèse de Magister (1997), U.F.A Sétif-Algérie-.

[10] A. KERAGHEL: “Etude adaptative et comparative des principales variantes dans l’algorithme de Karmarkar”, Thèse de Doctorat Université Joseph Fourier Grenoble (1989).

[11] N. MEGIDDO: “Pathways to the opyimal set in linear programming”, Progress in Mathématical Programming interior-point (1989).

[12] N. MEGIDDO, M. KOJIMA, S. MIZUNO: “A primal-dual infeasible interior point algorithm for linear programming”, Mathématical Programming 61(1993) 263-280.

[13] MINOUX: “Programmation mathématique théorie et algorithmes” T.1, (1984), Dunod, Paris.

[14] S. MIZUNO: “Polynomiality of infeasible interior point algorithms for linear programming”, Mathématical Programming 67(1994) 109-119.

[15] S. MIZUNO, M. KOJIMA, A. YOSHISE: “A primal-dual interior-point algorithm for linear programming”, Report NO.B188, TOKYO (1987).

[16] H. ROUMILI: “Etude qualitative des méthodes de points intérieurs non réalisables pour la programmation linéaire”,Thèse de Magister(1998), U.F.A Sétif-Algérie.

[17] M. TODD: “potential-reduction methods in mathematical programming”, Mathématical Programming serieB-january 1997.

[18] S.J. WRIGHT: “Primal-dual interior-point methods”, Copyright (1997) by SIAM.

[19] Y. ZHANG, D. ZHANG: “Superlinear convergence of infeasibal-interior point methods for linear programming”, Mathématical Programming 66 (1994) 361-377.

[20] Y. ZHANG: “On the convergence of a class of infeasibal-interior point algorithms for the horizontal linear complementarity problem”, SIAM journal on optimisation 4 (1994) 208-227.

Department of Mathematics, University of Setif 19000, Algeria
E-mail address: `r_nouha@yahoo.fr`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

COMODI: GUIDELINES FOR A COMPONENT-BASED FRAMEWORK FOR SCIENTIFIC COMPUTING

ZSOLT I. LÁZÁR, BAZIL PÂRV, ANDREEA FANEA, JOUKE R. HERINGA, AND SIMON W. DE LEEUW

Abstract. The present work discusses the aspects pertaining to the change of scientific software development practices towards the paradigm of component-based programming [1]. It summarizes the symptoms that indicate the necessity of a renewal in computational sciences. The main ingredients for the solution are identified and a vision on how effective code sharing can affect future scientific research is presented. Starting from the premises of today's scientific software development a set of requirements for the framework, component descriptor language, component wiring and component repository are formulated. We claim that the community rather needs a useful tool even if of restricted use than an ultimate high-tech solution that will remain unaccessible to a community not willing to change overnight those programming practices it has been accustomed to for decades.

1. Introduction

1.1. **The problem.** Today, most scientists can program and do it as an everyday activity. A rich variety of hardware architectures, operating systems, software libraries, protocols, standards, languages, etc. are in place. The scientist, as computer user, trying to communicate and share "business logic" with fellow scientists has to fight the ubiquitous incompatibilities at a day-to-day basis. A large segment of the community writes its own software tools. Consequently, several people

Received by the editors: September 2, 2004.

2000 *Mathematics Subject Classification.* 68U99, 68N99.

1998 *CR Categories and Descriptors.* D.2.12 [**Software Engineering**]: Interoperability – *Interface definition languages*; D.2.11 [**Software Engineering**]: Software Architectures – *Languages, Patterns*; D.2.6 [**Software Engineering**]: Programming Environments – *Graphical environments, Integrated environments, Interactive environments, Programmer workbench*; G.4 [**Mathematics and Computing**]: Mathematical Software – *User interfaces*; J.2 [**Computer Applications**]: Physical Sciences and Engineering – *Aerospace, Archaeology, Astronomy, Chemistry, Earth and atmospheric sciences, Electronics, Engineering, Mathematics and statistics, Physics* .

and research groups around the globe code for the same problem without knowing about each other. With the available source code the task is still major due to the fact that most programs are written for one specific problem and with no reusability considerations in mind. Scientific programs are often not properly structured and even if so, ad hoc binding standards are followed. Even if they used the same programming language, which is not really typical, adapting third party modules for personal use would prove to be a lengthy endeavor.

Many theoretical publications report on results of computer simulations. Unless, some widely popular software is used, reproducing the simulation by interested parties is more effort than most scientists would take.

This situation is referred to by Douglas Post as an actual crisis in computational sciences [2]. His report analyzes the symptoms, causes and possible solutions to this crisis. One of his conclusions is that high-performance computing should move towards newer technologies that provide better efficiency in terms of development efforts even on the cost of loosing some performance.

1.2. **The solution.** Component-based programming is one of todays' hottest topics in computer science [3]. Many view it as the holy grail of software reuse. Sharing code would provide computational sciences with at least three major benefits:

(1) *Efficiency* due to enhanced reusability, diversity and availability
(2) *Quality* due to the strong focus of expert groups, ranking systems
(3) *Reproducibility* of computer experiments by the author, referees and third party scientists

The means for achieving these goals are as follows:

(1) development environment for high-level visual programming
(2) standardized scientific component descriptor language (CDL)
(3) component developer tools facilitating the "componentization" of existing and future scientific software
(4) interactive project management and monitoring
(5) distributed component repository

Basically, a framework is needed which allows the assembling of scientific modules into a computational project depending on the services they offer. These services are defined via their interfaces documented in an XML-based Component Descriptor Language. These components are stored in a global component repository. Additional developer tools should facilitate adapting newly developed and existing code to the framework.

1.3. **The benefits.** Reuse oriented research would truly revolutionize computational sciences. In the component oriented future it is expected that...

- scientists program from scratch only when developing new algorithms and domain specific models
- reuse and integrate seamlessly other's components to support their own research
- find other's computational works in an instant and select from a large repository
- share their code with others
- reproduce results of computational studies published anywhere
- double check their own results using the same or similar algorithms developed by other authors
- have better control of complex computational projects
- component-technology will fully integrate with GRID technologies
- a top quality component layer will sediment in a few years by "natural selection"

2. State of the art

The first concrete steps towards component-based scientific computing have been made before the turn of the century. [4] describes a "standard for interoperability among high-performance scientific components". They touch upon most fundamental concepts of component-based programming in the context of high-performance computing and suggest a standard that they term as "Common Component Architecture" (CCA). Their recommendation for an interface definition language (IDL) closely follows the CORBA principles. The ideas therein are further developed in [5]. This group, however, favors an XML-based language for describing component interfaces. In both cases Java is chosen as the implementation language for the integration framework. Most of later works focus on the integration of the component-based approach into the realm of distributed computing in general and grid computing in particular. Notable efforts have been made by several member institutes of the Common Component Architecture Forum [6] for delivering different implementations of a CCA framework [7, 8, 9]. Most of the activity is centered around the Babel language interoperability tool [10] of the Lawrence Livermoore National laboratory [11]. Babel uses the Scientific Interface Definition Language (SIDL) for defining the interfaces of components implemented in any of the programming languages encountered in scientific computing. The palette includes C, Fortran and variants but also higher-level languages such as Java and Python.

Alexandria [12] is meant to be the future repository for CCA compliant components. As per now it contains no components.

Efforts of more restricted scope can be encountered in different fields of science such as life sciences [13], chemistry [14], nuclear physics [15], to name a few. The tremendous need in all computational sciences for sharing code is apparent if we consider the popularity of Netlib [16]. Netlib is a collection of mathematical software, papers, and databases with hundreds of numerical libraries available for download. There is an enormous number of over 270 million requests that have been made to the repository. And Netlib only includes mathematical software, no modeling or simulation packages built over them. Unfortunately, there is no uniform way for reusing this large variety of modules originating from different places and being the result of independent and uncorrelated efforts.

OpenDX is a powerful, full-featured software package for the visualization of scientific, engineering and analytical data [17]. Its open system design is built on familiar standard interface environments. Its sophisticated data model offers great flexibility in creating visualizations by providing hundreds of built-in specialized functions. The user can drag and drop different data filters onto the canvas and connect output to inputs for setting up the data flow diagram (Figure 1). OpenDX automatically checks the compatibility of the interfaces. The functions in OpenDX are hard-coded into the application. One can contribute with new components by applying the prescriptions that are set for the interfaces. The new component will become available after the recompilation of the application.

