Anul XLVII 2002

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

2

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

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

A WEIGHTED-PATH-FOLLOWING METHOD FOR LINEAR OPTIMIZATION

ZSOLT DARVAY

Abstract. In a recent paper [4] we introduced a new method for finding search directions for interior point methods (IPMs) in linear optimization (LO), and we developed a new polynomial algorithm for solving LO problems. It is well-known that using the self-dual embedding we can find a starting feasible solution, and this point will be on the central path. We proved [3] that this initialization method can be applied for the new algorithm as well. However, practical implementations often don't use perfectly centered starting points. Therefore it is worth analysing the case when the starting point is not on the central path. In this paper we develop a new weighted-path-following algorithm for solving LO problems. We conclude that following the central path yields to the best iteration bound in this case as well.

1. Introduction

In this paper we discuss a generalized form of path-following IPMs. The field of IPMs is an active research area, since Karmarkar [8] has developed the first IPM in 1984. For a survey of results see the following books [1, 2, 6, 11, 13, 14]. In this paper we generalize the algorithm introduced in [4], and we develop a new weighted-path-following algorithm. It is well known that with every algorithm which follows the central path we can associate a target sequence on the central path. This observation led to the concept of *target-following* methods introduced by Jansen et al. [7]. A survey of target-following algorithms can be found in [11] and [6]. Weighted-path-following methods can be viewed as a particular case of target-following methods. These methods were studied by Ding and Li [5] for primal-dual linear complementarity problems, and by Roos and den Hertog [10] for primal problems. In this paper we consider the LO problem in the following

2000 *Mathematics Subject Classification.* 90C05.

1998 *CR Categories and Descriptors.* G.1.6. [**Mathematics of Computing**]: Numerical Analysis – *Linear programming.*

standard form

$$(P) \qquad \begin{aligned} &min\ c^T x \\ s.t.\quad &Ax = b, \\ &x \geq 0, \end{aligned}$$

where $A \in \Re^{m\times n}$ with $rank(A) = m$, $b \in \Re^m$ and $c \in \Re^n$. The dual of this problem can be written in the following form

$$(D) \qquad \begin{aligned} &max\ b^T y \\ s.t.\quad &A^T y + s = c, \\ &s \geq 0. \end{aligned}$$

We assume that the *interior point condition* (IPC) holds for these probelms.

Assumption 1 (Interior point condition). There exist (x^0, y^0, s^0) such that

$$\begin{aligned} Ax^0 &= b, \qquad x^0 > 0, \\ A^T y^0 + s^0 &= c, \qquad s^0 > 0. \end{aligned}$$

Using the self-dual embedding method a larger LO problem can be constructed in such a way that the IPC holds for that problem. Hence, the IPC can be assumed without loss of generality. Finding the optimal solutions of both the original problem and its dual, is equivalent to solving the following system

$$(1) \qquad \begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ xs &= 0, \end{aligned}$$

where xs denotes the coordinatewise product of the vectors x and s, hence

$$xs = [x_1 s_1, x_2 s_2, \ldots, x_n s_n]^T.$$

We mention that in this paper for an arbitrary function f, and an arbitrary vector x we will use the notation

$$f(x) = [f(x_1), f(x_2), \ldots, f(x_n)]^T.$$

The first and the second equations of system (1) serve for maintaining feasibility, hence we call them the *feasibility conditions*. The last relation is the *complementarity condition*, which in IPMs is generally replaced by a parameterized equation, thus we obtain

$$(2) \qquad \begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ xs &= \mu e, \end{aligned}$$

where $\mu > 0$, and e is the n-dimensional all-one vector, hence $e = [1, 1, \ldots, 1]^T$. If the IPC is satisfied, then for a fixed $\mu > 0$ the system (2) has a unique solution.

This solution is called the μ-*center* (Sonnevend [12]), and the set of μ-centers for $\mu > 0$ formes the *central path*. The target-following approach starts from the observation that the system (2) can be generalized by replacing the vector μe with an arbitrary positive vector w^2. Thus we obtain the following system

$$\begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ xs &= w^2, \end{aligned} \tag{3}$$

where $w > 0$. If the IPC holds then the system (3) has a unique solution. This feature was first proved by Kojima et al. [9]. Hence we can apply Newton's method for the system (3) to develop a primal-dual target-following algorithm. In the following section we present a new method for finding search directions by applying Newton's method for an equivalent form of system (3).

2. New Search-Directions

In this section we introduce a new method for constructing search directions by using the system (3). Let $\Re^+ = \{x \in \Re \mid x \geq 0\}$, and consider the function

$$\varphi \in C^1, \qquad \varphi : \Re^+ \to \Re^+.$$

Furthermore, suppose that the inverse function φ^{-1} exists. Then, the system (3) can be written in the following equivalent form

$$\begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ \varphi(xs) &= \varphi(w^2), \end{aligned} \tag{4}$$

and we can apply Newton's method for the system (4) to obtain a new class of search directions. We mention that a direct generalization of the approach defined in [4] would be the following variant. The system (3) is equivalent to

$$\begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ \varphi\left(\frac{xs}{w^2}\right) &= \varphi(e), \end{aligned} \tag{5}$$

and using Newton's method for the system (5) yields new search directions. For our purpose it is more convenient the first approach, hence in this paper we use the system (4). Let us introduce the vectors

$$v = \sqrt{xs} \quad \text{and} \quad d = \sqrt{xs^{-1}},$$

and observe that these notations lead to

$$d^{-1}x = ds = v. \tag{6}$$

Suppose that we have $Ax = b$, and $A^T y + s = c$ for a triple (x, y, s) such that $x > 0$ and $s > 0$, hence x and s are strictly feasible. Applying Newton's method for the system (4) we obtain

$$\begin{aligned} A\Delta x &= 0, \\ A^T \Delta y + \Delta s &= 0, \\ s\varphi\prime(xs)\,\Delta x + x\varphi\prime(xs)\,\Delta s &= \varphi(w^2) - \varphi(xs)\,. \end{aligned} \tag{7}$$

Furthermore, denote

$$d_x = d^{-1}\Delta x, \qquad d_s = d\Delta s,$$

and observe that we have

$$v(d_x + d_s) = s\Delta x + x\Delta s, \tag{8}$$

and

$$d_x d_s = \Delta x \Delta s. \tag{9}$$

Hence the linear system (7) can be written in the following equivalent form

$$\begin{aligned} \bar{A} d_x &= 0, \\ \bar{A}^T \Delta y + d_s &= 0, \\ d_x + d_s &= p_v, \end{aligned} \tag{10}$$

where

$$p_v = \frac{\varphi(w^2) - \varphi(v^2)}{v\varphi\prime(v^2)}, \tag{11}$$

and $\bar{A} = Adiag(d)$. We also used the notation

$$diag(\xi) = \begin{bmatrix} \xi_1 & 0 & \dots & 0 \\ 0 & \xi_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \xi_n \end{bmatrix},$$

for any vector ξ. In the following section we will develop a new primal-dual weighted-path-following algorithm based on one particular search direction.

3. The Algorithm

In this section we let $\varphi(x) = \sqrt{x}$, and we develop a new primal-dual weighted-path-following algorithm based on the appropriate search directions. Thus, making the substitution $\varphi(x) = \sqrt{x}$ in (11) we get

$$p_v = 2(w - v). \tag{12}$$

Now for any positive vector v, we define the folowing proximity measure

$$\sigma(v, w) = \frac{\|p_v\|}{2\min(w)} = \frac{\|w - v\|}{\min(w)}, \tag{13}$$

where $\|\cdot\|$ is the Euclidean norm (l_2 norm), and for every vector ξ we denote $\min(\xi) = \min\{\xi_i \mid 1 \leq i \leq n\}$. We introduce another measure

$$\sigma_c(w) = \frac{max(w^2)}{min(w^2)},$$

where for any vector ξ we denote $\max(\xi) = \max\{\xi_i \mid 1 \leq i \leq n\}$. Observe that $\sigma_c(w)$ can be used to measure the distance of w^2 to the central path. Furthermore, let us introduce the notation

$$q_v = d_x - d_s,$$

observe that from (10) we get $d_x^T d_s = 0$, hence the vectors d_x and d_s are orthogonal, and thus we find that

$$\|p_v\| = \|q_v\|.$$

Consequently, the proximity measure can be written in the following form

$$\sigma(v, w) = \frac{\|q_v\|}{2\min(w)}, \tag{14}$$

thus we obtain

$$d_x = \frac{p_v + q_v}{2}, \qquad d_s = \frac{p_v - q_v}{2},$$

and

$$d_x d_s = \frac{p_v^2 - q_v^2}{4}. \tag{15}$$

Making the substitution $\varphi(x) = \sqrt{x}$ in (7) yields

$$\begin{aligned} A\Delta x &= 0, \\ A^T\Delta y + \Delta s &= 0, \\ \sqrt{\frac{s}{x}}\Delta x + \sqrt{\frac{x}{s}}\Delta s &= 2(w - \sqrt{xs}). \end{aligned} \tag{16}$$

Now we can define the algorithm.

Algorithm 3.1 *Suppose that for the triple (x^0, y^0, s^0) the interior point condition holds, and let $w^0 = \sqrt{x^0 s^0}$. Let $\epsilon > 0$ be the accuracy parameter, and $0 < \theta < 1$ the update parameter (default $\theta = \frac{1}{5\sqrt{\sigma_c(w^0)n}}$),*

begin
  $x := x^0;\ y := y^0;\ s := s^0;$
  $w := w^0;$
  while $x^T s > \epsilon$ **do begin**
    $w := (1 - \theta)w;$
    Compute $(\Delta x, \Delta y, \Delta s)$ from (16)
    $x := x + \Delta x;$
    $y := y + \Delta y;$
    $s := s + \Delta s;$

end
end.

In the next section we shall prove that this algorithm is well defined for the default value of θ, and we will also give an upper bound for the number of iterations performed by the algorithm.

4. Convergence Analysis

In the first lemma of this section we prove that if the proximity measure is small enough, then the Newton process is strictly feasible. Denote $x_+ = x + \Delta x$ and $s_+ = s + \Delta s$ the vectors obtained by a full Newton step, and let $v = \sqrt{xs}$ as usual.

Lemma 4.1 *Let $\sigma = \sigma(v, w) < 1$. Then the full Newton step is strictly feasible, hence*

$$x_+ > 0 \quad \textit{and} \quad s_+ > 0.$$

Proof: For every $0 \leq \alpha \leq 1$ let $x_+(\alpha) = x + \alpha\Delta x$ and $s_+(\alpha) = s + \alpha\Delta s$. Hence

$$x_+(\alpha)s_+(\alpha) = xs + \alpha(s\Delta x + x\Delta s) + \alpha^2\Delta x\Delta s$$

Now using (8) and (9) we find that

$$x_+(\alpha)s_+(\alpha) = v^2 + \alpha v(d_x + d_s) + \alpha^2 d_x d_s,$$

and from (10) and (15) we obtain

$$x_+(\alpha)s_+(\alpha) = (1-\alpha)v^2 + \alpha(v^2 + vp_v) + \alpha^2\left(\frac{p_v^2}{4} - \frac{q_v^2}{4}\right).$$

Moreover (12) yields

$$v + \frac{p_v}{2} = w,$$

and thus

$$v^2 + vp_v = w^2 - \frac{p_v^2}{4}.$$

Consequently

$$x_+(\alpha)s_+(\alpha) = (1-\alpha)v^2 + \alpha\left(w^2 - (1-\alpha)\frac{p_v^2}{4} - \alpha\frac{q_v^2}{4}\right), \tag{17}$$

thus the inequality $x_+(\alpha)s_+(\alpha) > 0$ certainly holds if

$$\left\|(1-\alpha)\frac{p_v^2}{4} + \alpha\frac{q_v^2}{4}\right\|_\infty < min(w^2),$$

where $\|\cdot\|_\infty$ denotes the Chebychev norm (l_∞ norm). Using (13) and (14) we get

$$\left\|(1-\alpha)\frac{p_v^2}{4}+\alpha\frac{q_v^2}{4}\right\|_\infty \le (1-\alpha)\frac{\|p_v^2\|_\infty}{4}+\alpha\frac{\|q_v^2\|_\infty}{4} \le$$

$$\le (1-\alpha)\frac{\|p_v\|^2}{4}+\alpha\frac{\|q_v\|^2}{4} = \sigma^2\min(w^2) < min(w^2).$$

Hence, for any $0 \le \alpha \le 1$ we have $x_+(\alpha)s_+(\alpha) > 0$. As a consequence we observe that the linear functions of α, $x_+(\alpha)$ and $s_+(\alpha)$ do not change sign on the interval $[0,1]$. For $\alpha = 0$ we have $x_+(0) = x > 0$ and $s_+(0) = s > 0$ thus we obtain $x_+(1) = x_+ > 0$ and $s_+(1) = s_+ > 0$, and this implies the lemma. ■

In the next lemma we prove that the same condition, namely $\sigma < 1$ is sufficient for the quadratic convergence of the Newton process.

Lemma 4.2 *Let $x_+ = x + \Delta x$ and $s_+ = s + \Delta s$ be the vectors obtaind after a full Newton step, $v = \sqrt{xs}$ and $v_+ = \sqrt{x_+ s_+}$. Suppose $\sigma = \sigma(v,w) < 1$. Then*

$$\sigma(v_+, w) \le \frac{\sigma^2}{1+\sqrt{1-\sigma^2}}.$$

Thus $\sigma(v_+, w) < \sigma^2$, which means quadratic convergence of the Newton step.

Proof: From Lemma 4.1 we get $x_+ > 0$ and $s_+ > 0$. Now substitute $\alpha = 1$ in (17) and get

$$v_+^2 = w^2 - \frac{q_v^2}{4}. \tag{18}$$

Using (18) we obtain

$$\min(v_+^2) \ge \min(w^2) - \frac{\|q_v^2\|_\infty}{4} \ge \min(w^2) - \frac{\|q_v\|^2}{4} = \min(w^2)(1-\sigma^2),$$

and this realtion yields

$$\min(v_+) \ge \min(w)\sqrt{1-\sigma^2}. \tag{19}$$

Furthermore, from (18) and (19) we get

$$\sigma(v_+, w) = \frac{1}{\min(w)}\left\|\frac{w^2 - v_+^2}{w + v_+}\right\| \le \frac{\|w^2 - v_+^2\|}{\min(w)\,(\min(w) + \min(v_+))} \le$$

$$\le \frac{\|q_v^2\|}{(2\min(w))^2\left(1+\sqrt{1-\sigma^2}\right)} \le \frac{1}{1+\sqrt{1-\sigma^2}}\left(\frac{\|q_v\|}{2\min(w)}\right)^2 = \frac{\sigma^2}{1+\sqrt{1-\sigma^2}}.$$

Consequently, we have $\sigma(v_+, w) < \sigma^2$, and this implies the lemma. ■

In the following lemma we give an upper bound for the duality gap obtained after a full Newton step.

Lemma 4.3 *Let $\sigma = \sigma(v, w)$. Moreover, let $x_+ = x + \Delta x$ and $s_+ = s + \Delta s$. Then*

$$(x_+)^T s_+ = \|w\|^2 - \frac{\|q_v\|^2}{4},$$

hence $(x_+)^T s_+ \leq \|w\|^2$.

Proof: From

$$x_+ s_+ = w^2 - \frac{q_v^2}{4},$$

we obtain

$$(x_+)^T s_+ = e^T(x_+ s_+) = e^T w^2 - \frac{e^T q_v^2}{4} = \|w\|^2 - \frac{\|q_v\|^2}{4},$$

and this proves the lemma. ■

In the following lemma we discuss the influence on the proximity measure of the Newton process followed by a step along the weighted-path. We assume that each component of the vector w will be reduced by a constant factor $1 - \theta$.

Lemma 4.4 *Let $\sigma = \sigma(v, w) < 1$ and $w_+ = (1 - \theta)w$, where $0 < \theta < 1$. Then*

$$\sigma(v_+, w_+) \leq \frac{\theta}{1 - \theta}\sqrt{\sigma_c(w)n} + \frac{1}{1 - \theta}\sigma(v_+, w).$$

Furthermore, if $\sigma \leq \frac{1}{2}$, $\theta = \frac{1}{5\sqrt{\sigma_c(w)n}}$ and $n \geq 4$ then we get $\sigma(v_+, w_+) \leq \frac{1}{2}$.

Proof: We have

$$\sigma(v_+, w_+) = \frac{1}{\min(w_+)}\|w_+ - v_+\| \leq \frac{1}{\min(w_+)}\|w_+ - w\| + \frac{1}{\min(w_+)}\|w - v_+\| =$$

$$= \frac{1}{(1 - \theta)min(w)}\|\theta w\| + \frac{1}{1 - \theta}\sigma(v_+, w) \leq \frac{\theta}{1 - \theta}\sqrt{\sigma_c(w)n} + \frac{1}{1 - \theta}\sigma(v_+, w).$$

Thus the first part of the lemma is proved. Now let $\theta = \frac{1}{5\sqrt{\sigma_c(w)n}}$, observe that $\sigma_c(w) \geq 1$, and for $n \geq 4$ we obtain $\theta \leq \frac{1}{10}$. Furthermore, if $\sigma \leq \frac{1}{2}$ then from Lemma 4.2 we deduce $\sigma(v_+, w) \leq \frac{1}{4}$. Finally, the above relations yield $\sigma(v_+, w_+) \leq \frac{1}{2}$. The proof of the lemma is complete. ■

Observe that $\sigma_c(w) = \sigma_c(w^0)$ for all iterates produced by the algorithm. Thus, an immediate result of Lemma 4.4 is that for $\theta = \frac{1}{5\sqrt{\sigma_c(w^0)n}}$ the conditions $(x, s) > 0$ and $\sigma(v, w) \leq \frac{1}{2}$ are maintained throughout the algorithm. Hence the algorithm is well defined. In the next lemma we calculate an upper bound for the total number of iterations performed by the algorithm.

Lemma 4.5 *Assume that x^0 and s^0 are strictly feasible, an let $w^0 = \sqrt{x^0 s^0}$. Moreover, let x^k and s^k be the vectors obtained after k iterations. Then the inequality $(x^k)^T s^k \leq \epsilon$ is satisfied for*

$$k \geq \left\lceil \frac{1}{2\theta} \log \frac{(x^0)^T s^0}{\epsilon} \right\rceil.$$

Proof: After k iterations we get $w = (1-\theta)^k w^0$. Using Lemma 4.3 we find that

$$(x^k)^T s^k \leq \|w\|^2 = (1-\theta)^{2k} \|w^0\|^2 = (1-\theta)^{2k} (x^0)^T s^0,$$

hence $(x^k)^T s^k \leq \epsilon$ holds if

$$(1-\theta)^{2k} (x^0)^T s^0 \leq \epsilon.$$

Taking logarithms, we obtain

$$2k \log(1-\theta) + \log((x^0)^T s^0) \leq \log \epsilon.$$

Using the inequality $-\log(1-\theta) \geq \theta$ we deduce that the above relation holds if

$$2k\theta \geq \log((x^0)^T s^0) - \log \epsilon = \log \frac{(x^0)^T s^0}{\epsilon}.$$

The proof is complete. ■

For the default value of θ specified in Algorithm 3.1 we obtain the following theorem.

Theorem 4.6 *Suppose that the pair (x^0, s^0) is strictly feasible, an let $w^0 = \sqrt{x^0 s^0}$. If $\theta = \frac{1}{5\sqrt{\sigma_c(w^0)n}}$ then Algorithm 3.1 requires at most*

$$\left\lceil \frac{5}{2} \sqrt{\sigma_c(w^0)n} \log \frac{(x^0)^T s^0}{\epsilon} \right\rceil$$

iterations. For the resulting vectors we have $x^T s \leq \epsilon$. ■

5. Conclusion

In this paper we have developed a new weighted-path-following algorithm for solving LO problems. Our approach is a generalization of [4] for weighted-paths. We have transformed the system (3) in an equivalent form by introducing a function φ. We have defined a new class of search directions by applying Newton's method for that form of the weighted-path. Using $\varphi(x) = \sqrt{x}$ we have developed a new primal-dual weighted-path-following algorithm, and we have proved that this algorithm performs no more than

$$\left\lceil \frac{5}{2} \sqrt{\sigma_c(w^0)n} \log \frac{(x^0)^T s^0}{\epsilon} \right\rceil$$

iterations. Observe, that this means that the best bound is obtained by following the central path. Indeed, we have $\sigma_c(w^0) = 1$ in this case, and we get the well-known iteration bound

$$O\left(\sqrt{n}\log\frac{(x^0)^T s^0}{\epsilon}\right).$$

If the starting point is not perfectly centered, then $\sigma_c(w^0) > 1$ and thus the iteration bound is worse.

References

[1] N. Andrei. *Mathematical Programming. Interior Point Methods*. Editura Tehnică, Bucureşti, 1999. (In Romanian).

[2] Zs. Darvay. *Interior Point Methods in Linear Programming*. ELTE, Budapest, 1997. (In Hungarian).

[3] Zs. Darvay. A new algorithm for solving self-dual linear optimization problems. *Studia Universitatis Babeş-Bolyai, Series Informatica*, 47(1):15–26, 2002.

[4] Zs. Darvay. A new class of search directions for linear optimization. In *Proceedings of Abstracts, McMaster Optimizations Conference: Theory and Applications held at McMaster University Hamilton, Ontario, Canada*, page 18, August 1-3, 2002. Submitted to *European Journal of Operational Research*.

[5] J. Ding and T.Y. Li. An algorithm based on weighted logarithmic barrier functions for linear complementarity problems. *Arabian Journal for Science and Engineering*, 15(4):679–685, 1990.

[6] B. Jansen. *Interior Point Techniques in Optimization. Complexity, Sensitivity and Algorithms*. Kluwer Academic Pubishers, 1997.

[7] B. Jansen, C. Roos, T. Terlaky, and J.-Ph. Vial. Primal-dual target-following algorithms for linear programming. *Annals of Operations Research*, 62:197–231, 1996.

[8] N.K. Karmarkar. A new polinomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[9] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise. *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*, volume 538 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Germany, 1991.

[10] C. Roos and D. den Hertog. A polinomial method of approximate weighted centers fo linear programming. Technical Report 89-13, Faculty of Technical Mathematics and Informatics, TU Delft, NL-2628 BL Delft, The Netherlands, 1994.

[11] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization. An Interior Approach*. John Wiley & Sons, Chichester, UK, 1997.

[12] Gy. Sonnevend. An "analytic center" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. In A. Prékopa, J. Szelezsán, and B. Strazicky, editors, *System Modelling and Optimization: Proceedings of the 12th IFIP-Conference held in Budapest, Hungay, September 1985*, volume 84 of *Lecture Notes in Control and Information Sciences*, pages 866–876. Springer Verlag, Berlin, West-Germany, 1986.

[13] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, USA, 1997.

[14] Y. Ye. *Interior Point Algorithms, Theory and Analysis*. John Wiley & Sons, Chichester, UK, 1997.

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., RO-3400 Cluj-Napoca, Romania

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

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

LASG - A LOGIC ARCHITECTURE FOR INTELLIGENT AGENTS

GABRIELA ŞERBAN

Abstract. It is well-known that one of the concrete architectures for intelligent agents is the logic one. In our opinion, the symbolic representations for the intelligent behavior are very important, and the logic approach is elegant and has a pure semantic. The aim of this paper is to present a new logic architecture for intelligent agents (LASG - a Logic Architecture based on Stacks of Goals). This architecture combines the traditional logic architecture with a planning architecture [3]. The advantages of the proposed architecture are shown in the paper.
Keywords: intelligent agents, logic.

1. Introduction

The logic approach is a topic of Symbolic Artificial Intelligence and has its own importance in the field of intelligent agents, even if it is well-known the controversy between the traditional approach and the intelligent calculus in the field of Artificial Intelligence.

Moreover, the only intelligence requirement we generally make for the agents is that [2] they can make an acceptable decision about what action to perform next in their environment, in time for this decision to be useful. Other requirements for intelligence will be determined by the domain in which the agent is applied: not all agents will need to be capable of learning, for example.

In such situations, a logic architecture is very appropriate, and offers, in our opinion, a simple and elegant representation for the agent's environment and desired behavior.

According to the traditional approach [2], the symbolic representations are *logical formulae*, and the syntactic manipulation corresponds to *logical deduction*, or theorem proving.

In such a logic approach, the agent could be considered as a *theorem prover*(if ϕ is a theory that explains how an intelligent agent should behave, the system might generate a sequence of steps - actions - that leads to ϕ, in fact a proof for ϕ).

2000 *Mathematics Subject Classification.* 68T27.

1998 *CR Categories and Descriptors.* I.2[**Computing Methodologies**]: Artificial Intelligence – *Logic.*

However, some disadvantages of the logic approach are:

- the computational complexity of a theorem proving process raises the problem if the agents represented this way can really operate in time-restricted environments;
- the process of decision making in such logic architectures is based on the assumption that the environment does not change its structure, essentially, during the decision process (a decision that is correct at the beginning of the process, will be correct at the end of it, too);
- the problem of representation and reasoning in complex and dynamic environments is an open problem, as well.

2. A Logic Architecture Based on Stacks of Goals (LASG)

We will consider, in the following, the case of an agent which goal is to solve a given problem (to bring the problem from an initial to a final state), based on a set of operators (rules) that could be applied on a given moment [4].

In a LASG architecture, we will use the *declarative representation* of the knowledge.

Let L be a set of sentences from the first-order logic, and $D = \mathscr{P}(L)$ the set of *L-databases* (the set of sets of *L-formulae*). In the model that we propose, the *internal state* of the agent will be given by an element from D (for simplicity, we will consider it as a formula in a conjunctive normal form).

2.1. **Case Study: Searching a maze.**

In this section we will consider the following problem: we have a maze that has a rectangular form; in some positions there are obstacles; a robotic agent starts in a given state (the initial state) and tries to reach a final (goal) state, avoiding the obstacles; in a certain position on the maze the agent could move in four directions: north, south, east, west (there are four possible actions). We will assume that the dimensions of the maze are known: **M** is the number of rows, **N** is the number of columns.

In the example that we choose, the environment (the maze) is not dynamic (it suffers no modifications after the agent's actions). However, this assumption is not essential, it has no significant influence on the agent's behavior.

We consider that:

- a position on the maze is identified by a pair (X, Y) (the line, respectively the column);
- the left up corner of the maze is marked as the position (1, 1).

The four actions that the robotic agent could execute are the following:

NORTH(X, Y, M, N) - from the position (X, Y) the robot moves in the north direction. The positions (X, Y) and (X-1, Y) must be into the maze and must not contain obstacles.

EAST(X, Y, M, N) - from the position (X, Y) the robot moves in the east direction. The positions (X, Y) and (X, Y+1) must be into the maze and must not contain obstacles.

SOUTH(X, Y, M, N) - from the position (X, Y) the robot moves in the south direction. The positions (X, Y) and (X+1, Y) must be into the maze and must not contain obstacles.

WEST(X, Y, M, N) - from the position (X, Y) the robot moves in the west direction. The positions (X, Y) and (X, Y-1) must be into the maze and must not contain obstacles.

In order to specify both the conditions in which the operations hold and the results of executing the operations, we will use the following predicates:

FREE(X, Y) - the position (X, Y) is *free* (does not contain an obstacle).
IN(X, Y) - the robotic agent is in the position (X, Y).
VALID(X, Y) - the position (X, Y) is *valid* (is into the maze).
POSSIBLE(X, Y) - the position (X, Y) is *free* and *valid*.

We notice that:

$$POSSIBLE(X,Y) <=> FREE(X,Y) and VALID(X,Y) \tag{1}$$

In such a logic representation, some logic declarations are valid. For example,

$$not\, FREE(X,Y) and VALID(X,Y) \rightarrow not\, IN(X,Y) \tag{2}$$

$$not\, FREE(X,Y)\, or\, not\, VALID(X,Y) \rightarrow not\, IN(X,Y) \tag{3}$$

As in a planning system, in a LASG architecture must be realized the following functions:

(1) how to detect the best rule to apply, based on the best (possible heuristic) information available;
(2) how to apply the chosen rule in order to compute the new problem's state;
(3) how to detect if a solution was found;
(4) how to detect if the system was blocked, in order to abandon the blocked paths and the system's effort to be directed in most interesting directions.

