DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Técnico

RT-MAC-9517

AUTOMATIC PROGRAMMING OF MMACH'S FOR OCR*

Junior Barrera, Routo Terada,
Flávio Soares Corrêa da Silva
and Nina Sumiko Tomita,

Dezembro 95

AUTOMATIC PROGRAMMING OF MMACH'S FOR OCR*

JUNIOR BARRERA, ROUTO TERADA,
FLÁVIO SOARES CORRÊA DA SILVA and NINA SUMIKO TOMITA
Universidade de São Paulo, Departamento de Ciência da Computação
Cidade Universitária "Armando de Salles Oliveira"
CP 66281 - CEP 05389-970 - São Paulo, SP, Brazil
E-mail: jb@ime.usp.br

Abstract.
Binary Image Analysis problems can be solved by set operators implemented as programs for a Morphological Machine (MMach). These programs can be generated automatically by the description of the goals of the user as a collection of input-output image pairs and the estimation of the target operator from these data. In this paper, we present a software, installed as a Toolbox for the KHOROS system, that implements this technique and some impressive results of applying this tool in shape recognition for OCR.

Key words: Morphological Machines, OCR, learning of set operators

1. Introduction

Optical Character Recognition (OCR) refers to a process in which printed documents are transformed into ASCII files for the purpose of compact storage, editing, fast retrieval, and other file manipulations through the use of a computer [8].

A key problem in OCR is the recognition of characters by their shapes. The techniques applied for this task must be robust and flexible to deal with different letter fonts in different contexts (distinct serifs, styles, noise, etc.)

A natural model of a procedure for shape recognition is a set operator applied on a Discrete Random Set [6]. Mathematical Morphology (MM) is a general framework to study set operators [2].

An important aspect of MM is the description of set operators by a formal language that is complete and expressive [3]. Since the sixties special machines, the Morphological Machines (MMach's), have been built to implement this language. However, designing useful MMach programs is not an elementary task.

Recently, much research effort has been addressed to automating the programming of MMach's. The goal is to find suitable knowledge representation formalisms to describe operations over geometric structures and to translate them into MMach programs. We have proposed [4, 5] the use of Machine Learning theory [1, 11] as a framework for the automatic programming of MMach's. In this approach, the goals of the user are represented as a collection of input-output pairs of images and the target operator is estimated from these data.

In this paper, we present a software that performs the automatic programming of MMach's by Machine Learning and some impressive results of applying this tool in

* The authors have received partial support of Olivetti do Brasil, CNPq, grants PROTEM-CC-ANIMOMAT and PROTEM-CC-TCPAC, and Cooperation USP-COFECUB

shape recognition for OCR.

Following this introduction, section 2 shows how to approach the problem of shape recognition by set operators. Sections 3 and 4 recall, respectively, the canonical representation of set operators and the formulation of the problem of learning set operators. Section 5 describes the software developed. Section 6 describes the strategies of learning employed in the experiments. Section 7 presents some experimental results. Finally, we discuss some aspects of this work and present some possible future steps of this research.

2. Shape Recognition by Set Operators

Let $\mathcal{P}(E)$ be the collection of all subsets of a non empty subset E . The set E is assumed to be an Abelian group with respect to a binary operation denoted by $+$. The zero element of $(E,+)$ is called the *origin* of E and it is denoted o.

Let W be a finite subset of E and Ψ_W denote the set operators on $\mathcal{P}(E)$ that are *translation invariant* (t.i.) and *locally defined* (l.d.) *within the window* W, that is, $\psi \in \Psi_W$ iff, $\forall X \in \mathcal{P}(E)$ and $\forall h \in E$,

$$\psi(X + h) = \psi(X) + h$$

and

$$h \in \psi(X) \Longleftrightarrow h \in \psi(X \cap (W + h)).$$

Let M be a finite subset of E. A shape $\mathcal{S}$ in M is a collection of subsets of M. A set $X \in \mathcal{S}$ is called a *set of shape* $\mathcal{S}$. A classical problem in Image Analysis is the problem of *shape recognition*.

Let I be a set of indices. Given a collection of shapes $\{\mathcal{S}_i : i \in I\}$, such that $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for $i \neq j$, $i, j \in I$, and a set X, such that $X \in \cup\{\mathcal{S}_i : i \in I\}$, of unknown shape, what is the shape of X?