3. Scientific vs. commercial software development

In spite of the determining influence of natural sciences on information technology, the maturing process of computer science started with the rise of interest of the private sector. The economical factors imposed several restrictions on using and developing software. The need for optimizing all aspects fueled studies that were beyond the goal of natural sciences. Important aspects included development and maintenance costs implying internal qualities such as reusability and external qualities such as ease of use. As the necessities of science regarding computer technologies are different from the needs of businesses and home users, their tools and programming methodologies are also different. We can even say that though natural sciences drive the evolution of IT they lag behind when it comes to using mature software development methodologies. Below, we have summarized a few of the main distinctive features of programming in natural sciences. The reader should be aware that these statements describe trends not laws. In most cases exceptions are available abundantly:

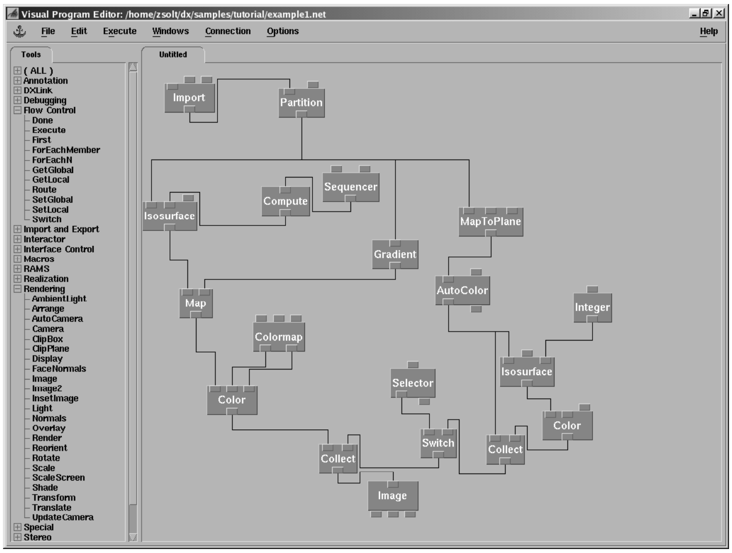


FIGURE 1. Example of flow-based programming. Representation of a data visualization project in Open Data eXplorer [17]

- requires scientific background of the programmer
- for a specific purpose, therefore of limited applicability
- for internal use
- does not have the constraints of commercial software development (budgets, deadlines, marketing strategies)
- performance critical
- developers use low-level languages such as C and Fortran
- high-level abstractions such as OOP concepts are rare
- requires more computer resources than commercial software (CPU, storage)
- lower complexity in terms of function points
- higher complexity algorithms
- less focus on reuse
- less investment in design
- requires less user interaction
- projects involve small to medium sized developer groups

Even though one can identify interdependencies between several items in the above list it is hopeless trying to disentangle them into cause-effect sequences. Any project that aims at changing many decades of programming traditions in computational sciences has to bear in mind these differences. Otherwise, attempts to make up the "grand unified way" will not have the proposed impact on the scientific community and will be downgraded to research of purely academic interest.

4. Requirements

Let us now formulate the expectations that we set against those constituents of a complete solution, which together will make component-based programming in computational sciences possible.

The framework should:

(1) be portable: since one of the main goals is to reduce environment dependence, a platform-independent programming is required such as Java, Python or other interpreted language.
(2) allow for high-performance computation: the implementation language of the framework should offer the facilities necessary for binding low level languages such as C and Fortran. The portability of the framework implies a higher level interpreted language. Therefore a careful design should assure a minimum involvement of the framework during project runtime.
(3) make component development and deployment easy
(4) allow for easy customization of user interfaces to domain specific needs
(5) be user friendly
(6) be open source

The component descriptor language should:

(1) describe both syntactically and semantically the component
(2) support the programming style of computational scientists as far as data structures
(3) be extensible

The component wiring mechanism should:

(1) constitute a low overhead in terms of execution time
(2) require little or no glue-code writing from the developer
(3) not require such changes in the implementation or interface of components that would make them unusable outside the framework

Certain amount of glue-code is usually inevitable. This task, however, should be the responsibility of additional developer tools.

The repository should:

(1) be distributed

(2) offer uniform and transparent access to every site storing components
(3) expose both direct user interfaces and transparent web service type access points
(4) employ a comprehensive and unambiguous classification scheme of components
(5) provide advanced search facilities
(6) provide ranking facilities of components as to popularity, user satisfaction, etc.
(7) allow for easy upload and download of components

5. Approach

Present section will discuss those guidelines that have been set forth for the COmputational MODule Integrator (COMODI) project [1]. COMODI is an international and interdisciplinary initiative attempting to offer a viable solution to computational sciences for moving towards the promised land of component technology. Henceforth, depending on the context, the word "COMODI" will refer either to the project itself, comprising the participants, ideas and means, or to the framework software with or without additional ingredients such as the component descriptor language and developer tools. Considering the faint success of present attempts it is apparent that the main challenges are not so much of technical nature but rather consist in finding the solution that is most likely to receive acceptance from the conservative community of computational scientists. COMODI's aim is to shape present programming practices such that they could metamorphose into future paradigms. In the first stage, COMODI will try to accommodate to present trends in scientific software development. This will be followed by a new generation of components that will adapt to the most recent paradigms.

COMODI's first target is the scientific software developer segment. The reasoning behind this strategy is to build up a component repository that by its sheer size will provide the variety and quality of components that will be appealing to most users. Besides, due to the level of proficiency in computer related issues, the developer segment will tolerate with more patience the less user-friendly and buggy versions in the initial phase of COMODI and will be able to contribute with important amount of know-how to its development.

Most present efforts to bring under the same roof the problematics of component-based, distributed and parallel programming are in their infancy. Indeed, this unification will most probably be ultimate solution for computational sciences. However, the surveys made by COMODI show that even though many scientists rely on parallel code the parallelism therein is usually transparent. Most don't participate in the development of parallel code and are aware of grid computing to the extent they are aware of quantum computing. Therefore we claim that effort should be invested into developing a framework, CDL and repository without mixing it with the paradigms of distributed computing.

Where does COMODI want to be better?

- short learning curve for scientific component developers
- no constructs of high-level abstraction, no new languages required, leaving that to a later stage
- bottom-up construction of the framework
- automatic glue-code generation
- zero source code line change for adapting scientific routines to COMODI

5.1. **Structure of the framework.** In order to make COMODI itself easily extensible it has to be component-based. This requires the separation of the framework into a core layer and several other modules built on the top of it. It is possible to enforce a very general view on this component architecture and deal uniformly with computational components and components that are intimately related to the framework itself. In this approach, anything apart from the core is a component, be it a simple numerical component or a heavyweight GUI. However, this uniformity, while simplifying the integration of components vital to the proper functioning of COMODI, will hit back by compromising the postulated simplicity of wiring computational components by users. Even though the customizability of COMODI to the needs of different user groups is a priority we can build upon the premise that the group of those that will contribute to COMODI will be a fraction of those that will use it or contribute to the repository. Therefore it is sensible not to sacrifice the support of user and component developer activities to those related to the development of COMODI.

COMODI will expose a core level API allowing the independent development of satellite modules such as GUIs and batch system handlers. The component architecture of the framework software itself would permit the contribution of several parties to the development of COMODI. More importantly, it will diversify the user experience allowing such "packagings" that best fit the purposes of a user group of a particular profile or closely resemble such visual development environments that this group is accustomed to, such as LabView, Simulink or OpenDX.

Under this cover COMODI, the CDL and the wiring mechanisms will be able to develop without exposing the users to major interface functionality disruptures between versions.

5.2. **Granularity.** There is no consensus as to what exactly a component and a component's port is [3]. This is due to the differences between the various languages concerning structuring code and passing data between processing units. In our case, a component can be viewed from the perspective of the computational project wherein the constituting elements are functions and procedures. If regarded from the point of view of the data that the project operates with, components should be objects in the sense of the OOP terminology. And finally, it is common to use the word "component" to a bundle of software entities, interfaces, classes, data, deployed as a unit. In other words, the physical properties of the software define the boundaries of a component.

At a lowest granularity level, where functions can be considered as components, each argument in the function signature can be considered a port. At a level of a class method, calls stand for the elementary access point while at a physical level deployment, packages can be considered the unit of assembly and whole interfaces represent the entry points. The table below is a summary of the different levels of granularity.

Component	**Port**
Function	argument of the signature
Class	function signature
Package	interface/pure abstract class

COMODI works at the lowest level using functions as atomic components. Even though it is meant to be a support for low-level programming, higher levels can also be emulated. Alternatively, COMODI can come with higher level APIs built over the base API. The higher level API's can be used for components that obey the rules of OOP.

5.3. **Component wiring.** In order to satisfy the requirement of low overhead in performance it is necessary that the framework does not intermediate the communication between components. Instead, it will wire up the connections by setting direct component-to-component references [10]. Therefore the components will have to comply with certain rules as to the interfaces they expose. These rules, however, should be set such that the guidelines formulated in section 4 are closely followed. [18] discusses in more detail several alternatives.