2.2. **How to select the rules.** The most used technique for choosing the appropriate rules is to determine a set of differences between the desired final state and the current state, and then to identify the relevant rules for reducing the differences. If more rules are identified, a variety of heuristic information could be exploited, in order to chose the rule to be applied. This technique is based on the means-end analysis.

2.3. **How to apply the rules.** A possibility to apply the rules is to describe for each possible action the changes that it brings to the state's description. Moreover, some declarations are needed, in order to state that the rest of the description remains the same. A solution for this problem could be to describe a state as a set of predicates representing the facts that are valid in the given state. Each state is explicitly represented as an argument of the predicates. For example, we assume that the current state S is characterized by the following predicate

$$POSSIBLE(X, Y, S) and\, IN(X, Y, S) and\, POSSIBLE(X - 1, Y, S) \tag{4}$$

and the rule that describes the operator **NORTH(X, Y)** will be

$$POSSIBLE(X, Y, S) and\, IN(X, Y, S) and\, POSSIBLE(X - 1, Y, S) \rightarrow IN(X - 1, Y, DO(NORTH(X, Y), S)) \tag{5}$$

In the above equation DO is a function which specifies the state that results after applying a given action in a given state.

For assuring the correctness of the deduction mechanisms, it will be necessary a set of rules to describe those components of the states that are not affected by the operators (the so named *frame axioms*). The advantage of this approach is that a unique mechanism, the resolution, could realize all the operations needed to describe the states. However, the disadvantage is the big number of axioms, if the states' descriptions are complex.

In the architecture that we propose in this section, the number of explicit frame axioms that should be used is not so big.

Each operator will be described by a list of new predicates that the operator makes true and a list of old predicates that the operator makes false. The two lists are named ADD, respectively DELETE. Moreover, for each operator is specific a third list, PRECONDITION, which contains all the predicates that must be true in order to apply the operator. The frame axioms are implicitly specified in LASG. Each predicate that is not included in the ADD or DELETE lists of an operator, is not affected by that operator.

The LASG operators that correspond to the operations presented above are shown in Figure 1. For simplicity, we numbered the four moving possibilities of the robotic agent from a given position (X, Y) as follows: 1- North, 2 - East, 3 - South, 4 - West. We also consider two vectors **dx = (-1, 0, 1, 0)** and **dy = (0, 1, 0, -1)** which gives the moves relative on line and column corresponding to the four actions. Thus, the operator corresponding to the **k**-th move from the position (X, Y) could be described as below:

The application of an operator **O** on a state **S** (given as a logic formula ϕ) means that the predicates from the ADD list of the operator should be added in ϕ. On the other hand, the return to the state before applying the operator **O** (the

O(X, Y, K)
P: POSSIBLE(X, Y) and IN(X, Y) and POSSIBLE(X+dx[k], Y+ dy[k])
A: IN(X+dx[k], Y+ dy[k])
D: IN(X, Y)

FIGURE 1. The operators' description

backtracking) means that the predicates from the DELETE list of the operator should be deleted from ϕ.

3. The LASG algorithm

The idea of the algorithm is to use a stack of goals (a unique stack that contains both goals and operators proposed for solving those goals). The problem solver is also based on a database that describes the current situation (state) and o set of operators described by the PRECONDITION, ADD and DELETE lists. For illustration, we will apply this method on the example shown in Figure 3.

At the beginning of the problem solving process, the stack of goals contains

IN(1, 3)

We have to find an operator which makes true the predicate from the top of the stack (in other words, the predicate **IN(1, 3)** must appear in the ADD list of the operator). We find (by variables' bounding) two possibilities: the operator **O(1, 2, 3)** and **O(2, 3, 1)**. We separate the initial stack into two stacks, we place in the top of the corresponding stack (instead of **IN(1, 3)**) the operator that was found and the predicates from it's PRECONDITION list.

IN(2, 3)	IN(1, 2)
FREE(2, 3)	FREE(1, 2)
VALID(2, 3)	VALID(1, 2)
O(2, 3, 1)	**O(1, 2, 3)**
(1)	(2)

For each stack, we repeat the operations described above with the predicate from the top of the stack. At a given moment, there are four possibilities:

- in the top of the stack is an operator; in this case we remove it from the top, and we retain the operator as part of the problem's solution;
- the predicate from the top of the stack is true - in this case we remove it from the top;
- the predicate from the top of the stack is false - in this case we have to find operators that make the predicate true; we ramify the stack; we add the operators (with their preconditions) in the stack;

- the predicate from the top of the stack can not be satisfied, which means that the system was blocked; in this case we have to abandon the current path, because it will not lead to a solution.

The operation is repeated until the stack became empty (a solution of the problem was found), or until all the possibilities were blocked (in this case the problem solving fails).

If we continue to apply the algorithm on our example, two solutions will be reported:

1. O(4, 1, 1)	1. O(4, 1, 1)
2. O(3, 1, 1)	2. O(3, 1, 1)
3. O(2, 1, 2)	3. O(2, 1, 2)
4. O(2, 2, 2)	4. O(2, 2, 1)
5. O(2, 3, 1)	5. O(1, 2, 2)

In fact, the algorithm consists in a process of **backward reasoning** (we starts from the final state), method known in the literature as a **goal directed reasoning**.

We assume that are given:

- *SI* (the initial state for the agent);
- *SF*(the final state that the agent tries to reach) - there could be a set of final states;
- a set of *operators* $O = \{O_1, O_2, \cdots O_k\}$ that are available to the problem solving agent. For each operator O_i the agent knows the three lists: PRECONDITION, ADD and DELETE.

The agent's goal is to reach the final state *SF*, starting from the initial state *SI*, keeping a history H of the visited states ($H = \{S_1, S_2, \cdots S_m\}$, where $S_1 = SI$ and $S_m = SF$), or of the applied operators ($H = \{O_{i,1}, O_{i,2}, \cdots O_{i,m-1}\}$). In the case that the problem has no solution, H will be empty.

The algorithm which determines a solution of the problem (if exists a solution) is described in Figure 2.

The non-determinism of the step 4 from the above described algorithm has to be implemented as a kind of search procedure (a limited depth-first search).

4. Comparison between LASG an the traditional logic architecture

The Logic Architecture based on Stack of Goals improves the traditional logic architecture for intelligent agents, in the following directions:

- in comparison with the traditional logic architecture, which requires a big number of frame axioms in order to realize a correct inference, the LASG architecture reduce this number, and that is why the space complexity is reduced;

(1) • we create a stack of goals S (the solution stack) that initially contains the predicates that should be satisfied in the final state SF. In other words, if the final state could be written as a conjunction of logic sentences $SF = \phi_1 and\, \phi_2 and \cdots and\, \phi_n\}$, then $S = \{\phi_1, \phi_2, \cdots \phi_n\}$;
• SC (the current state):= SI (the initial state);
• H:=empty;

(2) **If** S is empty and the final state SF was reached, **then** the algorithm stops and the final solution is reported; **else**, **go to** step **3**;

(3) **3.1 If** the top of the stack contains an operator O_i, on add the operator in H, on remove the top of stack, on recalculate the current state SC at which on add the predicates from the A list of the operator O_i; **go to** step **2**; **else**, **go to** step **3.2**;

3.2 On choose the predicate from the top of the stack (ϕ_1). **If** ϕ_1 is satisfied in SC, we remove it from the top of the stack; **go to** step **2**; **else**, **go to** step **4**;

(4) We look for the operator O_i (the operators, if are several) that makes ϕ_1 true. If there are several operators $O_{i,1}, O_{i,2}, \cdots O_{i,s}$, on ramify the solution (on obtain s stacks) **for** j=1,s (for each of the s stacks) we add on the top of the stack S_j the predicates from the PRECONDITION list of the operator $O_{i,j}$; **go to** step **3**.

FIGURE 2. The algorithm to determine a solution in a LASG architecture

- because a limited depth-first search is used, the time complexity is, also, reduced (a disadvantage of the traditional logic architecture is that the computational complexity is big);
- the representation is very simple and elegant.

5. EXPERIMENT

Because the above described architecture is based on logic and because the algorithm described in Figure 2 needs backtracking for finding all solutions, the implementation was made in Visual Prolog. It is well-known that the declarative programming languages (as Prolog) have a built-in control mechanism which allows finding all the solutions of a problem.

We have to say that the stack (stacks) of goals that we have to create for applying the algorithm (Figure 2) are retained implicitly by the control strategy of Prolog (a mechanism which allows backtracking).

For applying the algorithm we consider the environment shown in Figure 3.

The positions filled with black on the maze contains obstacles.

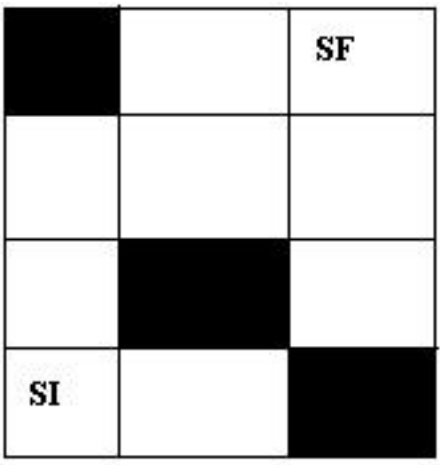


FIGURE 3. The agent's environment

5.1. **The program.** We implemented a Prolog program (Figure 4), which basic non-deterministic predicate is **path(Xi, Yi, Xf, Yf, M, N, L)**, having the flux model **(i, i, i, i, i, i, o)**, and the following signification for the arguments:

- **Xi, Yi** - the coordinates (line and column) of the initial position (the starting position for the agent);
- **Xf, Yf** - the coordinates (line and column) of the final position (the position that the agent tries to reach);
- **M, N** - number of lines and respectively columns of the maze;
- **L** - the list of positions visited by the agent for reaching **SF** starting from **SI** (if the problem has no solution, the list will be empty).

For solving the problem, we considered a LASG architecture, we applied the algorithm described in Figure 2, using the following auxiliary predicates:

- the non-deterministic predicate **candidat(Xf, Yf, X, Y, M, N) (i, i, o, o, i, i)** generates a possible candidate to the solution at a given moment: a state (X, Y) from which the agent could reach the current state (Xf, Yf);
- the non-deterministic predicate **solution(Xi, Yi, L1, L, M, N) (i, i, i, o, i)** collects the elements of a solution in L ($L1$ is the former generated list).

The goal has the form

goal: **path(4, 1, 1, 3, 4, 3, L)**

and the solutions are two:

L=[[4, 1], [3, 1], [2, 1], [2, 2], [2, 3], [1, 3]]

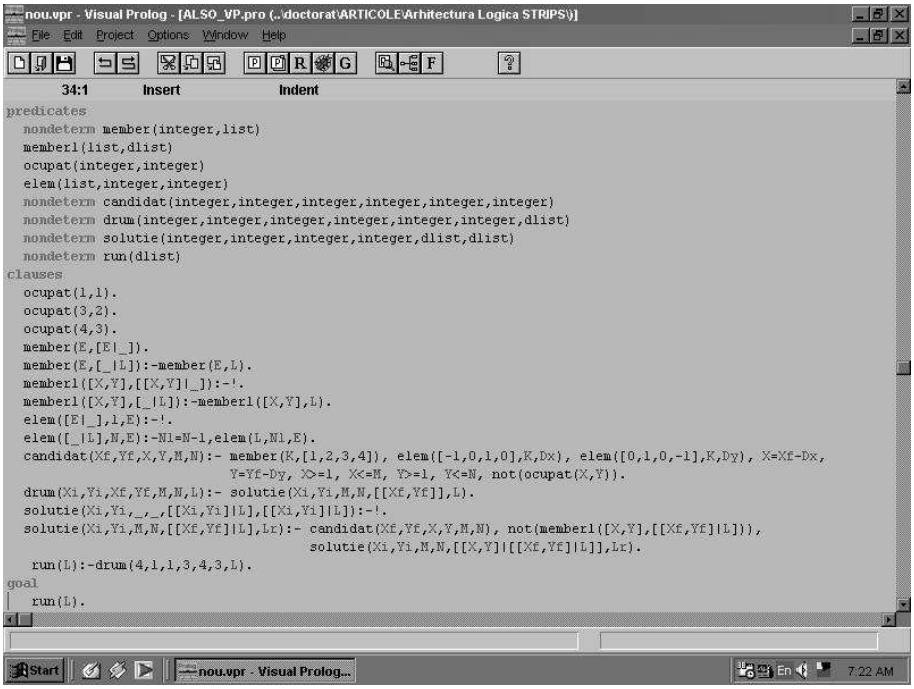


FIGURE 4. The Prolog program

L=[[4, 1], [3, 1], [2, 1], [2, 2], [1, 2], [1, 3]]

6. Conclusions and further work

In a logic based architecture, the intelligent behavior is generated by a symbolic representation of the environment and the agent's behavior, and by a symbolic manipulation of this representation.

In the logic based approach, the process of decisions' applying is, in fact a *deduction*, so the program part of the agent (the strategy of decisions' applying) is codified as a logic theory. That is why this approach is very elegant and has a pure (logic) semantic.

Further work is planned to be done in the following directions:

- in which way some heuristic information could be used in order to reduce the time complexity of the deduction process;
- in which way we can combine the traditional logic architecture with other planning architecture (TWEAK, hierarchical planning architectures).

References

[1] Weiss, G. "Multiagent systems – A Modern Approach to Distributed Artificial Intelligence", The MIT Press, Cambridge, Massachusetts, London, 1999
[2] Wooldridge, M., "Agent-Based Software Engineering", Mitsubishi Electric Digital Library Group, London, 1997
[3] Rich, E., Knight, K. "Artificial Intelligence", Mc Graw Hill, New York, 1991
[4] Winston, P., "Artificial Intelligence", Addison Wesley, Reading, MA, 1984, 2nd ed

Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: gabis@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

MINIMUM COST PATH IN A HUGE GRAPH

ION COZAC

Abstract. Suppose we have a weighted graph $G = (V, E, c)$, where V is the set of vertices, E is the set of arcs, and $c : E \to \mathbf{R}_+$ is the cost function. Determining a minimum cost path between two given nodes of this graph can take $\boldsymbol{O}(m \log n)$ time, where $n = |V|$ and $m = |E|$. If this graph is huge, say $n \approx 700000$ and $m \approx 2000000$, determining a minimum cost path can be a serious time consuming task. So we must develop an algorithm that quickly determines a path having the cost near the optimum one.

Keywords: minimum cost path, huge graph, strongly connected component

1. Introduction

If we develop a route planning application, it is very important to use efficient algorithms that determine a path between two distinct nodes. But what if the application manages a huge graph? This is the case of a complete roads map of a medium country, like Romania. In this case we simply ask to find a path that has the cost near the optimum one, but this path must be found very quickly. A fast algorithm is very important if the application is running on a server, and must satisfy the requests that come from many users by Internet.

To develop the algorithm proposed in this paper, we need some remarks, such as:

(i) each link (arc) can be either a main road or a secondary road;
(ii) the number of main roads (class MR) is very small as compared to the secondary ones (class SR); suppose the cardinality of MR is 8-10% of the cardinality of $MR \cup SR$;
(iii) the number of vertices that belong to a main road (class MV) is very small as compared to the number of vertices that belongs to a secondary road (class SV); suppose the cardinality of MV is 8-10% of the cardinality of $MV \cup SV$;
(iv) the main roads are uniformly scattered among the secondary ones.

2000 *Mathematics Subject Classification.* 05C40.

1998 *CR Categories and Descriptors.* G.2.2 [**Mathematics of Computing**]: Discrete Mathematics – *Graph Theory*.

We need to exploit the following idea (see figure 1). Given two distinct vertices s and t, each of them being of SV type, we first determine a minimum cost path from s to the nearest vertex s_1 that is of MV type. We also determine a minimum cost path from t to the nearest vertex t_1 (reversing!) that is of MV type. Next we determine a minimum cost path from s_1 to t_1, using only the main roads. The solution of the problem is the union of these three paths. This algorithm is very fast because:

- the paths from s to s_1 and from t_1 to t can be quickly determined: see remark (iv);
- the path from s_1 to t_1 can also be quickly determined: see remarks (ii) and (iii).

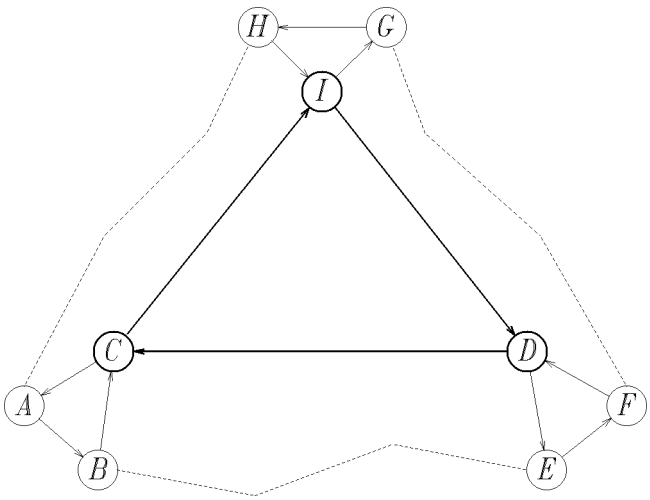


FIGURE 1. Examples of paths determined using the N*earOptimumPath algorithm.* From A to C: A - B - C, from C to D: C - I - D, from B to F: B - C - I - D - E - F

In order to use the algorithm sketched above, we have to prepare two supplementary structures.

We scan the original graph to find all the vertices of MV type. Using these nodes we build a partial subgraph that has only nodes of MV type and arcs of MR type. Let this partial subgraph be $Gs = (Vs, Es)$. We also build the graph $Gi = (V, Ei)$ - the inverse graph of G, where the set Ei is defined as follows:

if $(x, y) \in E$ then $(y, x) \in Ei$, id est, for each arc (x, y) from E we insert the inverse arc (y, x) to Ei.

We describe below the proposed algorithm.

Algorithm *NearOptimumPath*;
Input. The original graph G, the inverse graph Gi, the partial subgraph Gs; two distinct nodes s and t.
Output. Near optimum path from s to t.
begin
* **if** (s is not of MV type) **then**
determine, in graph G, using the algorithm of Dijkstra and selection trees, a minimum cost path D_1 from s to the nearest node s_1 of MV type;
if (t is detected before reaching a node of MV type) **then**
stop: we found the searched path;
else (s is of MV type)
let $s_1 := s$; $D_1 := \emptyset$;
* **if** (t is not of MV type) **then**
determine, in graph Gi, using the algorithm of Dijkstra and selection trees, a minimum cost path D_2 from t to the nearest node t_1 of MV type;
else (**t** is of MV type)
let $t_1 := t$; $D2 := \emptyset$;
* determine, in graph Gs, using the algorithm of Dijkstra and selection trees, a minimum cost path D_3 from s_1 to t_1;
* report the union of these three paths: $D := D_1 \cup$ reverse $(D_2) \cup D_3$;
end (algorithm).

When can we use this algorithm? The following theorem below answer this question.

Theorem 1. *The algorithm* NearOptimumPath *can find a path between any two distinct vertices of the graph G if and only if G and G_s are both strongly connected.*

Proof. These two conditions are obviously sufficient, and the graph G must also be strongly connected. We have to prove that the graph Gs must also be strongly connected. Indeed, suppose that the graph Gs is not strongly connected. It is possible that the algorithm wrongly reports that there is no path between two given nodes, even if such a path exists - the graph G is strongly connected. Let examine the figure 2:

- one can find two distinct nodes s and t, each of them being of MV type, but there is no path from s to t having only arcs of MR type;
- one can find two distinct nodes s and t, at least one being of SV type, and the algorithm find two nodes s_1 and t_1, but there is no path from s_1 to t_1 having only arcs of MR type. □

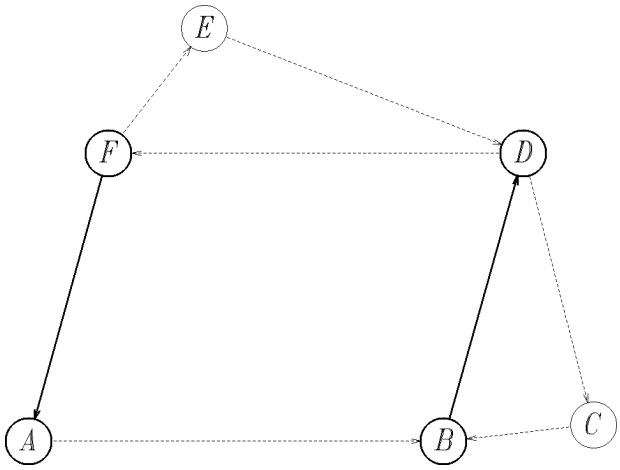


FIGURE 2. The algorithm can not find any path from A to D or from A to E, because the partial subgraph engendered by the arcs (F, A) and (B, D) is not strongly connected

How quickly can we determine a path using this algorithm? The running time of an implementation that uses this algorithm, as compared to the original Dijkstra's algorithm, is proportional to the percentage of the number of the main roads and main vertices.

We saw that this method needs to arrange the information into an organized structure to accelerate searching. This preprocessing phase is necessary because queries will be performed repeatedly on the same graph; these are so called repetitive-mode queries. How much time is needed to arrange the data for searching? To answer this question we present below an algorithm that determines the strongly connected components of a directed graph. This presentation is a review of the algorithm presented in [3].

Algorithm *StronglyConnectedComponents*;
Input. A directed graph $G = (V, E)$.
Output. An array C : the strongly connected components, each vertex being marked with the component number that contains it.
begin
  for (each vertex $x \in V$) **do**
    $Mk[x] := False$;
  $Md := 0$;
  for (each vertex $x \in V$) **do**

```
    if (Mk[x] = False) then
      ScanMark(x);
  Sort D on decreasing order, storing for each mark the associated vertex in X;
  for (each vertex x ∈ V) do
    C[x] := False;
  Build the inverse graph G' corresponding to the graph G;
  nc := 0;
  Warning! The procedure ScanCnx manages the graph G';
    for (each vertex x ∈ X) do
      if (C[x] = 0) then begin
        nc := nc + 1; ScanCnx(x);
        end
end (algorithm).
Procedure ScanMark(vertex x);
begin
  Mk[x] := True;
    for (each vertex y, successor of x) do
    if (Mk[y] = 0) then      ScanMark(y);
  Md := Md + 1; D[x] := Md;
end (procedure).
Procedure ParcCnx(vertex x);
begin
    C[x] := nc;
    for (each vertex y, successor of x) do
    if (C[y] = 0) then
      ScanCnx(y);
end (procedure).
```

The determination of the strongly connected components of a directed graph needs $\mathbf{O}(n \log n + m)$ time, and building the inverse graph Gi and the partial subgraph Gs takes $\mathbf{O}(m)$ time. So we have

Theorem 2. *The supplementary structures used by the algorithm NearOptimumPath are determined in* $\boldsymbol{O}(n \log n + m)$ *preprocessing time.*

We can use the following technique for a parallel architecture. The searching process starts two execution threads: the first thread uses the algorithm *NearOptimumPath*, and the second thread uses the original Dijkstra's algorithm. If one of these two threads finds a path (which may be optimal or not), it must stop the other thread. In this case we don't need to impose a very restrictive condition: it is not necessary that the partial subgraph Gs be strongly connected.

References

[1] Michel Gondran, Michel Minoux, Graphes et algorithmes, Editions Eyrolles, Paris 1979.
[2] Harry R Lewis, Larry Denenberg, Data Structures and Their Algorithms, Harper Collins Publishers, 1991.
[3] Dumitru Dan Burdescu, Analiza complexităţii algoritmilor, Editura Albastră, 1998.
[4] Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest, Introduction to Algorithms, MIT Press, Cambridge, Massachusetts, 1998.

"Petru Maior" University of Tg. Mureş
E-mail address: cozac@uttgm.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

AN EFFICIENT ID-BASED GROUP SIGNATURE SCHEME

CONSTANTIN POPESCU

Abstract. We present an efficient group signature scheme which make use of elliptic curves identity-based signature scheme. The performance of the generated group signature scheme is similar to the performance of the underlying ID-based signature scheme.

Keywords: Group Signature, ID-based Signature schemes, elliptic curves

1. Introduction

The concept of identity-based cryptography is due to Shamir [10]. An identity based crypto-system [2, 10] is a system that allows a publicly known identifier (email address, IP address, name) to be used as the public key component of a public/private key pair in a crypto-system. The scheme assumes the existence of a trusted authority whose sole purpose is to compute for each user the private key associated with the identifier they want to use as public key. The scheme is ideal for closed groups of users. Several ID-based signature schemes have been proposed in the last years [7, 9, 10]. Some of these schemes use Elliptic Curve (EC) algorithms and are therefore particularly efficient.

A group signature, introduced by Chaum and van Heyst [5], allows any member of a group to digitally sign a document such that a verifier can confirm that it came from the group but does not know which individual in the group signed the document. The scheme assumes the existence of a group manager whose sole purpose is to compute for each user a private key that the user should use when signing a message on behalf on the group. A user verifies a signature with the group public key that is usually constant and unique for the whole group (i.e. independent of the members). Many group signature schemes have been proposed [1, 3, 6, 8, 12]. However all of them are much less efficient that regular signature schemes (such as DSA or RSA). Designing an efficient group signature scheme is still an open research problem. In this paper we show that ID-based signature schemes [7] can be used to implement an efficient group signature scheme. Such group signature has the same performance than the performance of the ID-based

2000 *Mathematics Subject Classification.* 94A60.

1998 *CR Categories and Descriptors.* D.4.6. [**Software**]: Operating Systems – *Security and Protection.*

signature scheme it is derived from. This makes our proposal very attractive since it is probably the most efficient group signature scheme that exists today.

2. Identity-Based Signature Scheme

An identity based crypto-system [2, 10] is a system that allows a publicly known identifier to be used as the public key component of a public/private key pair for the purposes of digital signature [7, 9, 10], encryption [2] and key agreement [11]. The private key component is computed by the trusted authority and sends to the corresponding node via a secure and authentic channel.

Definition 1. *An identity based signature scheme is a digital signature scheme specified by the following four algorithms:*

SETUP*: An algorithm, executed by the trusted authority, that takes a random parameter l as input and generates from it system parameters and master key. System parameters is publicly known, while master key is only known to the trusted authority.*

EXTRACT*: An algorithm, executed by the trusted authority, that takes as input system parameters, master key and an arbitrary $ID_i \in \{0,1\}^*$, provided by a user, U_i, and returns a private key x_i. ID_i is an arbitrary string that is used as a public key and x_i is the corresponding private key.*

SIGN*: An algorithm that takes as input system parameters, x_i and a message, $m \in \{0,1\}^*$ and returns a signature σ.*

VERIFY*: An algorithm that takes as input a message $m \in \{0,1\}^*$ and its signature σ, the system parameters and a public key ID_i.* ***VERIFY*** *outputs* 0 *if the signature is invalid and* 1 *if the signature is valid.*

A secure ID-based signature scheme must at least satisfy the following properties:

Correctness: Signatures produced by a user using **SIGN** must be accepted by **VERIFY.**

Unforgeability: It is computationally hard for everyone that do know the secret key x_i of U_i to forge his signatures. As a consequence, it must be computationally hard for everyone to retrieve from system parameters the corresponding master key.

Coalition-resistance: A colluding subset of users, that have received their private key from the same trusted authority and system parameters, cannot generate a valid signature that the trusted authority cannot link to one of the colluding users.

3. ID-based Signatures from Pairings on Elliptic Curves

In this section we review the ID-based signature scheme from [7] which makes use of bilinear pairings on elliptic curves.

3.1. **Setup.** We use the same notation as in [7]:

(1) We let G_1 be an additive group of prime order q and G_2 be a multiplicative group of the same order q.
(2) We assume the existence of a bi-linear map $\widehat{e}$ from $G_1 \times G_1$ to G_2 with the property that the discrete logarithm problems in both G_1 and G_2 are hard. Typically, G_1 will be a subgroup of the group of points on an elliptic curve over a finite field, G_2 will be a subgroup of the multiplicative group of a related finite field and the map $\widehat{e}$ will be derived from the Weil or Tate pairing on the elliptic curve.
(3) We also assume that an element $P \in G_1$ satisfying $\widehat{e}(P,P) \neq 1_{G_2}$ is known. We refer to [2, 7] for a fuller description of how these groups, maps and other parameters should be selected in practice for efficiency and security.
(4) We let ID_i be a string denoting the identity of a user U_i and H_1, H_2 and H_3 be public cryptographic hash functions. We require $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ and $H_3 : G_1 \to \mathbb{Z}_q$.
(5) A trusted authority chooses a random integer $s \in \mathbb{Z}_q$ which is a system-wide master secret.
(6) We also assume that the value $P_{pub} = s \cdot P$ is publicly known.