A collection $\{\psi_i : i \in I\}$ of set operators can be used to solve this problem. A set operator ψ_i indicates if X is of shape $\mathcal{S}_i$ or not, respectively, if it satisfies the properties: $\psi_i(X) \neq \emptyset, \forall X \in \mathcal{S}_i$, and $\psi_i(X) = \emptyset, \forall X \in \cup\{\mathcal{S}_j : j \in I, j \neq i\}$.

The operator ψ_i is called the *marker* of the shape $\mathcal{S}_i$.

Let W and X be subsets of M. The model of X through W is the collection $X_W = \{W + h \cap X, h \in E\}$. A shape recognition problem is said to be of dimension W if, for all $i \in I$, there exists $\mathcal{M}_i \subset \mathcal{P}(W)$, $\mathcal{M}_i \neq \{\emptyset\}$, such that $\forall X \in \mathcal{S}_i$, $\mathcal{M}_i \subset X_W$ and $\forall j \in I$, $i \neq j, \forall Y \in \mathcal{S}_j, \mathcal{M}_i \not\subseteq Y_W$.

This condition implies that there exists a collection of t.i. operators l.d. within the window W that can solve the shape recognition problem.

The elements of $\mathcal{P}(W)$ will be called *patterns*.

3. Set operators representation

Let $\psi \in \Psi_W$. The set $\mathcal{K}_W(\psi) = \{X \in \mathcal{P}(W) : o \in \psi(X)\}$ is called the *kernel* of ψ.

Let $A, B \in \mathcal{P}(W)$, such that $A \subseteq B$. The set $[A, B] = \{X \in \mathcal{P}(W) : A \subseteq X \subseteq B\}$ is called a *closed interval*.

The set of maximal intervals contained in $\mathcal{K}_W(\psi)$ is called the *basis of* ψ and is denoted $\boldsymbol{B}_W(\psi)$.

Let $A, B \in \mathcal{P}(W)$, such that $A \subseteq B$. The operator $\lambda^W_{(A,B)}$ defined by

$$\lambda^{W}_{(A,B)}(X) = \{x \in E : A + x \subseteq X \cap (W + x) \subseteq B + x\},$$

for all $X \in \mathcal{P}(E)$, is called the *locally defined sup-generator operator* characterized by the pair $((A, B), W)$.

Any operator $\psi \in \Psi_W$ can be represented [2] as

$$\psi(X) = \cup\{\lambda^{W}_{(A,B)}(X) : [A, B] \in \boldsymbol{B}_W(\psi)\},$$

for all $X \in \mathcal{P}(E)$. This representation is called the *canonical representation* of the operator ψ.

Equivalently, the operator ψ can be represented by the Boolean function f_ψ defined by, for all $X \in \mathcal{P}(W)$,

$$f_\psi(X) = 1 \Leftrightarrow \exists [A, B] \in \boldsymbol{B}_W(\psi) : X \in [A, B].$$

4. Machine Learning

We understand a *concept* as a subset of objects in a predefined domain, structured by a probability distribution. An *example* of a concept is an object from the domain together with a label indicating whether the object belongs to the concept. If the object belongs to the concept it is a *positive example*, otherwise it is a *negative example*. *Concept learning* is the process by which a learner constructs a good statistical approximation to an unknown concept, given a relatively small number of examples and some prior information about the concept to be learned [1, 11]. In the following, we formalize these ideas.

Let $\mathcal{D}$ be a finite domain with a distribution μ. A concept c is a Boolean function from $\mathcal{D}$ to $\{0, 1\}$. A particular concept is the function f_ψ that represents the operator ψ in the domain $\mathcal{D} = \mathcal{P}(W)$.

For an object $X \in \mathcal{D}$, an example (X, b) is a *positive example* if $b = 1$ and a *negative example* if $b = 0$.

The set of all possible concepts to be learned will be referred to as the *hypothesis space* and denoted by H. The concept $t \in H$ to be determined is called the *target concept*. The problem is to find a *concept* $h \in H$, called *hypothesis* which is a good approximation for t.

A *training sample* of size m for a concept t is a sequence $(X_1, b_1), ..., (X_m, b_m)$.

