better understand how to put the constraints, local (total number and location of the control points) and global (global geometric constraints, boundedness, connectivity, etc.), coarse (first-order distance approximation), and fine (higher-order), at the beginning for the LP engine instead of using the trial-and-error method as discussed.

ACKNOWLEDGMENTS

This work was partially supported by U.S. National Science Foundation Grant #IRI-9224963. We thank reviewers for helpful suggestions.

REFERENCES

[1] G. Taubin, "Estimation of Planar Curves, Surfaces and Nonplanar Space Curves Defined by Implicit Equations, With Applications to Edge and Range Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, pp. 1,115-1,138, Nov. 1991.

[2] G. Taubin, "An Improved Algorithm for Algebraic Curve and Surface Fitting," *Proc. Fourth Int'l Conf. Computer Vision*, Berlin, Germany, pp. 658-665, May 1993.

[3] S. Sullivan, L. Sandford, and J. Ponce, "Using Geometric Distance Fits for 3-D Object Modeling and Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 12, pp. 1,183-1,196, Dec. 1994.

[4] Z. Lei, D. Keren, and D.B. Cooper, "Computationally Fast Bayesian Recognition of Complex Objects Based on Mutual Algebraic Invariants," *Proc. Int'l Conf. Image Processing*, pp. 635-638, Washington, D.C., Oct. 1995.

[5] J. Mundy and A. Zisserman, eds. *Geometric Invariance in Computer Vision*. Cambridge, Mass.: MIT Press, 1992.

[6] D. Keren, D.B. Cooper, and J. Subrahmonia, "Describing Complicated Objects by Implicit Polynomials," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 38-53, Jan. 1994.

[7] R.J. Vanderbei, *LOQO User's Manual*. Program in Statistics and Operations Research, Princeton Univ., Nov. 1992.

[8] G.B. Dantzig, *Linear Programming and Its Extensions*. Princeton, N.J.: Princeton Univ. Press, 1963.

[9] J. Subrahmonia, D.B. Cooper, and D. Keren, "Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 5, pp. 505-519, May 1996.

[10] Z. Lei and D.B. Cooper, "New, Faster, More Controlled Fitting of Implicit Polynomial 2D Curves and 3D Surfaces to Data," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 514-519, San Francisco, Calif., June 1996.

[11] Z. Lei and D.B. Cooper, "Linear Programming Fitting of Implicit Polynomials With Applications in Object Recognition and Interactive Computer Graphics," *LEMS Tech. Report 146*, Brown Univ., Oct. 1995.

Decomposition of Arbitrarily Shaped Binary Morphological Structuring Elements Using Genetic Algorithms

Giovanni Anelli, Alberto Broggi, *Member, IEEE*, and Giulio Destri, *Student Member, IEEE*

Abstract—A number of different algorithms have been described in the literature for the decomposition of both *convex* binary morphological structuring elements and a specific subset of *nonconvex* ones. Nevertheless, up to now no deterministic solutions have been found to the problem of decomposing *arbitrarily shaped* structuring elements. This work presents a new stochastic approach based on Genetic Algorithms in which no constraints are imposed on the shape of the initial structuring element, nor assumptions are made on the elementary factors, which are selected within a given set.

Index Terms—Mathematical morphology, arbitrarily shaped structuring element decomposition, genetic algorithms.

1 INTRODUCTION

MATHEMATICAL morphology [1], [2], [3] concerns the study of shape using the tools of set theory. Mathematical morphology has been extensively used in low-level image processing and analysis applications, since it allows to filter and/or enhance only some characteristics of objects, depending on their morphological shape. A lot of tutorials [3], [2], [4], [1], [5], [6], [7] can be found in the literature.

Within the mathematical morphology framework, a binary image A is defined as a subset of the two-dimensional Euclidean space E^2 $(Z \times Z)$:

$$A = \{a = (a_i, a_j) \mid a_i, a_j \in Z\} \tag{1}$$

In [3], monadic transforms acting on a generic image A (*complement*, *reflection*, and *translation*) and dyadic operators between sets (*dilation*, *erosion*, *opening*, and *closing*) are defined. In the following only the definitions of operators used throughout this paper are recalled, such as *translation*

$$A_t \triangleq \left\{x \in E^2 \,\middle|\, x = a + t, \text{ for some } a \in A\right\} \text{ with } t \in E^2 \tag{2}$$

and dilation

$$A \oplus B \triangleq \left\{x \in E^2 \,\middle|\, x = a + b, \text{ for some } a \in A,\ b \in B\right\} \tag{3}$$

where A represents the image to be processed, and B is called *Structuring Element* (SE), namely, another subset of E^2 whose shape parameterizes each operation.

An SE B is said to be *convex with respect to a given set of morphological operations (e.g., dilation) with a given set of SEs (factors)* $\{F_i, i = 1, \ldots, M\}$ if it can be expressed as a chain of dilations of the F_i elements:

$$B = F_{k_1} \oplus F_{k_2} \oplus F_{k_3} \oplus \ldots \oplus F_{k_m}, \text{ with } k_j \in [1, M], \text{ for } j = 1, \ldots, m \tag{4}$$

Otherwise B is said to be *nonconvex with respect to the same set of SEs*, and, thus, it can only be expressed as a chain of Boolean opera-

• *The authors are with the Dipartimento di Ingegneria dell'Informazione, Universitá di Parma, I-43100 Parma, Italy.*
E-mail: broggi@ce.unipr.it.

Manuscript received 15 Apr. 1996; revised 17 Apr. 1997. Recommended for acceptance by R. Chin.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 104926.

tions (e.g., unions and/or intersections) between convex elements (called *partitions*):

$$B = C_1 \odot C_2 \odot C_3 \odot \ldots \odot C_z \tag{5}$$

where $\odot$ represents any Boolean operation (such as unions $\cup$, intersections $\cap$, ...) and C_i are convex elements that can be expressed as chains of dilations, as shown in (4).