3.2. **Extract.** A user's public key for signature verification is $Q_{ID_i} = H_1(ID_i)$, while his secret key for signature generation is $D_{ID_i} = s \cdot Q_{ID_i}$. These keys are the same as in the encryption scheme of [7]. If desired, encryption and signature keys can be separated simply by concatenating the string ID_i with extra bits which identify the keys' intended functions.

3.3. **Sign.** To sign a message $m \in \{0,1\}^*$, a user U_i uses the following algorithm:

- Chooses a random $k \in \mathbb{Z}_q^*$.
- Computes $(R,S) \in G_1 \times G_1$, where

$$\begin{aligned} R &= k \cdot P \\ S &= k^{-1}\left(H_2(m) \cdot P + H_3(R) \cdot D_{ID_i}\right). \end{aligned}$$

  Here k^{-1} is the inverse of k in $\mathbb{Z}_q^*$.
- Output the signature (R,S).

3.4. **Verify.** Checking whether a pair (R,S) is a valid signature on a message $m \in \{0,1\}^*$ with respect to the public key Q_{ID_i} can be done as follow:

- Computes $\widehat{e}(U,V)$, where (U,V) is a purported signature on message m.
- Check whether $\widehat{e}(U,V) = \widehat{e}(P,P)^{H_2(m)} \cdot \widehat{e}(P_{pub}, Q_{ID_i})^{H_3(R)}$.
- The signature is accepted if these values in G_2 match and rejected otherwise.

4. Group Signature Scheme

A group signature, introduced by Chaum and van Heyst [5], allow any member of a group to sign on behalf of the group. Group signatures are publicly verifiable and can be verified with respect to a single group public key. Only a designated group manager, can revoke the anonymity of a group signature and find out the identity of the group member who issued a given signature. Furthermore, group signatures are unlinkable which makes it computationally hard to establish whether or not multiple signatures are produced by the same group member. At the same time, no one, including the group manager, can misattribute a valid group signature.

Group signature schemes are defined as follows. (See [4] for more details).

Definition 2. *A group signature scheme is a digital signature scheme comprised of the following:*

(1) ***Setup****: On input of a security parameter* 1^l *this probabilistic algorithm outputs the initial group public key* PK *and the secret key* SK *for the group manager.*
(2) ***Join****: An interactive protocol between the group manager and a user that results in the user becoming a new group member.*
(3) ***Sign****: An interactive protocol between a group member and a user whereby a group signature on a user supplied message is computed by the group member.*
(4) ***Verify****: An algorithm for establishing the validity of a group signature given a group public key and a signed message.*
(5) ***Open****: An algorithm that, given a signed message and a group secret key, determines the identity of the signer.*

A secure group signature scheme must satisfy the following properties:

(1) *Correctness*: Signature produces by a group member using **Sign** must be accepted by **Verify**.
(2) *Anonymity*: Given a signature, identifying the actual signer is computationally hard for everyone but the group manager.
(3) *Unlinkability*: Deciding whether two different signatures were computed by the same group member is computationally hard.
(4) *Unforgeability*: Only group members are able to sign messages on behalf of the group.
(5) *Exculpability*: Even if the group manager and some of the group members collude, they cannot sign on behalf of non-involved group members.
(6) *Traceability*: The group manager can always establish the identity of the member who issued a valid signature.
(7) *Coalition-resistance*: A colluding subset of group members cannot generate a valid group signature that cannot be traced.

5. Our Group Signature Scheme from a ID-based Signature

In this section we present how a ID-based signature scheme [7] can be used to implement an efficient group signature scheme. If we consider that, in the ID-based signature scheme [7], all users that get a private key (from their ID) from the same system and master key parameters form a group, the concepts of ID-based signatures and group signatures are very similar. In this description, the group manager is also a trusted authority.

5.1. **The scheme.**

- **Setup:** The group manager executes the steps from the subsection 3.1. The initial group public key is

$$PK = (q, P, P_{pub}, Q_{ID_i}, H_1, H_2, H_3, \widehat{e})$$

and the secret key is $SK = s$.

- **Join**: Suppose now that a user U_i wants to join the group. We assume that communication between the group member and the group manager is secure, i.e., private and authentic. To obtain his membership certificate, each user U_i must perform the following protocol with the group manager:
  - The user U_i sends ID_i to the group manager.
  - The group manager computes $S_i = s \cdot Q_{ID_i}$ and then S_i is communicated secretly to the user U_i.
- **Sign**: In our scheme, ID_i is the public component of a RSA signature public/private key pair generated by the user itself. This public/private key pair will be referred as (ID_i, d_i) in the remainder of this paper. First, the user U_i signs a message $m \in \{0,1\}^*$ with its RSA private key d_i and the corresponding RSA signature scheme:

$$SigRSA = m^{d_i} \, (mod \; n) \, ,$$

where n is an RSA-like modulus. Then, the group member U_i can generate anonymous and unlinkable group signatures on a message $m \in \{0,1\}^*$ as follows:
  - Chooses a random $k \in \mathbb{Z}_q^*$.
  - Computes $(R, S) \in G_1 \times G_1$, where

$$\begin{aligned} R &= k \cdot P \\ S &= k^{-1} \left(H_2 \left(m \right) \cdot P + H_3 \left(R \right) \cdot S_i \right) , \end{aligned}$$

    where k^{-1} is the inverse of k in $\mathbb{Z}_q^*$.
  - The group signature Sig is then the concatenation of the previously generated signatures $SigRSA$, (R, S) with the U_i's public key ID_i

$$Sig = m^{d_i} \, (mod \; n) \, ||x_R||x_S||ID_i$$

where x_R is the x-coordinate of R and x_S is the x-coordinate of S.

- **Verify**: First, a user verifies that the signature was generated by the group by verifying using the algorithm specified in Section 3.4 that (R, S) is valid and therefore the user U_i is an authorized member of the group:

$$\begin{aligned}\widehat{e}(R,S) &= \widehat{e}\left(k\cdot P, k^{-1}\left(H_2(m)\cdot P + H_3(R)\cdot S_i\right)\right)\\ &= \widehat{e}\left(P, H_2(m)\cdot P + H_3(R)\cdot S_i\right)\\ &= \widehat{e}(P,P)^{H_2(m)}\cdot\widehat{e}\left(P_{pub}, Q_{ID_i}\right)^{H_3(R)}\end{aligned}$$

  where we have used the bi-linearity properties of $\widehat{e}$. Second, a user verifies that the signature was generated by U_i and not by the group manager by verifying using the U_i's public key ID_i and the corresponding RSA signature that $SigRSA$ is valid:

$$m = SigRSA^{ID_i}\,(mod\ n)\,.$$

  Since the group manager does not know the private key d_i it will not be able to generate a valid $SigRSA$.
- **Open**: The group manager knows for each ID_j the identity of the user U_j that is associated with it. This binding is established during the **Join** phase. As a result, it is easy for a group manager, given a message and a valid group signature Sig, to determine the identity of the signer.

5.2. **Security Considerations.** In this section, we access the security of the group signature scheme defined in Section 5 according to the security properties defined in Section 4.

Correctness: This property is guaranteed since the ID-based signature scheme [7] must guaranteed it too.

Unforgeability: This property is guaranteed since the ID-based signature scheme [7] must guaranteed it too.

Anonymity: In our scheme, a group signature is the concatenation of the identity based signature with the user's public key (i.e. ID). Therefore if the underlying identity based signature provides anonymity and if the user's public key does not reveal any information about the user, anonymity is guaranteed by the group signature scheme.

Unlinkability: In our scheme, a group signature is the concatenation of the identity based signature with the user's public key (i.e. ID). As a result, all the signatures generated by a user will contains his public key. Therefore unlinkability is not provided. However if the underlying identity-based signature provides unlinkability and if a user uses a different public/private key pair for each signature, unlinkability is then provided. This solution might not be very practical if the user has to sign a lot of messages (because it needs to get and store a lot of public/private key pairs) but is acceptable otherwise.

Exculpability: In our group signature scheme, a group member can not sign on behalf of other members because it does not know the other members' private keys. The group manager knows each users' private key S_i, but he do not knows the users' RSA private key d_i. Therefore, exculpability is provided.

Traceability: Since, in our proposal, the group manager generates each member private keys from their public keys, it can easily identify the actual signer of a valid signature by looking at the public key component in the group signature. Traceability is therefore provided.

Coalition-resistance: This property is guaranteed since the ID-based signature scheme [7] must guaranteed it too.

Our ID-based group signature scheme has a performance cost since it adds one RSA signature. Furthermore even with this extra cost, we believe that our scheme is still more efficient that any existing group signatures.

6. Conclusion

This paper describes an efficient group signature scheme from an elliptic curves identity based signature scheme. The generated group signature can handle large groups since the group public key and parameters are constant and do not depend on the group members. The security of such a group signature depends on the security of the ID based signature scheme it was derived from. The generated group signature performance is similar to the performance of the underlying ID based signature scheme.

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, A practical and provably secure coalition-resistant group signature scheme, Advances in Cryptography, CRYPT0 2000, vol. 1880, Lecture Notes in Computer Science, Springer Verlag, pp. 255-270, 2000.

[2] D. Boneh and M. Franklin, Identity based Encryption from Weil pairing, Advances in Cryptography CRYPT0 2001, Springer-Verlag, Lecture Notes in Computer Science, vol. 2139, pp. 213-229, 2001.

[3] J. Camenisch, M. Stadler, Efficient group signature schemes for large groups, Advances in Cryptology, CRYPTO'97, Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1296, pp. 410-424, 1997.

[4] J. Camenisch and M. Michels, A group signature with improved efficiency, Advances in Cryptography, ASIACRYPT'98, Springer-Verlag, Lecture Notes in Computer Science, vol. 1514, pp. 160-174, 1998.

[5] D. Chaum and E. Van Heyst, Group signatures, Advances in Cryptography, EUROCRYPT'91, Springer-Verlag, Lecture Notes in Computer Science, vol. 547, pp. 257-265, 1991.

[6] L. Chen and T.P. Pedersen, New group signature schemes, Advances in Cryptography, EUROCRYPT'95, Springer-Verlag, Lecture Notes in Computer Science, vol. 950, pp. 171-181, 1995.

[7] K. Paterson, Id-based signatures from pairings on elliptic curves, Tech. Rep., IACR Cryptology ePrint Archive: Report 2002/004, http://eprint.iacr.org/2002/004/, 2002.

[8] C. Popescu, Group signature schemes based on the difficulty of computation of approximate e-th roots, Proceedings of Protocols for Multimedia Systems (PROMS 2000), Poland, pp. 325-331, 2000.
[9] R. Sakai, K. Ohgishi, and M. Kasahara, Cryptosystems based on pairing, Proceedings of Symposium on Cryptography and Information Security, Japan, Okinawa, pp. 26-28, 2000.
[10] A. Shamir, Identity-based cryptosystems and signature schemes, Advances in Cryptography, CRYPTO'84, Springer-Verlag, Lecture Notes in Computer Science, vol. 196, pp. 47-53, 1984.
[11] N. P. Smart, An Identity based Authenticated Key Agreement Protocol based on the Weil Pairing, Tech. Rep., IACR Cryptology ePrint Archive: Report 2001/111, http://eprint.iacr.org/2001/111/, 2001.
[12] Y.M. Tseng and J.K. Jan, A novel id-based group signature, Workshop on Cryptology and Information Security, Tainan, pp. 159-164, 1998.

University of Oradea, Department of Mathematics, Str. Armatei Romane 5, Oradea, Romania

E-mail address: cpopescu@uoradea.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

A WORD SENSE DISAMBIGUATION EXPERIMENT FOR ROMANIAN LANGUAGE

GABRIELA ŞERBAN AND DOINA TĂTAR

Abstract. The task of disambiguation is to determine which of the senses of an ambiguous word is invoked in a particular use of the word [5, 8]. It is known that the statistical methods produce high accuracy results for semantically tagged corpora [2]. Also, Word Net is a good source of information for WSD [3, 4]. Since for Romanian language does not exist neither a corpus nor something similar with WordNet, we make an experiment for WSD, using an algorithm for WSD [8], which requires only information that can be extracted from untagged corpus. This algorithm learns to make predictions based on local context with only a few labeled contexts and many unlabeled ones.

Keywords: Word sense disambiguation, corpus.

1. Introduction

In [9], Yarowsky observed that there are constraints between different occurrences of contextual features that can be used for disambiguation. Two such constraints are *one sense per discourse* and *one sense per collocation*. These mean that the sense of a target word is highly consistent within a given discourse (document) and the contextual features (nearby words) provide strong clues to the sense of a target word.

Notational conventions used in the following are: w is the word to be disambigued (*target word*), $s_1, \cdots, s_K$ are possible senses for w, $c_1, \cdots, c_I$ are contexts of w in a corpus, $v_1, \cdots, v_J$ are words used as contextual features for disambiguation of w. The contextual features $v_1, \cdots, v_J$ occur in a fixed position near w, in a *window* of fixed length, centered or not on w ("unrestricted collocations", in [6]).

Received by the editors: November, 20, 2002.

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

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

A Naive Bayes Classifier (NBC) realizes the calculus of the sense s', which for the target word w and a given context c satisfies the relation [5]: $s' = argmax_{s_k} P(s_k \mid c) = argmax_{s_k} \frac{P(c|s_k)}{P(c)} P(s_k) = argmax_{s_k} P(c \mid s_k) P(s_k)$. The Naive Bayes assumption is that the contextual features are all conditional independent. This is not generally true, but there is a large number of cases in which the algorithm works well. Concerning the probabilities $P(v_j \mid s_k)$ and $P(s_k)$, these are calculated from a labeled (annotated) corpus. In our algorithm the probabilities $P(v_j \mid s_k)$ are re-estimated until all the contexts are solved.

2. A Bootstrapping Algorithm (BA) for WSD

The BA algorithm begins by identifying a small number of training contexts. This could be accomplished by hand tagging with senses the contexts of w for which the sense of w is clear because some *seed collocations* [9, 10] occur in these contexts (for a detailed description of the BA algorithm see [8]).

The notational conventions are as above: $C = \{c_1, c_2, \cdots c_I\}$ are contexts (windows) of w, as obtained with query w and with an on-line corpus tool (at us *htdig* and a Romanian corpus). Each c_i has the form: $c_i = w_1, \cdots, w_t, w, w_{t+1}, \cdots, w_z$ where $w_1, w_2, \cdots, w_t, w_{t+1}, \cdots, w_z$ are words from the set $\{v_1, \cdots, v_J\}$ and t and z are selected by user.

Let us consider that the words $V = \{v^1, \cdots, v^l\} \subset \{v_1, \cdots, v_J\}$, where l is small (for example 2) are *surely* associated with senses for w, such that the occurrence of v^i in the context of w determines the choice of a sense s^i for w (one sense per collocation). Here $\{s^1, \cdots, s^l\}$ is a subset of $\{s_1, \cdots, s_K\}$.

These rules can be done generally as a decision list:

(1) **if** $\mathbf{v^i}$ *occurs in a context* **c** *of* **w** **then** *the sense of* **c** *is* $\mathbf{s^i}$, $s^i \in S$

So, from the set of contexts obtained as query results, some contexts can be solved.

For our algorithm, we define a relation $\delta \subset W \times P(W)$, where W is the set of all words and $P(W)$ is the power set of W . If $w \in W$ is a word and $c \in P(W)$ we say that $(w, c) \in \delta$ if $w \in c$ or, else, if exists a word $w1 \in c$ so that the words w and $w1$ have the same gramatical root (particularly c is a context).

So, a corresponding decision list has the following form:

(2) **if** $(v, c) \in \delta$ *and* **v** *has the sense* $\mathbf{s_i}$ **then** *the sense of the context* **c** *is* $\mathbf{s_i}$

3. The Application for Words Disambiguation

The application is written in Visual C++ 6.0 and its goal is to find the correct sense for a given word (the target word) in some given contexts using the algorithm described in section 2.

3.1. **Experiment.** Our aim is to use the BA algorithm for the romanian language, to disambiguate the word *poarta* in some contexts obtained with an on-line corpus tool (at us *htdig* and a Romanian corpus).

We make the following specifications:

- the target word *poarta* has, in romanian language, four possible senses (two nouns and two verbs);
- we experiment our algorithm starting with 38 contexts for the target word;
- we start with 6 words as contextual features for the disambiguation.

The input text file for our experiment is the following:

- the target word
poarta
- the senses of the target word
casa fotbal haine raspundere
- the words used as contextual features for the disambiguation and the indexes of the corresponding sense of the target word
lemn 1 casa 1 minge 2 blugi 3 raspundere 4 semnatura 4
- the contexts of the target word

- Respectivul Popa Nicolae Ioan a prezentat jandarmului de la **poarta** un buletin de identitate cu seria B.C., nr. 718609, aceasta in timp ce adevaratul Ioan Popa
- De cand s- a instalat in scaun ultimul primar, frenezia imperecherilor politice este de nestavilit. Se **poarta** negocieri secrete sau fatise, se nasc scenarii avortate dupa nici 24 de ore, se lanseaza nume alaturate te miri carei constructii politice
- hotul, natang in ce priveste alegerea modalitatii de a sustrage date de stricta confidentialitate, dar abil in a scoate pe **poarta** unei institutii, aflate in regim de paza militarizata, ditamai calculatorul

- avand rezolutia catre dl. consilier de stat Mihai Surcel, o dovedeste o alta adresa anexata la dosar, care este datata 15 aprilie 1999, **poarta** (cum se vede si in facsimilul alaturat) antetul Guvernului Romaniei, cabinetul primului-ministru, **poarta** semnatura sefei de cabinet Camelia Andrusenco si este destinata secretarului de stat Liviu Ionescu, din Ministerul de Interne
- Luptatorii SIAS s-au oprit din actiune la **poarta** unei ferme unde s-a refugiat infractorul, pe motiv ca nu aveau mandat de perchezitie
- ...

The accuracy of the BA algorithm in the proposed experiment is **60%**. We note that the *accuracy* of the disambiguation algorithm is calculated with the following formula

$$A = \frac{number\,of\,correctly\,solved\,contexts}{number\,of\,contexts} \tag{3}$$

The experiment at Hearst (1991) shows that to achieve a high precision in word sense tagging, the initial set must be large (20–30 occurrences for each sense).

We have to mention that, in our experiment, we associated a single occurrence for each sense. On the other hand, we observe that if the number of words used as contextual features for the disambiguation and the number of contexts grow, the accuracy of the BA algorithm grows, too.

3.2. **Experimental Comparison with the NBC Algorithm.** In the case of the algorithm described in section 2 (BA–Bootstrapping Algorithm), the relation δ described in Equation 2 is very important. In order to illustrate the efficiency of the BA algorithm (with an without δ), we ran at the same time the NBC algorithm for the experiment proposed in subsection 3.1. We note that "BA without relation" is the BA algorithm (Section 2), in which a decision list has the form described in Equation 1.

The comparative experimental results obtained are shown in Figure 1. In Figure 1, we give, for each algorithm, a graphical representation of accuracy/context. More exactly, for a given algorithm, for the i-th context we represent the accuracy (see Equation 3) of the algorithm for the first i contexts. From Figure 1, it is obvious that the most efficient is the BA algorithm with the relation δ (at each step, the BA algorithm's accuracy is maximum).

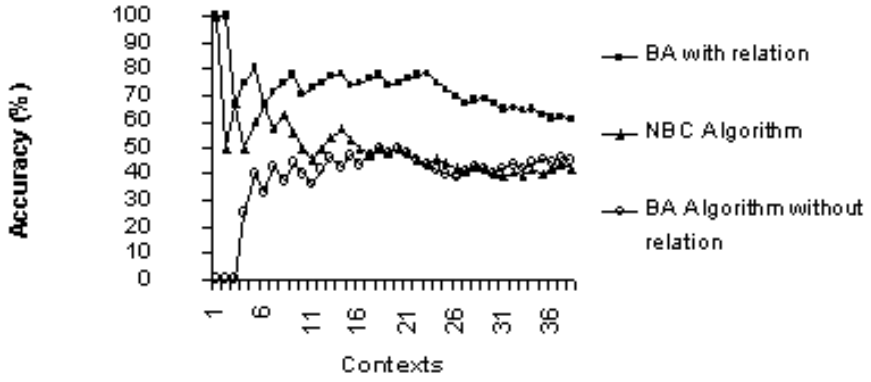


FIGURE 1. The comparative experimental results

4. Further Work

Further work is planned to be done in the following directions: for assuring a better efficiency of the disambiguation, we plain to retain in a database the results of the learning process. We plain to study our approach in the context of combining labeled and unlabeled data with Co-Training as in [1]. Our own goal is to solve with our method the disambiguation for a query in a future QA-system in Romanian which is now in construction.

References

[1] A. Blum, T. Mitchell: Combining Labeled and Unlabeled Data with C-Training. Proceedings of the 11th Annual Conference on Computational Learning Theory (1998) 92–100

[2] G. Escudero, L. Marquez, G. Rigau: Boosting applied to WSD. ACML,Barcelona, Spain (2000)

[3] R. Mihalcea, D. Moldovan: An iterative Approach to WSD. Proceedings of FLAIRS (2000)

[4] R. Mihalcea, D. Moldovan: A method for WSD of unrestricted text. Proceedings of the 37th Annual Meeting of the ACL, Maryland, NY (1999)

[5] C. Manning, H. Schutze: Foundation of statistical natural language processing. MIT (1999)

[6] T. Pedersen, R. Bruce: Knowledge Lean WSD. Proceedings of the Fifteenth National Conference on AI. Madison, WI (1998)

[7] P. Resnik, D. Yarowsky: Distinguishing Systems and Distinguishing sense: new evaluation methods fot WSD. Natural Language Engineering, **1** (1998)

[8] D. Tatar, G. Serban: A new algorithm for WSD. Studia Univ. Babes-Bolyai, Informatica. **2** (2001) 99–108

[9] D. Yarowsky: Hierarchical Decision Lists for WSD. Kluwer Academic Publishers (1999)

[10] David Yarowsky: Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. Proceedings of ACL'95 189–196

Babeş-Bolyai University, Cluj-Napoca, Romania

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

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

THE ROAD TO REAL MULTIMEDIA DATABASES – EMERGING MULTIMEDIA DATA TYPES

HOREA TODORAN

Abstract. This paper describes our view on multimedia data types, which are the fundamentals of real multimedia database management systems. Most of the research efforts in previous work have been focused on audio-visual data and their impact on the design and implementation of multimedia systems. We also take into account the emergence of new media (generically called 'non audio-visual media'), which have the potential to revolutionize the human-computer interaction and bring multimedia database management systems in a new era.

Keywords: Multimedia databases, Multimedia data types, Digital smell, Digital taste, Digital touch

1. Introduction

Because of both the complexity of the term "multimedia" and the diversity of the application fields of the database technology, *multimedia database management systems (MMDBMS)* have different meanings for different groups of users. They are often identified by CD-ROMs storing multimedia information, or by video-on-demand systems allowing users to choose a movie from a database and play it on their own screens, or by document-imaging systems, or by other types of database systems (relational, object-relational, object-oriented, spatial) able to manipulate multimedia elements [Khoshafian96]. All of these illustrate important features of the multimedia database technology, but none of them is exhaustive.

In the referenced literature, a *real* MMDBMS should be able to:

(1) Operate with at least all the audio-visual multimedia data types (as defined in this paper);
(2) Fulfill all the requirements of a real database management system (data persistence, transactions, concurrency control, system recovery, queries, versioning, data integrity, data security, expandability);
(3) Manipulate huge data volumes (virtually no restriction concerning the number of multimedia structures and their size);
(4) Allow interaction with the user;

2000 *Mathematics Subject Classification.* 68P15.

1998 *CR Categories and Descriptors.* H.2.4. [**Information Systems**]: Database Management – *Systems*.

(5) Retrieve multimedia data based on their content (attributes, features and concepts);
(6) Efficiently manage data distribution over the nodes of a computer network (distributed architecture).

Fostered by the development of specific input and output devices, new, *non audio-visual digital media* emerge. 'Digital smell', 'digital taste' and 'digital touch' will become reality and will radically improve the human-computer interaction of the future. Computer-generated multimedia presentations will not be limited to image and sound as they are today. Instead, they will have a greater impact on user's perceptions, allowing for a computer-controlled ambient.

In terms of multimedia databases, the emergence of non audio-visual digital media will lead to new storage, retrieval and presentation challenges. Consequently, the definition of a real MMDBMS will have to be reconsidered, i.e. adapted to the new challenges. A first attempt is made in the next sections of this paper.

The rest of this paper is organized as follows: The next section gives an overview of the audio-visual multimedia data types, which are indispensable in a real MMDBMS. Emerging 'non audio-visual media' are introduced in section 3, together with specific devices and possible evolutions. Our first definition of a real MMDBMS is given in section 4. Then, we conclude and present future work.

2. Overview of adudio-visual multimedia data types

2.1. **Minimal data type requirements for MMDBMS.** Most of the DBMS developed in the last years which claim to be multimedia, have the capacity to operate with only one data type. Even if this only data type is video, audio or image, a system of this kind cannot be considered as a *real multi*media database system (MMDBMS).

For example, *image database management systems*, even if they are able to deal with very large collections of

images and to offer advanced techniques for content-based retrieval (e.g. the PIQ system, described in [18, 19, 3]), are not *real* multimedia database management systems, because of their limitation to only one data type. In our opinion, the same is true for video database systems, which offer advanced techniques for storing, archiving, querying and visualizing digital video – e.g. the VideoSTAR system, developed by the Norwegian Institute for Technology [5, 6], the HERMES/AVIA prototype from GMD Darmstadt [22, 7] and MMVIS from Michigan University [14, 4].

We do not intend to diminish the extremely important contribution that the above-mentioned systems and the related research bring to the development of various techniques, successfully used in the management of multimedia data.

Nonetheless, we believe that:

Statement 1:

A multimedia database management system (MMDBMS) must be able to operate with at least all of the following basic audio-visual multimedia data types: text, image, graphics, audio and video.

Let us define the set of basic audio-visual multimedia data types for further use and explain what does it mean for a DBMS to be able to *operate* with a specific data type.

Definition 2 - Set of basic audio-visual MM data types

The set of basic audio-visual multimedia data types (BAVT) is defined as:

$BAVT = \{TEXT, IMAGE, GRAPHICS, AUDIO, VIDEO\}$

Definition 3 - Operate with an abstract data type

A DBMS is able to ***operate*** *with a specific abstract data type (ADT) when instances of the ADT can be manipulated (i.e. created, updated, deleted, retrieved) independently from other types of data, by means of their own specific techniques.*

By way of combination of BAVT objects, new complex objects, which are multimedia them selves, can be created to be recognized and manipulated by the system.

2.2. **Classification of audio-visual data types.** Taking into account their time-dependency, audio-visual multimedia data are divided in two main categories, as follows:

Definition 4 - Discrete and continuous data

Data not depending on a time scale are called ***discrete data*** *or* ***static data****.*

Data depending on a time scale are called ***continuous data*** *or* ***dynamic data****.*

Text, graphics and image are discrete data, while audio and video are continuous.

Continuous data are more complex than discrete data, which implies the use of much better compression/decompression algorithms and more sophisticated operations for their interpretation and manipulation (see [24, 8, 15]).

The main features and concepts related to the basic audio-visual multimedia data types are described in [15, 10, 20].

2.3. **Generated media.** In [15, 11] some other multimedia data types are described, which are called *generated media.* They are different kinds of computer-generated presentations, the most popular being *animation* and *music.*

If they are stored in audio or video files, then there is practically no difference between generated media, on the one hand, and audio and video data, on the other hand. Yet, if they are generated during the presentation (*real-time*), using specific devices and instruments, we assert that they must be treated as distinct media.

Let us now give our definition for generated media:

Definition 5 - Generated media (GM)

Generated media are computer-generated real-time multimedia presentations based on human-computer interaction.

$$GM \supset \{ANIMATION, MUSIC\}$$

The main advantage of the generated media over audio and video data resides in a much better interaction with the user, which is crucial in the case of MMDBMS.

Generated media are *interactive media*. Their manipulation requires simultaneous control of devices and efficient interpretation of user-generated interrupts.

In terms of *Definition 4* generated media are *continuous*, as long as they essentially depend on a time scale.

