Genetic Optimization of Fuzzy Classification Systems — A Case Study

Ulrich Bodenhofer* and Erich Peter Klement†

*Software Competence Center Hagenberg
A-4232 Hagenberg, Austria

†Fuzzy Logic Laboratorium Linz-Hagenberg
Institut für Algebra, Stochastik und wissensbasierte math. Systeme
Johannes Kepler Universität, A-4040 Linz, Austria

Abstract. This contribution presents a fuzzy method for a particular kind of pixel classification. It is one of the most important results of the development of an inspection system for a silk-screen printing process. The classification algorithm is applied to a reference image in the initial step of the printing process in order to obtain regions which are to be checked by applying different criteria. Tight limitations in terms of computation speed have necessitated very specific, efficient methods which operate locally. These methods are motivated and described in detail in the following. Furthermore, the optimization of the parameters of the classification system with genetic algorithms is discussed. Finally, the genetic approach is compared with other probabilistic optimization methods.

Keywords. Fuzzy system, genetic algorithm, pixel classification, print inspection.

1 Introduction

The main goal of this project was to design an automatic inspection system which does not sort out every print with defects, but only those with visible defects which are really unacceptable for the consumer. It is clear that the visibility of a defect depends on the structure of the print in its neighborhood. While little spots can hardly be recognized in very chaotic areas, they can be disturbing in rather homogeneous areas. So, the first step towards a sensitive inspection is to partition the print into areas of different sensitivity which, consequently, should be treated differently.

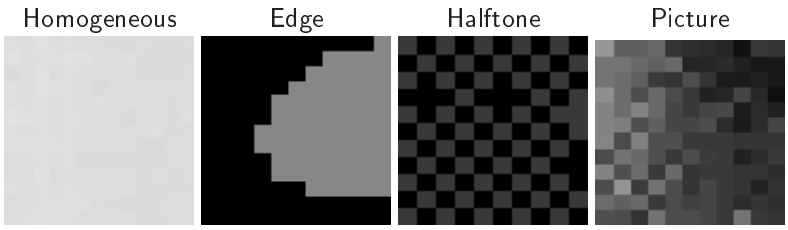

Fig. 1. Magnifications of typical representatives of the four types.

For certain reasons which can be explained with the special principles of this particular kind of silk-screen printing process it is sufficient to consider only the following four types:

Homogeneous area: uniformly colored area;
Edge area: pixels within or close to visually significant edges;
Halftone: area which looks rather homogeneous from a certain distance, although it is actually obtained by printing small raster dots of two or more colors;
Picture: rastered area with high, chaotic deviations, in particular small high-contrasted details.

The magnifications in Fig. 1 show how these areas typically look like at the pixel level. Of course, transitions between two or more of these areas are possible; hence, a fuzzy model is recommendable.

First of all, we should define precisely what, in our case, an image is:

Definition 1. An $N \times M$ matrix of the form

$$\Big((u_r(i,j), u_g(i,j), u_b(i,j))\Big)_{i=1,\dots,N}^{j=1,\dots,M}$$

with three-dimensional entries (additive RGB model)

$$(u_r(i,j), u_g(i,j), u_b(i,j)) \in \{0,\dots,255\}^3$$

is a model of a *24 bit color image* of size $N \times M$. A coordinate pair (i,j) stands for a *pixel*, where i is the *row index* and j is the *column index*; the values $(u_r(i,j), u_g(i,j), u_b(i,j))$ are called the *gray values* of the pixel (i,j).

It is near at hand to use something like the variance of gray values in the neighborhood of the pixels or an other measure for deviations to distinguish between areas which show only low deviations, such as, homogeneous areas and halftone areas, and areas with rather high deviations, such as, edges or pictures.