As discussed in the following section, the decomposition of a binary SE into a chain of operations involving only elementary factors is a key problem. So far, only *deterministic* solutions have been analyzed and proposed in the literature [8], [9], [10], each relying on different assumptions (such as *convex* SEs, specific sets of elementary operators, etc.); on the other hand the optimal decomposition (with respect to a given set of optimality criteria) of *nonconvex* generic SEs with a *deterministic* approach is still an open problem.

This paper addresses this problem utilizing a *stochastic* approach, based on Genetic Algorithms: starting from a population of potential solutions (individuals) determined through an exhaustive algorithm, an iterative process modifies the existing individuals and/or creates new ones in accordance to a set of genetic operators applied randomly. The individuals that minimize a given cost function tend to replace the others, and, after a sufficient number of iterations, the algorithm tends to converge toward the optimal solution. In particular, the main purpose of this work is to develop a tool able to give a preliminary answer to the problem of optimal decomposition of *nonconvex* SEs into concatenations of generic elementary operations.

This work is organized as follows: Section 2 motivates the need for SE decomposition and discusses some optimality criteria that can drive the decomposition; Section 3 briefly summarizes the Genetic Approach, its terminology and its notations, and describes the implementation of the decomposition algorithm and the data structures involved; Section 4 presents some results while Section 5 concludes the paper with some remarks and a discussion on future developments.

2 Structuring Element Decomposition

2.1 Motivation

The following two subsections motivate the need for SE decomposition on traditional *serial systems*, in which the use of a large SE is not efficient, and on *SIMD cellular systems* that allow the execution of only basic operations based on a neighborhood smaller than the size of the SE; the different characteristics of general-purpose (serial) and SIMD cellular (parallel) systems require different techniques in order to exploit the specific hardware characteristics of each system.

Hereinafter, a *dilation* between a generic image A and a complex SE B is considered; due to the different properties of unions and intersections discussed in [3], namely

$$A \oplus (B_1 \cup B_2) = (A \oplus B_1) \cup (A \oplus B_2) \tag{6}$$

$$A \oplus (B_3 \cap B_4) \subseteq (A \oplus B_3) \cap (A \oplus B_4) \tag{7}$$

in the following nonconvex SEs are decomposed using chains of unions of convex SEs (using the equality expressed by relation (6), instead of using chains of intersections or other Boolean operations (where no equality relations hold).

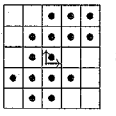
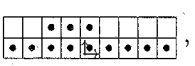
2.1.1 Serial Systems

General-purpose serial systems have no upper bound to the size of possible SEs: In fact, using a bitmapped image representation, the value of any image pixel can be accessed within a constant time. On the other hand, the computational complexity of a *serial* implementation of morphological operations depends on the number of elements which form the operands. As an example, the computation of $A \oplus B$ requires one vector sum and one logical union for each couple of elements $a \in A$ and $b \in B$, and, thus,

$$\mathcal{V}(A \oplus B) = \#(A) \cdot \#(B) \tag{8}$$

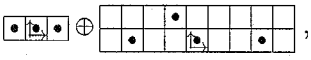
where $\mathcal{V}(\cdot)$ indicates the computational complexity (the number of vector operations) of a given operation, and $\#(\cdot)$ represents the number of elements in a set.

Using the well-known visual representation of morphological sets [3], the number of vector sums and logical unions required by dilation $R = A \oplus B$, where

$A =$  and $B =$  , (9)

according to (8), is given by $\#(A) \cdot \#(B) = 15 \cdot 12 = 180$ operations.

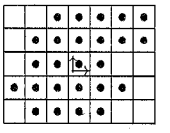
The structuring element B can be expressed as the dilation between subsets B_1 and B_2:

$B = B_1 \oplus B_2 =$  , (10)

and using the *chain rule* property [3],

$$R = A \oplus B = A \oplus (B_1 \oplus B_2) = (A \oplus B_1) \oplus B_2 = R' \oplus B_2 \tag{11}$$

where $R' \underset{=}{\Delta} A \oplus B_1$. The number of sums required to perform the first step of the processing

$R' = A \oplus B_1 =$  (12)

is given by $\#(A) \cdot \#(B_1) = 15 \cdot 3 = 45$, while the number of sums required to complete the processing ($R = R' \oplus B_2$) is given by $\#(R') \cdot \#(B_2) = 25 \cdot 4 = 100$. Thus, the decomposition shown in (10), while incrementing the total number of dilations from one to two, decreases the number of operations performed from 180 to 145.

2.1.2 SIMD Cellular Systems

When a bitmapped data representation is used, mathematical morphology operations involve repeated computations over large data structures, thus the use of parallel systems improves the overall performance. Both parallel architectures with *spatial parallelism* (cellular systems), based on a high number of Processing Elements (PEs) devoted to the simultaneous processing of different image areas, and parallel architectures with *operational parallelism* (pipeline systems), where the different PEs work in pipeline of the same image area, share common constraints. The planar surface of the silicon chip limits the hardware interconnections, thus reducing the complexity of the elementary operations (the size of the possible SEs) that can be performed by each single PE.