5.4. **Component Descriptor Language.** The problem of the CDL is the alpha and omega of any component-based framework. It should inspire from existing technologies such as CORBA, but at the dawn of grid computing web services are the most likely to set the standards. Given the many aspects that ought to be considered, [19], present paper will not even try to go deep into this topic.

COMODI's CDL is XML-based, even though a CDL file can also have an SIDL type of representation that is more appealing to the technically trained eye [10]. The CDL's complexity is expected to grow together with the user community and the number of application areas.

6. Conclusions and outlook

The tasks to be carried out for achieving COMODI's objectives of efficient code sharing are major. It will clearly need a large user and component developer base that will bring into motion the component repository. This goal can be achieved only with a sufficiently low accommodation effort threshold. COMODI is an attempt to minimize this threshold on the cost of temporarily sacrificing generality. As such, it is not meant to be long-lived in present form. However, higher-level constructs, when the times are ripe, can be built over, hiding such obscure details that in COMODI's initial form are required for assisting a community with deep roots in "performance mining" and low-level programming. The benefits of code sharing can put scientific research on a ground that challenges science-fiction. Nevertheless, the islands of isolated feeble initiatives should coalesce into a general awareness and coherent world-wide effort. Otherwise, computational sciences are doomed to spend at least another decade in the past.

7. Acknowledgments

This research is supported by the Netherland's Organization for Scientific Research (NWO) with grant no. 048.031.003 and by the National Research Council of Romania (CNCSIS) with grant no. 37/2004.

References

[1] *COMODI homepage,* http://phys.ubbcluj.ro/comodi/

[2] D. Post, *The Coming Crisis in Computational Science*, Proceedings of the IEEE International Conference on High Performance Computer Architecture: Workshop on Productivity and Performance in High-End Computing, Madrid, Spain, February 14, 2004

[3] C. Szyperski, D. Gruntz and S. Murer, *Component Software; Beyond Object Oriented Programming*, 2nd edition, Addison-Wesley (2002)

[4] R. Armstrong, Dennis Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker and, B. Smolinski, *Toward a Common Component Architecture for High-Performance Scientific Computing*, Proceedings of the 8th IEEE International Symposium on High-Performance Scientific Distributed Computing, August (1999)

[5] R. Bramley, K. Chiu, S. Diwan and D. Gannon, *A Component Based Services Architecture for Building Distributed Applications*, Ninth IEEE International Symposium on High Performance Distributed Computing, August 01-04, 2000 (http://www.extreme.indiana.edu/ccat/papers/hpdc2000.pdf)

[6] *Common Component Architecture (CCA) Forum homepage,* http://www.cca-forum.org

[7] *CCAFE homepage,* http://www.cca-forum.org/~baallan/ccafe

[8] *SCIRun homepage,* http://www.sci.utah.edu

[9] *XCAT homepage,* http://www.extreme.indiana.edu/xcat

[10] *Babel homepage,* http://www.llnl.gov/CASC/components/babel.html

[11] *Component Architecture for Scientific Computing homepage,* http://www.llnl.gov/CASC/components/

[12] *Alexandria component repository homepage,* http://www.llnl.gov/CASC/components/alexandria.html

[13] *IBM Life Sciences Framework homepage,* http://www-306.ibm.com/software/info/university/products/lifesciences/framework/

[14] *Octet Molecular Informatics Framework homepage,* http://octet.sourceforge.net/

[15] *ROOT homepage,* http://root.cern.ch/

[16] *Netlib homepage,* http://www.netlib.org/

[17] *Open Data Explorer homepage,* http://www.opendx.org/

[18] Zs.I. Lázár, B. Pârv, *COMODI: Component Wiring in a Framework for Scientific Computing*, Studia Universitatis Babes-Bolyai, Series Informatica 49 (2), 2004, pp. 103–110

[19] A. Mayer, S. McGough, M. Gulamali, L. Young, J. Stanton, S. Newhouse, J. Darlington, *Meaning and Behaviour in Grid Oriented Components*, Proceedings of the Third International Workshop on Grid Computing, Springer-Verlag (2002) (www.lesc.ic.ac.uk/iceni/pdf/Grid2002.pdf)

Department of Theoretical and Computational Physics, Faculty of Physics, Babeş-Bolyai University, Str. M. Kogălniceanu Nr. 1, RO 400084 Cluj-Napoca, Romania
E-mail address: `zlazar@phys.ubbcluj.ro`

Chair of Programming Languages and Methods, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Str. M. Kogălniceanu Nr. 1, RO 400084 Cluj-Napoca, Romania
E-mail address: `bparv@cs.ubbcluj.ro`

Chair of Programming Languages and Methods, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Str. M. Kogălniceanu Nr. 1, RO 400084 Cluj-Napoca, Romania
E-mail address: `afanea@cs.ubbcluj.ro`

Physical Chemistry & Molecular Thermodynamics, DelftChemTech, Technical University of Delft, Julianalaan 136, 2628 BL Delft, The Netherlands
E-mail address: `J.R.Heringa@tnw.tudelft.nl`

Physical Chemistry & Molecular Thermodynamics, DelftChemTech, Technical University of Delft, Julianalaan 136, 2628 BL Delft, The Netherlands
E-mail address: `S.W.deLeeuw@tnw.tudelft.nl`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

COMODI: COMPONENT WIRING IN A FRAMEWORK FOR SCIENTIFIC COMPUTING

ZSOLT I. LÁZÁR AND BAZIL PÂRV

Abstract. We present several alternatives for wiring atomic components, namely functions, within the COMODI framework [1]. The benefits and drawbacks of the different solutions are analyzed in the light of a few major guidelines that are set forth for COMODI. A mixture of pipe&filter and repository type of architectures is found to satisfy best the requirements for scientific computing. The role of connector components are also discussed.

1. Introduction

In [2], a few guidelines have been established for the possible wiring mechanisms. According to this paper the wiring should

- constitute a low overhead in terms of execution time
- require little or no glue-code writing from the developer
- not require such changes in the implementation or interface of components that makes them unusable outside the context of the framework

Even though the above requirements sound restrictive the forthcoming sections will show that it is possible to design such architectures. Let us first define the problem.

2. Calls & connectors

As described in [2] and ch. 8 of [4], components can be considered at different levels of granularity. For the points to be made in this paper, we shall distinguish

Received by the editors: September 2, 2004.

2000 *Mathematics Subject Classification.* 68U99, 68N99.

1998 *CR Categories and Descriptors.* D.2.12 [**Software Engineering**]: Interoperability – *Interface definition languages*; D.2.11 [**Software Engineering**]: Software Architectures – *Languages, Patterns*; D.2.6 [**Software Engineering**]: Programming Environments – *Graphical environments, Integrated environments, Interactive environments, Programmer workbench*; G.4 [**Mathematics and Computing**]: Mathematical Software – *User interfaces*; J.2 [**Computer Applications**]: Physical Sciences and Engineering – *Aerospace, Archaeology, Astronomy, Chemistry, Earth and atmospheric sciences, Electronics, Engineering, Mathematics and statistics, Physics* .

between physical and logical components. By the former we mean units of deployment, namely native library files, Java class files, etc. In view of the low-level languages used in high-performance computing, this is almost synonymous with units of compilation. Logical components are viewed from the perspective of the computational project assembled by the user. A *project* is a graph of interconnected components, as shown in Figure 1. Following the recommendations from [2], we shall work at the lowest level of granularity, i.e. the atomic components will consist in regular functions. Within the context of this work we shall use the words "component" and "function" interchangeably. *Connection* means data flow via function interfaces from the output of one component to the input of another.

We can distinguish between *horizontal* and *vertical communication*. The latter, e.g. 1.1 - 2.1.2, is the direct representation of the *pull model* for module communication. It consists in a function calling another, which returns a result to the former. This is basically the only model for communication in languages such as C or Fortran. In terms of connection architectures, it belongs to a client-server setup. On the other hand, horizontal calls imply the piping of an output into the input ports of another component. This pipe&filter architecture, and implicitely the *push model*, is not directly supported by low level languages. It is the representation of equivalent vertical calls at a high level of abstraction.

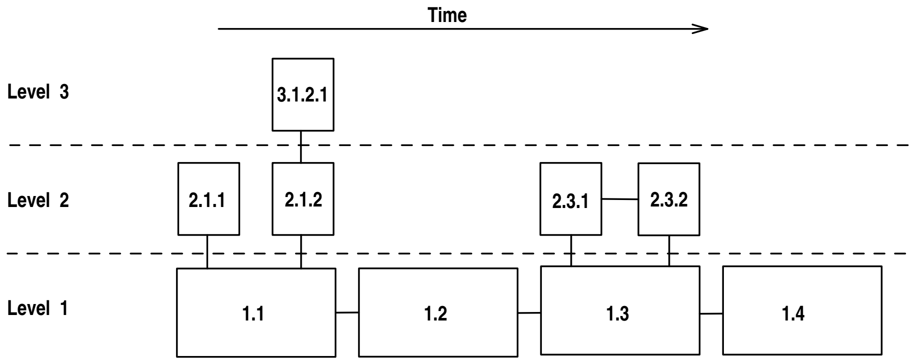


FIGURE 1. Structure of a project in a framework at a higher level of abstraction. See Figure 2 for more details.