On the contrary, it is intuitively clear that such a measure can never be used to separate edge areas from picture areas, because any geometrical information is ignored. Experiments have shown that well-known standard edge detectors, such as, the Laplacian or the Mexican Hat, but also many other locally operating filter masks [10], cannot distinguish sufficiently if deviations are chaotic or anisotropic. Another possibility we also took into consideration was to use wavelet transforms [3,13] or more sophisticated image segmentation methods [2,10]. Since we had to cope with serious restrictions in terms of computation speed, such highly advanced methods, although they are efficient, would require too much time. Finally, we found a fairly good alternative which is based on the discrepancy norm. This approach uses only, like the simplest filter masks, the closest neighborhood of a pixel. Figure 2 shows how the neighbors of pixel (i, j) are enumerated for the algorithm.

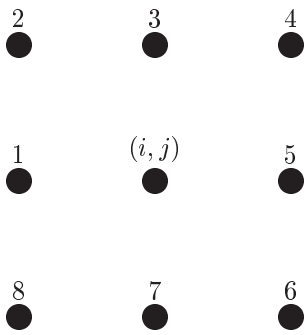

Fig. 2. Enumeration of the neighborhood of a pixel.

For an arbitrary but fixed pixel (i, j) we can define the enumeration mapping l as shown in Table 1. If we plot one color extraction of the eight neighbor pixels with respect to this enumeration, i.e $\big(u_x(l(k))\big)_{k\in\{1,\dots,8\}}$, where $x \in \{r, g, b\}$, we typically get curves like those shown in Fig. 3.

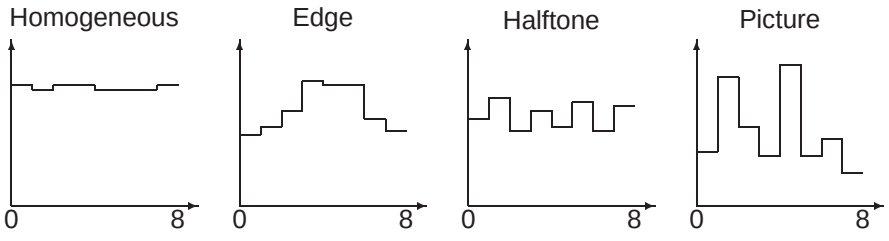

Fig. 3. Typical gray value curves of the form $u_x(l(.))$.

k	$l(k)$
1	$(\quad i \quad, j-1)$
2	$(i-1, j-1)$
3	$(i-1, \quad j \quad)$
4	$(i-1, j+1)$
5	$(\quad i \quad, j+1)$
6	$(i+1, j+1)$
7	$(i+1, \quad j \quad)$
8	$(i+1, j-1)$

Table 1. The enumeration mapping $l(.)$.

From these sketches, it can be seen easily that a measure for the deviations can be used to distinguish between homogeneous areas, halftones, and the other two types. On the other hand, the most eye-catching difference between the curves around pixels in pictures and edge areas is that, in the case of an edge pixel, the peaks appear to be more connected while they are mainly chaotic and narrow for a pixel in a picture area. So, a method which judges the shape of the peaks should be used in order to separate edge areas from pictures. A simple but effective method for this purpose is the so-called discrepancy norm.

2 The Discrepancy Norm

Definition 2. The mapping

$$\begin{aligned} \|.\|_D : \mathbb{R}^n &\longrightarrow \mathbb{R}^+ \\ \boldsymbol{x} &\longmapsto \max_{1 \le \alpha \le \beta \le n} \left| \sum_{i=\alpha}^{\beta} x_i \right| \end{aligned}$$

is called *discrepancy norm* on $\mathbb{R}^n$.

In words, $\|\boldsymbol{x}\|_D$ is the absolute value of the maximal sum of consecutive entries of the vector $\boldsymbol{x}$. Obviously, unlike conventional norms, the signs and the order of the entries play an essential role. Nevertheless, one easily verifies that the mapping $\|.\|_D$ is a norm on $\mathbb{R}^n$.