This fact is more evident in cellular systems, where the set of all possible operations performed by each single PE (known as *Instruction Set*, IS) is generally based on the use of 3×3 SEs. Thus, since operations based on large SEs cannot be performed, their decomposition into chains of simpler operations belonging to the IS becomes mandatory. The above dilation $R = A \oplus B$ shows the main difference between serial and cellular systems. On serial systems the dilation can be performed either directly (with a single dilation), $R = A \oplus B$ or, after the decomposition of B, as a chain of more than one dilation (as shown by (11)), thus leading to a different computational complexity. On the other hand, that dilation

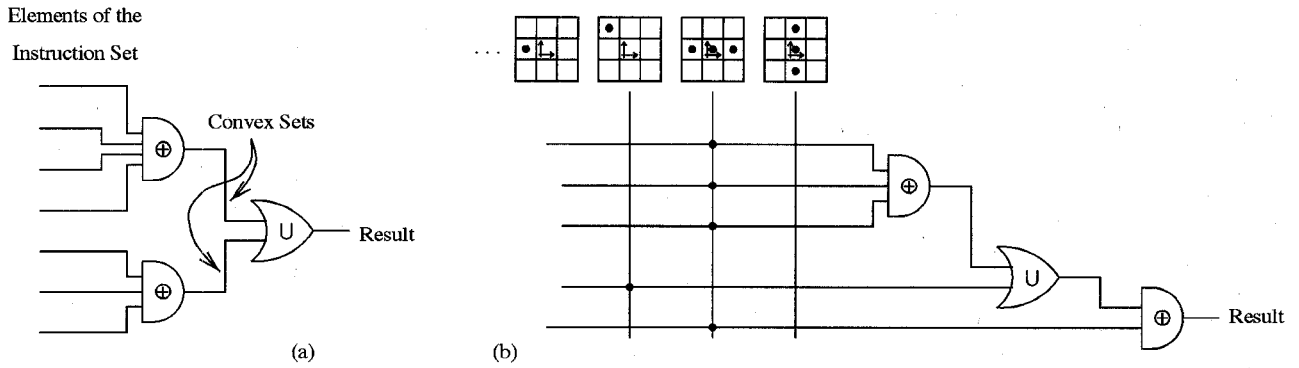


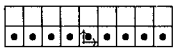
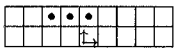
Fig. 1. (a) Union of dilations, also known as *sum of products*. (b) A two-levels solution.

cannot be performed directly on a cellular system since it is based on a SE not belonging to the IS. Thus, while in the first case (serial systems), the decomposition may be recommended for a number of reasons (such as the speed-up of the processing), it becomes mandatory in the second case (cellular systems).

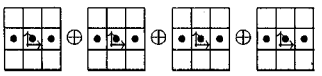
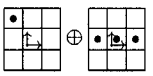
Assuming a system capable of performing horizontal and vertical dilations and translations in the eight main directions, B (as defined in (9)) is nonconvex with respect to the IS of the system; it may be expressed as a union of convex sets, for example

$$B = C_1 \cup C_2 \tag{13}$$

where

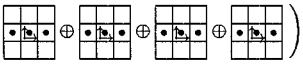
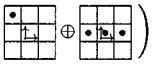
$C_1 =$  and $C_2 =$  . (14)

C_1 and C_2 are convex with respect to the IS of the system and can thus be expressed as:

$C_1 =$  and $C_2 =$ 

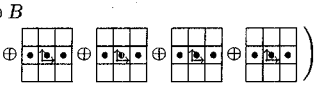
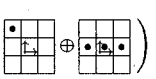
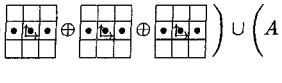
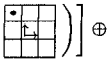
(15)

Thus, according to this decomposition, the initial dilation

$$\begin{aligned} R &= A \oplus B = A \oplus (C_1 \cup C_2) \\ &= (A \oplus C_1) \cup (A \oplus C_2) \\ &= \left(A \oplus \text{  }\right) \cup \left(A \oplus \text{  }\right) \end{aligned} \tag{16}$$

can be performed with six elementary dilations and one logical union. This is a *one-level* solution, involving only a single level of unions of dilations, also called *sum of products* (see Fig. 1a).

It is obvious that a multilevel solution may lead to a better result. For example using the chain rule property, $R = A \oplus B$ can be expressed in a two-levels solution:

$$\begin{aligned} R &= A \oplus B \\ &= \left(A \oplus \text{  }\right) \cup \left(A \oplus \text{  }\right) \\ &= \left[\left(A \oplus \text{  }\right) \cup \left(A \oplus \text{  }\right)\right] \oplus \text{  } \end{aligned} \tag{17}$$

This solution, depicted graphically in Fig. 1b, requires only five dilations and one logical union.

2.2 Optimality Criteria

The decomposition of a SE can be aimed to many different goals, such as:

- the minimization of the number of decomposing sets (to reduce the number of dilations);
- the minimization of the total number of computations (for speed-up reasons);
- the minimization of the total number of elements in the decomposing sets (to reduce the size of the data structures and thus also the memory requirements in serial systems);
- the possibility to implement complex morphological operations on cellular systems whose IS is based on simple, elementary operations (to overcome the problem caused by the simple interconnection topology that limits the size of possible SEs);
- or even the determination of factors with a given shape (to aid the recognition of 2D objects).

The optimality criterion addressed in this work can be changed acting on the parameters of a cost function.

3 The Stochastic Approach

Genetic Algorithms (GAs), widely used in various fields [11], are optimization algorithms based on a stochastic search [12], operating by means of *genetic operators* on a *population* of potential solutions of the considered problem (*individuals*). The main data structure is the *Genome* or *Chromosome*, that is composed of a set of *Genes* and of a *Fitness value*. In the population of possible solutions the set of new individuals generated by means of genetic operators is called *Offspring*.