We can define two levels of abstraction when modeling this communication. At higher level (Figure 1), we disregard the existence of the framework and certain elementary *connectors* that are components that perform simple tasks pertaining rather to the mediation of information between two or more components. This indirection can be due to reasons such as the necessity for satisfying certain syntactical requirements of component wiring. For a review on connectors see [3] and ch. 10 in [4]. At a lower level of abstraction (Figure 2), horizontal data flow is "translated" into vertical flow by the help of connector components that we will henceforth term as *propagators.*

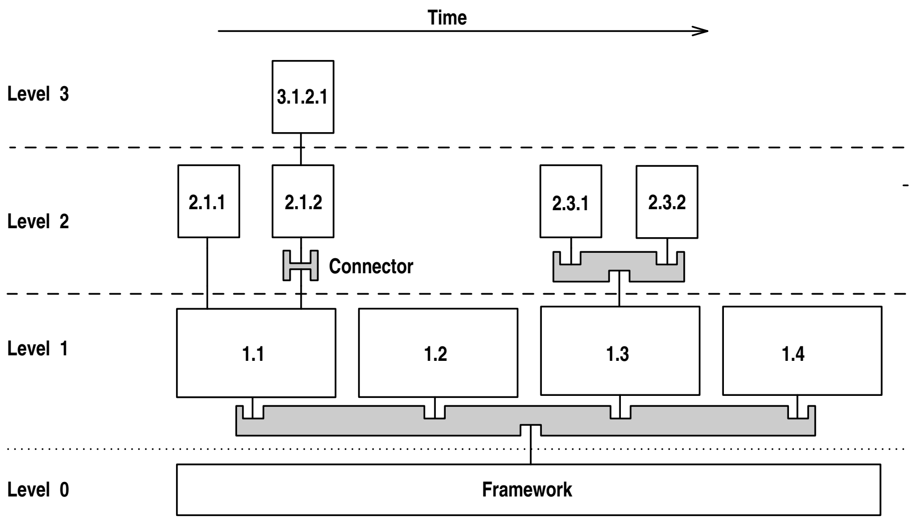


FIGURE 2. Structure of a project in a framework at a more concrete level of abstraction. The dotted line represents communication through interfaces that are discovered at runtime. The data flow between the upper layers, through the dashed line, takes place via interfaces that are established at compile time and necessarily have to be compatible. The components reach execution state starting from left to right, bottom to top. In this case the order is: framework, 1.1, 2.1.1, 2.1.2, 3.1.2.1, 1.2, 1.3, 2.3.1, 2.3.2, 1.4.

Excluding for now more advanced flow-control such as conditional branching and loops, the connection of components at low abstraction is tree-like since there are only vertical calls. At the lowest level, the framework calls the component positioned at level one. These can call others at higher level and so on. The layering is strict in the sense that there is communication only between adjacent levels. In the following, we will refer to components that are directly accessed by any given component as the latter's *child components.*

There are "framework aware" activities done at all three levels: at compile-, link- and runtime.

At compile time, there is extra coding necessary that will endow the physical component, be it a library or class, with the capabilities required by the framework. The definition of "dangling bonds" and statically set bindings is up to the developer. The glue-code can be written manually or generated automatically.

At link-time, the user decides on the particular wiring to be established between the components while the framework verifies the connections and sets the references. At this point, a variety of connectors need to be employed and configured, some manually by the user, some automatically by the framework. A typical example for manual configuration of a connector would be the case of two components that provide and require similar data but the correspondence between the parameters in the argument list is not obvious, e.g. (float pressure, float temperature) $\neq$ (float temperature, float pressure). However, the framework can set the propagator connectors by itself if the connected input and output ports are compatible. Other, special connectors fall in between these two extremes. They will require indications from the user while most of the settings are carried out by the framework.

In this approach, there is a uniform handling of calls at all levels of the hierarchy. As described in [2], it is necessary that runtime calls are done directly between the different components, that is, they are not intermediated by the framework. At the lowest level, the framework will call a component that will do all the invocations of the second level components. Following present practices this will often be a shell script or equivalent. In other cases it will be just an automatically generated propagator, such as in Figure 2.

3. Wiring architectures

We can distinguish between two largely different ways of managing the access of components to the references of their children:

(1) using a pipe&filter architectural style, wherein components posses only the reference data that exclusively concerns them and pass to their children the subtree of information that is necessary.
(2) using a repository-based architecture. Here a collectively accessible data repository is consulted by each component and the necessary information extracted.

Both architectures have beneficial and disadvantageous implications which we will explore in the following.

3.1. **A pipe&filter architecture.**

This alternative confers more autonomy to the logical constituents of a project, i.e., to the functions. We can even say that this approach favors the logical component view on the available component base. The interfaces of all logical components will need to be extended with a new argument through which the reference tree is communicated to the component, Figure 3a. We will refer to it as *paramString* as it can be a simple XML string or an equivalent representation of it such as a data aggregate (structure or object) describing an XML node. Every function will receive a *paramString* containing child information and invokes child components by sending, along other data, a subtree of the *paramString*. An important disadvantage of this architecture is that the signature of all functions will be unnaturally "distorted" because of the *paramString* argument that has to be passed along. Moreover, the implementation of the functions should be changed such that the parsing of the *paramString* could be done. It is feasible to parse the *paramString*s and set the references only once during the linking of the project by storing the references in internal static variables of each component.

3.2. **A repository architecture.** The suggested architecture enforces an object-oriented approach and confers a more important role to the physical aspect of components. The framework may create as many instances of a physical component (Java class, C library, etc.) as many times one of its methods are used in the project. Each instance will manage the reference data of its methods to other "child methods" internally, storing them in instance variables or global variables in case of a C library. The most important advantage is that the logical components, the methods, will not need to pass *paramString*s to each other during runtime and they will not need to have any implementation part dealing with *paramString.* Similarly to the pipe&filter architecture, the parsing of the *paramString* and setting the references is done only once per project. The beauty flaw that appears in this case is the global scope of the reference variables, which is usually not preferred by those writing computational library functions. On the other hand, existing libraries can be much more readily adapted to the framework.

In conclusion, the repository approach has the important benefit of not requiring modifications in the interfaces (signatures) of the logical components (functions). Moreover, unlike the pipe&filter architecture, the implementation of these components do not require any glue-code either. These benefits come at the cost of child reference variables with the scope of the whole physical component. In the pipe&filter case, XML parsing has to be included into all functions. There is more effort needed for implementing those indirections for each method, but some may prefer to pay this price for making the child reference variable's scope local.

Fortunately, the benefits of the two approaches can be collectd into an architecture that behaves like the pipe&filter variant during link-time and exhibits repository architecture traits during runtime.

3.3. **Mixed architecture.** There are different solutions sketched in Figure 3. Even though the last three show traits of the repository architecture, they are rather variants of the pipe&filter version. Each of them do a better or worse job in avoiding the three main problems that characterize this architecture: (i) the necessity of modifying the body of the function to include *paramString* parsing and the initialization of children references, (ii) the modified function signature that is used when being called and/or when calling the children, and (iii) the indirection introduced within the body of the physical component.

In Figure 3a, the signature includes an extra *paramString* entry and the body of the function is also modified.

Solution shown in Figure 3b moves the wiring into a separate function. In this way, the actual function can retain its most natural form both from the point of view of the interface and of the implementation. The drawback is the indirection that is introduced. Another imperfection is the fact that outgoing calls, i.e., calls to children will still require an outgoing interface that includes the *paramString*.

The third alternative, illustrated in Figure 3c, eliminates this problem by offering two different entry points at link-time and at runtime. However, this comes at the cost of an extra level of indirection. In this setup, function calls only require the passing of actual data.

One can eliminate the second level of indirection during runtime, the way it is depicted in Figure 3d. The runtime entry point is called once also during link-time and the children references are requested from the function in charge with wiring. As a result, the wiring function can be completely avoided during runtime. The actual business logic will be in the body of a function that receives all necessary information, including the child references, as parameters. This has the additional benefit of self-containment. The function is fully functional also outside the context of the framework without confusing extra arguments. On the other hand, there are two other functions containing glue-code of no interest for the component developer. However, the burden of writing glue-code can be taken over by auxiliary applications once the interface descriptor is available.