The connection to the concept of discrepancy in measure theory can be motivated as follows: The discrepancy between two measures μ and ν on $\mathbb{R}$ is usually defined as

$$\mathcal{D}(\mu, \nu) = \sup_{a \le b} |\mu([a, b]) - \nu([a, b])| .$$

If we have two discrete measures $\overline{\mu}$ and $\overline{\nu}$ on the set $\{1,\ldots,n\}$ and if we write $x_i = \overline{\mu}(i)$ and $y_i = \overline{\nu}(i)$, then $\mathcal{D}(\overline{\mu},\overline{\nu})$ equals $\|\boldsymbol{x}-\boldsymbol{y}\|_D$ [8,15]. Thus, it is reasonable to call $\|.\|_D$ a discrepancy norm in $\mathbb{R}^n$.

Obviously, the computation of $\|.\|_D$ by strictly using the definition requires $\mathcal{O}(n^2)$ operations. The following theorem allows us to compute $\|.\|_D$ with linear speed:

Theorem 1. *For all $\boldsymbol{x} \in \mathbb{R}^n$ we have*

$$\|\boldsymbol{x}\|_D = \max_{0\le\beta\le n} X_\beta - \min_{0\le\alpha\le n} X_\alpha,$$

where the values $X_j = \sum_{i=1}^{j} x_i$ denote the partial sums (with the additional setting $X_0 = 0$).

Proof. If we assign 0 to x_0 and x_{n+1} we obtain

$$\begin{aligned}
\|\boldsymbol{x}\|_D &= \max_{1\le\alpha\le\beta\le n+1}\Big|\sum_{i=\alpha}^{\beta} x_i\Big| = \max_{1\le\beta\le n+1}\max_{1\le\alpha\le n+1}\Big|\sum_{i=1}^{\beta} x_i - \sum_{i=1}^{\alpha-1} x_i\Big| \\
&= \max_{1\le\beta\le n}\max_{1\le\alpha\le n}\Big|\sum_{i=1}^{\beta} x_i - \sum_{i=1}^{\alpha} x_i\Big| = \max_{1\le\beta\le n}\max_{1\le\alpha\le n}\Big|X_\beta - X_\alpha\Big| \\
&= \max_{1\le\beta\le n} X_\beta - \min_{1\le\alpha\le n} X_\alpha,
\end{aligned}$$

which completes the proof. □

The following theorem clarifies the relationship between the discrepancy norm and conventional l_p vector norms and, by this way, provides a motivation why the discrepancy norm can be useful for our classification problem.

Theorem 2. *For all $p \in [1,\infty)$ and for all $\boldsymbol{x} \in \mathbb{R}^n$ we have*

$$n^{-\frac{1}{p}} \cdot \|\boldsymbol{x}\|_p \le \|\boldsymbol{x}\|_D \le n^{1-\frac{1}{p}} \cdot \|\boldsymbol{x}\|_p, \tag{1}$$

where $\|\boldsymbol{x}\|_p$ denotes the classical l_p-norm, i.e.,

$$\|\boldsymbol{x}\|_p = \Big(\sum_{i=1}^{n} |x_i|^p\Big)^{\frac{1}{p}}.$$

In the case $p = \infty$, we obtain

$$\|\boldsymbol{x}\|_\infty = \max_{1\le i\le n} |x_i| \le \|\boldsymbol{x}\|_D \le n \cdot \max_{1\le i\le n} |x_i|. \tag{2}$$