A *learning* or *training algorithm* is simply a function L which assigns to any training sample $\mathbf{s}$ for a target concept t a hypothesis $h \in H$. We write $h = L(\mathbf{s})$ and call $L(\mathbf{s})$ a *training* or *learning*.

Let ϵ and δ be two real numbers in the open interval $(0, 1)$. The *precision* of an algorithm L applied on a training sample of size m is

$$Prec(L, m, \epsilon) = P(\mu(\mathcal{D}(m)) < \epsilon),$$

where $\mathcal{D}(m) = \{X_1, X_2, ..., X_m\}$ and P is the probability on $\mathcal{D}^m$ inherited from the distribution μ on $\mathcal{D}$.

For a pair (ϵ, δ) fixed, the size m of the training sample must be such that

$$Prec(L, m, \epsilon) > 1 - \delta.$$

A training sample is *consistent* if $X_i = X_j$ implies $b_i = b_j$. A learning algorithm L for H is *consistent* if, given any consistent training sample $\mathbf{s}$ for a target concept $t \in H$, the output hypothesis agrees with t on the examples in $\mathbf{s}$, that is, $h(X_i) = t(X_i)$, for all $i \in [1, m]$.

When the algorithm L is consistent it is also called *Probably Approximately Correct* (PAC) [9] and a theoretical lower bound for m is

$$m(\epsilon, \delta) = \frac{1}{\epsilon} \ln(\frac{|H|}{\delta}),$$

where $|H|$ denotes the cardinality of the set H. For approximately consistent sample sets, the theoretical lower bounds for m are even bigger, since they need to take into account the contradictions in the training sample.

5. A Toolbox for the Automatic Programming of MMach's

We have developed a software for the automatic programming of MMach's that was installed as a toolbox for the KHOROS system [7]. We briefly describe the software modules, as shown in figure 1:

1. *Sample acquisition*: the modules vwin and vpat are to compile positive and negative examples. More specifically we have: (a) vwin: to specify the size and format of the window W; (b) vpat: to collect pairs (X, b) of positive and negative examples, where X is a pattern from X_i.

2. *Learning*: the modules vxpl, vinterv, and vlearn are to learn the operator ψ, from the examples acquired in the previous step. More specifically: (a) vxpl: to generate a reduced table of examples, i.e., eliminating pairs (X, b) which occurred more than once or choose between contradictory ones; (b) vinterv: to generate the initial intervals for the ISI learning algorithm (see [5]); (c) vlearn: to learn the operator ψ using the ISI algorithm.The resulting intervals correspond to the learned operator basis, i.e., the sup-generators that constitute its minimal representation.

3. *Application on new inputs*: the module vunisup is used to apply the learned operator ψ on new inputs Z.

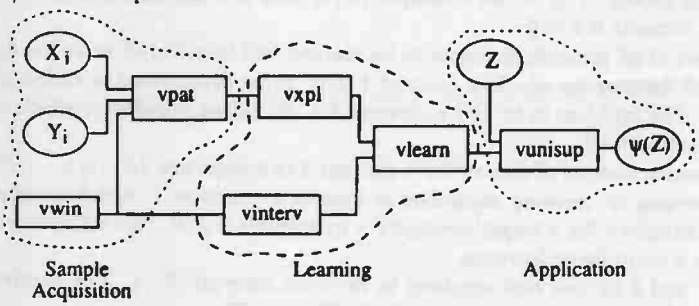

Fig. 1. Software Modules

6. Shape Recognition for OCR

Our goal is to learn a marker ψ that separates the characters that represent the letter of interest from the others. The marker ψ will be learned from pages of a book and applied on other pages of the same book.

We have performed the following steps to prepare the data : a1) scanning some pages of a book; a2) processing these data with gray-scale morphological operators in order to segment the images (i.e. transform the gray-scale image into a binary image, where the one pixels represent the characters and the zero pixels represent

the background); a3) separate some pages of the scanned data and extract by hand (i.e. using some image editing tools) all the occurrences of a given character. Each experiment consists of the following steps: b1) learning a set operator ψ from the data prepared in a3; b2) estimating the precision of ψ, from images not used in b1.

In b2, we have defined two type of errors : *missing errors* and *errors by excess*. The former is verified when the learned operator misses (i.e. doesn't mark) a character that should be recognized and the second is verified when it marks a character that should not be recognized. The sum of these two types of error, in relation to the total of characters in the pages considered in b2, determines the relative error of the learned operator.