The genetic search is driven by the *fitness* function: Each individual is evaluated to give some quantitative measure of its *fitness*, that is the "goodness" of the solution it represents. At each iteration (*generation*) the fitness evaluation is performed on all individuals. Then, at the following iteration, a new population is generated, starting from the individuals with the highest fitness, and replacing, completely or partially, the previous generation. The genetic operators used to generate new individuals are subdivided into two main categories: *unary* operators, creating new individuals and replacing the existing ones with a modified version of them (e.g., *mutation*, introduction of random changes of genes), and *binary* operators, creating new individuals through the combination of data coming from two individuals (e.g., *crossover*, exchange of genetic material between two individuals). Each iteration step is called *generation*.

The study of GAs led to the more general *Evolution Programs* (EPs) [13], or *Generalized GAs*. In "standard" GAs an individual is represented by a fixed-length binary string, encoding the parameters set, which corresponds to the solution it represents; the genetic operators act on these binary codes. In EPs, individuals are

represented by generalized data structures without the fixed-length constraint [14], [15]. In addition, ad-hoc operators are defined to act on these data structures.

EPs perfectly match the requirements of the SE decomposition problem, since the varying number of elementary items forming a solution does not allow to know a priori the size of a generic solution, that is the length of the coding of a generic individual. In fact, for an efficient implementation, the data structure representing a decomposition must explicitly encode both the number and the shape of each single elementary operation composing the solution. Moreover, this coding must also allow fast and easy processing and evaluation phases. For these reasons it has been necessary to develop an ad hoc EP with specific genetic rules, exploiting a method similar to the one presented in [16] for the solution of the bin-packing problem. Up to now, the number of iterations is chosen by the user, but different termination criteria are under evaluation (such as the percentage of improvement or the number of different individuals) [11], [13].

3.1 Data Structure

As stated above, the data structure representing an individual has to describe in a *flexible* and *compact* way the convex elements of (5), showing its shape and decomposition into factors, but it has also to make the evaluation phase fast and simple. This representation has to be variable in length, since the number z of possible partitions involved in the decomposition of a generic individual I has no maximum bound; on the other hand, recalling (4), the number m and the shape of factors F_{kj}, which form element C_k, depend directly on C_k.

For these reasons an individual is represented by an arbitrarily long *chain of genes*, each gene representing a *partition* of the input SE (see Fig. 2). The *logical union* of all genes produces the individual.

The simplest implementation consists in representing each individual with a data structure whose fields contain all the above information. Conversely, a more complex, hierarchical data structure has been developed in order both to use a lower amount of memory for each individual and to ease and speed-up the determination of new better solutions. Although the handling of this data structure is definitely complex, it allows to detect possible overlappings among the individuals of a population. Each level of the hierarchy encodes only the information strictly necessary to that level. Three are the levels of the hierarchy, as shown in Fig. 2:

- **Factor level**: The basic components are the elementary morphological operations (i.e., the instruction set elements): an integer indicates which element of the IS is used, while a pointer allows to follow the chain of elements.
- **Gene level**: A gene is composed of one or more factors and it corresponds to a *dilation chain of factors*; an integer gives the origin of the partition described by the gene, thus specifying the translation required to fit the gene onto the initial SE (the origin) and a pointer identifies the next gene.
- **Individual level**: One or more genes form the individual that corresponds to a *union of dilation chains*, corresponding to a decomposition or, more often, to a part of a decomposition. An integer gives the total number of genes forming the individual, a pointer gives the position of the first gene of the chain, while a double precision number contains the fitness value of the individual.

3.2 Initialization of the Population

In order to understand this fundamental step, some definitions are introduced.

DEFINITION 1. *The IS is a set of M factors:* $IS = \{F_i, i = 1, ..., M\}$.

DEFINITION 2. *Notation* $H_{(x,y)}$ *stands for* $H \oplus \{(x, y)\}$, *namely, it represents a translation, as in (2).*

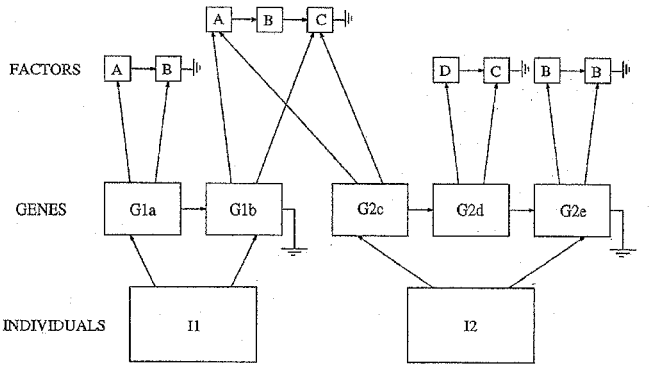


Fig. 2. The data structure representing two individuals.

DEFINITION 3. *For a generic image H,*

$$nH \triangleq H \oplus \ldots \oplus H \ (n \text{ terms}, \ n \geq 0)$$

DEFINITION 4. *For a generic image H,* O_H *represents its origin.*

In the following, B is the input SE, B_i is a generic subset of B with the same origin, and H represents any generic set, convex with respect to the IS:

$$H = q_1F_1 \oplus q_2F_2 \oplus \cdots \oplus q_iF_i \oplus \cdots \oplus q_MF_M \tag{18}$$

If $O_{Fi} \in F_i$ for every F_i belonging to the IS,[1] then $O_H \in H$. The process starts with the identification of every element of set $C(B)$, which is defined as

$$C(B) \triangleq \left\{ H \middle| H_{(h,k)} \subseteq B \text{ for some } (h,k) \right\} \tag{19}$$

since every element of $C(B)$ may represent a possible gene; this search has to be *deterministic* and *exhaustive*. Since $O_{F_i} \in F_i$, the set of possible pairs (h, k) is given by the set of all elements of B; for a generic image H, if $(h, k) \in B \ominus H$, then $H_{(h,k)} \in C(B)$. Therefore, the algorithm scans all the pixels of the SE and determines which factors can form a legal chain of dilations starting from that pixel. The gene obtained so far needs an additional shift in order to overlap its origin with the origin of B.