Proof. Trivially,

$$\max_{1\le i\le n} |x_i| \le \|\boldsymbol{x}\|_D \le \sum_{i=1}^{n} |x_i|.$$

Then the assertion follows from the relations

$$n^{-\frac{1}{p}} \cdot \|\boldsymbol{x}\|_p \le \|\boldsymbol{x}\|_\infty,$$
$$n^{1-\frac{1}{p}} \cdot \|\boldsymbol{x}\|_p \ge \|\boldsymbol{x}\|_1,$$

which can be proved using the Hoelder inequality. □

For the vectors

$$\boldsymbol{x}_1 = \big(1, -1, 1, \dots, (-1)^{n-2}, (-1)^{n-1}\big),$$
$$\boldsymbol{x}_2 = \big(1, 1, 1, \dots, 1\big),$$

we obtain the following:

$$\|\boldsymbol{x}_1\|_p = n^{\frac{1}{p}} \qquad \|\boldsymbol{x}_1\|_D = 1$$
$$\|\boldsymbol{x}_2\|_p = n^{\frac{1}{p}} \qquad \|\boldsymbol{x}_2\|_D = n$$

From these results, it can be seen easily that, for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, in the inequalities (1) and (2), we indeed have equalities and that there is no monotonic relationship between the discrepancy norm and any l_p-norm with $p \in (1, \infty)$. Furthermore, it can be seen that the more entries with equal signs appear successively, the higher the discrepancy norm is. On the contrary, for sequences with alternating signs it is close to the supremum norm $\|.\|_\infty$. Therefore, $\|.\|_D$ can be used for judging the connectedness of the peaks with equal signs.

3 The Fuzzy System

For each pixel (i, j), we consider its nearest eight neighbors enumerated as defined in Table 1 which yields three vectors of gray values with 8 entries — one for each color extraction. If we denote the mean values of all three gray value curves as

$$\bar{r}(i,j) = \frac{1}{8} \cdot \sum_{k=1}^{8} u_r(l(k)),$$
$$\bar{g}(i,j) = \frac{1}{8} \cdot \sum_{k=1}^{8} u_g(l(k)),$$
$$\bar{b}(i,j) = \frac{1}{8} \cdot \sum_{k=1}^{8} u_b(l(k)),$$

the sums of quadratic deviations of the gray values can be computed as

$$v_r(i,j) = \sum_{k=1}^{8} \big(u_r(l(k)) - \bar{r}(i,j)\big)^2,$$
$$v_g(i,j) = \sum_{k=1}^{8} \big(u_g(l(k)) - \bar{g}(i,j)\big)^2,$$
$$v_b(i,j) = \sum_{k=1}^{8} \big(u_b(l(k)) - \bar{b}(i,j)\big)^2.$$

Now we can take the sum of these three values as a measure for the size of the deviations in the neighborhood of the pixel:

$$v(i,j) = v_r(i,j) + v_g(i,j) + v_b(i,j)$$

On the other hand, the sum of the discrepancy norms of the vectors, where we subtract each entry by the mean value of all entries, can be used as a criterion whether the pixel is within or close to a visually significant edge:

$$\begin{aligned} e(i,j) = &\|u_r(l(.)) - (\bar{r},\ldots,\bar{r})\|_D + \|u_g(l(.)) - (\bar{g},\ldots,\bar{g})\|_D + \\ &\|u_b(l(.)) - (\bar{b},\ldots,\bar{b})\|_D \end{aligned}$$

Of course, e itself can be used as an edge detector. Figure 4 shows how good it works compared with the commonly used Mexican Hat filter mask.

The fuzzy decision is then carried out for each pixel (i,j) independently: First of all, the characteristic values $v(i,j)$ and $e(i,j)$ are computed. These values are taken as the input of a small fuzzy system with two inputs and one output. Let us denote the linguistic variables on the input side with v and e. Since the position of the pixel is of no relevance for the decision in this specific application, indices can be omitted here. The input space of the variable v is covered by three fuzzy sets which are labeled "low", "med", and "high". Analogously, the input space of the variable e is covered by two fuzzy sets which are labeled "low" and "high". Experiments have shown that $[0,600]$ and $[0,200]$ are appropriate universes of discourse for v and e, respectively. For the decomposition of the input domains simple Ruspini partitions [11] consisting of trapezoidal fuzzy subsets were chosen, where a family of fuzzy subsets $(\mu_1,\ldots,\mu_k)$ of a domain X is called *Ruspini partition* if and only if, for all $x \in X$, the equation

$$\sum_{i=1}^{k} \mu_i(x) = 1$$

holds. The typical shape of these partitions is shown in Fig. 5.