Let I be a set of indices and let $\{X_i : i \in I\}$ and $\{Y_i : i \in I\}$ be the collection of images generated, respectively, in a2 and a3. A *first marker* ψ_1 was learned from these data. This training considers as positive examples just the patterns that were observed in the sets of the shape of interest and were not observed in the sets of the other shapes.

We have noticed that almost all errors observed in $\psi_1(X_j)$ (where X_j was not used in the training) were by excess. This fact led us to suggest a second training stage, based on the pairs of images $(\psi_1(X_i), Y_i)$, where X_i was not used to train ψ_1, to get a second operator ψ_2 that acts as a filter to reduce the errors by excess [10]. In this case, the marker is $\psi = \psi_2\psi_1$.

We have also noticed that we could apply a succession of filters to get improved results. We will generically use the term n-stage training to express the learning of a marker built by the composition of a first marker with $(n-1)$ filters.

We have used two variants, L_1 and L_2, of the ISI algorithm (see [5]) to perform the training. These variants are such that $L_1(\mathbf{s})(X_j) = L_2(\mathbf{s})(X_j)$, for all pattern X_j in the training sample $\mathbf{s}$. However, they have different *generalizations*, that is, there exists a pattern $Y \in \mathcal{P}(W)$ such that $L_1(\mathbf{s})(Y) \neq L_2(\mathbf{s})(Y)$.

7. Experimental Results

We have chosen two old books, referred to as Book 1 and 2 (written in Portuguese), to experiment our tools to solve the problem of shape recognition for OCR. We have performed some experiments on both books to recognize the lower case letters "s" and "a". Figure 2 shows images of these books. In this figure, the pixels in black are the markers produced by ψ.

7.1. One-stage Training

In tables I and II we describe the results of experiments with one-stage training performed with Book 1. In all the tables presented in this paper, the time of training was measured in hours(h), minutes(m) and seconds(s).

One can notice the number of examples used in the training affects rightly the relative error (see rows 1,2 and 4 in table II). However, the time of training increases with the size of the training sample (see rows 1 and 2 in table II).

We can observe in rows 2 and 4 in table I, or rows 3 and 5 or rows 2 and 6 in table II, that there is no conclusive relationship between the window size and the relative error.

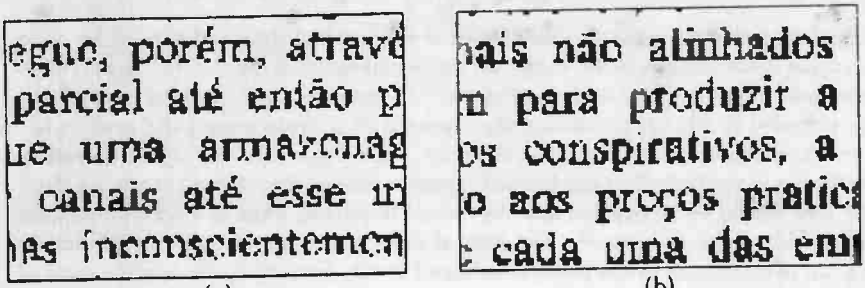

Fig. 2. a) Book 1. b) Book 2.

window size	number of examples	type of algorithm	size of basis	time of training	relative error (%)
5 x 5	270,267	2	1,560	5h20m	6.2
5 x 5	79,049	3	416	20m53s	7.66
7 x 7	79,040	2	1,348	1h46m	14.85
7 x 7	79,040	3	284	25m	5.12

TABLE I
Book 1 - letter "s"

Furthermore, since the variants of ISI give different number of elements in the basis to the same training sample, it affects the relative error of the learned operator, as we can see in rows 3 and 4 in table I or in rows 2 and 3 or rows 5 and 6 in table II.

We have repeated some of these experiments to Book 2 and the best relative error obtained for it were 4.1 and 11.8, respectively, for the letters "s" and "a".

7.2. Multiple-stage training

In multiple-stage training, we have performed some experiments with different numbers of stages. In each stage we used a square window, reduced by 2 pixels their sides in relation to the former stage. In tables III and IV we show some results concerned with multiple-stage training, extending the training described in 7.1.