2.4. **Speech.** Due to the recent development of advanced techniques for *speech recognition* and *speech understanding* (see [12, 13, 21]), *speech data* are likely to be treated independently from audio data.

The main difference between *speech recognition* and *speech understanding* resides in the fact that the latest implies action taken by the system in response to the vocal command of the user. Current speech recognition systems have better than 95% accuracy and the errors that might occur are very easy to correct. Speech understanding is more complex, especially when the semantic of the command plays an important role.

Speech is also a continuous medium.

2.5. **Synthesis of audio-visual data types.** Based on the previous definitions, a synthetic view on audio-visual media is presented in the table bellow:

	BAVT	**GM**	**Sp.**
Discrete media	TEXT GRAPHICS IMAGE		
Continuous media	AUDIO VIDEO	ANIMATION MUSIC	SPEECH

TABLE 1. Table 1: Audio-visual data types

3. Emerging non audio-visual multimedia data types

Most of the research in the field of human-computer interaction has been focused, until recent years, solely on audio-visual technologies. Up to a certain point, this can be explained by the natural evolution of the human societies, built on the communication between their members, which is mainly based on signes (writing) and sounds (oral communication). Television, the most important mass-media of the last century, is also made of image and sound. Therefore, more and more sophisticated audio and video devices have been developed by the electronic industry, followed by the associated software tools. This evolution has also had a great impact on the database technology, leading to the development of today's MMDBMS.

But, as far as human beings are endowed with five senses, why concentrate the whole effort only on two of them? Why not trying to further improve the human-computer interaction by means of adding the strength of *smell*, *taste* and *touch*? A positive answer is given by the development, in the last few years, of a new generation of hardware devices and software applications, which we briefly

describe in the next subsections of this paper. They will lead to the emergence of new media, that are non audio-visual and that will probably have the same impact on human-computer interaction as audio-visual media have had in the early 1980s.

To what extent will affect the new non audio-visual media the database field is almost impossible to accurately predict. However, we give our vision on the consequences of the new non audio-visual media for the multimedia database design and implementation.

3.1. **Olfactory input and output interfaces.** Olfactory and tasting interfaces seem to be the least developed among the human-computer interaction technologies. This is mainly due to the lack of useful applications, comparing with the other sense-based technologies. However, the use of scents and taste in military (chemical and biological warfare detectors), medicine (surgical training) and electronic commerce (sample of groceries, cosmetics, household products) has fostered the research on olfactory and tasting systems in the last few years.

There are two types of *olfactory interfaces*, briefly described below: olfactory input interfaces and olfactory delivery (output) systems.

Olfactory input interfaces, also called *electronic noses*, are used to collect and interpret odours (very useful in product quality control and warfare detectors). There are three basic approaches to this kind of input devices:

gas chromatography: separation, identification and determination of chemical components in a complex mixture using the differences in migration rates among the sample components;

mass spectrometry: detects patterns of the molecules using the difference in mass-to-charge ratio of ionised atoms;

chemical sensor arrays: based on the multisensing principle, in which the distributed response of an array of chemical sensors is used to identify the constituents of a gaseous environment (eg. ENOSE by JPL and Caltech).

Olfactory delivery systems are a combination of at least four different processing steps: odour storage (liquid, gels, microencapsulation), odour selection, evacuation and cleaning of exhaled air and odour display. Olfactory delivery systems are already available for the consumer market – e.g. the SENX scent device from TriSenx (`http://www.trisenx.com`).

The Sniffman portable scent system from Ruetz Technologies (Germany) has already been adapted for a multimedia entertainment application – Duftkino - the smell cinema (`http://www.duftkino.de`). "One Day Diet" is the first movie for the nose ("ein Film für die Nase"), allowing audience also to smell the action.

France Télécom R&D and the Burgundy wine industry association (BIVP) are creating a website that lets visitors take an olfactory stroll through the famous vineyards of Burgundy. Aromas, pictures and sounds will be brought together to recreate the atmosphere of the vineyards.

3.2. **Tasting interfaces.** Tasting systems, frequently called *electronic tongues*, mimic their natural counterparts, being already able to distinguish between sweet,

sour, salty and bitter[1], and having the potential to respond to a dazzling array of subtle flavours. Even more, e-tongues can also "taste" cholesterol levels in blood, cocaine concentration in urine, or toxins in water, which means that they can return both qualitative and quantitative results. Most of the applications of electronic tongues are in the field of quality control (flavours, beverages, fragrances, pharmaceuticals) and medicine (blood and urine tests).

Recent examples of e-tongue prototypes include:

The e-tongue prototype developed at University of Texas[2] is made of polymer microbeads positioned on a silicon chip of about $1cm^2$ and arranged in tiny pits to represent taste buds. Each pit is marked with dye to create a RGB color bar, which changes when in contact with a chemical. A camera connected to a computer examines the colors and performs a RGB analysis to determine what tastes are present.

The "Astree Liquid & Taste Analyzer" produced by Alpha-MOS (Web address `http://www.alpha-mos.com`). This analyses a liquid matrix using sensor reactions and different statistical pattern recognition techniques to classify tastes. It was the first tasting system commercially available.

The hand-held device produced by Antonio Riul at Embrapa Instrumentação Agropecuária in São Carlos, Brazil (`http://www.embrapa.br`). It is able to detect low levels of impurities in water and discriminate between Cabernet Sauvignons of the same year from two different wineries, and between those from the same winery but different years. It can also spot molecules such as sugar and salt at concentrations too low for human detection [1].

3.3. **Haptic interfaces.** Haptic interfaces are devices that measure the motion of, and provide sensory stimulus to, the users' hands and fingers. A haptic device provides information to the computer based on the device's position (the way a mouse does) and stimulates users' sense of touch by supplying output in the form of force feedback and tactile, or haptic, feedback. Haptic devices make it possible for users to "touch", feel, manipulate, create, and/or alter with their own hands and fingers, objects presented on computer displays as if they were real physical objects. This is done by carefully calculating the forces one would feel when touching a real object and then presenting these forces to users by using the force feedback and tactile display capability of a haptic device. When done properly, this creates the illusion of "touching" the object.

Haptic interfaces can be used to train physical skills such as those jobs requiring specialized hand-help tools (e.g. surgeons, astronauts, mechanics), to provide haptic-feedback modelling of three dimensional objects without a physical medium (such as automobile body designers working with clay models), or to mock-up developmental prototypes directly from CAD databases (rather than in a machine shop).

---

[1]Recently, a fifth candidate basic taste was identified: *umami*, the taste of monosodium glutamate, characteristic of protein-rich foods (`http://www.umami.it`)

[2]Further information at: `http://weewave.mer.utexas.edu/MED_files/MED_research/MEMS_chem_snsr/beads/bead_sensor.html`

Based on the interaction between de user and the machine, haptic devices can be classified as:

Finger-based: attached to user's finger and responding to its movements. Examples include PHANToM (developed at MIT, but commercialised by SensAble), the pen based device from University of Washington, Rutgers Masters (RM-I, RM-II), Feelit Mouse by Immersion.

Hand-based: users interact with the device by grasping a rigid tool. The machine gives the human arm the sensation of forces associated with various arbitrary manoeuvres. Prototypes have been developed at several universities (Carnegie Mellon, McGill, Northwestern, Rutgers and so on); commercial products: TouchSense by Immersion, Cyberglove and CyberTouch of Virtual Tech.

Exoskeletal: track the movements of user's arm, shoulder or even of the whole body, allowing high interactivity, but at extremely high prices. These machines are mostly used in medicine, for people with disabilities, and military. Examples of commercially available products include Cybergrasp by Virtual Technologies, Dextrous Arms and Hands from Sarcos, Arm Master by Exos.

Inherently passive devices (or intelligent assist devices): these are passive, therefore safe, robots for direct physical interaction with human operators within shared environments. They use intelligent microcontrollers, servo-motors and an advanced "sense/process/actuate" control concept to quantify the speed and direction of motion that the user wants. This information is then processed and the proprietary algorithms direct the movement of the device, no time lapse between the machine's sensing and its response being noticed by the operator (see `http://cobot.com`).

Besides force feedback, other tactile display technologies include:

Vibration: Vibration can be used to transmit information about texture, puncture, slip, and impact. Since vibrations often are sensed as being diffuse or unlocalised, a single vibrating device for each finger or skin area is often sufficient.

Thermal display: Thermal perceptions of an object are based on a combination of thermal conductivity, thermal capacity, and temperature. This enables users to infer material composition as well as temperature.

Small-scale shape or pressure distribution: The most frequently used devices have an array of closely spaced pins that can be raised or lowered individually against the skin to approximate a shape. To conform to the human ability to perceive tactile sensations, the pins must be spaced within a few millimetres of one another.

Other tactile display technologies: These include electrorheological devices (materials that use a "smart" fluid which can change viscosity in an electrical field) combined with sensors, electrocutaneous stimulators

(using electrodes to stimulate cutaneous nerve endings), ultrasonic friction displays, and rotating disks for creating the sensation of slip. The MEMICA prototype from Rutgers University is a good example.

While online scent services are almost ready to be launched on the market, taste and touch seem to be harder to address.

3.4. **Steps towards the integration of smell, taste and touch in real MMDBMS.** In our view, the first and very important step is to *increase the number of application areas* for digital smell, taste and touch and *gain public acceptance*. In [9] possible uses of computer-generated scent are revealed, both in the public spaces (malls, theme parks, retail spaces) and the individual sphere (high-end gamers, aromatherapy to enhance memorization and learning, ubiquitous computing, individuals with special needs – blind, deaf). For this to happen, low-cost standardized devices are needed.

Taking into account the experience gained in image, audio and video, the next step will probably be to develop *specialized databases*. We presume that smell databases, taste databases and touch databases will evolve separately and specific storage devices, querying techniques and presentation methods will be built for each of the three media. Because of the wide effort required for such systems to be put together, it is very realistic for any research team to focus on only one of these non audio-visual media at a time.

The process of integration of digital smell, taste and touch in a real MMDBMS will most likely continue in the form of a *multimedia federated database system*. By means of *wrappers* diverse audio-visual data sources have been already integrated in coherent systems where high-performance query optimization is achieved (see [2, 17]). The same strategy is likely to be adopted by developers also in the case of emerging non audio-visual media rather then building new multimedia systems from scratch. A great challenge will be to send these media across networks, which requires infrastructure upgrading.

On the top of a MMDBMS that also includes smell, taste and touch, amazing *five-senses multimedia applications* will be built.

4. Our first definition of a real MMDBMS

In the view of the topics discussed above, we give our first definition of a real MMDBMS, which is an extension of various definitions found in the referenced literature:

Definition 6: Real MMDBMS

A real multimedia database management system has the following characteristics:

(1) *(Data types) Is able to operate with the following multimedia data types:*
   (a) *basic audio-visual data types: TEXT, GRAPHICS, IMAGE, AUDIO, VIDEO;*
   (b) *generated media: ANIMATION, MUSIC;*
   (c) *SPEECH*
   (d) *non-audio-visual data types: SMELL, TASTE, TOUCH*

(2) *(Database features) Fulfill all the requirements of a real database management system (data persistence, transactions, concurrency control, system recovery, queries, versioning, data integrity, data security, expandability);*
(3) *(Storage) Manipulate huge data volumes (virtually no restriction concerning the number of multimedia structures and their size);*
(4) *(Interaction) Allow interaction with the human operator through all the five senses;*
(5) *(Queries) Retrieve multimedia data based on their content (attributes, features and concepts);*
(6) *(Distribution) Efficiently manage data distribution over the nodes of a computer network (distributed architecture).*

Note: This definition is likely to change in the future, according to advanced results in digital smell, taste and touch.

5. Conclusions and future work

In this paper we have discussed a basic aspect of real multimedia database management systems: the multimedia data types. We have had a brief overview of the audio-visual multimedia data types, which we have defined and classified. Then, we have presented various input and output technologies, which foster the emergence of new multimedia data types, called here "non audio-visual data types": smell, taste, touch. Integrating these new data types in multimedia database systems is a very difficult task. We have imagined three incipient steps. We have also given our first definition of a real MMDBMS.

In the future we plan to analyse the opportunity of building smell database and then integrating smell into federated multimedia database management systems.

6. Acknowledgements

Dr. Thomas Skordas is acknowledged for contributions to this paper and for giving me a first view on real human-computer interaction.

References

[1] Ball, Philip – "Electronic tongue has a good taste", Nature Science Update, 9 January 2002. `http://www.nature.com/nsu/020107/020107-3.html`
[2] Berthold, Henrike – *"A Federated Multimedia Database System"*, Proc. ACM Multimedia '99 Doctoral Symposium, Orlando, Florida, 1999.
[3] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft – *"When is Nearest Neighbor Meaningful?"*, Proc. ICDT, 1999.
[4] Hibino, Stacie Lynn – *"MultiMedia Visual Information Seeking (MMVIS): A Multimedia Interactive Visualization Environment for Exploration and Spatio-Temporal Analysis of Video Data"*, PhD thesis, Michigan University, Ann Arbor, 1998.
[5] Hjelsvold, Rune – *"Video Information Contents and Architecture"*, Proceedings of the 4th International Conference on Extending Database Technology, Cambridge, 1994.
[6] Hjelsvold, Rune; Roger Midtstraum and Olav Sandsta, *"Searching and Browsing a Shared Video Database"*, Proceedings of the 1995 International Workshop on Multi-Media Database Management Systems, 1995.

[7] Hollfelder, Silvia; Everts, André; Thiel, Ulrich – *"Designing for Semantic Access: A Video Browsing System"*, in Proc. of IEEE Int. Conf. On Multimedia Computing and Systems (ICMCS), 1999.

[8] Kamath, Mohan; Ramamritham, Krithi; Towsley, Don – *"Continuous Media Sharing in Multimedia Database Systems"*, Proc. of the 4th International Conference on Database Systems for Advanced Applications (DASFAA'95), Singapore, 1995.

[9] Kaye, Joseph Nathaniel – "Symbolic Olfactory Display", PhD Thesis, MIT Media Lab, May 2001. `http://web.media.mit.edu/~jofish`

[10] Khoshafian, Setrag; Baker, Brad -, *"Multimedia and Imaging Databases"*, Morgan Kuafmann Publ., USA, 1996.

[11] Klas, W., Aberer, K. - *"Multimedia Applications and Their Implications on Database Arhitecture"*, GMD Technical Report, Darmstadt, Germany, 1995.

[12] *** - *"Emerging technologies that will change the world – The Technology Review Ten"*, Massachusetts Institute of Technology's magazine of Innovation, Jan/Feb 2001. `http://www.technologyreview.com`.

[13] *** - *"Voice recognition improves - but still suffers from selective hearing"*, 16 Jan 2001. `http://www.msnbc.com/news/516591.asp`.

[14] Nwosu, C. K.; Thuraisingham, B.; Berra, P.B. – *"Multimedia Database Systems, Design and Implementation Strategies"*, Kluwer Academic Publishers, USA, 1996.

[15] Rakow, Thomas C.; Neuhold, Erich J.; Löhr, Michael – *"Multimedia Database Systems – The Notions and the Issues"*, in G. Lausen, editor, Tagungsband GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW, Informatik Aktuell, p. 1-29, Springer, March 1995.

[16] Rakow, Thomas C.; Klas, Wolfgang; Neuhold, Erich J. – *"Research on Multimedia Database Systems at GMD-IPSI"*, IEEE Multimedia Newsletter 4(1): 41-46, April 1996.

[17] Roth, Mary Tork; Özkan, Fatma; Hass, Laura M. – *"Cost Models DO Matter: Providing Information for Diverse Data Sources in a Federated System"*, Research Report, IBM Research Division, IBM Almaden Research Center, San Jose, California, 1999.

[18] Shaft, Uri; Ramakrishnan, Raghu – *"Data Modeling and Querying in the PIQ Image DBMS"*, Bulletin of the IEEE Computer Science Technical Committee on Data Engineering, 1996.

[19] Shaft, Uri; Ramakrishnan, Raghu - *"Content-Based Queries in Image Databases"*, Data Engineering Bulletin, February 1997.

[20] Smeureanu, Ion; Drulă, Georgeta – *"Multimedia – concepte şi practică"*, CISON Publishers, Bucharest, 1997 (in Romanian).

[21] Sumedha Kshirsagar, Chirs Joslin, Won-Sook Lee, Nadia Magnenat-Thalmann, *"Personalized Face and Speech Communication over the Internet"*, in IEEE Virtual Reality 2001.

[22] Thiel, Ulrich; Hollfelder, Silvia; Everts, André – *"Multimedia Management and Query Processing Issues in Distributed Digital Libraries: A HERMES Perspective"*, Proceedings of 9th International Workshop on Database and Expert Systems Applications (DEXA 98), August 26-28, 1998; Vienna, Austria, Los Alamitos, California: IEEE Computer Society, 1998, pp. 84-89.

[23] *"Towards Natural Human Computer Interaction"*, Jorge Santos (European Commission, DG Information Society), 29 June 2000.

[24] TURAU, Volker; RAKOW, Thomas C. – *"A Schema Partition for Multimedia Database Management Systems"*. `http://www.gmd.de`

[25] Ţâmbulea, Leon – *"Structuri de date şi bănci de date"*, "Babeş-Bolyai" University, Cluj-Napoca, Romania, 1992 (in Romanian).

Faculty of European Studies, Babes-Bolyai University, Cluj-Napoca
E-mail address: `htodoran@euro.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

A STUDY OF DEPENDENCE OF SOFTWARE ATTRIBUTES USING DATA ANALYSIS TECHNIQUES

MILITON FRENŢIU AND HORIA F. POP

Abstract. The dependence between software attributes is studied, using the projects written by second year students as a requirement in their curriculum. The dendrogram, factorial analysis and principal components methods are used as Data Analysis Technique. Also, some consequences on the education activity are considered.

Keywords: software metrics, measurement, fuzzy clustering, data analysis techniques, education

1. Introduction

The main purpose of Software Metrics is to evaluate the needed resources and to improve the Software development process [6]. Software Metrics are also useful to evaluate the quality of a software product [15]. And, as we show and in this paper, Software Metrics are useful in education. The future programmer will respect an adequate programming methodology if he is thaught to do so.

The dependence between some software product attributes was discussed by many authors [1, 2, 8, 9, 20]. The effect of programming style on some software product attributes was analysed in [7]. Here we consider again this problem, taking in account 29 software attributes and using the Principal Components Method to study the dependence between these attributes.

The fact that code indentation increases software programs readability has been recognized and underlined for a long time [11, 14]. This was later proved by other authors through statistical experiments [16, 17, 19]. Also, it was proved [16] that excessive indentation is useless, that the best result for increasing readability is obtained when 2–4 spaces are used for indentation.

Oman and Cook [17] showed through an experiment that maintenance was performed better by the subjects that had to maintain their programs in book format, than those that had traditional programs. Also, they showed that the

---

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.3 [**Software**] : Software Engineering – *Coding Tools and Techniques*; I.5.1 [**Computing Methodologies**] : Pattern Recognition – *Models* – *Fuzzy set*; G.3 [**Mathematics of Computing**] : Probability and Statistics – *Data analysis*.

use of typographic style reduces the maintenance effort, improving programmer performance and program comprehension.

There is a close dependence between the clarity, readability and correctness of a program [7]. We all expect that a strong correlation exists between comprehensibility and good design, and this is confirmed again by our experiment. A study of licence examination results based on fuzzy clustering, showing the relationships to programming habitudes is presented in [9].

We have observed that there is a strong dependence between almost all considered attributes. We will try to explain the reason in the Conclusion part of this paper.

2. The experiment

The study is based on 29 projects produced by second year undergraduate students as part of their requirements curriculum. These projects were analysed observing the attributes described in Table 1, and the primary data is given in Table 2.

Attribute	Description	Attribute	Description
A1:	requirements description	A16:	readability
A2:	good specification	A17:	comprehensibility
A3:	function points	A18:	changeability (modifiability)
A4:	design clarity	A19:	structuredness
A5:	design correctness	A20:	testability
A6:	design completeness	A21:	reliability
A7:	design diagrams	A22:	efficiency
A8:	modules specification	A23:	extensibility
A9:	algorithms description	A24:	adaptability
A10:	LOC	A25:	documentation clarity
A11:	no. of comments	A26:	documentation completeness
A12:	good use of comments	A27:	maintainability
A13:	good use of free lines	A28:	simplicity
A14:	indentation	A29:	quality
A15:	good names		

Table 1. Attributes description

The attributes A10 and A11 were measured automatically by computer. All the others were estimated by postgraduate students. All metrics have the values in the interval [0, 10], where 0 stands for "very bad" (or not present at all), and 10 for "excellent". These values are the impressions of the students about the corresponding attributes. Certainly, these values are subjective, but we consider

that this fact does not affect the dependency between the attributes, all values for a project being given by the same person. After all, "subjective measures are cheap and worth using" [5]. Also, we may accept that the postgraduate students are not experienced programmers, but they have finished a similar project three years ago, and another two projects in their third and fourth year. More, half of them are working at Software companies.

These students form a Master group that study the subject "Software Metrics". The definitions of the above considered attributes were given there. These definitions and are inspired from and can be found in [6].

The attribute A12 refers to the documentation done by comments. It is not based on the number of comment lines of the programs. We may write as many comment lines as we like and sometimes the comments contradict the code, or do not reflect what the code does. The measure for this attribute takes in account if the specification of each module is reflected through comments, if the meaning of each variable and object is explained by comments, if the invariants and other important explanations are given by comments.

In [7] a measure for comprehensibility was defined by

$$m(\text{comprehensibility}) = w_1 \cdot m(\text{readability}) + w_2 * m(\text{design}) \tag{1}$$

where $w_1 = 0.4$, and $w_2 = 0.6$, and

$$m(\text{readability}) = [m(\text{comments}) + m(\text{indentation}) + m(\text{names}) + m(\text{spaces})]/4 \tag{2}$$

If we want to verify this hypothesis we may use Chi-square test for our data. For this we must compute the sum

$$\chi^2 = \sum_{i=1}^{n} (c_i - d_i)^2$$

where c_i are the observed values for comprehensibility, and d_i are the computed values using the formula (1). We have considered here that m(design) is the average of m(A3) and m(A7) since the clarity of design and the needed diagrams affect comprehensibility. For 28 degrees of freedom, the critical value of χ^2 at 0.05 level of probability is 41.34. The computed value for our experiment is 7.56, therefore the test is passed.

3. Study of variables dependence

3.1. **Correlation Matrix.** First of all, the correlation coefficients for all pairs (A_i, A_j) were computed[1].

We remark a strong dependence between almost all pairs of attributes. We may observe that the largest correlation coeficients are cor(A3, A10) = 0.98, cor(A2,

[1]Due to space restrictions, the correlation matrix and other results are not printed in this paper. Instead, they are available, together with all other numerical data, on Internet, at the address `http://www.cs.ubbcluj.ro/~mfrentiu/articole/projdat.html`.

Prj	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	6	5	38	5	6	5	6	2	0	759	268	6	8	7	5	6	5	4	4	4	3	5	5	4	6	5	5	6	4
2	6	7	97	6	6	7	5	6	4	2695	22	2	8	5	5	5	5	5	5	4	4	4	5	4	4	5	4	4	4
3	8	8	145	8	8	8	7	9	3	3550	575	7	7	5	5	6	7	5	6	5	5	6	6	6	7	7	6	5	5
4	6	6	58	3	6	5	4	1	0	1600	0	0	5	3	5	4	4	5	5	5	4	5	6	5	5	6	5	6	5
5	8	8	80	7	7	6	7	5	2	2144	130	5	8	8	7	8	7	8	9	8	7	7	7	7	8	7	8	7	7
6	9	9	159	9	8	8	7	6	2	4027	200	6	9	9	5	7	7	7	7	6	7	8	6	7	7	7	6	6	6
7	6	6	61	8	7	7	6	6	2	1649	6	1	9	8	5	5	6	6	8	7	7	9	6	6	7	7	6	6	6
8	9	8	29	7	8	8	6	4	6	653	0	0	6	8	5	5	5	4	4	6	5	4	3	3	5	5	3	5	5
9	9	8	60	7	8	8	7	5	1	1542	0	0	3	8	5	5	7	5	5	5	5	4	6	6	6	6	5	4	5
10	8	9	47	8	7	7	8	8	1	1164	5	1	8	4	5	5	7	5	5	7	5	5	6	7	7	6	5	7	6
11	7	7	90	8	7	7	5	6	2	2816	108	4	7	6	7	7	7	5	7	8	6	6	5	5	8	7	5	7	6
12	8	8	59	7	7	7	6	8	1	1145	68	3	7	5	6	6	6	6	6	5	5	6	6	5	8	7	6	5	5
13	8	8	74	8	8	9	8	9	1	1880	8	1	9	8	7	7	7	7	8	7	7	8	8	7	9	8	7	7	8
14	7	6	68	7	7	8	7	6	5	1680	0	0	7	6	5	5	6	6	7	6	6	7	6	5	8	7	7	6	7
15	8	7	35	7	8	8	6	9	9	715	0	0	4	5	5	4	6	8	7	7	7	8	8	7	8	8	7	8	7
16	6	6	34	6	7	7	6	4	5	780	32	2	6	3	6	5	5	5	5	6	5	5	6	5	7	6	5	6	5
17	9	8	55	8	8	8	8	9	2	1460	30	2	9	7	8	7	8	7	8	8	7	8	7	8	7	7	7	8	8
18	9	8	45	8	8	8	5	8	0	871	0	0	9	9	8	7	6	9	9	8	8	9	8	8	8	7	8	9	9
19	8	9	57	8	9	7	8	8	8	1553	0	0	9	7	8	7	7	6	8	9	8	8	8	8	7	7	8	4	8
20	9	8	49	9	9	8	8	9	6	1289	15	1	9	7	8	7	8	7	9	9	8	9	9	7	8	8	8	9	9
21	9	9	61	9	8	9	8	9	8	1466	4	0	6	7	6	6	7	8	7	9	8	8	8	6	8	7	7	8	8
22	8	7	60	6	7	9	6	6	9	1653	0	0	4	3	8	4	5	5	5	9	8	7	4	4	6	6	5	5	6
23	9	9	86	8	9	9	8	9	7	2105	120	3	8	5	8	7	7	7	7	8	8	8	7	7	9	8	8	7	8
24	8	7	37	7	8	7	7	4	8	752	6	0	9	3	9	6	7	6	6	8	7	8	7	6	9	8	7	7	7
25	8	8	57	7	7	8	8	8	5	1680	192	5	8	7	7	7	8	7	8	6	7	7	7	6	7	8	7	7	7
26	7	6	34	5	6	6	6	2	0	605	38	2	8	7	6	6	6	4	5	4	4	5	5	5	6	7	5	6	6
27	6	6	28	6	7	7	6	3	0	526	6	0	5	5	7	5	5	5	6	6	5	6	5	6	5	5	5	6	5
28	7	6	32	7	7	6	6	3	0	914	0	0	7	6	5	5	5	6	6	6	7	7	7	6	7	7	6	7	6
29	7	7	39	7	8	7	8	7	0	778	153	4	7	7	7	7	7	7	7	6	7	8	7	7	7	7	7	7	7

TABLE 2. The attribute values for 35 projects

A27) = 0.97, cor(A20, 21) = 0.95, cor(A25, 26) = 0.95. The first one is expected, since the size of the product strongly depends of the atribute A3 (function points).

3.2. **Dendrogram.** A dendrogram is a tree that depicts a hierarchical dependence between the attributes, starting from the correlation matrix [12, 13]. Since we refer downwards to this dendrogram, we have drawn it and it can be seen at the above mentioned address (see Figure 1).

3.3. **Factorial Analysis.** A factorial analysis [21] was also performed. The result of this analysis is printed below in Tables 3.a, 3.b and 3.c.