The output space is a set of linguistic labels, namely "Ho", "Ed", "Ha", and "Pi", which are, of course, just abbreviations of the names of the four

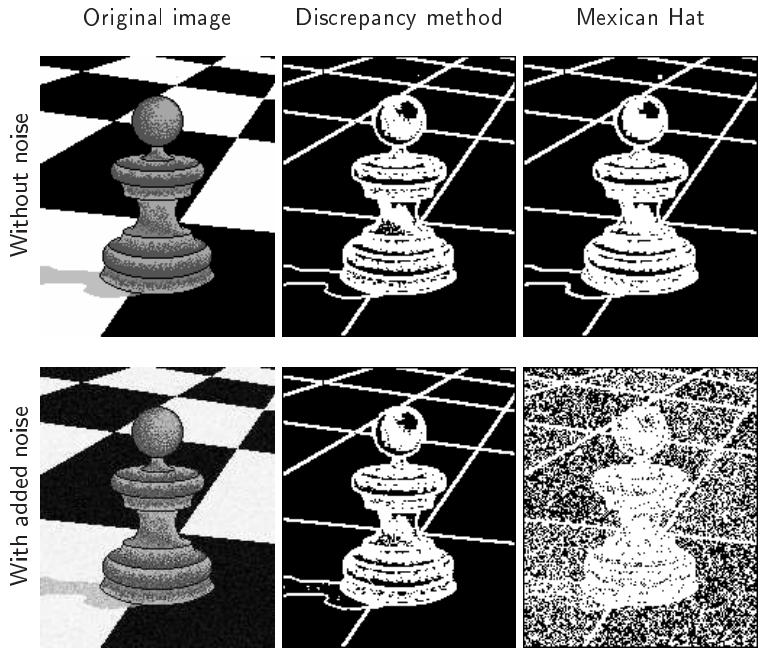

Fig. 4. Comparison between e and a standard 3×3 filter mask.

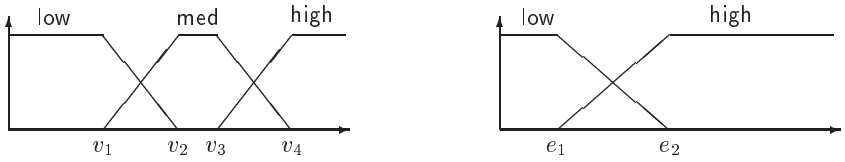

Fig. 5. The fuzzy variables v and e.

types. Let us denote the output variable itself with t. Finally, the output of the system for each pixel (i, j) is a fuzzy subset of {"Ho", "Ed", "Ha", "Pi"}. This output set is computed by processing the values $v(i, j)$ and $e(i, j)$ through a rule base with five rules, which cover all the possible combinations:

IF v is low THEN t = Ho
IF v is med AND e is high THEN t = Ed
IF v is high AND e is high THEN t = Ed
IF v is med AND e is low THEN t = Ha
IF v is high AND e is low THEN t = Pi

In this application, ordinary Mamdani min/max-inference is used. Finally, the degree to which "Ho", "Ed", "Ha", or "Pi" belong to the output set

can be regarded as the degree to which the particular pixel belongs to area Homogeneous, Edge, Halftone, or Picture, respectively.

In our application, the images are taken by an RGB video camera with a resolution of 720×576 pixels. We consider a clipping with approximately 250000 pixels. The A/D converter provides a resolution of 8 bit in each color channel (compare with Def. 1). For such an image, the classification takes at most two seconds on the hardware which had to be used (standard workstations with RISC CPUs, clock rates between 133MHz and 200MHz).