There is a considerable decrease of relative error from stage 1 to stage 2, while the increase of time (spent to train ψ_2) and of the basis size (due to the basis of ψ_2) are not significant. Here, we note that the initial window size affects the final relative

window size	number of examples	type of algorithm	size of basis	time of training	relative error (%)
7 x 7	270,192	2	5,059	267h12m	15.5
7 x 7	79,040	2	2,311	19h20m	25.56
7 x 7	79,040	3	644	2h47m28s	10.4
7 x 7	37,201	2	1,447	5h18m	35.31
9 x 9	79,019	3	551	3h45m	12.75
9 x 9	79,019	2	2,798	14h42m	31.45

TABLE II
Book 1 - letter "a"

first stage window size	number of examples	number of stages	total size of basis	time of training	relative error (%)
5 x 5	79,049	1	116	20m53s	7.66
5 x 5	86,111	2	429	20m54s	1.31
7 x 7	79,040	1	284	24m31s	5.12
7 x 7	83,237	2	354	24m49s	0.49
7 x 7	87,288	3	388	24m50s	0.35

TABLE III
Book 1 - letter "s"

first stage window size	number of examples	number of stages	total size of basis	time of training	relative error (%)
7 x 7	79,040	1	644	2h47m28s	10.4
7 x 7	88,333	2	726	2h47m55s	1.38
7 x 7	96,532	3	762	2h47m56s	0.50
9 x 9	79,019	1	551	3h45m	12.75
9 x 9	88,275	2	700	3h47m52s	0.80
9 x 9	96,121	3	760	3h48m13s	0.47
9 x 9	103,530	4	781	3h48m14s	0.38

TABLE IV
Book 1 - letter "a"

error (see rows 2 and 4 in table III or rows 2 and 5 in table IV).

We have repeated some of these experiments to Book 2, and we achieved the relative errors 0.39 in stage 3 and 0.15 in stage 2, respectively, for the letters "a" and "s". In the case of letter "s", a third stage presented a significant increase of missing errors in relation to a small decrease of errors by excess, resulting in a realtive error 0.16. This fact establishes a limit to the number of stages applicable to multiple-stage training.

We have concluded that a multiple-stage training is a very suitable and efficient way to improve the results without increase the sample size, despite of the limitation on the number of training stages.

8. Discussion

We have gotten exceptional results (more than 99.5 % correct) applying learning algorithms to the design of MMach programs for character recognition.

Analysing these results a number of natural questions arise: Why the relative small number of examples used are enough to get such a performance? Why the multi-stage learning is much better than the one-stage learning? How to choose the size of the windows? How to define the number of stages? How to estimate the number of examples needed? How to choose good generalizations?

Answering all these questions with solid mathematical arguments is fundamental for the development of a strong Mathematical theory for the design of MMach programs from examples. Our attempts to answer them indicate they are hard problems and, for the moment, all that we can give are some informal comments, as follows :

At first, we observe the distance between realistic bounds and the theoretical bounds for the sample size. The training samples that we have used are not consistent, but just to give an idea of the discrepancy let us compare the bound for PAC algorithms with the size of the training samples that we have used. For example, using a 7×7 image window (i.e. $|W| = 49$) and adopting $\epsilon = \delta = 0.25$, the theoretical bound is

$$m(\epsilon, \delta) = \frac{1}{0.25} \ln(\frac{2^{2^{49}}}{0.25}) \sim 10^{15},$$

while in the corresponding experiment we have used $m(\epsilon, \delta) = 270,000$ and got an equivalent precision.

It seems that the reason for this discrepancy comes from the fact that the domain of a single book is a very restricted context $\mathcal{A} \subseteq \mathcal{P}(W)$, because the letters have just particular patterns and not all the possible patterns in $\mathcal{P}(W)$. This should imply in extraordinary reductions in the size of the hypothesis space H. Another point is that even restricted to $\mathcal{A}$ there are extremely rare shapes that have practically no influence in the precision rates.

The multi-stage training is another fundamental and intriguing point. It seems that each stage reduces the context making easier the work of the next stage. This reduction of context is so remarkable that even the size of the training sample diminishes dramatically relative to the one-stage training. Besides this method gives hybrid representations (parallel-sequential) that are much simpler (use a smaller number of sup-generating operators) than the strictly parallel ones.