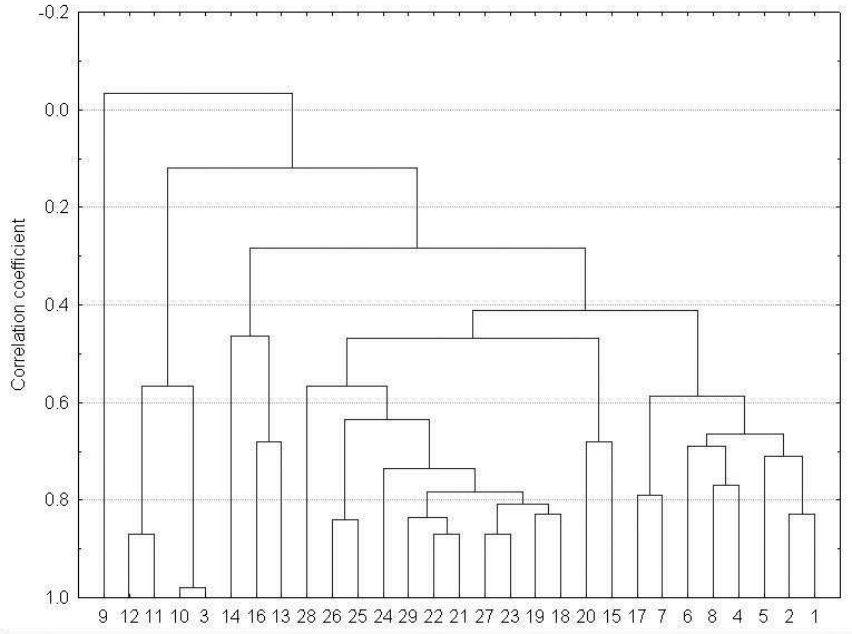


FIGURE 1. Dendrogram for 29 software attributes

To explain the attributes with the probability 0.60 we need two factors, and the relation between attributes and these factors is given in Table 3.a.

Only the attributes A3, A10, A11, and A12 depend more on the second factor than on the first one. All the others depend on the first factor, and we consider this factor the general knowledge of the students. The second "factor" is not a single factor, as can be seen analysing the dendrogram. This dendrogram clearly highlights that A3 and A10 are grouped together, A11 and A12 are also grouped together, but these two groups have quite a low dependency.

We have repeated the factorial analysis for a probability of 0.70 (see Table 3.b). At this level three factors are needed, but, still, the attributes A3, A10, A11, A12, A13, A14 depend stronger on "the second factor". The third factor influences the attributes A1–A8, which characterise the design of the projects.

For the probability 0.80 we need six factors. Now, the second one influences the attributes A3 and A10, and we consider it "the complexity of the problem". The attributes that depend stronger on the third factor are A1, A2, A4, A5, A6, A7, and A17. We can see that the dendrogram classifies these attributes together, in the following subtree: [(((1, 2), 5), ((4, 8), 6)), (7, 17)]. We remark that these

	0.60			0.70				0.80						
Attr.	F1	F2	C	F1	F2	F3	C	F1	F2	F3	F4	F5	F6	C
1	**0.65**	0.24	0.48	0.30	0.12	**-0.76**	0.68	0.21	0.04	**-0.85**	0.11	0.07	0.10	0.79
2	**0.59**	0.44	0.54	0.24	0.32	**-0.77**	0.75	0.15	0.22	**-0.85**	0.12	-0.05	0.13	0.82
3	0.01	**0.86**	0.73	-0.10	**0.82**	-0.30	0.77	0.00	**0.92**	-0.20	0.06	-0.34	-0.05	0.95
4	**0.73**	0.41	0.7	0.47	0.31	**-0.67**	0.76	0.40	0.28	**-0.72**	0.20	0.01	0.16	0.81
5	**0.78**	0.14	0.62	0.45	0.02	**-0.74**	0.75	0.39	0.00	**-0.77**	-0.07	0.08	0.20	0.78
6	**0.57**	0.14	0.34	0.11	0.00	**-0.88**	0.78	0.20	0.27	**-0.76**	-0.26	0.25	-0.04	0.79
7	**0.63**	0.29	0.48	0.44	0.21	**-0.52**	0.51	0.30	-0.26	**-0.73**	0.02	-0.38	0.23	0.79
8	**0.73**	0.33	0.64	0.44	0.23	**-0.71**	0.74	0.49	0.22	**-0.70**	-0.09	-0.09	-0.02	0.78
9	**0.38**	-0.20	0.18	-0.03	-0.31	**-0.70**	0.58	0.11	0.11	-0.42	**-0.80**	0.28	0.16	0.77
10	0.00	**0.81**	0.66	-0.10	**0.78**	-0.29	0.70	-0.02	**0.95**	-0.16	0.04	-0.27	0.00	0.93
11	-0.17	**0.79**	0.65	-0.08	**0.81**	0.06	0.66	-0.06	0.37	-0.04	-0.02	**-0.85**	-0.04	0.82
12	-0.15	**0.84**	0.72	0.03	**0.88**	0.19	0.80	-0.05	0.37	0.05	0.19	**-0.87**	0.15	0.89
13	**0.40**	0.37	0.30	**0.59**	0.39	0.11	0.52	0.32	0.11	-0.02	0.42	-0.28	**0.65**	0.68
14	0.30	**0.37**	0.23	**0.38**	0.37	-0.03	0.29	0.15	0.14	-0.28	**0.94**	0.06	0.09	0.82
15	**0.64**	-0.17	0.44	**0.57**	-0.21	-0.27	0.44	0.35	-0.19	-0.22	-0.26	0.09	**0.86**	0.86
16	**0.56**	0.55	0.62	**0.68**	0.54	-0.08	0.76	0.37	0.09	-0.29	0.47	-0.43	**0.66**	0.93
17	**0.72**	0.48	0.75	**0.60**	0.41	-0.47	0.75	0.44	-0.01	**-0.66**	0.16	-0.42	0.32	0.86
18	**0.83**	0.06	0.70	**0.80**	0.01	-0.31	0.73	**0.86**	0.16	-0.27	0.17	0.17	0.02	0.87
19	**0.84**	0.18	0.73	**0.87**	0.14	-0.22	0.83	**0.81**	0.22	-0.21	0.28	0.09	0.33	0.91
20	**0.80**	-0.20	0.68	0.55	-0.30	**-0.59**	0.73	0.49	0.07	-0.42	-0.28	0.43	**0.51**	0.86
21	**0.90**	-0.06	0.81	**0.67**	-0.15	-0.60	0.83	**0.68**	0.19	-0.44	-0.15	0.36	0.35	0.91
22	**0.86**	-0.01	0.75	**0.83**	-0.06	-0.30	0.79	**0.84**	0.16	-0.19	-0.02	0.19	0.33	0.88
23	**0.82**	-0.02	0.67	**0.84**	-0.05	-0.22	0.75	**0.88**	-0.14	-0.25	0.07	-0.05	0.01	0.86
24	**0.79**	0.15	0.64	**0.79**	0.11	-0.24	0.70	**0.73**	-0.06	-0.35	0.30	-0.07	0.13	0.75
25	**0.77**	0.11	0.61	**0.72**	0.05	-0.33	0.63	**0.75**	-0.02	-0.30	-0.19	-0.18	0.21	0.74
26	**0.78**	0.14	0.63	**0.75**	0.09	-0.31	0.66	**0.82**	0.04	-0.27	-0.18	-0.20	0.15	0.80
27	**0.88**	0.08	0.78	**0.89**	0.04	-0.25	0.86	**0.87**	-0.01	-0.25	0.00	-0.09	0.29	0.90
28	**0.65**	-0.18	0.46	**0.76**	-0.18	-0.01	0.61	**0.75**	-0.24	-0.04	0.11	0.05	0.12	0.64
29	**0.96**	-0.09	0.92	**0.85**	-0.16	-0.43	0.93	**0.76**	-0.10	-0.42	0.06	0.22	0.38	0.94
	(a)			(b)				(c)						

TABLE 3. Factorial analysis for a probability of 0.60, 0.70, and 0.80 respectively

attributes are those connected to the design (similarily to the case of probability 0.70), plus the comprehensibility!

But there are other factors, the fourth one that strongly influences the attribute "indentation", the fifth one, that influences the attributes A11, A12 (comments), and the sixth one influences A13, A15, A16 (mainly the readability of programs).

3.4. **Principal Components Analysis.** Principal Components Analysis (PCA) is designed to reduce the number of variables that need to be considered to a small

number of axes called the principal components, that are linear combinations of the original variables. The new axes lie along the directions of maximum variance thus containing most of the information. PCA provides an objective way of finding attributes of this type so that the variation in the data can be accounted for as concisely as possible. Moreover, due to this space rotation, PCA is often used as a dimensionality reduction method: very few principal components provide a good coverage of all the original variables.

PCA has been applied on the set of variables (i. e. the transpose of the original data set) in order to produce a visual representation of the variables. Table 4 describes the reduction coefficients produced by considering the 29 software attributes and 29 student projects, and Figure 2 presents the scores corresponding to the first two principal components.

No.	Eigenvalue	Successive diff.	Proportion	Cummulative prop.
1	28.8374	28.6766	0.994392	0.994392
2	0.160824	0.160002	0.00554565	0.999938
3	0.000821365	0.000529606	2.83229e-005	0.999966
4	0.000291759	9.32732e-005	1.00606e-005	0.999976
5	0.000198485	4.18986e-005	6.84432e-006	0.999983
6	0.000156587	3.55741e-005	5.39954e-006	0.999989
7	0.000121013	6.58028e-005	4.17285e-006	0.999993

TABLE 4. Reduction coefficients for 29 software attributes and 29 student projects (the remaining eigenvalues are less than 0.0001, and less important)

From an analysis of Figure 2 we remark the three isolated points (corresponding to software attributes A3, A10 and A11). Because of the agglomeration of points in the remaining region, we will repeat the procedure, but this time will ignore the three above mentioned attributes. The results are depicted in Table 5 and Figure 3.

3.5. **Fuzzy Hierarchic Clustering.** The theory of fuzzy sets was introduced in 1965 by Lotfi A. Zadeh [22] as a natural generalization of the classical set concept. Let X be a data set, composed of n data items characterized by the values of s characteristics. A fuzzy set on X is a mapping $A : X \to [0, 1]$. The value $A(x)$ represents the membership degree of the data item $x \in X$ to the class A. The advantage of this approach is that it allows a data item x to be a member of more classes, with different membership degrees, according to certain similarity criteria.

Clustering algorithms based on fuzzy sets have proved their superiority due to their ability to deal with imprecise sets, imprecisely-defined boundaries, isolated points, and other delicate situations. The class of fuzzy clustering algorithms based

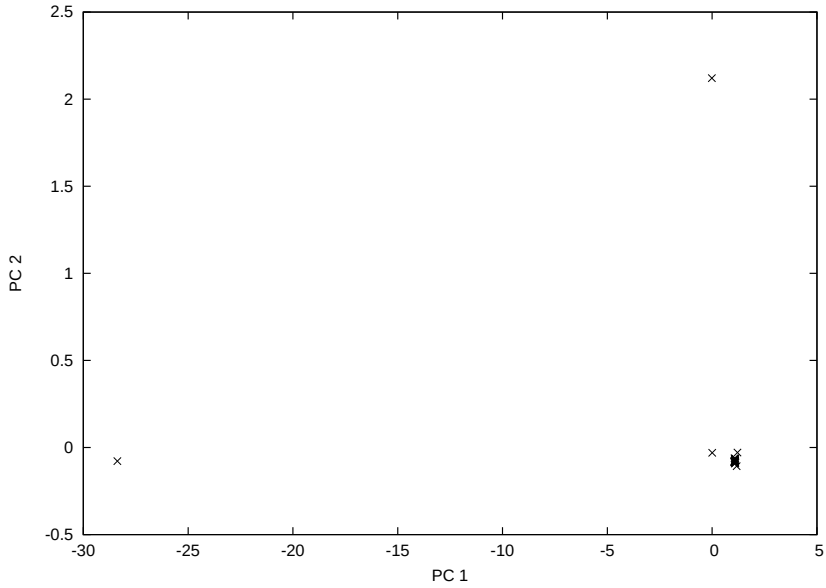


FIGURE 2. Representation of scores corresponding to PC1 and PC2 for 29 software attributes

on fuzzy objective functions [3] provides a large share of geometrical prototypes and combinations thereof, to be used according to the data substructure. On the other hand, the Fuzzy Divisive Hierarchical scheme [4, 18] provides an in-depth analysis of the data set, by deciding on the optimal subcluster cardinality and the optimal cluster substructure of the data set.

The visual representations in Figures 2 and, especially, 3 enable us to further analyse the attributes set. Due to the obvious linear structure of the data, we consider the Fuzzy Divisive Hierarchic Clustering (FDHC) algorithm with linear prototypes. In order to fully understand the relationships between the software attributes, we have used both the original set of 29 software attributes, as well as the smaller set of 26 attributes, where attributes A3, A10 and A11 have been ommitted.

The classification tree and the final binary partition produced by FDHC with linear prototypes using the set of 29 normalized attributes are depicted in Figure 4.

The classification tree and the final binary partition produced by FDHC with linear prototypes using the set of 26 normalized attributes are depicted in Figure 5. The corresponding fuzzy membership degrees to the classes from the final fuzzy partition are displayed in Table 6.

No.	Eigenvalue	Successive diff.	Proportion	Cummulative prop.
1	15.77	11.0031	0.543793	0.543793
2	4.7669	2.66606	0.164376	0.708169
3	2.10084	0.720066	0.0724429	0.780612
4	1.38078	0.34621	0.047613	0.828225
5	1.03457	0.272993	0.0356747	0.863899
6	0.761573	0.168301	0.0262612	0.890161
7	0.593272	0.090646	0.0204577	0.910618
8	0.502626	0.109051	0.0173319	0.92795
9	0.393575	0.0607744	0.0135716	0.941522
10	0.332801	0.0378924	0.0114759	0.952998
11	0.294909	0.0311804	0.0101693	0.963167
12	0.263728	0.0901184	0.00909408	0.972261
13	0.17361	0.00443054	0.00598655	0.978247
14	0.169179	0.0390419	0.00583377	0.984081
15	0.130137	0.0416428	0.0044875	0.988569
16	0.0884946	0.0165013	0.00305154	0.99162
17	0.0719933	0.0172449	0.00248253	0.994103
18	0.0547483	0.0127307	0.00188787	0.995991
19	0.0420176	0.0133489	0.00144888	0.99744
20	0.0286687	0.00901874	0.000988575	0.998428
21	0.0196499	0.00936073	0.000677584	0.999106
22	0.0102892	0.00124848	0.0003548	0.999461
23	0.00904073	0.00496486	0.000311749	0.999772
24	0.00407587	0.00154728	0.000140547	0.999913
25	0.00252859	0.00252859	8.71928e-005	1
26	2.5631e-016	1.55938e-016	8.83826e-018	1
27	1.00372e-016	6.85124e-016	3.4611e-018	1
28	-5.84752e-016	9.65261e-017	-2.01639e-017	1
29	-6.81278e-016	-6.81278e-016	-2.34923e-017	1

TABLE 5. Reduction coefficients for 26 software attributes and 29 student projects

By analysing the Figures 4 and 5, we remark that the cluster substructure of the set of attributes became more detailed after the three isolated attributes (A3, A10, A11) have been removed. This is consistent with the preceding remark on the quality of the PCA projection obtained without the same three atttributes, as we can see from Figures 2 and 3.

A completely different remark may be drawn by analysing Table 6. We see there that many of the attributes have very close fuzzy membership degrees to the four

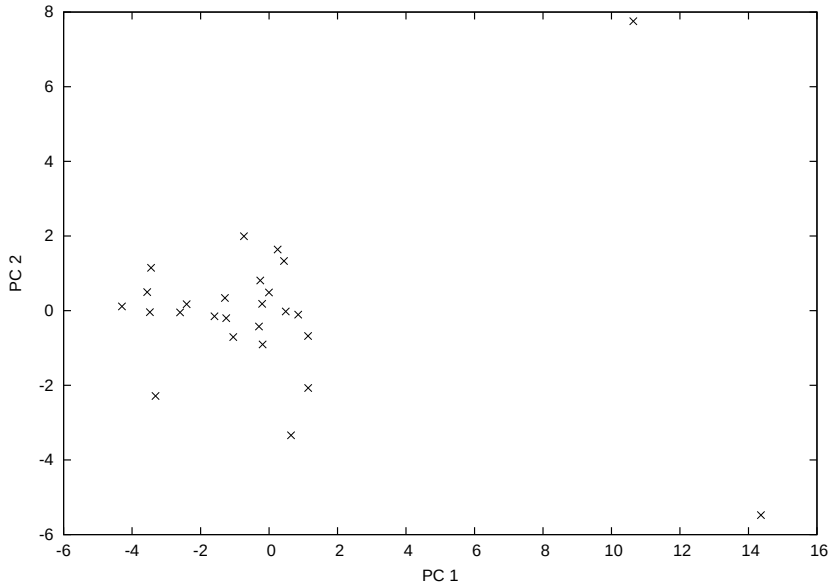
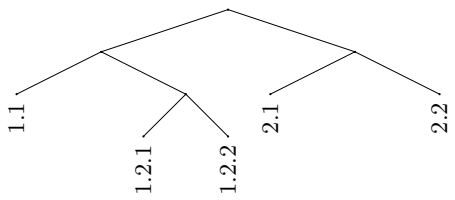


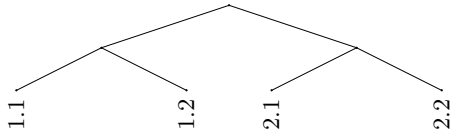
FIGURE 3. Representation of scores corresponding to PC1 and PC2 for 26 software attributes



Class	Members
1.1.	1 2 4 5 6 7 12 13 14 15 16 17
1.2.1.	3 18 19 22 23 24 25 26 27
1.2.2.	8 9 20 21 28 29
2.1.	11
2.2.	10

FIGURE 4. Classification tree and final partition for the set of 29 normalized attributes

final classes. More specifically, 10 attributes, out of the total of 26 (A7, A8, A14, A15, A20, A23, A24, A25, A26, A28), have dominant fuzzy memberships between 0.25 and 0.35. Because of this very strong fuzziness, these attributes should be considered as being shared by all the four classes. They do not contribute to shaping the cluster substructure, and, effectively, will be associated to a fifth class, a sort of 'unclassified' class. Out of the remaining 16 attributes, ten have membership degrees between 0.30 and 0.50 (A4, A5, A6, A17, A18, A19, A21, A22, A27, A29), four have membership degrees between 0.50 and 0.80 (A1, A2, A13, A16) and only two have membership degrees between 0.80 and 1.00 (A9,



Class	Members
1.1.	8 9 15 19 20 21 22 28
1.2.	13 18 23 24 27 29
2.1.	1 2 5 6
2.2.	4 7 12 14 16 17 25 26

FIGURE 5. Classification tree and final partition for the set of 26 normalized attributes (without A3, A10 and A11)

Attr.	1.1.	1.2.	2.1.	2.2.
1	0.0975649	0.0962468	**0.640445**	0.165744
2	0.110721	0.130625	**0.559055**	0.199599
4	0.146171	0.165414	0.336343	**0.352072**
5	0.0959909	0.100097	**0.455519**	0.348393
6	0.112062	0.121853	**0.49245**	0.273635
7	0.175628	0.206314	0.297583	**0.320475**
8	**0.254044**	0.246184	0.249927	0.249846
9	**0.997947**	4.30E-05	0.000966477	0.0010432
12	0.00224497	0.00176989	0.000225756	**0.995759**
13	0.0122362	**0.513947**	0.217884	0.255933
14	0.219962	0.228581	0.273131	**0.278326**
15	**0.284285**	0.278515	0.229546	0.207654
16	0.12194	0.137694	0.238524	**0.501841**
17	0.148125	0.168134	0.257769	**0.425972**
18	0.278445	**0.362585**	0.244657	0.114314
19	**0.384798**	0.308419	0.179179	0.127605
20	**0.315049**	0.292766	0.211903	0.180283
21	**0.380813**	0.378577	0.1516	0.0890098
22	**0.376193**	0.313267	0.177344	0.133196
23	0.30867	**0.347824**	0.22084	0.122666
24	0.239836	**0.29741**	0.3165	0.146254
25	0.251962	0.22761	0.244006	**0.276423**
26	0.226493	0.244955	0.259432	**0.26912**
27	0.323874	**0.381824**	0.207184	0.0871175
28	**0.297997**	0.295768	0.233317	0.172919
29	0.377936	**0.400709**	0.149011	0.0723431

TABLE 6. Fuzzy membership degrees to the final partition for the set of 26 normalized attributes (boldfaces indicate the membership degree to the major defuzzified class)

A12). These last 16 attributes form, actually, the core of the data substructure. The final partition, modified as described, is presented in Table 7.

Class	Members
1.1.	9 19 21 22
1.2.	13 18 27 29
2.1.	1 2 5 6
2.2.	4 12 16 17
(unclassified)	7 8 14 15 20
	23 24 25 26 28

TABLE 7. Final partition for the set of 26 normalized attributes, modified by isolating the attributes with dominant fuzzy membership degrees between 0.25 and 0.35

We also conclude that our fuzzy clustering analysis shows three different kinds of attributes, based on the fuzzy membership distributions. The first main set of attributes is formed by the three strongest attributes removed in the first instance and the two attributes with very large membership degrees (A3, A9, A10, A11, A12). These attributes, also forming the core of the 29 attributes classification (as we see from Figure 4), are the best separated attributes. A second set of attributes correspond to the attributes having resonably high fuzzy membership degrees, and is formed by the attributes from classes 1.1, 1.2, 2.1 and 2.2 from Table 7, other than A9 and A12. These attributes have been classified with a high degree of certainty, but are not as crisp as those in the first set. Finally, the set of 'unclassified' attributes, as presented in Table 7, have fuzzy membership degrees distributed almost evenly between the four fuzzy classes, suggesting that they are not quite suitable to help discriminating among the analysed set of student projects.

4. Concluding remarks

As we expected, very few correlation coefficients are close to zero, and this shows that there is a strong dependence between almost all attributes. As the factorial analysis proved, with the exception of the three above mentioned attributes, plus A9 and A12, these attributes are dependent on the general knowledge of the programmers, which is the main factor that influences all attributes. But there are other factors, and the factorial analysis revealed the complexity of the solved problems and the "discipline", i. e. the wish and habituation of respecting the methodology of programming; it is not sufficient to know it, we must respect it. We may conclude that a good programming style and a correct programming habit must be taught in parallel.

Also, we must observe some anomalies, and, for educational purposes, take some measure to eliminate them. First, we can observe that the smallest coefficients correspond to the comments (second column), and one factor of factorial analysis

is strongly connected to this attribute. By analysing the primary data, we may observe that students do not like writing comments (9 projects out of 29 have no comments at all)! We all know that software documentation is generally poor, often the only information we have is the source code, and we see the reason.

As was observed earlier [9], "the indentation rules are much better obeyed. There is one more reason for this. At all lectures, when the teachers write algorithms or code, they respect these rules in all lines. But only sometimes they write comments." But, also, we must observe a progress: the students are more aware than one year ago [7] that they must write comments in their programs.

We have analysed Software products made by undergraduate students. We are confident that the results cannot be extrapolated to large software systems, but they may be used to provide better instruction for the students, and may be used as effective didactic materials, especially for the course on "Software Metrics". Even if at the level of the first year of study we insist on the necessity of a personal style in programming, and of fulfilling a series of important rules [7, 8], the students are skeptical, they are happy that their programs "work". They do not like to write comments, or to insist on a good design and Pseudocode algorithms, or documentation.

The masters students have seen completely differently the necessity to have a full and correct documentation, its usefulness, the effect of an adequate programming style on the final software products.

By studying the primary data obtained by the masters students we have observed for four projects large discordances between attributes A3 (function points) and A10 (size). By making a more careful analysis of these projects we noticed that they are actually incomplete, since they have only fully implemented a part of the functions required at the specification phase. By not being finalised projects, they have been eliminated from the subsequent data manipulation phases. But this fact in itself underlined another usefulness of assesment of these attributes, as well as the necessity of having some adequate tools for student projects assesment.

Acknowledgements

We would like to acknowledge the support of postgraduate students of the group "Component Based Programming" for their help in analysing the projects.

References

[1] Ronald Baecker, Aaron Marcus, *Design Principles for the Enhanced Presentation of Computer Program Source Text*, CHI'86 Proceedings, 51–58.

[2] Victor R. Basili, Richard W. Selby, David H. Hutchens, *Experimentation in Software Engineering*, IEEE Transactions on Software Engineering, Vol. Se-12 (1986), no.7, 733–743.

[3] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[4] D. Dumitrescu, *Hierarchical pattern classification*, Fuzzy Sets and Systems 28 (1988), 145–162.

[5] A. Dunsmore, M. Roper, *A Comparative Evaluation of Program Comprehension Measures*, EfoCS-35-2000, Department of Computer Science, University of Strathelgde, Glasgow, 2000.
[6] N.E. Fenton, *Software Metrics*. A Rigorous Approach, Int. Thompson Computer Press, London, 1995.
[7] M. Frenţiu, *On programming style – program correctness relation*, Studia Univ. Babes-Bolyai, Series Informatica 45, 2 (2000), 60–66.
[8] M. Frenţiu, *The Impact of Style on Program Comprehensibility*, "Babeş-Bolyai" University of Cluj-Napoca, Research Seminar on Computer Science, 2002, pp. 7–12
[9] M. Frenţiu, H. F. Pop, *A study of licence examination results using Fuzzy Clustering techniques*, Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Research Seminars, Seminar on Computer Science, 2001, 99–106.
[10] Don Gesink, *Software Metrics – The Art of Measuring Software Quality*, 1995, `http://www.interex.org/pubcontent/interact/apr95/pp56.html` and `http://www.interex.org/pubcontent/interact/oct95/05softqu/softqu.html`
[11] D. Gries, *The Science of Programming*, Springer Verlag, Berlin, 1981.
[12] John W. Harbaugh, Daniel F.Merriam, *Computer applications in stratigrafic analysis*, John Wiley & Sons, NewYork, 1968.
[13] G. Lazarovici, M. Frenţiu, *Methods for Automated Classification Use in Archaeology. An Application to Neolithic Graves and Ornaments*, First Romanian Conference on the Application of Physics Methods in Archaeology, Cluj-Napoca, 5–6 November 1987.
[14] H.F. Ledgard, *Programming Proverbs for Fortran Programers*, Hayden Book Company, Inc., New Jersey, 1975.
[15] Steve McConnell, *Software Quality at Top Speed, Software Development*, 1996, `http://www.construx.com/stevemcc`
[16] Richard J. Miara, Joyce A. Musselman, Juan A. Navarro, and Ben Shneiderman, *Program Indentation and Comprehensibility*, Comm. A.C.M., 26 (1983), no.2, 861–867.
[17] Paul W. Oman and Curtis R. Cook, *Typographic Style is More than Cosmetic*, Comm.A.C.M., 33 (1990), no. 4, 506–520.
[18] H.F. Pop, *SAADI: Software for fuzzy clustering and related fields*, Studia Universitatis Babeş-Bolyai, Series Informatica 41, 1 (1996), 69–80.
[19] Armstrong A. Takang, Penny A. Grubb and Robert D.Macredie, *The effects of comments and identifier names on program comprehensibility: an experimental investigation*, Journal of Programming Languages, vol. 4 (1996), 143–167.
[20] Iris Vessey, Ron Weber, *Some Factors Affecting Program Repair Maintenance: An Empirical Study*, Comm. A.C.M., 26 (1983), no. 2, 128–134.
[21] S.Watanabe, H.Haven, *Karhunen-Loeve expansion and Factor analysis. Theoretical remarks and applications*, in Transactions of the Fourth Prague Conference on Information Theory, Statistical Decision Functions, Random Processes, Prague 1967.
[22] L.A. Zadeh, *Fuzzy sets*, Information and Control 8 (1965), 338–353.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, RO-3400 Cluj-Napoca, Romania

E-mail address: `mfrentiu@cs.ubbcluj.ro, hfpop@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

A COST MODEL FOR THE AND-PARALLEL EXECUTION OF LOGIC PROGRAMS

MONICA VANCEA AND ALEXANDRU VANCEA

Abstract. Almost all the results regarding the automatic parallelization of logic programs assume ideal execution environments, focusing only on implicit parallelism detection and not taking into account practical computing system overheads. Trying to overcome such a drawback, we propose in this paper a cost model for the AND-parallel execution of logic programs, which is able to insert at compile time some cost functions which will estimate at run time the parallel execution costs involved. The cost functions are defined based on the particular computing system properties combined with the parallelization process features. If the conditions evaluated by these cost functions are met, the program is allowed to proceed in parallel. If not, it means that parallel execution may even require extra time compared with the sequential execution, so the code will be executed sequentially. We believe that our model is of a very practical importance allowing the run time environment to take the adequate decision with respect to the possibility of AND parallel execution of the (implicit) parallelism present in the logic programs.