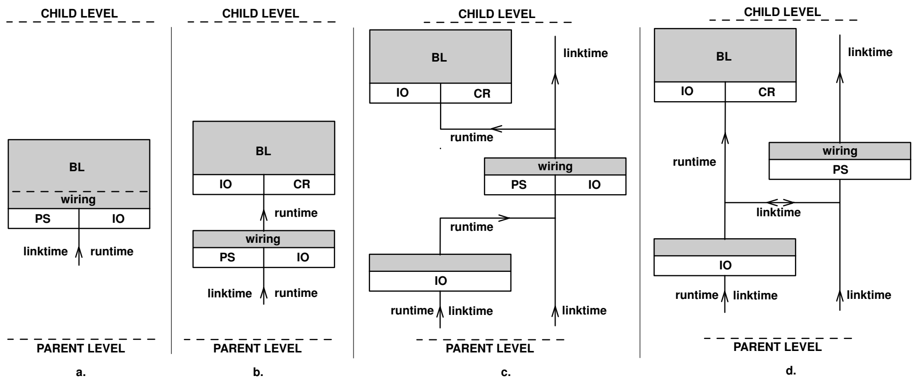


FIGURE 3. Different binding solutions for the pipe&filter architecture. Notation: PS = *paramString*, IO = input/output, BL = business logic, CR = children references

3.4. **Putting it all together.** At times, the different components will share a small amount of data, sometimes large chunks of it. The exchange may happen once in a while or frequently. As long as small amounts of data is flowing occasionally from one component to the other any binding will do it. If the amount is large or the communication is frequent or both, performance and storage issues should be considered.

In Figure 1 we saw two different communication types: framework-component and component-component. Since the framework-component connection is done dynamically through reflection, the whole communication process imposes a much larger overhead than direct connection between components. In C, sharing large amounts of data is done in a straightforward manner as methods pass each other pointers to the array they want to share. This has the benefit of avoiding data replication. Assuming that the framework is implemented in Java, the Java Native Interface (JNI) also passes objects by reference. However, this does not solve the problem of large arrays. They get copied to another location in the memory and upon leaving the native function, the original data gets updated with the modified one. This makes JNI in its original form unfit for the job. Fortunately, since version 1.4 of the JDK, JNI has been renewed with new features targeting exactly the problem of performance. Previously, Java introduced a new package of IO classes which performs much better than the original one. The new JNI makes use of this package and provides more low level approach to data communication between object methods. This, to some extent, solves the problem of passing large chunks of data via the framework.

Frequent data passing occurs at higher levels in the hierarchy between components which heavily use the specialized services of some lightweight child components.

When connecting components, a simple syntactic check of primitive data types is not sufficient. Data types relevant to science, such as temperature, electric resistance, particle number, represent a certain amount of semantics. The types preferred by the computer are: double, long, etc. One alternative for including semantics would be the definition of classes and therein all properties of the given quantity. This would reduce the semantic problem to a regular syntactic one. The main problem of this approach is that a consistent organization of the inheritance tree of all quantities used in natural sciences may not be feasible. Moreover, the used low level languages do not offer a natural way of dealing with objects.

As for now, the connection could be done by the user tying explicitly outputs and inputs that are semantically compatible. The syntactical validation can be done simultaneously by the framework. There are reversed situations when two entry points are compatible semantically but different data types are used. For instance, electric charge is described using integer data type in one component and floating point in another. This will require simple connector objects.

4. Conclusions and outlook

We have shown that there is a way to harmonize all requirements for a wiring mechanism. The suggested architecture neither causes execution overhead, nor requires extra implementation from the developer, nor distorts in any way the signatures of functions when adapting them to COMODI. As a next step, one has to look into the actual form of the glue-code that is generated so that it could cope with tasks such as callbacks to the framework for exception and error handling, project monitoring, and user interaction management. Achieving this for the relevant languages in the spirit of Babel [5] is expected to be a considerable technical challenge but none that is insurmountable.

5. Acknowledgments

This research is supported by the Netherland's Organization for Scientific Research (NWO) with grant number 048.031.003 and by the National Research Council (CNCSIS) with grant code 37/2004. One of the authors (Zs.I.L) is especially indebted to Prof. Dr. Simon de Leeuw from the Physical Chemistry and Molecular Thermodynamics Group at the Department of Chemical Technologies of the Technical University of Delft, The Netherlands for sharing his own and his group's extensive experience in computer simulation. The insightful comments of dr. Jouke Heringa are gratefully acknowledged.

References

[1] *COMODI homepage,* http://phys.ubbcluj.ro/
[2] Zs.I. Lázár and B. Pârv, *COMODI: Guidelines for a Component Based Framework for Scientific Computing*, Studia Universitatis Babes-Bolyai, Seria Informatica 49 (2), 2004, pp. 91–102
[3] A. Mayer, S. McGough, M. Gulamali, L. Young, J. Stanton, S. Newhouse, J. Darlington, *Meaning and Behaviour in Grid Oriented Components*, Proceedings of the Third International Workshop on Grid Computing, Springer-Verlag (2002) (www.lesc.ic.ac.uk/iceni/pdf/Grid2002.pdf)
[4] C. Szyperski, D. Gruntz and S. Murer, *Component Software; Beyond Object Oriented Programming*, 2nd edition, Addison-Wesley (2002)
[5] *Babel homepage,* http://www.llnl.gov/CASC/components/babel.html

Department of Theoretical and Computational Physics, Faculty of Physics, Babeş-Bolyai University, Str. M. Kogălniceanu Nr. 1, RO 400084 Cluj-Napoca, Romania
E-mail address: zlazar@phys.ubbcluj.ro

Chair of Programming Languages and Methods, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Str. M. Kogălniceanu Nr. 1, RO 400084 Cluj-Napoca, Romania
E-mail address: bparv@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

DATA ANALYSIS WITH FUZZY SETS: A SHORT SURVEY

HORIA F. POP

Abstract. This paper develops a short survey of the fuzzy sets theory and how it can contribute to better, more robust data analysis methods. We briefly cover the major fuzzy sets definitions, the rough sets alternative, fuzzy clustering, fuzzy classification, fuzzy regression, data visualisation and projection. We conclude that the fuzzy sets represent a very important advance for intelligent data analysis.

1. Introduction

The fuzzy sets represent a mathematical theory suitable for modelling imprecision and unclearness. Generally, unclearness is associated to the difficulty of making precise statements with respect to a certain topic. On the other side, in the *Fuzzy Sets Theory*, the hard alternative yes - no is indefinitely nuanceable. From this point of view, the fuzzy sets theory is not only a theory dealing with ambiguity; it is also a theory of fuzzy reasoning.

The interpretation of *Fuzzy Logic* is twofold. In a narrow sense, fuzzy logic is a logical system that may be viewed as an extension and a generalization of classical logic. In a wider sense, fuzzy logic is almost synonymous with the theory of fuzzy sets, encompassing the 'strict' fuzzy logic.

The fundamental fact that lies behind fuzzy logic is that any field and any theory may be fuzzified by replacing the concept of crisp set with the concept of fuzzy set. Thus have appeared theoretic fields such as fuzzy arithmetic, fuzzy topology, fuzzy graph theory, fuzzy probability theory, 'strict' fuzzy logic, a.o. Similarily, applied fields that suffered generalizations are fuzzy neural network theory, fuzzy pattern recognition, fuzzy mathematical programming, a.o.

Received by the editors: December 10, 2004.

2000 *Mathematics Subject Classification.* 03E72, 62-07, 62Hxx.

1998 *CR Categories and Descriptors.* I.5.1 [**Computing Methodologies**] : Pattern Recognition – *Models – Fuzzy set*; I.5.2 [**Computing Methodologies**] : Pattern Recognition – *Design methodology – Classifier design and evaluation, Feature evaluation and selection, Pattern analysis*; H.3.1 [**Information Systems**] : Information Storage and Retrieval – *Content analysis and indexing* .

What is gained through fuzzyfication is *greater generality*, *higher expressivity*, an *enhanced ability to model* real-world problems, and a methodology for exploiting the *tolerance for imprecision* [6].

2. Fuzzy sets

Fuzzy sets were introduced as generalization of the classical crisp sets, as a means of representing and manipulating imprecise data. However, not all the properties that are valid for operations on crisp set are valid for fuzzy sets, and the inability to deal with this may result in improper use of fuzzy sets [13].

Professor Lotfi A. Zadeh used the following words to describe the importance of fuzzy sets:

> "The fuzzy set was conceived as a result of an attempt to come to grips with the problem of pattern recognition in the context of imprecisely defined categories. In such cases, the belonging of an object to a class is a matter of degree, as is the question of whether or not a group of objects form a cluster."