We hope that these examples have shown the enormous potential of the automatic programming of MMach's by learning algorithms and motivate other researchers to work on these fundamental and intriguing questions.

References

1. M. Anthony and N.Biggs. *Computational Learning Theory. An Introduction.* Cambridge Univ. Press, 1992.
2. G. J. F. Banon and J. Barrera. Minimal representation for translation invariant set mappings by Mathematical Morphology. *SIAM J. Appl. Math.*, V. 51, pp. 1782-1798, 1991.
3. J. Barrera and G. J. F. Banon. Expressiveness of the Morphological Language. In *Image Algebra and Morphological Image Processing III*, V. 1769 of SPIE Proceedings, pp. 264-275, San Diego, CA, 1992.
4. J. Barrera, F.S.C. da Silva, and G. J. F. Banon. Automatic programming of binary morphological machines. In *Image Algebra and Morphological Image Processing*, V. 2300 of SPIE Proceedings, pp. 229-240, San Diego, 1994.
5. J. Barrera, N. S. Tomita, F.S.C. da Silva, and R. Terada. Automatic programming of binary morphological machines by PAC learning. In *Neural and Stochastic Methods in Image and Signal Processing*, V.2568 of SPIE Proceedings, pp.233-244, San Diego, 1995.
6. J. Goutsias. Morphological analysis of discrete random shapes. *Journal of Mathematical Image and Vision*, V. 2, pp. 193-215, 1992.
7. K. Konstantinides and J. Rasure. The khoros software development environment for image and signal processing. *IEEE Transactions on Image Processing*, V. 3, N. 3, pp. 243-252, 1994.
8. S. Mori, C. Y. Suen and K. Yamamoto. Historical review of OCR research and development. *IEEE Proceedings*, V. 80, N. 7, pp. 1029-1058, 1992.
9. L. Valiant. A theory of the learnable. Comm. ACM, V. 27, pp. 1134-1142, 1984.
10. R. E. Dougherty. Optimal Mean-Square n-Observation Digital Morphological Filters. *CVGIP:Image Understanding*,55:36-54,1992.
11. D. Haussler. Decision Theoretic Generalization of the PAC Model for Neural Nets and Other Learning Applications. *Information and Computation*, 100, 78-150, 1992.

RELATÓRIOS TÉCNICOS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Instituto de Matemática e Estatística da USP

A listagem contendo os relatórios técnicos anteriores a 1992 poderá ser consultada ou solicitada à Secretaria do Departamento, pessoalmente, por carta ou e-mail(mac@ime.usp.br).

J.Z. Gonçalves, Arnaldo Mandel
COMMUTATIVITY THEOREMS FOR DIVISION RINGS AND DOMAINS
RT-MAC-9201, Janeiro 1992, 12 pp.

J. Sakarovitch
THE "LAST" DECISION PROBLEM FOR RATIONAL TRACE LANGUAGES
RT-MAC 9202, Abril 1992, 20 pp.

Valdemar W. Setzer, Fábio Henrique Carvalheiro
ALGOTRITMOS E SUA ANÁLISE (UMA INTRODUÇÃO DIDÁTICA)
RT-MAC 9203, Agosto 1992, 19 pp.

Claudio Santos Pinhanez
UM SIMULADOR DE SUBSUMPTION ARCHITECTURES
RT-MAC-9204, Outubro 1992, 18 pp.

Julio M. Stern
REGIONALIZAÇÃO DA MATRIZ PARA O ESTADO DE SÃO PAULO
RT-MAC-9205, Julho 1992, 14 pp.

Imre Simon
THE PRODUCT OF RATIONAL LANGUAGES
RT-MAC-9301, Maio 1993, 18 pp.

Flávio Soares C. da Silva
AUTOMATED REASONING WITH UNCERTAINTIES
RT-MAC-9302, Maio 1993, 25 pp.

Flávio Soares C. da Silva
ON PROOF-AND MODEL-BASED TECHNIQUES FOR REASONING WITH UNCERTAINTY
RT-MAC-9303, Maio 1993, 11 pp.

Carlos Humes Jr., Leônidas de O.Brandão, Manuel Pera Garcia
A MIXED DYNAMICS APPROACH FOR LINEAR CORRIDOR POLICIES
(A REVISITATION OF DYNAMIC SETUP SCHEDULING AND FLOW CONTROL IN MANUFACTURING SYSTEMS)
RT-MAC-9304, Junho 1993, 25 pp.