Since the length of the best solution is not known a priori and since randomly linking together some genes seldom yields a legal solution, we have decided to use each element of $C(B)$, which comprises all the workable genes, as constituting an individual with a chromosome composed of a single gene only. It is also possible, as an option, to include in the population multiple copies of each individual so formed. In this way the set of individuals forming the initial population is obtained.

3.3 The Fitness Function

The *fitness* function $f(I)$ is used to evaluate each individual I in order to drive the algorithm during the search. A cost function $f_C(I)$ has been introduced, which is identical to $f(I)$ if and only if $\bigcup_{i=1}^{N} B_i = B$, where B_i is the general partition forming solution I of length N. This function must have several properties:

1) for legal solutions it is equal to $f(I)$;
2) it is defined also for nonlegal solutions (i.e., solutions not covering perfectly the original SE), thus widening the search space;
3) it is easily implemented as a penalty function.

1. In the current implementation, each factor is limited in size to 3×3 and $O_{F_i} \in F_i$.

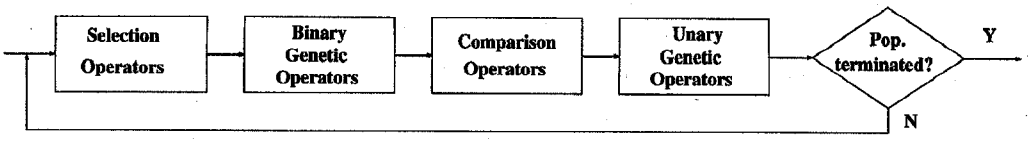


Fig. 3. The generational cycle.

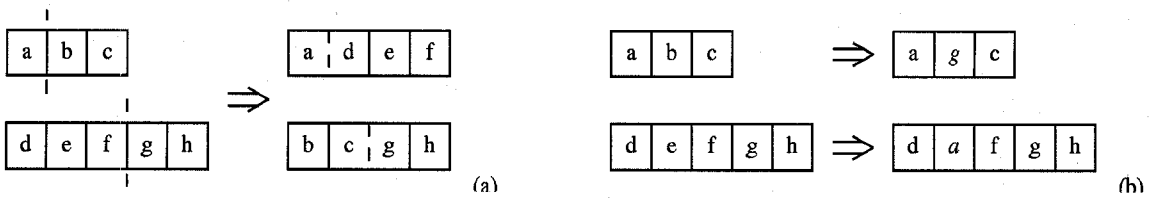


Fig. 4. Examples of the *cut and splice*. (a) *Replacing crossover*. (b) Operators.

Penalty functions are used in highly constrained problems when the need of evaluating nonlegal solutions is met by penalizing them with respect to the legal ones. The cost function thus includes a penalty term:

$$f_C(I) = af(I) + bf_P(I) \tag{20}$$

where a is equal to 1 if and only if $\bigcup_{i=1}^{N} B_i = B$, as stated above, otherwise a is expressed by a term proportional to the ratio between the number of elements present in the solution and the total number of elements in B. The term b is related, via user-defined parameters, to the current population size, and f_P is the *penalty function*, that is still related to the percentage of elements of B covered by the decomposition contained in the considered individual.

Assuming that the goal of the process is to obtain the decomposition that minimizes the number of operations required to compute the dilation of a generic image with SE B, the *fitness* function $f(I)$ is mainly constituted by the sum of the cost of every partition B_i; in addition, it takes into account also the number of logical union operations required, weighted by an appropriate coefficient, and the additional saving allowed by a multilevel solution. The program, in fact, lets the user choose from among three different optimization levels, thus leading to different final results: Level 0 does not perform any optimization; level 1 performs a first packing, based on the methods explained at the end of Section 2.1.2; level 2 tries to pack again the solutions obtained at level 1. The use of optimization levels 1 and 2 becomes of basic importance when the target architecture has the capability to store temporary results, since it allows to achieve solutions with sensibly lower costs.

3.4 The Genetic Search

The structure of the algorithm is depicted in Fig. 3; a single processing cycle is composed of four stages:

- **Selection Operators**: Choose two individuals among population for reproduction purpose;
- **Binary Genetic Operators**: Combine, in various ways, the parents' chromosomes in order to get offspring (i.e., one child or two children);
- **Comparison Operators**: Set up a competition between parents and offspring for inclusion in the next generation;
- **Unary Genetic Operators**: Mutate the chromosomes of the individuals winning at the previous stage.

When the list containing the individuals of the current population is exhausted, i.e., every individual has been chosen for reproduction, the population just formed undergoes the same processing: this generation-formation process stops when the maximum number of allowed generations is reached.[2] In the following, a brief review of the operators is presented.

3.4.1 Selection Operators

The operator is based on the *tournament selection scheme* as exposed in [17] with a tournament size of two. The scheme implemented makes use of a slightly modified version of the *genic selective crowding* technique [17] which forces individuals to compete with those who have at least

$$v(I_1, I_2) = \frac{\#\left(\bigcup_i I_{1,i}\right) \cdot \#\left(\bigcup_i I_{2,i}\right)}{\#(B)} \tag{21}$$

pixels in common with them. In this way, a pressure is maintained for similar to compete with similar, incrementing in this way the significance of the tournament.

3.4.2 Binary Genetic Operators

In order to obtain individuals that cover the SE in the initial phase we need an operator which varies appreciably the length of individuals' chromosomes. Conversely, after this phase, when the average length of individuals is roughly close to the optimal one, we are concerned about the *quality* of the chromosome; for this reason two different operators have been conceived.