1. Preliminaries

Automatic parallelization is the most suitable technique at the moment for exploiting the inherent parallelism from programs written in logic programming languages [Sehr92].

Among the parallel execution models employed at the level of logic programming languages, the AND parallel model raises the most complex problems [Chass84, Herm89, Lin88]. That is because this model is confronted with data dependence relations that appear frequently between the subgoals of a clause [Chang85, Shen92]. That is why, even if from a technically point of view we succeed to solve the actual automatic parallelization problem, the practical costs assumed by such an action (involving intensive data dependence testing followed by the generation of the equivalent parallel code) could be prohibitive.

2000 *Mathematics Subject Classification.* 68Q17, 68Q10, 68M20.

1998 *CR Categories and Descriptors.* D.1.6. [**Software**]: Programming Techniques – *Logic Programming*; D.1.3. [**Software**]: Programming Techniques – *Parallel Programming*; D.2.9. [**Software**]: Software Engineering – *Cost Estimation*; H.3.4. [**Information Systems**]: Information Storage and Retrieval – *Performance evaluation.*

When developing and analyzing parallel logic execution models we assume in general an ideal execution environment, making abstraction of a significant number of supplimentary practical costs (overheads). As examples of such overheads we mention here the task creation overhead, the time cost of task switching between processors, communication costs etc. The drawbacks implied by such practical execution factors may lead not only to the diminishing of the parallel execution performance, but even to a parallel execution time much greater than the corresponding sequential one! If such a situation is met, then parallel processing becomes an inappropriate decision for the program's execution.

In logic programming these issues are much more present in the case of AND parallelism where the possible data dependencies and their management may imply significant costs compared to other types of logic parallelism (like OR parallelism for example where such issues are not so critical [Chass94, Lusk90]).

These are the reason for which we propose in this paper a cost model for the AND parallelization of a sequential logic program. By applying this model we can estimate if the implied parallelization effort and the envisioned execution costs may keep also in practice the theoretical advantages of the parallel execution on the sequential one.

More exactly, we will propose ***a cost model for deriving sufficient conditions for deciding if the AND parallel execution of a particular logic program is an appropriate decision***.

Execution costs control can't be performed entirely as a compile time activity because in the general case these costs depend on the input data. On the other hand, a cost analysis performed entirely at run time risks to conclude non parallel execution in a great number of practical cases, taking into account a too large run time overhead. So, our strategy will be to divide in a reasonable way the workload between compile time and run time phases.

We will define and generate some *cost functions* at compile-time. Their mission will be to *estimate* at run time the total cost of parallel execution relatively to the particular size and nature of the input data, information which will be known at that moment, so possible of be practically taken into account.

2. Definitions and notations

Let S be the goal which we want to be analyzed and let suppose that it is composed of the (sub)goals $(s_1, \ldots, s_n)$ so $S = (s_1, \ldots, s_n)$. For the goal S we denote by:

- C_{seq} – the cost of its sequential execution;
- C_{par} – the cost of its parallel execution.

And we denote by C_i the execution cost for the s_i goal.

The sequential execution of the goal S may be performed only in one way, implying only obbeying the sequential execution order of goals $s_1, \ldots, s_n$ for the constituent subgoals. The parallel execution however, may be performed in many ways, its particular history and development depending on many influences, among these being the number of available processors, the implemented scheduling and memory allocation techniques etc. So when we refer to the cost of the sequential execution it is obvious what we mean, because this is a unique value at the level of a particular computing system. But we cannot refer to a single well defined value when we refer to the cost of the parallel execution, because it can follow diverse paths, one particular execution being selected upon some dynamic criteria.

For this reason and for our analysis to be enough general we will denote by *Cpar* the maximum cost of all possible parallel execution alternatives, that is, *the cost of the most costly parallel execution possibility for goal S.*

Our analysis intends to establish if the subgoals $s_1, \ldots, s_n$ justify their parallel execution. More exactly, from the viewpoint of the parallel execution opportunity, the cost analysis has to verify if the relation $C_{par} \leqslant C_{seq}$ holds or not.

Because of the way in which a parallel execution proceeds (as mentioned above based first of all on dynamic decisions) we cannot really *compute* the value C_{par}, but we have to *estimate* and/or *approximate* it (the notion of *execution time* itself assumes that the exact relation between C_{par} and C_{seq} can be established only after the execution of goal S).

We must decide further the approximation technique to be used. Let $C_{par}^{\sup}$ be an upper limit for the parallel execution cost (in fact we already assumed that $C_{par} = C_{par}^{\sup}$) and let $C_{seq}^{\inf}$ be a lower limit for the sequential execution (the cost of the fastest possible sequential execution of the goal S).

Let's notice that if we succeed to prove for S that $C_{par}^{\sup} \leqslant C_{seq}^{\inf}$ then the same relation holds trivially between the actual execution times also, so running in parallel the subgoals $s_1, \ldots, s_n$ is a correct decision. Mathematically, the relation $C_{par}^{\sup} \leqslant C_{seq}^{\inf}$ becomes a sufficient condition for having $C_{par} \leqslant C_{seq}$, so $C_{par}^{\sup} \leqslant C_{seq}^{\inf}$ ***is a sufficient condition for the decision to run in (AND) parallel the subgoals of a given goal.***

3. Model description

We assume that we have k processors available for the execution of the n subgoals of the S goal.

Definition 3.1. We define ***the processor's average computing load*** as beeing the value $\Im^{=}_{md}\lceil n/k \rceil$, i.e. the number of subgoals that a processor must solve on the average.

Definition 3.2. We define as an upper bound of the total cost of the parallel execution

$$C^{\sup}_{par} = \Re^{\sup}_{c} + T^{\sup}_{par}$$

where $\Re^{\sup}_{c}$ is an upper bound for the creation cost of the tasks associated with the clause's subgoals (we will call it the *task creation overhead*) and $T^{\sup}_{par}$ is an upper bound for the parallel execution time taken by the goal S.

$\Re^{\sup}_{c}$is an architecture dependent value which can be experimentally determined. In general, such a value is a constant or a function depending on some parameters such as the number or size of the input data, the number of manageable tasks etc. We want in the following to approximate the value $T^{\sup}_{par}$.

Let $T^{\sup}_{i}$be an upper limit for the execution time of the goal s_i and let $T^{\sup}_{\max} = \max(T^{\sup}_{1}, \ldots T^{\sup}_{n})$. Obviously, we have then

$$T^{\sup}_{par} \leqslant m\, T^{\sup}_{\max}$$

Also, for every subgoal we have

$$T^{\sup}_{i} = P^{\sup}_{i} + C^{\sup}_{i}$$

where $P^{\sup}_{i}$represents the *scheduling overhead* for the goal s_i (the time passed between the corresponding task creation and the actual starting of its execution) and $C^{\sup}_{i}$ denotes *the effective run-time cost* for the goal s_i, without taking into consideration task creation overhead or the scheduling overhead.

Regarding the $T^{\inf}_{seq}$ value, this can be approximated as follows:

$$T^{\inf}_{seq} = T^{\inf}_{s_1} + \ldots + T^{\inf}_{s_n},$$

where $T^{\inf}_{seq(s_i)}$ denotes a lower bound for the cost of s_i's sequential execution (the best sequential execution for this subgoal).

All the above reasoning can be resumed by the following lemma.

Lemma 3.3. If the following relation holds

$$P^{\sup} + C^{\sup}_{par} \leqslant T^{\inf}_{s_1} + \ldots + T^{\inf}_{s_n}$$

then we have also $C_{par}^{\sup} \leqslant C_{seq}^{\inf}$.

This result can be further relaxed as we will show in the theorem 3.5.

Definition 3.4. By ***goal execution overhead*** we denote the total time taken by the corresponding task creation plus the time taken by the scheduling overhead for a particular goal, that is

$$\Re_e = \Re_c + \Re_P$$

where the goal scheduling overhead is approximated by $\Re_P = \Im_{md} \cdot P_i^{\sup}$.

The main result of this section is presented in theorem 3.5. and it establishes some sufficient conditions for the AND parallel execution of a logic program's clauses.

Theorem 3.5. Let $(s_1,\ldots,s_n)$ be the S goal's subgoals and let $m = \Im_{md}$. If among these subgoals there exists at least $m{+}1$ goals such as $\forall i = 1,\ldots, m{+}1$, $\Re_e \leqslant T_{s_i}^{\inf}$, then we have $C_{par} \leqslant C_{seq}$.

Proof. If we have at least $m{+}1$ subgoals such that $\forall i = 1,\ldots, m{+}1$, $\Re_e \leqslant T_{s_i}^{\inf}$ then it follows that we have at least one subgoal s_j , $j = m{+}1,\ldots, n$, such that $\Re_e \leqslant T_{s_j}^{\inf}$, from where we conclude that even more it holds

$$\Re_e \leqslant T_{s_{m+1}}^{\inf} + \ldots + T_{s_n}^{\inf} \leqslant T_{s_{m+1}} + \ldots + T_{s_n}$$

By adding to the both members the running time for the goals $1\ldots m$ we obtain

$$\Re_e + T_{s_1} + \ldots + T_{s_m} \leqslant T_{s_1} + \ldots + T_{s_m} + T_{s_{m+1}} + \ldots + T_{s_n} \tag{1}$$

Let's recall that $m = \Im_{md}$ (the average computation load for a processor) indicates the average number of subgoals that will be sequentially processed by a processor during the parallel execution of the initial goal.

Because relation (1) holds for any m subgoals for which the execution time is present as a term in the left hand side, it holds in particular also for the case in which those m subgoals are those with the longest execution time. In this latter case, the left hand side of the inequality (1) obviously represents an upper bound for the parallel execution time of the initial goal S (because the parallel execution will take maximum the time taken by the sequential execution of the longest m subgoals at a processor plus the goal execution overhead for the entire S goal). It follows that we have

$$C_{par}^{\sup} \leqslant T_{s_1} + \ldots + T_{s_m} + T_{s_{m+1}} + \ldots + T_{s_n}$$

but the right hand side is nothing else than the sequential execution cost of the goal S, so we have

$$C_{par}^{\sup} \leqslant C_{seq}^{\inf}$$

which shows that the conditions which we gave as hypothesis are truly sufficient conditions for the AND parallel execution of logic programs.

Example 3.6. Let's consider the following program code sequence:

```
q([ ], [ ]).
q([H|T], [X|Y]) :-
        X is H + 1,
        q(T,Y).

r([ ], [ ]).
r([X|RX], [X2|RX1]) :-
        X1 is X * 2,
        X2 is X1 + 7,
        r(RX,RX1).
```

We will consider as an estimation of the execution cost (or more precisely as a unit of measure for this cost) of a goal the number of resolution steps required for proving it. Then the execution costs for the predicated q and r may be estimated upon the following cost functions:

```
Cost q(n) = 2n + 1
Cost r(n) = 3n + 1
```

We consider the AND parallel goal ... q(X,Y) & r(X)... expressed as in [Herm91] in which the argument list represents the set of the input arguments and not the arity of those predicates.

Based on the results of the theorem 3.5 the initial code sequence can be translated to

... length(X, LX), cost_q is LX2=1, cost_r is LX3+1,

(cost_q > Re(q), cost_r > Re(r) → q(X,Y) & r(X); q(X,Y), r(X)),...

where Re(q) and Re(r) denote respectively the parallel goal execution overhead and cost_q and cost_r represent the sequential execution costs for goals q and r respectively. The adnotation of the initial code sequence with such a condition allows in this moment to decide at run time whether or not to AND parallel execute the goals that follow the tested condition.

To conclude: the practical usefullness of our results from theorem 3.5 resides in the possibility to apply source code transformations at compile time which will insert the necessary tests to be performed at run time. These tests will decide upon the adequacy of running in AND parallel the adnotated sequence of goals.

4. Conclusions

Ideal execution environments are assumed when methods for automatic parallelization of logic programs are studied. In developing such methods, the focus is directed towards implicit parallelism detection and only very few models are taking into account the costs implied by practical computing system overheads. Estimating such costs are nevertheless of a critical importance because we can meet situations in practice for which parallel execution would be more time consuming than the equivalent sequential one. That is why we proposed in this paper a cost model for the AND-parallel execution of logic programs, which using adnotations capabilities inserts at compile time some cost functions which will perform at run time a good estimation of the parallel execution costs involved. The cost functions are defined and generated based on the computing system properties and on parallelization process features. If at run time the conditions evaluated by these cost functions are met, the program is allowed to proceed in parallel. If not, it means that parallel execution will not provide the expected speedup, so the code will be executed sequentially. We believe that our model is a very practical one, allowing the run time environment to take the adequate decision with respect to the possibility of AND parallel execution of the (implicit) parallelism present in the logic programs.

References

[Chang85] J-H. Chang, A.M. Despain, D. DeGroot, *And-Parallelism of Logic Programs based on Static Data Dependency Analysis*, in Digest of Papers of COMPCON, Spring 1985, pp. 218–225.

[Chass94] J. Chassin de Kergommeaux, P. Codognet, *Parallel Logic Programming Systems*, in ACM Computing Survey, vol.26, no.3, Sept.94, pp. 295–336.

[Herm89] M.V. Hermenegildo, F. Rossi, *On the Corectness and Efficiency of Independent AND-Parallelism in Logic Programs*, in Proc. of the 1989 North American Conf. on Logic Programming, 1989, pp. 369–389.

[Herm91] M.V. Hermenegildo and L.Greene, *The &-prolog System: Exploiting Independent And Parallelism*, New Generation Computing, 9 (3, 4), 1991 pp. 233–257.

[Lin88] Y. J. Lin, V. Kumar, *AND-parallel execution of Logic Programs on a shared Memory Multiprocessor: A Summary of Results*, in Fifth International Logic Programming Conference, Seatle, WA, 1988, pp. 1106–1120.

[Lusk90] E. Lusk, S. Haridi, D.H.D. Warren et. al., *The Aurora OR-Prolog System*, New Generation Computing, vol.7, no.2,3, 1990 pp. 243–273.

[Sehr92] D.C. Sehr, *Automatic Parallelization of Prolog Programs*, Ph.D. dissertation, Univ.of Illinois at Urbana-Champaign, 1992.

[Shen92] K. Shen, *Exploiting Dependent And-Parallelism in Prolog: The Dynamic, Dependent And-Parallel Scheme*, in Proc. Joint Int'l. Conf. and Symp. On Logic Prog. MIT Press, 1992, pp. 717–731.

Faculty of Economic Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: vancea@econ.ubbcluj.ro

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

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

ON THE CONVERGENCE OF ASYNCHRONOUS BLOCK NEWTON METHODS FOR NONLINEAR SYSTEMS OF EQUATIONS

IOAN LAZĂR

Abstract. Convergence of asynchronous block Newton methods for solving nonlinear systems of equations of the form $F(x) = 0$ are studied. Sufficient conditions to guarantee their local convergence are given. Our analysis emphasizes the connection between the conditions on F involved in local convergence theorems for sequential and synchronous block Newton's method, and our settings for asynchronous block Newton methods. Our results are similar to the results of Szyld and Xu, obtained in an asynchronous nonlinear multisplitting context.

Keywords: numerical analysis, iterative methods, nonlinear system of equations, Newton methods

1. Introduction

Consider the parallel solution of nonlinear systems of equations of the form

$$F(x) = 0, \tag{1}$$

where $F = (f_1, \ldots, f_n) : \Omega \subseteq \mathbf{R}^n \to \mathbf{R}^n$ is a nonlinear operator. Newton's method is based on the approximation $F(x) \approx F(x^k) + F'(x^k)(x - x^k)$, and is given by the iteration

$$x^{k+1} = x^k - F'(x^k)^{-1}F(x^k), \tag{2}$$

for $k = 0, 1, \ldots$, where x^0 is an initial guess.

Each linear system (2) can be solved in parallel using some kind of block iterative methods [10]. Block iterative methods are studied using the concept of multisplittings [10] and the application of block iterative methods for solving the systems (2) at each Newton step k was considered in [14].

The application of the concept of multisplittings directly to the nonlinear system (1) were considered in [6] and [2] These methods are called parallel synchronous nonlinear multisplitting methods. The methods are called synchronous in the sense

---

2000 *Mathematics Subject Classification.* 65H10, 65Y05.

1998 *CR Categories and Descriptors.* G.1.0 [**Numerical analysis**] General: *parallel algorithms*; G.1.5 [**Numerical analysis**] Roots of Nonlinear Equations: *Iterative methods, Convergence, Systems of equations*.

that all processors have to wait at some synchronization point before proceeding to the next iteration.

The asynchronous nonlinear multisplitting methods were considered in [1] and [12], i.e. methods where no synchronization barrier is present (see [5, 3, 7] for some general discussions on asynchronous methods). Bahi et all [1] studied asynchronous nonlinear multisplitting methods in a general context for nonlinear fixed point problems, while Szyld and Xu [12] studied these methods for problems of the form (1), and extended the study to the case of overlapping blocks, i.e., certain variables are updated by more than one processors.

Our framework presented here is similar to the framework used by Xu [15] for the study of asynchronous block quasi-Newton methods. Our analysis emphasizes the connection between the conditions on F involved in local convergence theorems for sequential and synchronous block Newton's method, and those used for asynchronous block Newton methods.

This paper is organized as follows: in section 2 we give a brief review of block Newton methods, a computational model for asynchronous block (Newton) methods and a corresponding mathematical model. The main result is presented in section 3, after a brief review of the tools used or study both synchronous and asynchronous cases. Finally some connections with different asynchronous block Newton type methods are discussed.

2. Asynchronous Block Newton Methods

Suppose F and x are conformally partitioned as follows $F = (F_1, \ldots, F_L)$, $x = (x_1, \ldots, x_L)$, where $x_i = (x_{i_1}, \ldots, x_{i_{n_i}}) \in \mathbf{R}^{n_i}$ and $F_i : \mathbf{R}^n \to \mathbf{R}^{n_i}$, $i = 1, \ldots, L$. Suppose the partition $S_i = \{i_1, \ldots, i_{n_i}\}$, $i = 1, \ldots, L$ is chosen such that $\cup_{i=1}^{L} S_i = \{1, \ldots, n\}$ and $S_i \cap S_j = \emptyset$ for $i \neq j$, $i, j = 1, \ldots, L$.

The system (1) can be rewritten

$$F_l(x_1, \ldots, x_l, \ldots, x_L) = 0, l = 1, \ldots, L. \tag{3}$$

and we consider the following nonlinear block method. Given initial values $x = (x_1, \ldots, x_L)$, repeat the following procedure until convergence

For $l = 1, \ldots, L$

$$\begin{cases} \text{Solve for } y \text{ in } F_l(x_1, \ldots, x_{l-1}, y, x_{l+1}, \ldots, x_L) = 0, \\ \text{Set } x_l = y. \end{cases} \tag{4}$$

In (4) the order in which the block are updated could be arbitrary. The classical nonlinear block-Jacobi method and block-Gauss-Seidel method [4, 11] are special cases of such methods. For the purpose of parallel processing the nonlinear block-Jacobi method is nearly ideal, since up to L processors can each perform one of the iterations in (4). Such iterations are synchronous in the sense that to begin the computation of the next iterate, each processor has to wait until all processors have completed their current iteration. By removing the synchronization and letting the pocessors continue their calculations according to the information currently available, we obtain asynchronous parallel methods.

Let the Jacobian of F be partitioned conformally with x, and define

$$F'(x) = \begin{pmatrix} \frac{\partial F_1(x)}{\partial x_1} & \cdots & \frac{\partial F_1(x)}{\partial x_L} \\ \cdots & \cdots & \cdots \\ \frac{\partial F_L(x)}{\partial x_1} & \cdots & \frac{\partial F_L(x)}{\partial x_L} \end{pmatrix}, \frac{\partial F_i(x)}{\partial x_j} = \begin{pmatrix} \frac{\partial f_{i_1}(x)}{\partial x_{j_1}} & \cdots & \frac{\partial f_{i_1}(x)}{\partial x_{j_{n_j}}} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_{i_{n_i}}(x)}{\partial x_{j_1}} & \cdots & \frac{\partial f_{i_{n_i}}(x)}{\partial x_{j_{n_j}}} \end{pmatrix},$$

The block diagonal matrix of $F'(x)$ is denoted by

$$D(x) = diag(\frac{\partial F_1(x)}{\partial x_1}, \ldots, \frac{\partial F_L(x)}{\partial x_L}).$$

Using the above notations, one Newton step applied to the system (4) and starting from the initial value x is

$$y_l = x_l - \left(\frac{\partial F_l(x)}{\partial x_l}\right)^{-1} F_l(x). \tag{5}$$

Note that when solving (5) we are only interested in the components x_l corresponding to S_l. This means we work with a system of dimension n_l, although the initial system (1) is of dimension n. The evaluation of $F_l(x)$ in (5) is dependent on the entire vector x, that is the processor solving the equation (5) needs the components evaluated by other processors.

2.1. **Computational Model.** Denote the (approximate) solution of $F_l(x_1, \ldots, x_{l-1}, y, x_{l+1}, \ldots, x_L) = 0$ by $y_l = G_l(x)$, $l = 1, \ldots, L$. Applying one step of Newton method gives the operator defined by (5).

Assume we are working with a (shared memory) parallel computer with L processors and associate a block of components with each processor. Then a parallel variant of (3) can be implemented as in Algorithm 1. If the processors would wait for each other to complete each run through the loop we would get a parallel synchronous implementation of the procedure (3).

Here the processors continue the loop by collecting the needed vectors computed by the other processors according to the information available at the moment. A computational model for the asynchronous block method can be written as the pseudocod of Algorithm 1 shows.

Since the processors do not wait for each other, the processors get out of phase due to different run times for each loop. At a given time point, different processors will have achieved different number of iterations. In this context, the iteration number k in (2) looses its meaning.

Using a direct linear solver for step 4 in Algorithm 1, for exemple an LU factorization of F', we obtain the *asynchronous block Newton method*

4'a: Factor $F_l'(x) = LU$
4'b: Solve $LUs = -F_l(x)$
4'c: $y_l := x_l + s$

Any other appropriate factorization such as QR or Cholesky could be used as well.

Algorithm 1 Pseudocode for the lth processor ($l = 1, \ldots, L$). x represents the initial guess x_j, $j = 1, \ldots, L$. x and *convergence* are global variables written in common memory.

```
1: read(converge)
2: while not converge do
3:    read(x)
4:    y_l = G_l(x)
5:    x_l := y_l; overwrite(x_l)
6:    read(converge);
7: end while
```

2.2. **Mathematical Model.** In order to analyse the asynchronous computational model presented in Algorithm 1 we consider a counter k which is updated every time a new vector is computed by some processor and let $x_l^0 = x^0$, $l = 1, \ldots, L$.

Let $I^k \subseteq \{1, \ldots, L\}$ denotes all updated block components, then the asynchronous block Newton iteration is defined by

$$x_i^{k+1} = \begin{cases} x_i^{s_i(k)} - \left(\frac{\partial F_i(u)}{\partial x_i}\right)^{-1} F_i(u) & \text{for } i \in I^k, \\ x_i^k & \text{for } i \notin I^k, \end{cases} \tag{6}$$

for $i \in \{1, \ldots, L\}$, $k = 0, 1, \ldots$, where $u = (x_1^{s_1(k)}, \ldots, x_L^{s_L(k)})$.

The iteration counts $s_i(k)$, $i = 1, \ldots, L$ indicate the iteration, prior to k, when the ith block component was computed.

Let $S = \{(s_1(k), \ldots, s_L(k)) \in \mathbf{N}^L\}_{k \in \mathbf{N}}$ where $\mathbf{N} = 0, 1, \ldots$ denotes the set of natural numbers. The *standard assumptions* for $I = \{I^k\}_{k \in \mathbf{N}}$ and S are:

$$\forall i \in \{1, \ldots, L\}, \forall k \in \mathbf{N}, s_i(k) \leq k, \tag{7}$$

$$\forall i \in \{1, \ldots, L\}, \lim_{k \to \infty} s_i(k) = \infty, \tag{8}$$

$$\forall i \in \{1, \ldots, L\}, \text{ the set } \{k \in \mathbf{N} | i \in I^k\} \text{ is infinite.} \tag{9}$$

The next definitions are similar to those considered by El Tarazi in [13] and will be used in our proofs.

We define the sequence $\{s(k)\}_{k \in \mathbf{N}} \subset \mathbf{N}$ by

$$s(k) = \min_i s_i(k). \tag{10}$$

We obtain immediately from (7) and (8)

$$s(k) \leq k \text{ and } \lim_{k \to \infty} s(k) = \infty. \tag{11}$$

Suppose that (7)–(9) are satisfied, then we can define an increasing sequence $\{k_l\}_{l\in\mathbf{N}}$ having the properties

$$\bigcup_{0\le s(k)\le k<k_0} I^k = \{1,\dots,L\}, \tag{12}$$

$$\bigcup_{k_l\le s(k)\le k<k_{l+1}} I^k = \{1,\dots,L\}. \tag{13}$$

The proofs given by Baudet [3] and El Tarazi [13] for general asynchronous iterations use the sequence $\{k_l\}$ defined above. This sequence says that the asynchronous iteration (6) updates all block components at least once at the steps $k_0, k_1, \dots$.

If $k_{l+1} - k_l = L$ for all l, we get a synchronous block Gauss-Seidel iteration, and if the sequence of differences $\{k_{l+1} - k_l\}$ is bounded then we get a partially asynchronous algorithm.

3. Local Convergence

3.1. **Synchronous Newton Methods.** The standard assumptions on F in synchronous (or sequential) case are:

(C1): Equation (1) has a solution x^*.
(C2): $F' : \Omega \to \mathbf{R}^{n\times n}$ is Lipschitz continuous on Ω, with Lipschitz constant γ, i.e., $\|F'(x) - F'(y)\| \le \gamma\|x - y\|$, for all $x, y \in \Omega$.
(C3): $F'(x^*)$ is nonsingular.

These assumptions can be weakened without sacrificing convergence results presented here. However the classical result on quadratic convergence of Newton's method requires them.

The main result concerning the local convergence of Newton's method is presented in the next theorem.

Theorem 3.1. [11] *Let the standard assumptions (C1)–(C3) hold. Then there are $K > 0$ and $\delta > 0$ such that if $\|x^0 - x^*\| < \delta$ then the Newton iterates $\{x^k\}$ defined by (2) converge q-quadratically to the solution x^* of (1).*

The convergence results on Newton's method follow from the basic results given in Lemma 3.2 and 3.3 (a variant of Banach lemma).

Lemma 3.2. [11] *Assume F satisfies (C2). Then for all $x, y \in \Omega$,*

$$\|F(y) - F(x) - F'(x)(y - x)\| \le \frac{\gamma}{2}\|x - y\|^2. \tag{14}$$

In the context of Theorem 3.1, the inequality (14) is used to obtain the estimates

$$\|x^{k+1} - x^*\| \le K\|x^k - x^*\|^2,\ k = 0, 1, \dots. \tag{15}$$

Lemma 3.3. [11] *Let $A, C \in \boldsymbol{R}^{n\times n}$, A nonsingular and $\|A^{-1}\| \leq \alpha_1$, $\|C - A\| \leq \alpha_2$ with $\alpha_1\alpha_2 < 1$. Then C is a nonsingular matrix, and*

$$\|C^{-1}\| \leq \frac{\alpha_1}{1 - \alpha_1\alpha_2}.$$

The hypothesis of theorem 3.1 does not give sufficient conditions for solving subsystems $F_l(x) = 0$, $l = 1, \ldots, L$ of the system $F(x) = 0$, since the subsystem $F_l(x) = 0$ is solved only in respect to the components of block l.

The asynchronous iteration (6) is more close related to other Newton type methods which consider some splitting of the Jacobian,

$$F'(x) = B(x) - C(x). \tag{16}$$

and iterative processes

$$x^{k+1} = x^k - B(x^k)^{-1}F(x^k), k = 0, 1, \ldots. \tag{17}$$

The Newton-SOR and Newton-Jacobi belong to this family of iterative processes. Ortega and Rheinboldt establish the following result concerning the iteration (17).