Ana Flora P.C.Humes e Carlos Humes Jr.
STABILITY OF CLEARING OPEN LOOP POLICIES IN MANUFACTURING SYSTEMS (Revised Version)
RT-MAC-9305, Julho 1993, 31 pp.

Maria Angela M.C. Gurgel e Yoshiko Wakabayashi
THE COMPLETE PRE-ORDER POLYTOPE: FACETS AND SEPARATION PROBLEM
RT-MAC-9306, Julho 1993, 29 pp.

Tito Homem de Mello e Carlos Humes Jr.
SOME STABILITY CONDITIONS FOR FLEXIBLE MANUFACTURING SYSTEMS WITH NO SET-UP TIMES
RT-MAC-9307, Julho de 1993, 26 pp.

Carlos Humes Jr. e Tito Homem de Mello
A NECESSARY AND SUFFICIENT CONDITION FOR THE EXISTENCE OF ANALYTIC CENTERS IN PATH FOLLOWING METHODS FOR LINEAR PROGRAMMING
RT-MAC-9308, Agosto de 1993

Flavio S. Corrêa da Silva
AN ALGEBRAIC VIEW OF COMBINATION RULES
RT-MAC-9401, Janeiro de 1994, 10 pp.

Flavio S. Corrêa da Silva e Junior Barrera
AUTOMATING THE GENERATION OF PROCEDURES TO ANALYSE BINARY IMAGES
RT-MAC-9402, Janeiro de 1994, 13 pp.

Junior Barrera, Gerald Jean Francis Banon e Roberto de Alencar Lotufo
A MATHEMATICAL MORPHOLOGY TOOLBOX FOR THE KHOROS SYSTEM
RT-MAC-9403, Janeiro de 1994, 28 pp.

Flavio S. Corrêa da Silva
ON THE RELATIONS BETWEEN INCIDENCE CALCULUS AND FAGIN-HALPERN STRUCTURES
RT-MAC-9404, abril de 1994, 11 pp.

Junior Barrera; Flávio Soares Corrêa da Silva e Gerald Jean Francis Banon
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES
RT-MAC-9405, abril de 1994, 15 pp.

Valdemar W. Setzer; Cristina G. Fernandes; Wania Gomes Pedrosa e Flavio Hirata
UM GERADOR DE ANALISADORES SINTÁTICOS PARA GRAFOS SINTÁTICOS SIMPLES
RT-MAC-9406, abril de 1994, 16 pp.

Siang W. Song
TOWARDS A SIMPLE CONSTRUCTION METHOD FOR HAMILTONIAN DECOMPOSITION OF THE HYPERCUBE
RT-MAC-9407, maio de 1994, 13 pp.

Julio M. Stern
MODELOS MATEMATICOS PARA FORMAÇÃO DE PORTFÓLIOS
RT-MAC-9408, maio de 1994, 50 pp.

Imre Simon
STRING MATCHING ALGORITHMS AND AUTOMATA
RT-MAC-9409, maio de 1994, 14 pp.

Valdemar W. Setzer e Andrea Zisman
*CONCURRENCY CONTROL FOR ACCESSING AND COMPACTING B-TREES**
RT-MAC-9410, junho de 1994, 21 pp.

Renata Wassermann e Flávio S. Corrêa da Silva
TOWARDS EFFICIENT MODELLING OF DISTRIBUTED KNOWLEDGE USING EQUATIONAL AND ORDER-SORTED LOGIC
RT-MAC-9411, junho de 1994, 15 pp.

Jair M. Abe, Flávio S. Corrêa da Silva e Marcio Rillo
PARACONSISTENT LOGICS IN ARTIFICIAL INTELLIGENCE AND ROBOTICS.
RT-MAC-9412, junho de 1994, 14 pp.

Flávio S. Corrêa da Silva, Daniela V. Carbogim
A SYSTEM FOR REASONING WITH FUZZY PREDICATES
RT-MAC-9413, junho de 1994, 22 pp.