We recall here the basic definitions of fuzzy sets.

Let X be a non-empty crisp set. The *fuzzy set* A in X is characterised by its membership function, $A : X \to [0,1]$, where $A(x)$ is interpreted as the membership degree of element $x \in X$ in the fuzzy set A.

The *universal fuzzy set* in X, denoted by X, is defined by $X(x) = 1, \forall x \in X$.

The *empty fuzzy set* in X, denoted $\emptyset$, is defined by $\emptyset(x) = 0, \forall x \in X$.

The fuzzy sets A and B in X are said *equal*, and denoted $A = B$, if $A(x) = B(x), \forall x \in X$.

The fuzzy set A in X is *a subset* of the fuzzy set B in X, denoted $A \subset B$, if $A(x) \leq B(x), \forall x \in X$.

The *complement* of a fuzzy set A in X, denoted $\bar{A}$ is defined by $\bar{A}(x) = 1 - A(x), \forall x \in X$.

The *intersection* of fuzzy sets A and B in X, denoted $A \cap B$, is the fuzzy set defined as $(A \cap B)(x) = T(A(x), B(x)), \forall x \in X$, where T is a triangular norm (i.e. commutative, associative, non-decreasing in each argument, and $T(a, 1) = a, \forall a \in [0,1]$).

The *union* of fuzzy sets A and B in X, denoted $A \cup B$, is the fuzzy set defined as $(A \cup B)(x) = S(A(x), B(x)), \forall x \in X$, where S is a triangular conorm (i.e. commutative, associative, non-decreasing in each argument, and $S(a, 0) = a, \forall a \in [0,1]$).

For any triangular norm $T : [0,1] \times [0,1] \to [0,1]$, the dual triangular conorm $S : [0,1] \times [0,1] \to [0,1]$, defined by $S(x,y) = 1 - T(1-x, 1-y)$ is, itself, a triangular conorm.

In what follows we will analyse some of the properties of crisp sets from the point of view of applicability to fuzzy sets. We will exemplify with two widely-used t-norms and t-conorms:

Standard: $T_S(a,b) = \min\{a,b\}$ and $S_S(a,b) = \max\{a,b\}$
Lukasiewicz: $T_L(a,b) = \max\{a+b-1,0\}$ and $S_L(a,b) = \min\{a+b,1\}$

	Property	Crisp sets	T_S and S_S	T_L and S_L
(1)	Idempotence laws	Valid	Valid	Invalid
(2)	$A \cup \bar{A} = X$ and $A \cap \bar{A} = \emptyset$	Valid	Invalid	Valid
(3)	Distributivity laws	Valid	Valid	Invalid
(4)	De Morgan laws	Valid	Valid	Valid

Properties (1) are satisfied only by the standard t-norm and t-conorm. Properties (4) are satisfied by any t-norm and the dual t-conorm. Any t-norm and t-conorm, defined on nondegenerate fuzzy sets, that satisfy properties (2), do not satisfy properties (3). For any t-norm and t-conorm that verify the properties (2) and (3), the fuzzy sets have only crisp values, i.e. they reduce to crisp sets.

The properties above make the standard definition suitable for use in 'strict' fuzzy logic, and the Lukasiewicz definition suitable for use in fuzzy clustering.

3. Rough sets

A different generalisation of the concept of crisp set is the *rough set*. But while a fuzzy set comes as a membership function, the idea behind rough sets is to approximate sets using collections of sets [8].

Let us consider a collection of sets $C = \{C_1, C_2, \ldots\}$, and a set D. We define the *lower approximation of D by C*, denoted D_L, the set

$$D_L = \cup C_i \text{ such that } C_i \cap D = C_i.$$

We define the *upper approximation of D by C*, denoted D^U, the set

$$D^U = \cup C_i \text{ such that } C_i \cap D \neq \emptyset.$$

We define the *boundary of D by C*, denoted $D_L^U = D^U - D_L$.

A set D is said to be *rough* if it has a non-empty boundary when approximated by C. Otherwise, the set D is called *crisp*.

Fuzzy sets model the inherent vagueness in the data. As a difference, rough sets model ambiguity due to lack of information.

4. Fuzzy clustering

The subject of *Cluster Analysis* is the classification of objects into categories. Since most categories we use have vague boundaries, and may even overlap, the necessity of introducing fuzzy sets is obvious. The main issues approached by research in Fuzzy Clustering refer to the following [10]:

Data representation: Input data is obtained by measurements on the objects that are to be recognized. Each object is represented as a vector of measured values $x = (x_1, x_2, \ldots, x_s)$, where x_i is a particular characteristic of the object.

Feature extraction: Due to the large number of characteristics, there is a need to extract the most relevant characteristics from the input data, so that the amount of information lost in this way is minimal, and the classification realised with the projected data set is relevant with respect to the original data. In order to achieve this feature extraction, different statistical techniques, as well as the fuzzy clustering algorithms outlined here, may be used.

Clusters shape: Pattern recognition techniques based on fuzzy objective functions minimizations use objective functions which are particular to different clusters shapes. Ways to approach the problem of correctly identifying the clusters shape are the use of adaptive distances in a second run to change the shapes of the produced clusters so that all are unit spheres, and adaptive algorithms which dynamically change the local metrics during the iterative procedure in the original run, without the need of a second run.

Cluster validity: Another problem of such algorithms is that of determining the optimal number of classes that correspond to the cluster substructure of the data set. There are two approaches: the use of validity functionals which is a post-factum method, and the use of hierarchical algorithms, which produce not only the optimal number of classes (based on the needed granularity), but also a binary hierarchy that show the existing relationships between the classes.

Defuzzification of final fuzzy partition: Since humans need for their analysis crisp partitions, such procedures should be able to produce, together with the final fuzzy partition, a crisp version thereof. There are a number of techniques, which differ on their ability to produce a non-degenerate crisp partition (i.e. a crisp partition with all the member crisp sets non-empty), and on their ability to produce the crisp partition closest to the original fuzzy partition.

Method: Essentially, the method is based on defining a dissimilarity function between the data set and the prototypes (not necessarily vectors in the same space) of the fuzzy classes. A fuzzy objective function is defined based on this dissimilarity function. In order to minimize the fuzzy objective function, a two step iterative procedure is used: for certain prototypes, the optimal fuzzy partition is determined; reciprocally, for a certain fuzzy partition, the optimal prototypes are determined. This procedure continually decreases the value of the objective function.

In what follows we will recall the Fuzzy Clustering generic algorithm [1].

Let us consider a set of data items $X = \left\{x^1, \ldots, x^n\right\} \subset \mathbb{R}^s$, characterised in such a way that we may define a measure of their (dis)similarity. Our aim is to find a fuzzy partition $P = \{A_1, \ldots, A_c\}$ that best represents the cluster substructure of the data set X., i.e. data items of the same class should be as similar as possible, and data items of different classes should be as dissimilar as possible.

Let us suppose that the fuzzy sets A_i are represented by some prototypes L_i, and that we can define a function $D(x^j, L_i)$ that represents the dissimilarity between a certain data item x^j and the prototype L_i.

At this point, we do not specify the exact shape of the prototypes L_i. Of course, when the prototypes L_i are fully described, the dissimilarity $D(x^j, L_i)$ will need to be fully described, as well.

We define the representation inadequacy of the fuzzy partition P by the set of prototypes L as the function

$$J(P, L) = \sum_{i=1}^{c} \sum_{j=1}^{n} A_i(x^j)^m \cdot D(x^j, L_i)$$

where $m > 1$ is a constant defining the weight the fuzziness is taken into account with.

Let us observe that J is an objective function of the tipe of square errors sum. Our problem is to determine the fuzzy partition P and its prototype representation L that minimizes the function J. Because an algorithm to obtain an exact solution to this problem is not known, we will use an approximate method in order to determine a local solution. The minimum problem will be solved by using an iterative method where J is successively minimized with respect to P and L.

The *Fuzzy Clustering generic algorithm* is, thus, the following:

(1) Given: c, n, m, and $x^j, j = 1, \ldots, n$; $l = 0$;
(2) Initialize fuzzy partition $P^{(0)} = \{A_1, \ldots, A_c\}$;
(3) Compute prototypes L_i that minimize $J(P^{(l)}, \cdot)$; this depends on the definition of D, and may be costly;
(4) Compute fuzzy partition $P^{(l+1)}$ that minimizes $J(\cdot, L)$:

$$A_i^{(l+1)}(x^j) = \frac{1}{\displaystyle\sum_{k=1}^{c} \left(\frac{D(x^j, L_i)}{D(x^j, L_k)}\right)^{\frac{1}{m-1}}}$$

(5) Compare fuzzy partitions $P^{(l+1)}$ with $P^{(l)}$. If close enough, then STOP, else increase l by 1 and GOTO STEP 3.