Theorem 3.4. [11] *Let the standard assumptions (C1)–(C3) hold, and suppose $B : \Omega \to L(\boldsymbol{R}^n)$ is continuous in x^*, $B(x^*)$ is nonsingular and $\rho(B(x^*)^{-1}(F'(x^*) - B(x^*))) < 1$. Then $\{x^k\}$ defined by (17) and (16) converges q-linearly to x^* with q-order $\rho(B(x^*)^{-1}(F'(x^*) - B(x^*)))$.*

3.2. **Weighted maximum norms.** The assumptions (C1) and (C2) are also naturally for asynchronous block Newton method. The condition (C3) will be replaced by the following sufficient conditions which guarantee the existence of solutions of the subsystems and local convergence of the asynchronous method:

(C3'): All the matrices $\frac{\partial F_i(x^*)}{\partial x_i}$, $i = 1, \ldots, L$ are nonsingular, and

$$\rho(|D(x^*)^{-1}(F'(x^*) - D(x^*))|) < 1.$$

Remarks. (a) (see also [15]) Conditions (C1) and (C2) are standard for Newton methods, and (C3') is natural for the convergence of asynchronous methods. Consider the linear case, where $F(x) = Ax - b$ and $F'(x) = A$. If there exists A^{-1} then (C3') is necessary and sufficient for the convergence of the asynchronous block methods for the linear system $F(x) = 0$.

(b) Condition (C3') is also similar to the main requirement for the convergence given in the theorem 3.4, $\rho(B(x^*)^{-1}(F'(x^*) - B(x^*))) < 1$.

(c) (see also [15]) Condition (C3') holds when the Jacobian matrix $F'(x^*)$ is an H-matrix, since $F'(x^*) = D(x^*) - (D(x^*) - F'(x^*))$ is an H-splitting of $F'(x^*)$. Moreover, (C3') is equivalent to $F'(x^*)$ being an H-matrix if each block has only one component.

By the theory of nonnegative matrix, condition (C3') is quivalent to

(C3"): All matrices $\frac{\partial F_i(x^*)}{\partial x_i}$, $i = 1, \ldots, L$ are nosingular, and there exists $\rho_0 < 1$ and a vector $w > 0$ such that

$$\|D(x^*)^{-1}(F'(x^*) - D(x^*))\|_w < \rho_0.$$

The weighted maximum norms used in (C3") are defined as follows. Let $w \in R^n$, $w > 0$ and $A \in \mathbf{R}^{n\times n}$ be partitioned conformally with x, then we define

$$\|x\|_w = \max\{\|x_i\|_{w_i}, 1 \le i \le L\} = \max\{\frac{|x_{i_j}|}{w_{i_j}}, 1 \le j \le n_i, 1 \le i \le L\}. \tag{18}$$

and the induced matrix norm $\|A\|_w = \max\{\frac{\|Ax\|_w}{\|x\|_w} : x \in \mathbf{R}^n \setminus \{0\}\}$.

We return to the subsystems (5). Starting from x and applying one step of the Newton method for the ith block, we are interested to estimate

$$\begin{aligned} \|y_i - x_i^*\|_{w_i} &= \|x_i - x_i^* - \left(\frac{\partial F_i(x)}{\partial x_i}\right)^{-1} F_i(x)\|_{w_i} \\ &= \| \left(\frac{\partial F_i(x)}{\partial x_i}\right)^{-1} \left[F_i(x^*) - F_i(x) - \frac{\partial F_i(x)}{\partial x_i}(x_i^* - x_i)\right] \|_{w_i} \end{aligned} \tag{19}$$

As we can see from the right hand side of (19), the Lemma 3.2 cannot be applyed directly as for the sequential Newton method.

In order to obtain a similar lemma we can extend the weighted matrix norms for rectangular matrices as follows,

$$\|(A_{i1}, \ldots, A_{iL})\|_{w_i} = \max\{\frac{\|(A_{i1},\ldots,A_{iL})x\|_{w_i}}{\|x\|_w} : x \in \mathbf{R}^n \setminus \{0\}\},$$
$$\|A_{ij}\|_{w_i} = \max\{\frac{\|A_{ij}x_j\|_{w_i}}{\|x_j\|_{w_j}} : x_j \in \mathbf{R}^{n_j} \setminus \{0\}\}.$$

We immediately have,

$$\|A_{ii}(A_{i1}, \ldots, A_{iL})\|_{w_i} \le \|A_{ii}\|_{w_i} \cdot \|(A_{i1}, \ldots, A_{iL})\|_{w_i},$$
$$\||A|\|_w = \|A\|_w, \||(A_{i1}, \ldots, A_{iL})|\|_{w_i} = \|(A_{i1}, \ldots, A_{iL})\|_{w_i},$$
$$\|A\|_w = \max\{\|(A_{i1}, \ldots, A_{iL})\|_{w_i}, 1 \le i \le L\},$$
$$\|A_{ij}\|_{w_i} \le \|(A_{i1}, \ldots, A_{iL})\|_{w_i}, 1 \le i \le L.$$

These extensions were considered by Xu [15]. The following lemma will play a similar role for asynchronous block Newton methods as the lemma 3.2 for sequential Newton methods.

Because of the norm equivalence in finite dimensional spaces we can consider that the norm used in (C2) is the weighted norm $\|\cdot\|_w$, where w is the vector defined in (C3").

Lemma 3.5. [15] *Under the conditions (C1), (C2) and (C3') we have*

$$\|\frac{\partial F_i(x)}{\partial x_i} - \frac{\partial F_i(x^*)}{\partial x_i}\|_{w_i} \le \gamma\|x - x^*\|_w, \quad \forall x \in S(x^*, \epsilon), \tag{20}$$

and there exists $\epsilon > 0$ such that $S(x^*, \epsilon) \subset \Omega$ and

$$\| \left(\frac{\partial F_i(x^*)}{\partial x_i}\right)^{-1} \left(\frac{\partial F_i(x)}{\partial x_1}, \ldots, \frac{\partial F_i(x)}{\partial x_{i-1}}, 0, \frac{\partial F_i(x)}{\partial x_{i+1}}, \ldots, \frac{\partial F_i(x)}{\partial x_L}\right) \|_{w_i} \le \rho_0, \tag{21}$$

$$\begin{aligned}(22)\qquad &\|x_i - x_i^* - \left(\tfrac{\partial F_i(x^*)}{\partial x_i}\right)^{-1} F_i(x)\|_{w_i} \\ &\le \rho_0\|x - x^*\|_w + \tfrac{\gamma}{2}\|\left(\tfrac{\partial F_i(x^*)}{\partial x_i}\right)^{-1}\|_{w_i}\cdot\|x - x^*\|_w^2,\end{aligned}$$

for all $i = 1, \dots, L, x \in S(x^*, \epsilon)$.

3.3. **Asynchronous Newton Method.** The next theorem represents the main result of the paper.

Theorem 3.6. *Let the assumptions (C1), (C2) and (C3'), and also the conditions (7)–(9) hold. Then there exists* $\delta > 0$ *such that if* $x^0 \in S(x^*, \delta)$ *then the sequence generated by asynchronous block Newton method converges to* x^*.

Moreover, for $l = 0, 1, \dots$, *we have*

$$(23)\qquad \|x^k - x^*\|_w \le r^l\|x^0 - x^*\|_w, \forall k \ge k_l,$$

where $K > 0$, $r := \rho_0 + K\delta < 1$, *and the sequence* $\{k_l\}$*is defined by (12)–(13).*

Proof. We proceed in two steps: first we show that the sequence generated by asynchronous block Newton method is well defined and then it converges.

We consider $\beta > 0$ such that

$$\|F'(x^*)\|_w \le \beta, \quad \|(\frac{\partial F_i(x^*)}{\partial x_i})^{-1}\|_{w_i} \le \beta,\ i = 1, \dots, L.$$

First part. We choose $\delta > 0$ such that the matrices $\frac{\partial F_i(x)}{\partial x_i}$, $i = 1, \dots, L$ are nonsingular for all $x \in S(x^*, \delta)$. From (20),

$$\|\frac{\partial F_i(x)}{\partial x_i} - \frac{\partial F_i(x^*)}{\partial x_i}\|_{w_i} \le \gamma\|x - x^*\|_w, \quad \forall x \in \Omega,$$

and by (C3') there exists $\frac{\partial F_i(x^*)}{\partial x_i}^{-1}$. Let $\delta > 0$ be such that the hypothesis of Banach lemma 3.3 hold, so there exists $\frac{\partial F_i(x)}{\partial x_i}^{-1}$, for all $x \in S(x^*, \delta)$. Now we choose δ small enough such that the assumptions of Lemma 3.5 also hold.

We show that if $x^0 \in S(x^*, \delta)$ then the sequence $\{x^k\}$ remains in $S(x^*, \delta)$. Suppose that for all j, $0 \le j \le k$, $\|x^j - x^*\|_w \le \|x^0 - x^*\|_w$.

Let $u = (x_1^{s_1(k)}, \dots, x_L^{s_L(k)})$. For $i \in I^k$,

$$\begin{aligned}(24)\qquad \|x_i^{k+1} - x_i^*\|_{w_i} &= \|x_i^{s_i(k)} - x_i^* - \tfrac{\partial F_i(u)}{\partial x_i}^{-1} F_i(u)\|_{w_i} \\ &= \|x_i^{s_i(k)} - x_i^* - \tfrac{\partial F_i(x^*)}{\partial x_i}^{-1} F_i(u)\|_{w_i} + \\ &\quad \|\tfrac{\partial F_i(x^*)}{\partial x_i}^{-1} - \tfrac{\partial F_i(u)}{\partial x_i}^{-1}\|_{w_i}\cdot\|F_i(u)\|_{w_i}\end{aligned}$$

Since

$$\begin{aligned}(25)\qquad \|F_i(u)\|_{w_i} &= \|F_i(u) - F_i(x^*)\|_{w_i} \le \|F(u) - F(x^*)\|_w \\ &\le \|F(u) - F(x^*) - F'(x^*)(u - x^*)\|_w + \|F'(x^*)(u - x^*)\|_w \\ &\le \tfrac{\gamma}{2}\|u - x^*\|_w^2 + \beta\|u - x^*\|_w,\end{aligned}$$

$$\begin{aligned} \|\tfrac{\partial F_i(x^*)}{\partial x_i}^{-1} - \tfrac{\partial F_i(u)}{\partial x_i}^{-1}\|_{w_i} &= \|\tfrac{\partial F_i(x^*)}{\partial x_i}^{-1}(\tfrac{\partial F_i(x^*)}{\partial x_i} - \tfrac{\partial F_i(u)}{\partial x_i})\tfrac{\partial F_i(u)}{\partial x_i}^{-1}\|_{w_i} \\ &\le \beta^2\gamma\|u - x^*\|_w \end{aligned} \tag{26}$$

we get

$$\begin{aligned} \|x_i^{k+1} - x_i^*\|_{w_i} &\le \left[\rho_0 + \left(\beta\gamma(\tfrac{1}{2} + \beta^2) + \beta^2\gamma^2\|u - x^*\|_w\right)\|u - x^*\|_w\right]\|u - x^*\|_w \\ &= (\rho_0 + K\|u - x^*\|_w)\,\|u - x^*\|_w, \end{aligned} \tag{27}$$

where $K = \beta\gamma(\frac{1}{2}+\beta^2)+\beta^2\gamma^2\delta$. Again, if necessary, we choose δ small enough such that $r := \rho_0 + K\delta < 1$, then (27) gives $\|x_i^{k+1} - x_i^*\|_{w_i} \le \|x^0 - x^*\|_w$, for $i \in I^k$.

On the other hand, if $i \notin I^k$ then the ith component is not modified, $x_i^{k+1} = x_i^k$, and from the induction hypothesis it follows $\|x_i^{k+1}-x_i^*\|_{w_i} \le \|x^0-x^*\|_w$, for $i \notin I^k$. The last two inequalities together with the norm definitions implies

$$\|x^{k+1} - x^*\|_w \le \|x^0 - x^*\|_w,$$

that means the sequence $\{x^k\}$ generated by the asynchronous method is well defined and remains in $S(x^*, \delta)$ if $x^0 \in S(x^*, \delta)$.

Second part. Let $\{k_l\}$ be the sequence defined by (12) and (13). We show by mathematical induction that for all $l \in \mathbf{N}$

$$\|x^k - x^*\|_w \le r^l\|x^0 - x^*\|_w, \forall k \ge k_l. \tag{28}$$

hence $\{x^k\}$ is convergent, since $r = \rho_0 + K\delta < 1$.

Let $l = 0$. From the definition of $\{k_l\}$ it follows

$$\begin{aligned} &\forall k \ge k_0,\ \forall i \in \{1, \dots, L\},\ \text{there exists } j : 0 \le s(j) \le j < k \\ &\text{such that } x_i^k = x_i^{j+1} \text{ and } i \in I^j. \end{aligned}$$

Using (27) we get $\|x_i^k - x_i^*\|_{w_i} = \|x_i^{j+1} - x_i^*\|_{w_i} \le r\|x^0 - x^*\|_w$, for $i \in \{1, \dots, L\}$, and by the definition of weighted norms, $\|x^k - x^*\|_w \le r^1\|x^0 - x^*\|_w \le, \forall k \ge k_0$.

Now, suppose for fixed $l \in \mathbf{N}$ we have

$$\|x^k - x^*\|_w \le r^l\|x^0 - x^*\|_w, \forall k \ge k_l.$$

Using again the definition of $\{k_l\}$ we get

$$\begin{aligned} &\forall k \ge k_{l+1},\ \forall i \in \{1, \dots, L\},\ \text{there exists } j : k_l \le s(j) \le j < k \\ &\text{such that } x_i^k = x_i^{j+1} \text{ and } i \in I^j. \end{aligned}$$

Let $u = (x_1^{s_1(j)}, \dots, x_L^{s_L(j)})$. Using again (27), $\|x_i^k - x_i^*\|_{w_i} = \|x_i^{j+1} - x_i^*\|_{w_i} \le r\|u - x^*\|_w = r\|x_p^{s_p(j)} - x^*\|_{w_p}$, where p is an index for which the last equality holds (according to the definition of the norm $\|\cdot\|_w$). Since $s_p(j) \ge k_l$, it follows that $\|x_p^{s_p(j)} - x^*\|_{w_p} \le r^l\|x^0 - x^*\|_w$ and the proof is complete. □

Remark Under the assumptions of Theorem 3.6, the asynchronous block Newton method converges with a rate of convergence (see [3]).

$$R = \liminf_{k\to\infty}[(-\log\|x^k - x^*\|)/k] \ge \rho_0.$$

If F and F' are computed innacurately then the asynchronous iteration (6) becomes, for $i \in I^k$,

$$x_i^{k+1} = x_i^{s_i(k)} - \left(\frac{\partial F_i(u)}{\partial x_i} + \Delta(u)\right)^{-1} (F_i(u) + \epsilon(u)), \tag{29}$$

where $u = (x_1^{s_1(k)}, \ldots, x_L^{s_L(k)})$. A similar local convergence theorem can be shown for an asynchronous block Newton perturbed method defined by (29) (see [9]). As in the sequential case [8], one can use Newton perturbed method to derive local convergence results for other Newton methods (e.g. chord method).

References

[1] Jacques Bahi, Jean-Claude Miellou, and Karim Rhofir. Asynchronous multisplitting methods for nonlinear fixed point problems. *Numerical Algorithms*, 15:315–345, 1997.

[2] Zhong-Zhi Bai, Violeta Migallon, Jose Penades, and Daniel B. Szyld. Block and Asynchronous Two-Stage Methods for Midly Nonlinear Systems. *Num. Math.*, 82:1–21, 1999.

[3] Gérard M. Baudet. Asynchronous iterative methods for multiprocessors. *J. Association for Computing Machinery*, 25:226–244, 1978.

[4] Dimitri P. Bertsekas. *Distributed Asynchronous Computation of Fixed Points*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[5] D. Chazan and W. Miranker. Chaotic relaxation. *Linear Algebra and its Applications*, 2:199–222, 1969.

[6] Andreas Frommer. Parallel Nonlinear Multisplitting Methods. *Numerische Mathematik*, 56:269–282, 1989.

[7] Andreas Frommer and Daniel B. Szyld. On Asynchronous Iterations. *J. Computational and Applied Mathematics*, 123:201–216, 2000.

[8] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM Publications, 1995.

[9] Ioan Lazăr. On Asynchronous Two-Stage Newton Iterative Methods. *Rev. Anal. Numér. Théor. Approx.* submitted.

[10] D.P. O'Leary and R.E. White. Multi-Splitting of Matrices and Parallel Solution of Linear Systems. *SIAM J. Alg. Disc. Meth.*, 6:630–640, 1985.

[11] J.M. Ortega and W.G. Rheinboldt. *Iterative Solutions of Nonlinear Equations in Several variables*. Academic Press, New York, 1970.

[12] Daniel B. Szyld and Jian-Jun Xu. Convergence of Some Asynchronous Nonlinear Multisplitting Methods. *Numerical Algorithms*, 25:347–361, 2000.

[13] Mouhamed Nabih El Tarazi. Some Convergence Results for Asynchronous Algorithms. *Num. Math.*, 39:325–340, 1982.

[14] R.E. White. Parallel Algorithms for Nonlinear Problems. *SIAM J. Alg. Disc. Meth.*, 7:137–149, 1986.

[15] Jian-Jun Xu. Convergence of Partially Asynchronous Block Quasi-Newton Methods for Nonlinear Systems of Equations. *J. Computational and Applied Mathematics*, 103:307–321, 1999.

Babeş-Bolyai University, Faculty of Mathematics and Informatics, Cluj-Napoca, str. M. Kogălniceanu 1

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

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

FRINGED-QUADTREES: A NEW KIND OF DATA STRUCTURE

CLARA IONESCU

Abstract. In our everyday life we have to deal with different problems that, in most cases, need new data structures. At first sight, these structures do not look like any known data structure. This paper presents a data structure wehave called "fringed-quadtree". This structure is a tree with nodes that may be roots or leaves. A root-node may have at most four leaves. These trees may be built considering some rules that are presented in the paper. Due to the specific queries, the pointers will be ascending for the root-nodes and descending for the leaves. The time complexity of the described algorithms is logarithmic or linear and the memory space needed has the order $O(n)$.

1. Introduction

Data from database management systems are processed using special software programs. These have a lot of tools, but, in order to retrieve data, the users often need to create data structures, which will maintain the hierarchy between the elements and will perform searching as quick as possible. For example, the well-known *multilevel marketing systems (MLM)* are working on basis of various rules. These systems need to be able to retrieve records based on some hierarchy, (which are not stored explicitly in the database) in order to calculate the financial rights of persons from the system.

In this paper we present the fringed-quadtree, designed in order to have a suitable data structure for such a database. The quadtree, in a conventional approach, is a tree structure where every node may have at most four descendents. It was introduced for spatial data by Finkel and Bentley [Fink74].

This paper is organized as follows. We first define (section 2) the requirements of the model and the issues that must be considered in choosing a representation. This depends on the nature of the queries involving them, and on the type of operations that must be performed to answer them. Section 3 describes the manner of the building of such quadtrees. We discuss the implementation issues of the building in the subsection 4.0.1. For implementation we propose appropriate

2000 *Mathematics Subject Classification.* 68P05, 68P20.

1998 *CR Categories and Descriptors.* E.1 [**Data**]: Data Structures; H.2.1 [**Information Systems**]: Database Management – *Logical Design.*

search structures, for example balanced binary search trees [Adel62], [Knut73], [Wirt76] or B-trees [Come79]. Section 4 describes possible query types, and subsection 4.0.2 presents their implementations. Section 5 evaluates the performances of the algorithms (the storage and execution time requirements).

2. Problem Description

We will build a data structure having the following properties:

(1) Based on some specific rules, the elements are grouped in *buckets*. A bucket consists of at most five elements.
(2) The data associated to the nodes follows a hierarchy depending on the time-factor (the insertion time) and the parent-node.
(3) The structure contains two types of links (pointers) between its elements. The first link type specifies the *parent* of a given element. Obviously, each node has a single such pointer. The second link type is used for pointing to the descendents of a node; this is a *child*-type link. There will be four such pointers for each node, because they are used for retrieving the elements in the bucket corresponding to a parent-node.
(4) The structure contains two types of nodes: *roots* and *leaves*. *We define a root as a node that is referred as parent by at least one node, and a leaf as a node that is not referred as parent by any node in the structure.* The proper tree-structure consists of root-nodes, where each root is the nucleus of its own bucket. Such abucket may contain at most four other nodes that are leaves. These leaves are "hanged on the root like some fringes".
(5) The dynamics used for building the structure leads to a quad-tree spanned (using the pointers) from its bottom part to its upper part, because only the *parent* pointers may be linked between the root-type nodes of the tree.
(6) When a new element is inserted, its *parent* must be given. This way, the hierarchical position of the new node is specified. The insertion of a new element may cause some changes in the bucket of its parent, as follows:
    (a) If the *parent* is a leaf, then the new node becomes a leaf and takes the place of *its own parent*. After this "replacement", the *parent* pointers do not change, but the parent of the inserted node is no longer the child of a node (a leaf), but it becomes a root; hence, it is now a node in the quad-tree.
    (b) If the element is *not* the first leaf of its parent, it becomes one of the children (leaves) of the *parent*.

It follows that we have a quad-tree that is built based on ascending pointers and each node may have at most four descending pointers. We may notice that

a node may be referred as *parent* by five nodes, but the first such node "leaves" the parent's bucket. *The specificity of the dynamic used for building the buckets consists in the fact that the first child Z of a leaf Y becomes a child of the node X that referred Y as its child.* Hence, the parent of a leaf may be the node that contains the leaf in its bucket, or another root on the *parent pointers* path.

3. Building Fringed-Quadtrees

Initially, the structure consists of a single element, denoted by A. By convention, A is the only node predefined having root-type from the beginning. The bucket of the first element is built in a slightly different manner than the other buckets because the first element of the tree does not have a parent. Hence, it will be directly referred as *parent* by at most four nodes.

We suppose that, at each moment of time, only one node referring a certain parent may be inserted. Due to the fact that each existing node may be specified as *parent*, at a certain moment of time, the number of insertions may be equal to the number of nodes in the tree that do not have complete buckets.

Let we see the way a bucket is built. *After building the first bucket, no more leaves having as direct parent the root of the bucket may be inserted.* The structure of the buckets may change because when new insertions are performed, the *existent leaves* are replaced by their own new leaves.

We denote by X_0 the first node that refers X as its *parent* and by X_i, $i = 1 \dots 4$ the other nodes that refer X as their *parent*. The first child X_0 of X became child of the parent of X, and because A has no parent, A_0 does not exist. We denote the first child of X_i by X_{i0}, the others four children by X_{ij}, $j = 1 \dots 4$. For a better view of the notes we will rename the nodes wich became of root-type.

(1) The root A becomes the parent of the first leaf A_1.

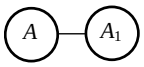


Figure 1: The first two nodes

(2) A is referred as the parent of the new node A_2. A_1 has a leaf, so it is referred as the parent of A_{10}; at this moment A_1 gains its independence and leaves the bucket of A, becoming a root (B). A_{10}, its first child, becomes a leaf of A.
(3) A refers to A_3 as its child; B (formerly A_1) is referred as parent by B_1 and A_2 is referred as parent by A_{20}; A_2 gains its independence and leaves the bucket of A becoming a root (C); A_{20} becomes a leaf of A. A_{10} is

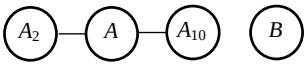


Figure 2: A_1 becomes a root (B)

referred as parent by A_{100}; A_{10} also gains independence and leaves the bucket of A becoming a root (D); its first child, A_{100}, becomes a leaf of A.

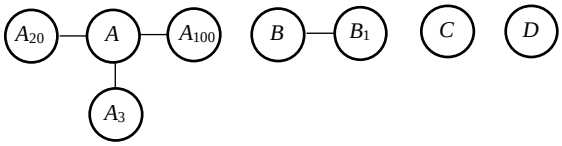


Figure 3: The structure has eight nodes; the bucket of A contains three leaves and the bucket of B contains one leaf; C and D do not yet have any leaf. A, B, C and D are root-nodes.

(4) A_4 is the new child of A, B_2 is the new child of B, C_1 is the new child of C and D_1 is the new child of D; then, A_{100} is referred as parent by A_{1000} and leaves the bucket of A, becoming a root (E), and A_{1000} becomes a leaf of A. A_{20} is referred as the parent by A_{200}; A_{20} becomes the root F and A_{200} becomes a leaf of A. A_3 is referred as parent by A_{30}; it becomes a root (G) and A_{30} becomes a leaf of A; B_1 is referred as parent by B_{10}, so it becomes a root (H) and B_{10} becomes a leaf of B.

The first loop ends here (the bucket of A is complete). One should not substitute "complete" with "finalized", because this term refers only to the number and it does not refer to the content of the bucket. A may not be referred as parent by any new node, but the content of its bucket changes due to the leaves of the leaves of A.

(5) We suppose that, in the next step, all nodes (except A, because its bucket is complete), roots and leaves, are referred as parents by new elements.

Figure 5 shows the bucket of A has the same number of elements, but A_{1000} was replaced by A_{10000}, A_{200} was replaced by A_{2000}, A_{30} was replaced by A_{300} and A_4 was replaced by A_{40}. Simultaneously, the sizes of the buckets of B, C, D, E, F, G, H increase and the leaves that became roots are I, J, K, L, M, N, P, O.

One may notice that at each step the number of nodes is increased by the number of root type nodes not having complete buckets and the leaves.

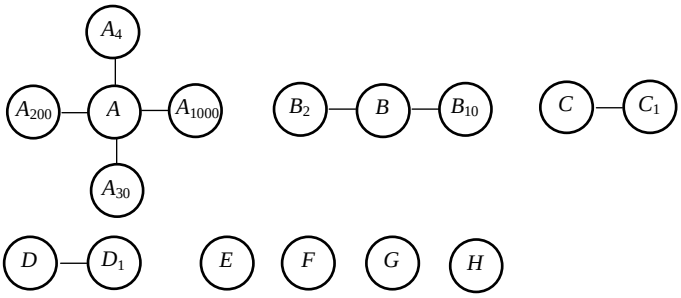


Figure 4: The links between the nodes after the fourth step of the insertions. There are eight roots (A, B, C, D, E, F, G, H) and eight leaves (A_{1000}, A_{200}, A_{30}, A_4, B_{10}, B_2, C_1, D_1).

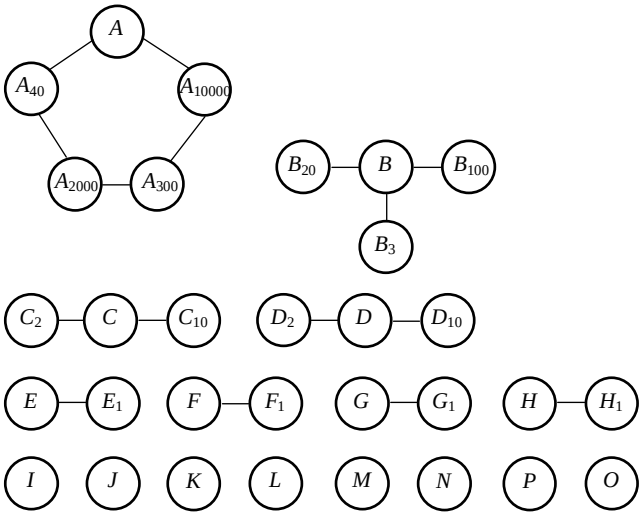


Figure 5: The leaves I, J, K, L, M, N, P, O became roots

The place (position) of an inserted node depends only on the node referred as parent by the node.

Obviously, the order in which the insertions are performed is not necessarily the same as in the example. It is possible to have some nodes referred as parents at most five times in a row and others to stay leaves for a long time.

4. Possible Queries

First of all, the problem of building such a structure must be considered.

Without reducing the generality, we may suppose that the information corresponding to a node is contained in a field called *name*. When designing such a data structure it is obvious that the possible queries that may be performed on the database modelled using this structure have to be considered. It is known that searching queries must be supported by such a database. We suppose that all queries, except for insertions, are performed on a single *name*. The insertion is performed using a search (we look for a node that has as field *name* the value specified as the parent of the new node); hence, two *names* will be given (for the new node and for the parent).

There are many types of queries, such as:

(1) Given a *name*, check whether it exists or not in the database.
(2) Given a *name*, find the *name* of its parent.
(3) Given a *name*, check whether it corresponds to a root or to a leaf; if it corresponds to a leaf, find the content of its bucket (the names of the leaves).
(4) Given a *name*, find all nodes that are ancestors of a node specified by its *name*. The *ancestors* of a node are the parent of the parent (the grandparent), the parent of the grandparent, etc., until the first inserted node is reached.
(5) Given a *name*, find all the names of the nodes for which the node corresponding to the *name* is an ancestor.