- The first one is based upon the *cut and splice* operator [18]. Let λ be the number of genes constituting a chromosome: This operator cuts the chromosome randomly in correspondence to one of the λ possible points. If one of the $\lambda - 1$ points connecting two consecutive genes is chosen, the chromosome is broken into two parts; otherwise, with probability $\frac{1}{\lambda}$, it is left unchanged. The two, three, or four chromosome segments of two different individuals are pushed in a stack; the *splice* stage either merges the first two top elements of the stack, creating in this way a single child, or promotes each element to a full individual. This shows how the number of individuals in the following generation can be altered. An example is shown in Fig. 4a.
- The second operator is the dual of the previous one: it attempts to improve the fitness of the parents by mixing their chromosomes, searching for a slight edge of improvement by trial and error. A gene composing the first parent is "injected" into the chromosome of the other parent, replacing one of its genes, thus not changing the length of the chromosome but altering only its content. The procedure is

2. As mentioned, other termination criteria are currently under evaluation.

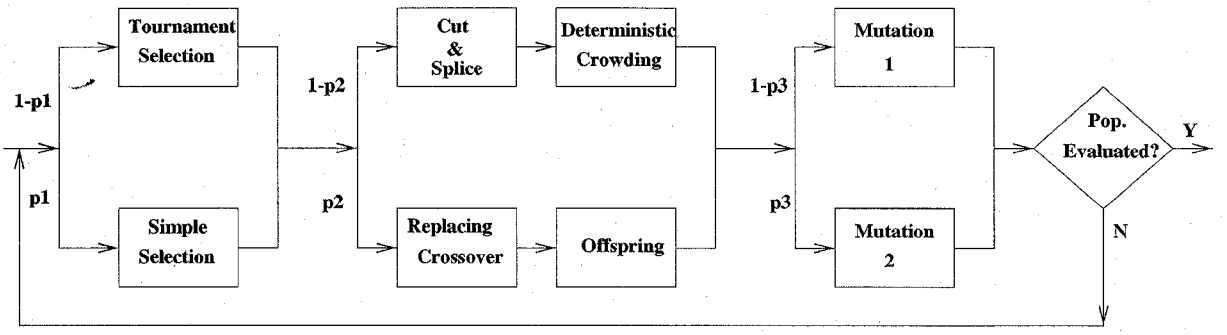


Fig. 5. A schematic representation of the generational process: at each generation individuals are taken in pairs, and in accordance to the respective probability values $p1$, $p2$ and $p3$, selection, binary operators and unary operators are applied.

run twice, swapping the two parents' roles. The number of offspring generated is always two, although in some cases they can coincide with a parent. An example is shown in Fig. 4b.

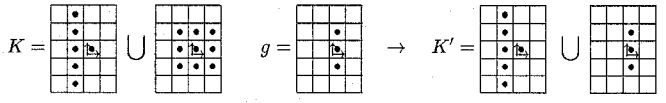
3.4.3 Comparison Operators

At this stage the operator chooses among parents and offspring the individuals to be inserted in the next generation. The scheme followed is based upon the *Deterministic Crowding* scheme presented in [19].

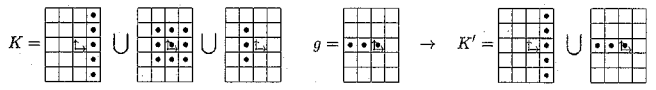
3.4.4 Unary Genetic Operators

In standard GA the unary operator is the *mutation*, that simply inverts randomly one or more bits of the string representing the chromosome. On the other hand, our implementation of mutation has the primary goal of reinserting genes previously discarded and otherwise definitively lost; typically this is the case of little partitions, whose contribution to the fitness improvement has been underestimated in the previous phases of the execution. This contribution can be essential later to achieve the covering of the whole B. Genes are drawn from an array of genes (containing all possible genes) and stored in memory so that every gene is chosen cyclically. Two operators have been created:

- MUTATION 1: This operator compares each gene forming the chromosome of the individual with the gene g coming from the array. The gene g substitutes the most similar one in the chain, that is the gene that maximizes the intersection between the two genes, as in the following example:



- MUTATION 2: This operator forces gene g to be included, along with the suppression of those which overlap with it. It can cause a big fitness worsening but it has the advantage to increase diversity in the chromosomes as a whole, as in the following example:



The complete process is shown in Fig. 5 where a simple selection (which does not use a tournament scheme) is added. It is possible to traverse the graph following 2^3 paths and the decisions are made according to the value of the respective probability values p_1, p_2, p_3. These parameters are computed at the beginning of every generation starting from parameters describing the *status* of the current generation; they can be regarded as *adaptive parameters* [20].

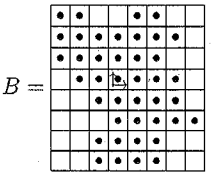
4 Analysis of the Results

4.1 Decomposition of Convex SEs

In accordance with the way the initial population is generated, the decomposition of a *convex* SE by means of this approach leads to the same results discussed in the literature (such as in [8]). This is due to the fact that the optimal solution is a member of the initial population, which consists of all possible (i.e., *legal*) decompositions of the SE, given an arbitrary set of elementary SEs.