In this specific application, the raster dots and the pixels are of about equal size and the images are, due to expensive high-end camera equipment, remarkably sharp. If these conditions are not fulfilled, the performance of the algorithm in terms of the quality of the decision can be considerably weaker. The proposed methods are especially suited for the needs of this concrete application — the price to be paid for the applicability of the methods under such heavy time constraints is a certain loss of universality.

4 The Integration of the Classification System in the Inspection Procedure

The speed of the printing machine is approximately one print per second. The process can be stopped for at most four seconds. Our implementation takes the first four prints to compute a reference from them, then the machine is stopped for four cycles in order to have time for computing the classification and for doing the other preparatory work, where the purpose of the classification is twofold:

- It determines regions which can be interpreted with different criteria in the further printing process.
- The second purpose, which has not yet been mentioned at all, is that the classification of a pixel is used for computing a tolerance interval. Such a tolerance interval determines to which extent the gray values of a print image at a certain pixel may deviate from the reference image. If the difference between a gray value of the reference and the image, which should be checked, is bigger than the tolerance interval in at least one of the three colors, a pixel is marked as suspicious and has then to be looked at more carefully (by applying techniques which also take the classification into account).

For the first point, a fuzzy classification seems to be unnecessary. The reason why a fuzzy classification is used stems from the second point. The tolerance interval of a certain pixel $\sigma(i,j)$ is computed as the sum of predefined default intervals σ_{Ho}, σ_{Ed}, σ_{Ha}, and σ_{Pi} for the four types of areas weighted with the

degree to which the pixel belongs to the respective area. Specifically,

$$\sigma(i,j) = \frac{\sum\limits_{X \in \{\mathsf{Ho},\mathsf{Ed},\mathsf{Ha},\mathsf{Pi}\}} t_X(i,j) \cdot \sigma_X}{\sum\limits_{X \in \{\mathsf{Ho},\mathsf{Ed},\mathsf{Ha},\mathsf{Pi}\}} t_X(i,j)} \tag{3}$$

which is, in some sense, a kind of Sugeno inference. Since Ruspini partitions are used for the decomposition of the input domains and since traditional Mamdani min/max inference is used for the evaluation of the rule base, the equation

$$\sum_{X \in \{\mathsf{Ho},\mathsf{Ed},\mathsf{Ha},\mathsf{Pi}\}} t_X(i,j) = 1 \tag{4}$$

holds, and (3) simplifies to

$$\sigma(i,j) = \sum_{X \in \{\mathsf{Ho},\mathsf{Ed},\mathsf{Ha},\mathsf{Pi}\}} t_X(i,j) \cdot \sigma_X .$$

Experiments have shown that, if crisp thresholds are used for the classification instead of the fuzzy transitions above, the quality of the decision can be unstable in transitional areas where at least one of the values $v(i,j)$ or $e(i,j)$ is close to a threshold value. Obviously, the tolerance intervals can flip abruptly in such areas. As a consequence, it can happen — in real printing, this is often the case — that pixels in transitional areas are either checked too tolerantly or too rigidly. In the fuzzy case, however, pixels in transitional areas are supplied with transitional tolerance intervals. From these considerations it might be clear that *the fuzzy model is indispensable.*

After computing the reference and its classification, the regular prints of the printing order are checked within the regular cycle time of one second mentioned above (see [1] for more details).

5 The Optimization of the Classification System

As apparent from Fig. 5, the behavior of the fuzzy system depends on six parameters, which determine the shape of the two fuzzy partitions. In the first step, these parameters were tuned manually. Of course, we have also taken into consideration the use of (semi)automatic methods for finding the optimal parameters.

The general problem is not to find an appropriate algorithm for doing that task, the difficulty is how to judge such a classification. Since the specification of the four types of areas is given in a vague, verbal form, no mathematical criterion is available for that. Hence, a model-based optimization process is,

because of the lack of a model, not applicable. The alternative is a knowledge-based approach, which poses the question how to generate this knowledge — the examples from which the algorithm should learn.