4.1. **Fuzzy clustering variants and improvements.** The first algorithm from this class, developed by Dunn [3], was the *Fuzzy c-Means* algorithm and used spherical prototypes. For an up-to-date discussion of the different variants and improvements of the Fuzzy c-Means algorithm, see [5].

Geometric prototypes:

- Fuzzy c-Means – Dunn (1974), Bezdek (1974)
- Fuzzy c-Varieties and Fuzzy c-Elliptotypes – Bezdek et.al. (1981)
- Fuzzy c-Ellipsoids – Lenart (1989)
- Adaptive Fuzzy Clustering – Dave (1989), Dumitrescu, Pop (1990)
- Use of fuzzy covariance matrix – Gustaffson, Kessel (1979)
- L_p Fuzzy c-Means – Miyamoto, Agusta (1998), Hathaway, Bezdek (2000)

Empty shell prototypes:

- Fuzzy c-Shells – Dave (1990), Krishnapuram, Nasraoui, Frigui (1992)
- Adaptive Fuzzy c-Shells – Dave, Bhaswan (1992)
- Fuzzy c-Ellipsoidal Shells – Frigui, Krishnapuram (1996), Gath, Hoory (1995)
- Fuzzy c-Quadric Shells – Krishnapuram, Frigui, Nasraoui (1993, 1995)
- Fuzzy c-Rectangular Shells – Höppner, Klawonn, Kruse (1997)

Fuzzy clustering with incomplete data:

- Unsupervised fuzzy competitive learning – Chung, Lee (1994)
- Whole data strategy – Hathaway, Bezdek (2001)
- Partial distance strategy – Hathaway, Bezdek (2001)
- Optimal completion strategy – Hathaway, Bezdek (2001)
- Nearest neighbour strategy – Hathaway, Bezdek (2001)

Other methods and models:

- Fuzzy divisive hierarchic clustering – Dumitrescu (1988)
- Cross-clustering – Dumitrescu, Pop (1995)
- Noise Clustering – Dave (1991), Dave, Sen (1997)
- Possibilistic, Probabilistic – Krishnapuram, Keller (1993), Gath, Geva (1989)

5. Fuzzy classification

Let us consider a set of objects, $X = \{x^1, \ldots, x^p\} \in \mathbb{R}^d$, classified with a fuzzy clustering algorithm of the type Fuzzy n-Means, and the fuzzy partition $P = \{A_1, \ldots, A_n\}$ coresponding to the cluster substructure of the set X (see [4]).

We rise the problem of including an extra-object $x^0 \notin X$ in the cluster structure of X. Of course, this would mean to determine the membership degrees of x^0 to the fuzzy sets members of the partition P. These degrees will provide sufficient information in order to classify the object x^0 with respect to the elements of X.

The algorithms that solve this kind of problems are called *fuzzy decision supervized classification algorithms* [9]. Supervised classification because the classification of the extra-object is realized using not only the data set X, but also the fuzzy partition obtained by classifying the set X. Fuzzy decision because, unlike the traditional classifiers, where the aim is to state in which classical subset the object may be included, now we are interested in the membership degrees of the object to the fuzzy sets members in the given fuzzy partition.

This category of classifiers has to be studied in comparison with the other two important classes. On one hand, the *classical classifiers*, where the supervised information is a crisp partition and the final descision is, as well, crisp (i.e. the 'to be or not to be' kind of question). On the other hand, we have the traditional fuzzy generalisations of the crisp classifiers, the so-called *fuzzy classifiers*, where the supervised information is a fuzzy partition, but the final decision is, still, crisp.

The simplest approach for the fuzzy decision supervised classifier is the classification of the extended set $X \cup \{x^0\}$ using one of the common fuzzy clustering algorithms, and the comparative analysis of the produced partition to the partition P. Unfortunately, this method is very costly considering the neccessary execution time, because it supposes the classification of the objects of X, set that in the real applications may be quite large. Also, this is not supervised classification, because the information provided by the fuzzy partition P is not used.

The alternate approach is to keep unchanged the membership degrees of the objects in X to the sets of the fuzzy partiton P, and to determine the membership degreess of x^0 as a consequence of the minimization of an objective function similar to those used for the algorithms of the Fuzzy n-Means type.

There are, as well, a few more straightforward algorithms of this type. These algorithms are fuzzy generalizations of the well-known *k nearest neighbours* and *nearest prototype*.

6. Fuzzy regression

The aim of regression techniques is to relate, corelate or model a measure response based on the value of a given variable.

The common approach has been to work with traditional schemes as, for example, the linear Least-Square method. But these methods come with important drawbacks: they assume that data is homoscedastic – y-direction error is independent of the controlled variable.

Unfortunately, most real-world data are heteroscedastic, and presence of outliers is, as well, common. In such cases, use of robust methods is desired. But most of current robust methods do not provide best results in all situations.

We aim at developing a class of fuzzy regression methods that overcome these negative issues.

Fuzzy clustering techniques are suitable for determining the optimal cluster substructure of a data set, and they suppose that such a substructure does exist. The problem at hand is, however, to be able to determine the one fuzzy set A and its prototype L that best describes the data set. In such a case, a regular fuzzy clustering algorithm will not work.

The fuzzy set that best corresponds to a data set, based on a prototype characterisation of the data, is a useful notion in the search for robust regression techniques, as well as for developing data analysis techniques where the data items are considered according to their goodness of fit (i.e. their membership degree to this fuzzy set).

We consider a binary fuzzy partition, $\{A, \bar{A}\}$, where $\bar{A}$ is a virtual class with a hypothetical prototype, characterized by the constant dissimilarity

$$D(x^j, \bar{L}) = \delta := \left(\frac{\alpha}{1-\alpha}\right)^{m-1}.$$

The optimal fuzzy set A, as defined by our problem, is determined by minimizing the following fuzzy objective function:

$$J(A, L) = \sum_{j=1}^{n} A(x^j)^m D(x^j, L) + \sum_{j=1}^{n} \bar{A}(x^j)^m \left(\frac{\alpha}{1-\alpha}\right)^{m-1}, \alpha \in (0,1).$$

The algorithm used to solve this problem has been called the *Fuzzy Regression* generic algorithm [11, 12]:

(1) Given α; Initialize $A^{(0)}(x) = 1$, $l = 0$;
(2) Compute prototype L that minimizes $J(A^{(l)}, \cdot)$;
(3) Compute fuzzy set $A^{(l+1)}$ that minimizes $J(\cdot, L)$:

$$A^{(l+1)}(x^j) = \frac{\dfrac{\alpha}{1-\alpha}}{\dfrac{\alpha}{1-\alpha} + D(x^j, L)^{\frac{1}{m-1}}}$$

(4) Compare fuzzy sets $A^{(l+1)}$ with $A^{(l)}$. If close enough, then STOP, else increase l by 1 and GOTO STEP 2.

As an improvement, in order to assure the independence of scale, in the equation from STEP 3, we will replace the dissimilarity $D(x^j, L)$ with the relative dissimilarity

$$D_r(x^j, L) = D(x^j, L) / \max_{j=1,n} D(x^j, L).$$

This is equivalent to setting δ initially to

$$D(x^j, \bar{L}) = \delta := \left(\frac{\alpha}{1-\alpha}\right)^{m-1} \max_{j=1,n} D(x^j, L).$$

6.1. **Properties of the Fuzzy Regression algorithm.** Let us suppose that X is a data set, and A and L are the optimal fuzzy set and its prototype representation, respectively. The following properties are valid [11].

i. *(Maximal membership degree)* $A(x) = 1 \Leftrightarrow D(x, L) = 0$
ii. *(Minimal membership degree)* $A(x) = \alpha \Leftrightarrow D_r(x, L) = 1$
iii. *(Membership degree interval)* $A(x) \in [\alpha, 1]$ for all $x \in X$

iv. *(Empty fuzzy set)* $\alpha = 0 \Leftrightarrow A(x) = 0$ for all $x \in X$
v. *(Degenerate fuzzy set)* $\alpha = 1 \Leftrightarrow A(x) = 1$ for all $x \in X$
vi. *(Strict monotony)* $A(x) < A(y) \Leftrightarrow D(x, L) < D(y, L)$
vii. *(Equality)* $A(x) = A(y) \Leftrightarrow D(x, L) = D(y, L)$

The constant α is an input parameter that has the role of setting the polarization of the fuzzy partition $\{A, \bar{A}\}$. The best results appear to be obtained with $\alpha \approx 0.10$.

Being based on the Fuzzy c-Lines algorithm, the linear version of this algorithm (Fuzzy Linear Regression) uses as dissimilarity the square distance to the line, as opposed to the y-distance, used by the classical Least Squares algorithm.