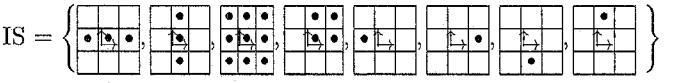
4.2 Decomposition of NonConvex SEs

Let us now consider the decomposition of the following nonconvex SE B,



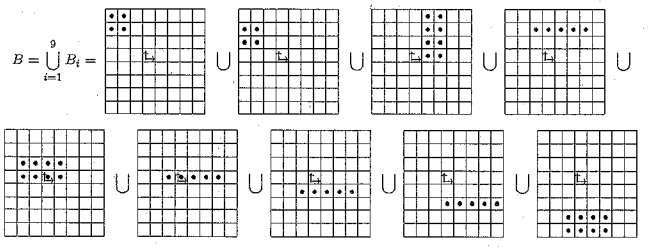
(22)

whose *optimal* decomposition using the following[3] IS



(23)

is definitely nontrivial. The stochastic decomposition led to the result shown in the following:



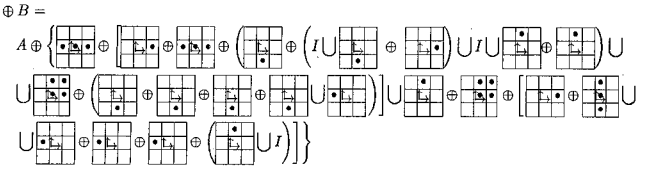
The dilation of a generic image A with B is then reduced to the following:

$$A \oplus B = A \oplus \left(B_1 \bigcup \ldots \bigcup B_9\right) \tag{24}$$

that, considering the IS shown in (23), takes a total of 50 elementary dilations and eight logical unions. Had the algorithm run with the optimization level set to one, (24) could be expressed as a sequence of 22 elementary dilations and eight logical unions:

3. This IS has been chosen to reflect the set of operations available on the PAPRICA system, a special-purpose architecture dedicated to the execution of morphological processings.

$A \oplus B =$



(25)

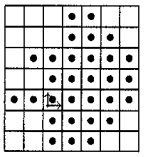
where I is the identity image.

In [8] the original SE needs to be *convex* and it is decomposed using a *given* set of factors. Conversely, in [9], a wider class of SEs is considered: The original SE can also be *nonconvex* but must be simply connected and must belong to a specific class $\mathcal{D}$ of *decomposable* SEs. In that paper, the decomposition of a generic SE S is defined by:

$$S = A^1 \oplus A^2 \oplus \ldots \oplus A^n \tag{26}$$

where A^i is a 3×3 or less simply connected factor. This represents an optimal decomposition when n is minimized, regardless of the shape of the A^i elements. To compare this algorithm with ours, let us choose a SE belonging to $\mathcal{D}$ as discussed in [9]:

$H =$



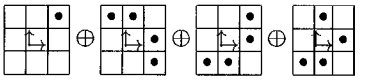
Hereinafter, when presenting a SE decomposition, we will use the following notation:

$$D_M(B, f_c) \tag{27}$$

where M indicates the decomposition method used, B is the input SE, f_c indicates the function giving the cost associated with each factor belonging to the IS.

The optimal decomposition of SE H proposed by Park and Chin (PC) in [9] is

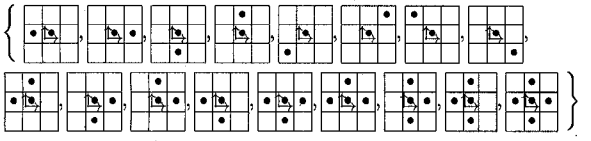
$D_{PC}(H, f) =$



(28)

where f is the four-connected shift cost function mentioned in [8]. Note that using this technique the morphology of the factors is not known a priori, but only at the end. This is in contrast with our approach that requires to specify a set of factors before running the algorithm. To overcome this we can synthesize every factor in (28) with factors belonging to a generic set and use the same set within our program. Obviously, when a factor is not *convex* with respect to such IS, Boolean operators must be used (in this case, logical unions). The IS used here is a modified version of the set specified in [8]:

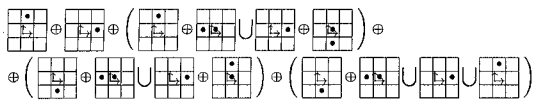
IS $=$



(29)

The resulting decomposition is:

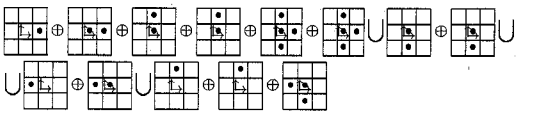
$D_{PC}(H, f) =$



(30)

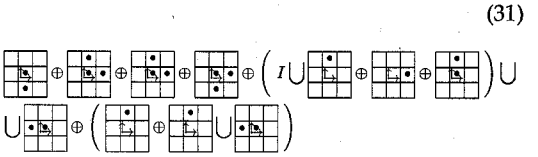
According to the four-connected shift cost function [8], $D_{PC}(H, f)$ has cost 14. In the following, the decomposition obtained with our (Anelli, Broggi, Destri, *ABD*) stochastic approach is presented,[4] where the superscript stands for the optimization level:

$D^0_{ABD}(H, f) =$



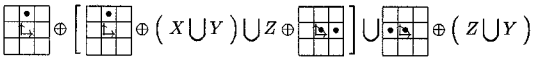
(31)

$D^1_{ABD}(H, f) =$



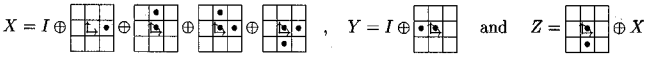
(32)

$D^2_{ABD}(H, f) =$



(33)

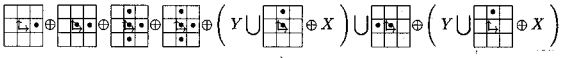
with the following partial results:



(34)

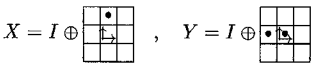
On the other hand, if the cost is set to one for every factor (cost function g), the total cost of $D_{PC}(H, g)$ is equal to the cost of $D_{PC}(H, f)$, while $D^2_{ABD}(H, g)$ becomes:

$D^2_{ABD}(H, g) =$



(35)

with the following partial results:



(36)

leading to a total cost of nine. When operating with level 2 of optimization, $A \oplus H$ can be easily computed replacing the identity image I with the image A in (34) and (36). Table 1 summarizes the cost of the solutions we have obtained for SE H for different optimization levels, and for cost functions f and g, along with the solution presented in [9].