At first sight, the structure may be viewed as groups of five elements where we should know the root of each group. We might keep the roots in an alphabetically sorted list and keep trace of the children. In this way, the hierarchy hidden in the model is lost. From a logical point of view, this hierarchy is due to the fact that each node is inserted in the structure as a child of another node. But, here there are no "bosses" and no "subordinates".

In addition, the first child Z of a node Y is "given" to the node X that referred to Y as one of its children. From the logical point of view, X is on a superior level with respect to Y, hence Z and Y become "siblings", even if it looks like Z should be a descendent of Y. Hence, an implementation that uses tree-like data structures in which child-type pointers are used, does not correspond to the real situation from a logical point of view.

We propose an implementation that allows the buckets to be part of the structure (the content of the buckets is changeable) and also allows the hierarchy to be saved.

The root-nodes are maintained in a tree-like structure in which each node represents a bucket consisting in at least one and at most five nodes. For a node, we have the information field (*name*) and six pointers. One of them points to the *parent* specified when the node was inserted. The other five are child-type pointers. The first of them points to the node that was replaced at the insertion time. The other four point to the leaves (the other nodes in the bucket).

The tree-like structure is a quad-tree because each bucket consists in five nodes, one of them being the root of the bucket; hence, each node has at most four "descendents".

4.0.1. *Building Fringed-Quadtrees. Implementation.* We describe the way the quadtree is built based on a sequence of insertions. The following figures show the insertions presented in the example from section 3.

At the first step A_1 is inserted as child of A. For the node A the *parent* pointer is `nil` and *child*[1] will point to A_1. The other child-type pointers are `nil`. For the node A_1, *parent* points to A and the other pointers are `nil`. We use the following convention: the pointer corresponding to the first child of A points to A, in order to have a value different than `nil`.

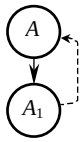


Figure 6: The first step

name	*parent*	*child*[0]	*child*[1]	*child*[2]	*child*[3]	*child*[4]
A	`nil`	A	A_1	`nil`	`nil`	`nil`
A_1	A	`nil`	`nil`	`nil`	`nil`	`nil`

For the node A the second child-type reference appears: *child*[2] points to A_2. The node A_2 is created in the same way as the node A_1 at the previous step. Due to the fact that A_1 refers to A_{10} as its first child, the pointer from the node A to A_1 is replaced by a pointer to A_{10}. Obviously, *parent* of the node A_{10} points to A_1. The node A_1 remains unchanged because its *parent* remains A. Its child-type pointers are `nil` because there are no leaves in its bucket.

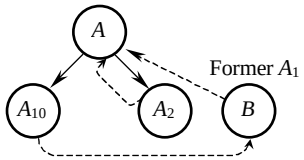


Figure 7: The second step

name	*parent*	*child*[0]	*child*[1]	*child*[2]	*child*[3]	*child*[4]
A	`nil`	A	A_{10}	A_2	`nil`	`nil`
B (formerly A_1)	A	A_{10}	`nil`	`nil`	`nil`	`nil`
A_2	A	`nil`	`nil`	`nil`	`nil`	`nil`
A_{10}	B	`nil`	`nil`	`nil`	`nil`	`nil`

A_3 is inserted as a leaf of A, so *child*[3] of the node A points to A_3. B (formerly A_1) receives its first leaf B_1 (the first node that referred it as *parent* became a leaf of A at the previous step), hence *child*[1] of the node B points to B_1. The pointer *parent* of the node B_1 points to B (the former A_1). A_{20}, the first descendent of A_2 is inserted so, *child*[2] of the node A points to A_{20}. The node A_2 (now C) remains unchanged, its *parent* field still points to A. A child of A_{10} (A_{100}) is inserted, hence the field *child*[1] from A is changed; now, it points to A_{100}. The node A_{100} is created in such a way that its *parent* field points to A_{10} (now D).

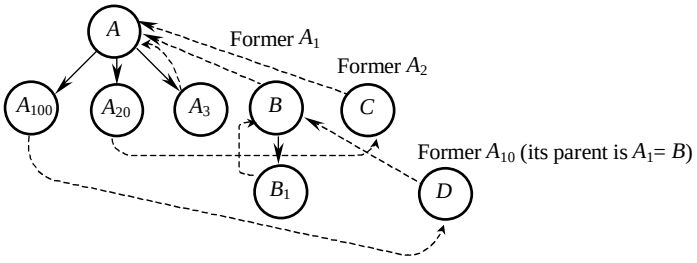


Figure 8: There are four root-type nodes and four leaves

name	*parent*	*child*[0]	*child*[1]	*child*[2]	*child*[3]	*child*[4]
A	`nil`	A	A_{100}	A_{20}	A_3	`nil`
B (formerly A_1)	A	D	B_1	`nil`	`nil`	`nil`
C (formerly A_2)	A	A_{20}	`nil`	`nil`	`nil`	`nil`
D (formerly A_{10})	B	A_{100}	`nil`	`nil`	`nil`	`nil`
A_3	A	`nil`	`nil`	`nil`	`nil`	`nil`
B_1	B	`nil`	`nil`	`nil`	`nil`	`nil`
A_{20}	C	`nil`	`nil`	`nil`	`nil`	`nil`
A_{100}	D	`nil`	`nil`	`nil`	`nil`	`nil`

At the next step, we suppose that each of the eight nodes in the structure is referred as *parent* by a new node that must be inserted. The tree has the pointers presented in the following table:

name	*parent*	*child*[0]	*child*[1]	*child*[2]	*child*[3]	*child*[4]
A	`nil`	A	A_{1000}	A_{200}	A_{30}	A_4
B	A	D	B_{10}	B_2	`nil`	`nil`
C	A	G	C_1	`nil`	`nil`	`nil`
D	B	H	D_1	`nil`	`nil`	`nil`
E (formerly A_3)	A	A_{30}	`nil`	`nil`	`nil`	`nil`
F (formerly B_1)	B	B_{10}	`nil`	`nil`	`nil`	`nil`
G (formerly A_{20})	C	A_{200}	`nil`	`nil`	`nil`	`nil`
H (formerly A_{100})	D	A_{1000}	`nil`	`nil`	`nil`	`nil`
A_4	A	`nil`	`nil`	`nil`	`nil`	`nil`
B_2	B	`nil`	`nil`	`nil`	`nil`	`nil`
C_1	C	`nil`	`nil`	`nil`	`nil`	`nil`
D_1	D	`nil`	`nil`	`nil`	`nil`	`nil`
A_{30}	E	`nil`	`nil`	`nil`	`nil`	`nil`
B_{10}	F	`nil`	`nil`	`nil`	`nil`	`nil`
A_{200}	G	`nil`	`nil`	`nil`	`nil`	`nil`
A_{1000}	H	`nil`	`nil`	`nil`	`nil`	`nil`

One may notice that the bucket of A is complete, that is A cannot have any more leaves that refer it as *parent*. But, when elements referring as parents the leaves of A are inserted, the new elements replace the leaves of A, becoming part of the bucket.

In order to clarify the way the fringed quad-tree is built, the next table presents an extra *optional* step.

name	*parent*	*child*[0]	*child*[1]	*child*[2]	*child*[3]	*child*[4]
A	nil	A	A_{10000}	A_{2000}	A_{300}	A_{40}
B	A	D	B_{100}	B_{20}	B_3	nil
C	A	G	C_{10}	C_2	nil	nil
D	B	H	D_{10}	D_2	nil	nil
E (formerly A_3)	A	M	E_1	nil	nil	nil
F (formerly B_1)	B	N	F_1	nil	nil	nil
G (formerly A_{20})	C	P	G_1	nil	nil	nil
H (formerly A_{100})	D	Q	H_1	nil	nil	nil
I (formerly A_4)	A	A_{40}	nil	nil	nil	nil
J (formerly B_2)	B	B_{20}	nil	nil	nil	nil
K (formerly C_1)	C	C_{10}	nil	nil	nil	nil
L (formerly D_1)	D	C_{10}	nil	nil	nil	nil
M (formerly A_{30})	E	A_{300}	nil	nil	nil	nil
N (formerly B_{10})	F	B_{100}	nil	nil	nil	nil
P (formerly A_{200})	G	A_{2000}	nil	nil	nil	nil
Q (formerly A_{1000})	H	A_{10000}	nil	nil	nil	nil
B_3	B	nil	nil	nil	nil	nil
C_2	C	nil	nil	nil	nil	nil
D_2	D	nil	nil	nil	nil	nil
E_1	E	nil	nil	nil	nil	nil
F_1	F	nil	nil	nil	nil	nil
G_1	G	nil	nil	nil	nil	nil
H_1	H	nil	nil	nil	nil	nil
A_{40}	I	nil	nil	nil	nil	nil
B_{20}	J	nil	nil	nil	nil	nil
C_{10}	K	nil	nil	nil	nil	nil
D_{10}	L	nil	nil	nil	nil	nil
A_{300}	M	nil	nil	nil	nil	nil
B_{100}	N	nil	nil	nil	nil	nil
A_{2000}	P	nil	nil	nil	nil	nil
A_{10000}	Q	nil	nil	nil	nil	nil

We recall that in this example we considered all possible insertions that *may be* performed at each moment of time even if this is not compulsory.

We now present an image of the fringed-quadtree described in the previous table. For a better view, the *parent* pointers of the leaves were removed.

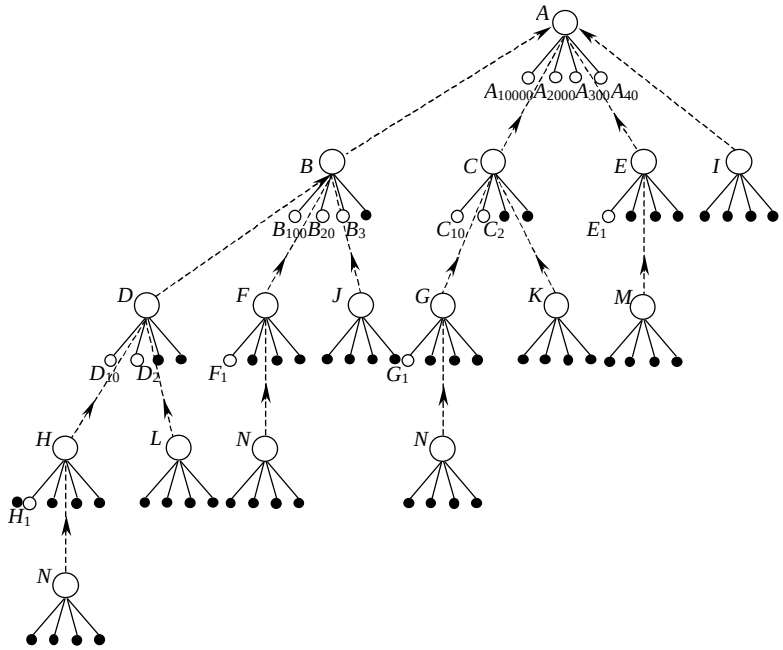


Figure 9: The fringed-quadtree

Due to the hierarchy established by the *parent* pointers, the spanning must be performed from the bottom side to the upper-side of the quad-tree. The spanning may be optimised if data is inserted in a search structure such as a balanced binary searching tree or a B-tree [Ione91]. This means that the memory space needed to store a node increases due to the new pointers that refer to the search structure and other needed fields (for example, the factor of balance).

4.0.2. *Queries implementation.* We suppose that we have as searching key the field *name* of the nodes.

(1) For the first type of query we must check whether a given *name* is contained in the database. A search for this *name* will be performed in the search structure.
(2) For the second type of query we must retrieve the *parent* of a given node. If the node is in the search structure, than the *name* field of the node referred by the pointer *parent* is returned.
(3) For the third type of query we must establish whether a node is a root or a leaf. If, for that node, we have $child[0] = \texttt{nil}$, it follows that the

node is a leaf; otherwise, the node is a root. This type of query also asks the children of a root (the bucket content) to be retrieved. For a leaf, the algorithm halts here. For a root, the *names* of the nodes referred by the pointers $child[i]$, $i = 1 \dots 4$ are returned. Obviously, it is not compulsory to have acomplete bucket, so will be returned the nodes pointed by $child[i] \neq$ `nil`.

(4) For the fourth type of query we must retrieve all the ancestors of a given node identified by its *name*.
   (a) We retrieve the given node (query of first type).
   (b) After finding the node we "climb" the tree until we reach the node having the value `nil` for the *parent* pointer. All the *names* of the nodes on the "way-up" are returned.

(5) For the fifth type of query we must retrieve all the nodes having as one of the ancestors a node identified by its *name*. Apparently, this is the inverse of the previous query where all the ancestors of a node had to be found. A closer look leads to the conclusion that, in fact, this is not an inverse, but a generalization. We must find paths that link a node (not the "oldest" ancestor) to certain nodes for which we *do not know* the *names*.
   (a) At the first step we retrieve the node corresponding to the *name* and return it.
   (b) If the pointer $child[0]$ has the value `nil`, the node is a leaf and the algorithm halts.
   (c) Otherwise, we return all the nodes that refer (directly or not) the node $child[0]$ as *parent*, which means we recursively call the algorithm for the node referred by $child[0]$. For the pointers $child[i]$, we "climb", using the *parent* pointers, until we reach a pointer to a child of the current node. At the next step we apply the spanning algorithm for this node.

```
1:  procedure SPANNING(r)
2:      write r^.name
3:      if r^.child[0] ≠ nil then
4:          SPANNING(r^.child[0])
5:          for i = 1, 4, 1 do
6:              if r^.child[i] ≠ nil then
7:                  p ← r^.child[i]
8:                  while p^.parent ≠ r do
9:                      p ← p^.parent
10:                 end while
```

```
11:                 SPANNING(p);
12:             end if
13:         end for
14:     end if
15: end procedure
```

5. Complexity Analysis

The memory space needed has the order of magnitude $O(n)$ because for each element we need a node. For the implementation we do not necessarly have to use dynamic memory allocation. We might build a database in which the records contain (apart from the corresponding information) six pointers. For the actual implementation we should find an efficient way to store the leaves (the nodes having all child-type pointers set to `nil`). This is not a waste of time because in a fringed quad-tree there may be $4n/5$ leaves.

Analysing the algorithms for the first three types of queries, it follows that the first step has a time complexity of $O(\log n)$, the time needed for a search in a structure similar to a balanced binary search tree [Ione91].

For the fourth query we have an algorithm running in $O(\log n + h)$ time, where h is the number of nodes returned. For the last query the algorithm runs in $O(\log n + m)$ time, where m is the number of nodes having the given node as ancestor. It follows that the algorithms for the fourth and fifths type of query have the order of magnitude $O(n)$ for the worst case.

For the fourth type of query the worst case is finding the ancestors of the only leaf in a fringed-quadtree in which all nodes (except for the leaf) have exactly one child (all the nodes of the fringed-quadtree must be returned).

For the fifth type of query the worst case is finding all the nodes having the fringed-quadtree root as ancestor (all the other nodes in the fringed-quadtree must be returned).

6. Further work

In this paper we described operations such insertion, search and several types of queries. Since deletion of a leaf-type node is trivial and the deletion of a root-type node is not allowed in the real model, we did not consider for the moment this operation. Obviously, we must be able to delete records from any kind of database, so the next issue will be to find a way of records deletion from a database implemented with fringed-quadtrees, without loosing the hierarchy. While developing the application, we will try to optimize as much as possible all the details regarding the implementation of the fringed-quadtrees.

References

[Adel62] **Adelson-Velskii, G.M., Landis, E.M.:** *An algorithm for the organisation of information*, Soviet Mathematics Doklady, 3:1259-1263, 1962.

[Come79] **Comer, D.:** *The ubiquitous B-tree*, ACM Computing Surveys 11, 2 (June 1979), 121–137.

[Fink74] **Finkel, R.A., i Bentley, J.L.:** *Quad trees: A data structure for retrieval on composite keys*, Acta Informatica, 4:1-9, 1974.

[Ione91] **Ionescu, C., Zsakó, I.** *Structuri arborescente cu aplicaţiile lor*, Editura Tehnică, Bucureşti 1991.

[Knut73] **Knuth, D.E.:** *The Art of Computer Programming, vol.I, Fundamental Algorithms*, Second Edition, Addison-Wesley, Reading, MA, 1973.

[Wirt76] **Wirth, N.:** *Algorithms + Data Structures = Programs*, Prentice Hall, 1976.

Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: clara@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 2, 2002

MATHEMATICAL MODELS FOR ORGANIZING DATA COLLECTIONS

ILEANA TĂNASE

Abstract. Mathematical organisation of data collections is based on three models: vector processing, logical and probabilistic. Vector processing model, materialised in the SMART system implementation has the best mathematical basis. In this model entities and queries have a vectorial representation and some similarities can be established between them based on the comparison of attached vectors. The similar entities will have answers for the same requests and will be searched together. On this observation the cluster hypothesis of van Rijbergen and Sparck is based. This hypothesis suggests detecting entities class as a way for increasing the efficiency of the search.

Key words: similarity measure, dissimilarity measure, clustering, criterion function.

1. Introduction

Mathematical organisation of data collections is based on three models: vector processing, logical and probabilistic. Vector processing model [4,5], materialised in the SMART system implementation has the best mathematical basis. In this model entities and queries have a vectorial representation and some similarities can be established between them based on the comparison of attached vectors. The similar entities will have answers for the same requests and will be searched together. On this observation the cluster hypothesis of van Rijbergen [7] and Sparck [6] is based. This hypothesis suggests detecting entities class as a way for increasing the efficiency of the search.

Consider a data collection $X = \{x^1, x^2, \ldots, x^d\}$. Each entity x^j is identified by one or more index terms. Each entity x^j is represented by a d-dimensional vector:

$$X^j = (x_1^j, x_2^j, \ldots, x_d^j),$$

where the values of x_i^j are restricted to 0 and 1 (x_i^j equals 0 if the i-th index terms is not assigned to the entity, and it equals 1 only if it is assigned to the entity).

2000 *Mathematics Subject Classification.* 62H30.

1998 *CR Categories and Descriptors.* I.5.3. [**Computing Methodologies**]: Pattern Recognition – *Clustering.*

For a better performance in retrieval of entities it is useful that entities be clusterised according to appropriate criteria. A space of entities could be represented as in Figure 1.

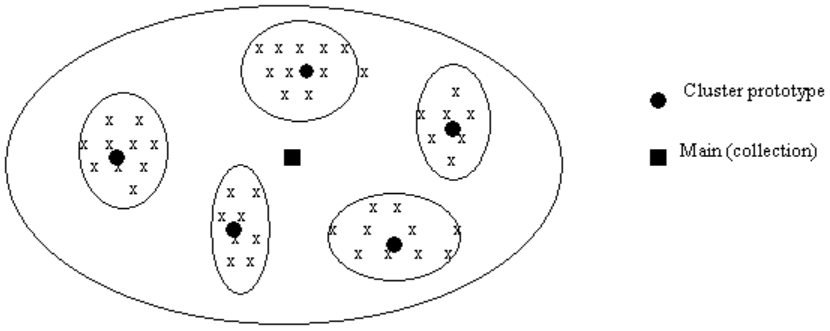


FIGURE 1. Collection and prototype representation

One must take into account that entities of the same class centre have similar characteristics. Thus, the best retrieval performance must be obtained for data collections consisting of individual compact classes, but with great distance between class prototypes.

Splitting a space of entities into classes can be done using several determinist classifying methods. These methods are mainly optimisation procedures of some criterion functions. In order to build a criterion function we may consider that each class is represented by a geometrical prototype. In the vector processing model [4, 5], in which classes have an approximately spherical shape, the prototypes will be points in the Euclidean $\mathbb{R}^d$ space.

2. Similarity measures

Let X be the space of entities to be classified. A similarity measure over X is a function $S : X \times X \to \mathbb{R}$, which satisfies the following axioms:

a) $S(x, y) \geqslant 0, \forall x, y \in X$,
b) $S(x, y) = S(y, x), \forall x, y \in X$,
c) $S(x, x) = S(y, y) > S(x, y), \forall x, y \in X, x \neq y$.

The most used similarity measure in vector processing model is considered the angle cossinus between two vectors:

$$S_1 = \frac{< x, y >}{\|x\| \cdot \|y\|} = \frac{x^T y}{\|x\| \cdot \|y\|}$$

But as shown before, the vectors x, y have binary components. When all the characteristics are binary, there is a set of well known similarity measures. These

measures are based on the following values:

$$s = \sum_{i=1}^{d} x_i \cdot y_i,$$

which represents the number of index terms that simultaneously exist in x and y, in the same way:

$$t = \sum_{i=1}^{d} (1 - x_i)(1 - y_i),$$

represents the number of index terms which simultaneously miss from the x and y entities,

$$u = \sum_{i=1}^{d} x_i(1 - y_i),$$

represents the number of index terms that exist in x, but they miss from y,

$$v = \sum_{i=1}^{d} y_i(1 - x_i),$$

represents the number of index terms that exist in y, but they miss from x.

It is easy to show that:

$$s + t = x^T x$$

and

$$s + v = y^T y$$

Taking account of the prior features, the meaning of the following similarity measures is easy to understand [2]:

$$S_2 = \frac{s}{s + \frac{1}{2}(u + v)},$$

$$S_3 = \frac{s}{s + 2(u + v)},$$

$$S_5 = \frac{st - uv}{st + uv}.$$

3. The criterion function

Let $X = \{x^1, x^2, \ldots, x^p\}$ be the entities set that must be classified. Our aim is to find the cluster structure of the given set. The cluster structure of the set X can be done by a partition $P = \{A_1, A_2, \ldots, A_n\}$ of X. Each member of the partition P will correspond to an entity class. Using a similarity measure we can build a criterion function. The classification problem is reduced to an optimization problem.

Each A_i class can be represented by a prototype L_i, and denote by $L = \{L_1, L_2, \ldots, L_n\}$. Consider the representation of the P partition. In the vector processing model the classes have almost spherical shape and a class prototype will be a point in $\mathbb{R}^d$. This point is the same with the centre of the class, as shown in Figure 1.

A dissimilarity measure on X is a function $D : X \times X \to \mathbb{R}$, that satisfies the following axioms:

a) $D(x, y) \geqslant 0, \forall x, y \in X$,
b) $D(x, x) = 0, \forall x \in X$,
c) $D(x, y) = D(y, x), \forall x, y \in X$.

The criterion function (J) may be defined as [2]:

$$J(P, L) = \sum_{i=1}^{n} \sum_{x \in A_i} D(x, L_i), \tag{1}$$

where D is a dissimilarity measure (for instance, a distance on $\mathbb{R}^d$).

4. The n-mean algorithm

The n-mean algorithm is a very popular clustering technique. The following dissimilarity measure is considered:

$$D(x, y) = \|x - y\|^2.$$

The dissimilarity between a point x and the L_i prototype can be interpreted as error when the point x is approximated by the class prototype L_i. This dissimilarity can be written down as follows:

$$D(x, L_i) = \|x - L_i\|^2.$$

The criterion function will be in this case:

$$J(P, L) = \sum_{i=1}^{n} \sum_{x \in A_i} \|x - L_i\|^2. \tag{2}$$

Using the notation:

$$A_{ij} = \begin{cases} i, & x^j \in A_i \\ 0, & otherwise, \end{cases} \tag{3}$$

the criterion function will be:

$$J(P, L) = \sum_{i=1}^{n} \sum_{j=1}^{p} A_{ij} \left\|x^j - L_i\right\|^2 . \tag{4}$$

Taking into account that in Euclidian space, the scalar product has the form

$$\langle x, y \rangle = x^T M y,$$

where M is a symmetrical and positive defined matrix, the criterion function becomes:

$$J(P,L)=\sum_{i=1}^{n}\sum_{j=1}^{p}A_{ij}(x^j-L_i)^TM(x^j-L_i). \tag{5}$$

From the minimum condition

$$\frac{\partial J(P,L)}{\partial P}=0,\ i=1,\ldots,n \quad , \tag{6}$$

we have

$$-2\sum_{j=1}^{p}A_{ij}M(x^j-L_i)=0,\ i=1,\ldots,n \quad . \tag{7}$$

But the matrix M is nonsingular. Thus we obtain:

$$\sum_{j=1}^{p}A_{ij}x^j-\sum_{j=1}^{p}A_{ij}L_i=0,\ i=1,\ldots,n \quad . \tag{8}$$

From (8) we obtain:

$$L_i=\frac{\sum\limits_{j=1}^{p}A_{ij}x^j}{\sum\limits_{j=1}^{p}A_{ij}}\ i=1,\ldots,n. \tag{9}$$

But $p_i=\sum\limits_{j=1}^{p}A_{ij}$ represents the number of elements of the class A_i. L_i can also be written as:

$$L_i=\frac{1}{p}\sum_{x\in A_i}x. \tag{10}$$

We can now see that the prototype L_i is the mass centre of the A_i class. The representation $L=\{L_1,L_2,\ldots,L_n\}$, where L_i is given by (9) induces a new partition. This partition is obtained using *the nearest neighbour (NN) rule.*

If

$$\left\|x^j-L_i\right\|<\left\|x^j-L_k\right\|,\ k=1,\ldots,n,k\neq i \tag{11}$$

then $x^j$is assigned to the class A_i.

We may also write:

$$A_{ij}=\begin{cases}1,\ if\ \left\|x^j-L_i\right\|\leqslant\left\|x^j-L_k\right\|,\forall k\neq i\\ 0,\ otherwise\end{cases} \tag{12}$$

The n-mean algorithm consists of applying iteratively the equalities (9) and (12), starting from an initial partition of the set X. This partition can be arbitrarily chosen.

As a conclusion, we may say that determinist clustering methods allow the entities arrangement into classes which verify the following conditions:

a) the similarity between entities in a class is high;
b) the average similarity between class centres is low.

References

[1] Chang Y.K., Cirillo C., *"Evaluation of feedback retrieval using modified freezing, residual collection, and test and control groups"*, Englewood Cliffs, Prentice Hall Inc., 1991
[2] Dumitrescu D., *"Mathematical principles of Classification Theory"*, Ed. Academiei Române, Bucuresti, 1999
[3] Popovici M., Rican G., *"Proiectare si implementare software"*, Teora, 1998
[4] Salton G., *"Automatic Information Organization and Retrieval"*, McGraw-Hill, New York, 1975
[5] Salton G., Yang C.S., *"Contribution to the theory of indexing"*, American Elsevier, New York, 1980
[6] Sparck J., *"Automatic Keyword Classification for Information Retrieval"*, Butterworts, London, 1981
[7] van Rijsbergen C.J., *"Information retrieval"*, Butterworths, London, 1982
[8] Wong A., *"An investigation of the effects of differnt indexing methods on the document space configuration"*, Cambridge, England, 1987
[9] Veryard R., *"Information modelling – practical guidance"*, Prentice Hall, 1992

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 1, 2004

APOLOGY ON PLAGIARISM PAPERS

THE EDITORS

Since the preceding issue has been send to print we have found out, and have been informed by more interested readers that the following papers are plagiates:

- D. Marcu, *The Chromatic Number of Triangle-Free Regular Graphs*, Studia Universitatis Babeş-Bolyai Series Informatica, 47 (1), 2002, p. 54–56.
- D. Marcu, *A Note on the Chromatic Number of a Graph*, Studia Universitatis Babeş-Bolyai Series Informatica, 47 (2), 2002, p. 105–106.
- D. Marcu, *A Note on the Chromatic and Independence Number of a Graph*, Studia Universitatis Babeş-Bolyai Series Informatica, 48 (2), 2003, p. 11–16.

According to practices currently in place, these papers have been reviewed, as always, by a panel of two experts. They have made all possible effort to ensure the scientific quality and accuracy of the papers submitted to the journal. However, we are not always able to verify the originality of every paper submitted, and, as usually, this rests with the responsibility of the author.

After a careful consideration, we have decided to retract the papers under scrutiny; the papers will be marked as such on the journal web page. As we have lost the confidence in Mr. Dănuţ Marcu, the author of these plagiates, we have decided to ban Mr. Marcu from publishing in our journal.

We are apologizing to the international scientific community for this situation. Despite this, we are ensuring our readers that we are continually working to ensure a high scientific quality for our journal. As such, they may continue to consider our journal as the journal of their choice.

The Editors

Faculty of Mathematics and Computer Science, Department of Computer Science, Babes-Bolyai University, 400084 Cluj-Napoca, Romania

E-mail address: studia-i@cs.ubbcluj.ro

Received by the editors: May 15, 2004.