Our optimization procedure consists of a painting program which offers tools, such as, a pencil, a rubber, a filling algorithm, and many more, which can be used to make a classification of a given representative image by hand. Then an optimization algorithm can be used to find that configuration of parameters which yields the maximal degree of matching between the desired result and the output actually obtained by the classification system.

Assume that we have N sample pixels for which the pairs of input values $(\tilde{v}_k, \tilde{e}_k)_{k\in\{1,\dots,N\}}$ are computed and that we already have a reference classification of these pixels

$$\tilde{t}(k) = (\tilde{t}_{\mathsf{Ho}}(k), \tilde{t}_{\mathsf{Ed}}(k), \tilde{t}_{\mathsf{Ha}}(k), \tilde{t}_{\mathsf{Pi}}(k)), \qquad k \in \{1, \dots, N\}.$$

Since, as soon as the values $\tilde{v}$ and $\tilde{e}$ are computed, the geometry of the image plays no role anymore, we can switch to one-dimensional indices here. Then one possibility to define the performance (fitness) of the fuzzy system would be

$$\frac{1}{N}\sum_{k=1}^{N} d(t(k), \tilde{t}(k)), \tag{5}$$

where

$$t(k) = (t_{\mathsf{Ho}}(k), t_{\mathsf{Ed}}(k), t_{\mathsf{Ha}}(k), t_{\mathsf{Pi}}(k))$$

are the classifications actually obtained by the fuzzy system for the input pairs $(\tilde{v}_k, \tilde{e}_k)$ with respect to the parameters v_1, v_2, v_3, v_4, e_1, and e_2; $d(.,.)$ is an arbitrary (pseudo-)metric on $[0,1]^4$. The problem of this brute force approach is that the output of the fuzzy system has to be evaluated for each pair (v_k, e_k), even if many of these values are similar or even equal. In order to keep the amount of computation low, we "simplified" the procedure by a "clustering process" as follows:

1. Choose a partition $(P_1, \dots, P_K)$ of the input space and count the number $(n_1, \dots, n_K)$ of sample points $\{p_1^i, \dots, p_{n_i}^i\}$ each part contains.
2. Then the desired classification of a certain part (cluster) can be defined as

$$\tilde{t}_X(P_i) = \frac{1}{n_i}\sum_{j=1}^{n_i} \tilde{t}_X(p_j^i),$$

where $X \in \{\mathsf{Ho}, \mathsf{Ed}, \mathsf{Ha}, \mathsf{Pi}\}$. Due to Eq. (4), we can conclude that, for all $i \in \{1, \dots, K\}$,

$$\sum_{X\in\{\mathsf{Ho},\mathsf{Ed},\mathsf{Ha},\mathsf{Pi}\}} \tilde{t}_X(P_i) = 1. \tag{6}$$

3. If ϕ is a function which maps each cluster to a representative value (e.g., its center of gravity), we can define the fitness (objective) function as

$$f(v_1, \ldots, v_4, e_1, e_2) = \frac{100}{N} \sum_{i=1}^{K} n_i \cdot \left(1 - \frac{1}{2} \cdot d_i\right), \tag{7}$$

with

$$d_i = \sum_{X \in \{\mathsf{Ho}, \mathsf{Ed}, \mathsf{Ha}, \mathsf{Pi}\}} \left(\tilde{t}_X(P_i) - t_X(\phi(P_i))\right)^2.$$

Note that, since Eqs. (4) and (6) hold, the value d_i always lies between 0 and 2.

If the number of parts is chosen moderately (e.g. a rectangular 64×32 net which yields $K = 2048$) the evaluation of the fitness function takes considerably less time than a direct application of Eq. (5).

In Eq. (7), the fitness is already transformed such that it can be regarded as a degree of matching between the desired and the actually obtained classification measured in percent. This value is always positive and has to be maximized.