Even though we had to rearrange the decompositions given in [9] in order to fit our requirements (thus altering its cost), this two examples show that the two approaches, although not directly comparable, give solutions with similar cost. In addition, in our approach the freedom of not knowing a priori the shape of the SEs composing the IS is replaced with the possibility to decompose also nonconvex SEs.

5 Conclusion

This paper presented a new approach to the decomposition of arbitrarily shaped binary morphological structuring elements into chains of elementary factors, using a stochastic technique. The application of this technique to *convex* structuring elements leads to the optimal decompositions discussed in the literature; in addition, this paper provides a way of decomposing also *nonconvex* SEs.

Extensive experimentations (not documented here due to space limitations) have shown that the amount of memory required by the system grows according to the size of the initial SE and with the number and size of the elementary SEs. Elements up to 16×16

4. In these examples, the cost of the union operation has been set to zero.

TABLE 1
COSTS OF THE DECOMPOSITIONS OF SE H

SE	$D_{PC}(H,f)$	$D_{PC}(H,g)$	$D^0_{ABD}(H,f)$	$D^1_{ABD}(H,f)$	$D^2_{ABD}(H,f)$	$D^2_{ABD}(H,g)$
H	14	14	18	14	12	9

have been decomposed using ISs composed of eight basic operations on a two processors HP 9000 with 128 megabytes of RAM; the decompositions took about six hours of CPU time and computed 200 generations starting with an average of 2,000 individuals; the computations required about 80 megabytes of memory.

Due to the extremely high computational load required by this iterative approach and to the large memory requirements, the genetic engine is now being ported to the MPI parallel environment: the decomposition is managed by a "master" process, which spawns child processes on the different nodes of a cluster of workstations: each child process is in charge of a specific portion of the processing, which is executed in parallel with all others. This parallel implementation allows to speed up the processing and to decompose very large SEs. Moreover, a graphical interface is also being designed to ease the definition of both the initial SE and the IS, as well as the introduction of parameters. Being based on the Java programming language, its integration into a Web page is straightforward, thus allowing remote users to define and run their own decompositions on our cluster of workstations.

In addition, the first release of the complete tool running on many different systems (SunOS, AIX, Linux, HP-UX, DOS, and Windows) will be shortly available as public domain software via anonymous FTP to researchers working in the mathematical morphology field.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Aurelio Piazzi for his valuable suggestions. This work was partially supported by the Italian National Research Council (CNR) under the frame of the "Progetto Finalizzato Trasporti 2."

REFERENCES

[1] P. Angeline, G. Saunders, and J. Pollack, "An Evolutionary Algorithm That Constructs Recurrent Neural Networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 54-65, Jan. 1994.

[2] A. Broggi, "Speeding-Up Mathematical Morphology Computations With Special-Purpose Array Processors," *Proc. 27th Hawaii Int'l Conf. System Sciences*, T.N. Mudge and B.D. Shriver, eds., vol. 1, pp. 321-330, Maui, Hawaii, Jan. 4-7 1994. Los Alamitos, Calif.: IEEE Computer Society.

[3] E. Falkenauer, "A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems," *Evolutionary Computation*, vol. 2 no. 2, 1994.

[4] D. Fogel, "An Introduction to Simulated Evolutionary Optimization," *IEEE Trans. Neural Networks*, vol. 5, pp. 3-14, Jan. 1994.

[5] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Readings, Mass.: Addison Wesley, 1989.

[6] D.E. Goldberg, B. Korb, and K. Deb, "Messy Genetic Algorithms: Motivation, Analysis, and First Results," *Complex Systems*, vol. 3, pp. 493-530, 1989.

[7] D.E. Goldberg, B. Korb, and K. Deb, "Messy Genetic Algorithms Revisited: Studies in Mixed Size and Scale," *Complex Systems*, vol. 4, pp. 415-444, 1990.

[8] R.M. Haralick, S.R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 532-550, Apr. 1987.

[9] J. Holland, *Adaption Natural and Artificial Systems*. Ann Arbor, Mich.: Univ. of Michigan Press, 1975.

[10] S.W. Mahfoud, "Crossover Interactions Among Niches," *Proc. First IEEE Conf. on Evolutionary Computation*, pp. 188-193, 1994.

[11] G. Matheron, *Random Sets and Integral Geometry*. New York: John Wiley, 1975.

[12] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer-Verlag,.1992.

[13] H. Park and R.T. Chin, "Optimal Decomposition of Convex Structuring Elements for a 4-Connected Parallel Array Processor," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, Mar. 1994.

[14] H. Park and R.T. Chin, "Decomposition of Arbitrarily Shaped Morphological Structuring Elements," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, Jan. 1995.

[15] J. Serra, *Image Analysis and Mathematical Morphology*. London: Academic Press, 1982.

[16] M. Srinivas and L. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithm," *IEEE Trans. System, Man, and Cybernetics*, vol. 24, no. 4, Apr. 1994.

[17] R. van den Boomgaard, *Mathematical Morphology: Extensions Towards Computer Vision*, PhD thesis, Universiteit Van Amsterdam, 1992.

[18] R. van den Boomgaard and R. van Balen, "Methods for Fast Morphological Image Transforms Using Bitmapped Binary Images," *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, vol. 54, no. 3, pp. 252-258, May 1992.

[19] S.S. Wilson, "Theory of Matrix Morphology," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 6, pp. 636-652, June 1992.

[20] X. Zhuang and R.M. Haralick, "Morphological Structuring Element Decomposition," *Computer Vision, Graphics, and Image Processing*, vol. 35, pp. 370-382, Sept. 1986.