Flávio S. Corrêa da Silva, Jair M. Abe, Marcio Rillo
MODELING PARACONSISTENT KNOWLEDGE IN DISTRIBUTED SYSTEMS
RT-MAC-9414, julho de 1994, 12 pp.

Nami Kobayashi
THE CLOSURE UNDER DIVISION AND A CHARACTERIZATION OF THE RECOGNIZABLE Z-SUBSETS
RT-MAC-9415, julho de 1994, 29pp.

Flávio K. Miyazawa e Yoshiko Wakabayashi
AN ALGORITHM FOR THE THREE-DIMENSIONAL PACKING PROBLEM WITH ASYMPTOTIC PERFORMANCE ANALYSIS
RT-MAC-9416, novembro de 1994, 30 pp.

Thomaz I. Seidman e Carlos Humes Jr.
SOME KANBAN-CONTROLLED MANUFACTURING SYSTEMS: A FIRST STABILITY ANALYSIS
RT-MAC-9501, janeiro de 1995, 19 pp.

C.Humes Jr. and A.F.P.C. Humes
STABILIZATION IN FMS BY QUASI- PERIODIC POLICIES
RT-MAC-9502, março de 1995, 31 pp.

Fabio Kon e Arnaldo Mandel
SODA: A LEASE-BASED CONSISTENT DISTRIBUTED FILE SYSTEM
RT-MAC-9503, março de 1995, 18 pp.

Junior Barrera, Nina Sumiko Tomita, Flávio Soares C. Silva, Routo Terada
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY PAC LEARNING
RT-MAC-9504, abril de 1995,16 pp.

Flávio S. Corrêa da Silva e Fabio Kon
CATEGORIAL GRAMMAR AND HARMONIC ANALYSIS
RT-MAC-9505, junho de 1995,17 pp.

Henrique Mongelli e Routo Terada
ALGORITMOS PARALELOS PARA SOLUÇÃO DE SISTEMAS LINEARES
RT-MAC-9506, junho de 1995, 158 pp.

Kunio Okuda
PARALELIZAÇÃO DE LAÇOS UNIFORMES POR REDUCÃO DE DEPENDÊNCIA
RT-MAC-9507, julho de 1995, 27 pp.

Valdemar W. Setzer e Lowell Monke
COMPUTERS IN EDUCATION: WHY, WHEN, HOW
RT-MAC-9508, julho de 1995, 21 pp.

Flávio S. Corrêa da Silva
REASONING WITH LOCAL AND GLOBAL INCONSISTENCIES
RT-MAC-9509, julho de 1995, 16 pp.

Julio M. Stern
MODELOS MATEMÁTICOS PARA FORMAÇÃO DE PORTFÓLIOS
RT-MAC-9510, julho de 1995, 43 pp.

Fernando Iazzetta e Fabio Kon
A DETAILED DESCRIPTION OF MAXANNEALING
RT-MAC-9511, agosto de 1995, 22 pp.

Flávio Keidi Miyazawa e Yoshiko Wakabayashi
POLYNOMIAL APPROXIMATION ALGORITHMS FOR THE ORTHOGONAL Z-ORIENTED 3-D PACKING PROBLEM
RT-MAC-9512, agosto de 1995, pp.

Junior Barrera e Guillermo Pablo Salas
SET OPERATIONS ON COLLECTIONS OF CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMINIG OF MORPHOLOGICAL MACHINES
RT-MAC-9513, agosto de 1995, 84 pp.

Marco Dimas Gubitoso e Jörg Cordsen
PERFORMANCE CONSIDERATIONS IN VOTE FOR PEACE
RT-MAC-9514, novembro de 1995, 18pp.

Carlos Eduardo Ferreira e Yoshiko Wakabayashi
ANAIS DA I OFICINA NACIONAL EM PROBLEMAS COMBINATÓRIOS: TEORIA, ALGORITMOS E APLICAÇÕES
RT-MAC-9515, novembro de 1995, 45 pp.

Markus Endler and Anil D'Souza
SUPPORTING DISTRIBUTED APPLICATION MANAGEMENT IN SAMPA
RT-MAC-9516, novembro de 1995, 22 pp.

nior Barrera, Routo Terada,
ávio Correa da Silva and Nina Sumiko Tomita
*UTOMATIC PROGRAMMING OF MMACH'S FOR OCR**
T-MAC-9517, dezembro de 1995, 14 pp.