Due to the use of fuzzy sets, the algorithm is efficient in all testing conditions, and is better than most methods it has been compared with [11].

The algorithm allows the detection of the type of data sets, i.e. homoscedasticity, heteroscedasticity, presence of outliers, or any combination thereof, with or without any other irregularities. This is done through repeated runs by analysing the graphical representation of the surface made by the coefficients vectors of the linear prototypes, as determined for α varied continuously in the interval $(0, 1)$. In the case of a two-dimensional data set, the curve is defined through the points (a_0, a_1), where $y = a_0 + a_1 x$ is the linear prototype of the data set defined.

7. Data projection and selection

7.1. **Visualisation of high-dimensional data items.**

The simplest method to visualise a data set is to plot profiles: two-dimensional graphs where the dimensions are enumerated on the x axis, and the corresponding values on y. An alternative, also largely used, is to plot two-dimensional representations of pairs of two original dimensions.

There are, also, methods that produce different curves based on the data items values. The most important drawback of such methods is that they do not reduce the amount of data, and thus they cannot be used effectively with large high-dimensional data sets. However, they can be used for illustrating data summaries.

7.2. **Projection methods.**

The main feature of clustering algorithms is that they reduce the amount of data items by grouping them in classes. The projection methods described below can be used for reducing the data dimensionality. The goal of these techniques is to represent the data set in a lower-dimensional space in such a way that certain properties of the data set are preserved as much as possible.

The following questions are important when discussing a method for large, high-dimensional data sets: what kind of structure the method extracts from the data set; how does it illustrate the structure; does it reduce dimensionality of data; does it reduce the number of data items.

7.3. **Principal Components Analysis with Fuzzy sets.** The aim of PCA is to achieve data dimensionality reduction by determining new, fewer variables. The new variables, called principal components, correspond to the axes of maximal elongation of data These principal components are linear combinations of the original variables.

It has been remarked that the number of principal components necessary to conserve 90% of data variance is considerably less than the size of data space. As such, it has become extremely important to determine the relevant variables. We could thus achieve not only a significant data projection, but as well a variable selection.

The use of fuzzy sets enables us to aproach the problem of isolated points with respect both to the data set and to the principal directions [2].

The PCA Algorithm. The PCA algorithm starts by computing the covariance or corelation matrix

$$\mathrm{Cov}_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(x_i^k - \bar{x}_i)\cdot(x_j^k - \bar{x}_j)$$

$$\mathrm{Cor}_{ij} = \frac{\mathrm{Cov}_{ij}}{s_i \cdot s_j}, s_i = \frac{1}{n-1}\sum_{k=1}^{n}(x_i^k - \bar{x}_i)^2$$

Then we compute the eigenvectors and eigenvalues of this matrix; these are the principal components and the scatter values. Based on these scatter values, select the necessary number of principal components.

Then we determine the values of data for the new variables (i.e. project the data set in the space of the selected principal components): $X'^T = X^T \cdot E$.

This is a very simple, but very effective algorithm indeed. Essentially, we achieve a clearer image about the data by only rotating the data space such that the new axes of coordinates coincide with the directions of maximal elongation of data.

Fuzzy PCA, first component. The major problem of the PCA algorithm rests, as always, with the isolated points. As a first possible way to handle this, we will take into account the points isolated with respect to the first principal component only.

We aim to introduce fuzzy membership degrees according to the distance to the first principal component. As such, we will use a scheme similar to the fuzzy regression algorithm, to determine the first fuzzy principal component and the corresponding fuzzy membership degrees.

We will thus replace the traditional covariance matrix by the fuzzy covariance matrix, given by

$$C_{ij} = \frac{\sum_{k=1}^{n} A(x^k)^m \cdot (x_i^k - \bar{x}_i)\cdot(x_j^k - \bar{x}_j)}{\sum_{k=1}^{n} A(x^k)^m}, \ i,j = 1,\ldots,p.$$

The major advantage is that the first principal component will count the merits of each data item; as such, will consider the isolated points with less significance.

Fuzzy PCA, orthogonal. The Fuzzy PCA algorithm discussed in the preceding section fuzzifies only the first component. In order to get a most effective method, we have to deal with the problem of fuzzifying all the components.

The main idea is to use a different approach: by projecting the data in smaller-sized spaces. After the first fuzzy eigenvector is determined, all data is projected to the hyperplane rectangular on it. The eigenvectors corresponding to the projected data will be orthogonal to the eigenvector determined above. As such, the second largest eigenvector of the original data will correspond to the largest eigenvector of the projected data. The projection continues further on, etc.; finally, the eigenvectors are rebuilt in the original space.

This method Advantage: the compotation of the other fuzzy components is reduced to the computation of the first fuzzy component of a smaller-sized matrix

7.4. **Multi-Dimensional Scaling.** Alternate projection methods aim to reduce the data dimensionality is to optimize the representation in the lower-dimension space so that the distances between points in the projected space are as similar as possible to the distances between the corresponding points in the original space [7].

We will present here a class of methods known as *multidimensional scaling (MDS)*. The aim of these methods is to project data from a pseudo-metric space (i.e. characterised by a dissimilarity measure) onto a metric space. Such metods are especially useful for preprocessing non-metric data in order to use them with algorithms only valid with metric input.

The first MDS method is the *metric MDS*, characterized by minimizing the squared error cost function:

$$E_M = \sum_{k \neq l} \left(d(k,l) - d'(k,l) \right)^2 ,$$

where, for the original items x_k and x_l, $d(k,l)$ is their dissimilarity, and $d'(k,l)$ is the distance between the corresponding vectors from the projected metric space.

If the components of the data vectors are expressed on an ordinal scale, a perfect reproduction of the Euclidean distances may not be the best goal. In such a situation, only the rank order of the distances between the vectors is meaningful. The error function is defined as

$$E_N = \frac{\sum\limits_{k \neq l} \left(f(d(k,l)) - d'(k,l) \right)^2}{\sum\limits_{k \neq l} \left(d'(k,l) \right)^2},$$

where f is a monotonically increasing function that acts on the original distances and always maps the distances to such values that best preserve the rank order.

Another non-linear mapping method, the *Sammon's mapping*, is closely related to the metric MDS. The only difference is that the errors in distance preservation are normalized with the distance in the original space. Thus, preservation of small distances is emphasized. The error function is defined as

$$E_S = \sum_{k \neq l} \frac{\left(d(k,l) - d'(k,l)\right)^2}{d(k,l)}.$$

8. Conclusions

This paper surveyed the different intelligent data analysis aspects, with an insight from the fuzzy sets perspective. The fuzzy sets, as a natural generalisation of the classical crisp sets, bring more power and refinement to data analysis. A lot of work has been done in developping better, more robust algorithms for data analysis. The different fuzzy data analysis algorithms are not only an issue for theoretical study, but a series of effective tools used in most diverse applications, from medicine to chemistry, physics, biology and engineering.

References

[1] James C. Bezdek, *Pattern Recongnition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981

[2] Thomas R. Cundari, Costel Sârbu, Horia F. Pop, Robust Fuzzy Principal Component Analysis (FPCA). A comparative study concerning interaction of carbon-hydrogen bonds with molybdenum-oxo bonds. *J. Chem. Inf. Comput. Sci.*, 42, 6, 2002, 1363–1369

[3] J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *Journal of Cybernetics* 3, 1974, 32–57

[4] D. Dumitrescu, Hierarchical pattern classification, *Fuzzy Sets Syst.*, 28, 1988, 145–162

[5] F. Höppner, F. Klawonn, R. Kruse, T. Runkler, *Fuzzy Cluster Analysis*, Wiley, New York, 1999

[6] George J. Klir, Bo Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, New Jersey, 1995

[7] J. B. Kruskal, M. Wish, *Multidimensional Scaling*, Sage Publications, Beverly Hills, CA, 1977

[8] Zdzislaw Pawlak, Rough sets, *Int. J. Comput. Inform. Sci.*, 11, 1982, 341–356

[9] Horia F. Pop, Fuzzy decision supervised classifiers, *Studia Universitatis Babes-Bolyai, Series Informatica*, 40, 3, 1995, 89–100

[10] Horia F. Pop, *Sisteme inteligente în probleme de clasificare*, Editura Mediamira, Cluj, 2004

[11] Horia F. Pop, Costel Sârbu, A New Fuzzy Regression Algorithm, *Anal. Chem.* 68, 1996, 771–778

[12] Costel Sârbu, Horia F. Pop, Fuzzy robust estimation of central location, *Talanta*, 54, 2001, 125–130

[13] Lotfi A. Zadeh, Fuzzy sets, *Inf. Control*, 8, 1965, 338–353

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., RO-400084 Cluj-Napoca, Romania

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