Figure 6 shows cross sections of such a fitness function, where, in each case, five parameters are kept constant and only one is varied. From this figure it seems obvious that f is continuous but not necessarily differentiable — a fact which can proved easily — and that there can be a lot of local maxima. As a consequence, all conventional continuous optimization methods, which make fundamental use of derivatives, such as, gradient descent, Newton or Quasi-Newton methods, cannot be applied. So, it was near at hand to use a probabilistic optimization method. This, first of all, requires a (binary) coding of the parameters. We decided to use a coding which maps the parameters v_1, v_2, v_3, v_4, e_1, and e_2 to a string of six 8-bit integers $s_1, \ldots, s_6$ which range from 0 to 255. The following table shows how the encoding and decoding is done:

$$\begin{array}{ll} s_1 = v_1 & v_1 = s_1 \\ s_2 = v_2 - v_1 & v_2 = s_1 + s_2 \\ s_3 = v_3 - v_2 & v_3 = s_1 + s_2 + s_3 \\ s_4 = v_4 - v_3 & v_4 = s_1 + s_2 + s_3 + s_4 \\ s_5 = e_1 & e_1 = s_5 \\ s_6 = e_2 - e_1 & e_2 = s_5 + s_6 \end{array}$$

If fuzzy sets of a more general shape are used, this coding is not applicable. Codings for such cases can be found for instance in [12] or [14].

A class of probabilistic optimization methods which has come into fashion in the last years are genetic algorithms (GAs). They can be regarded as simplified simulations of an evolution process, based on the principles of genetic

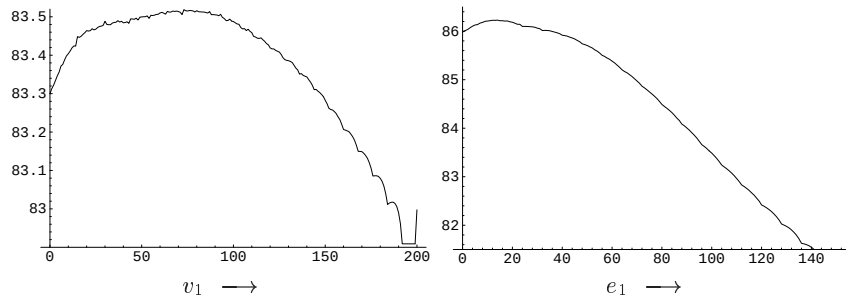

Fig. 6. Cross sections of functions of type (7).

reproduction employing mechanisms, such as, selection, mutation, and sexual reproduction. Another important difference between GAs and conventional optimization algorithms is that GAs do not operate on single points but on whole populations of points (which are, in this case, binary strings).

We first tried a standard GA [5,6] with proportional (standard roulette wheel) selection, one-point crossing over with uniform selection of the crossing point, bitwise mutation, and full replacement of the parent generation by its offsprings. The size of the population m was constant, the length of the strings was 48 (compare with the coding above, see [4] or [5] for an overview of more sophisticated variants of GAs). The following algorithm shows schematically how such a procedure works.

Algorithm 1.

$t := 0$;
Compute initial population $\mathcal{B}_0 = (b_{1,0}, \ldots, b_{m,0})$;

WHILE *stopping condition not fulfilled* **DO**
BEGIN
  FOR $i := 1$ **TO** m **DO**
    select an individual $b_{i,t+1}$ *from* $\mathcal{B}_t$;

  FOR $i := 1$ **TO** $m-1$ **STEP** *2* **DO**
    IF $\mathsf{Random}[0,1] \leq p_{\mathbf{C}}$ **THEN**
      cross $b_{i,t+1}$ *with* $b_{i+1,t+1}$;

  FOR $i := 1$ **TO** m **DO**
    eventually mutate $b_{i,t+1}$;

  $t := t + 1$
END

As apparent from Alg. 1, crossing over is only done with a probability p_{C}. Figure 7 shows graphically how standard one-point crossing over works. In the mutation step each bit is negated with a fixed, yet small, probability p_{M}.

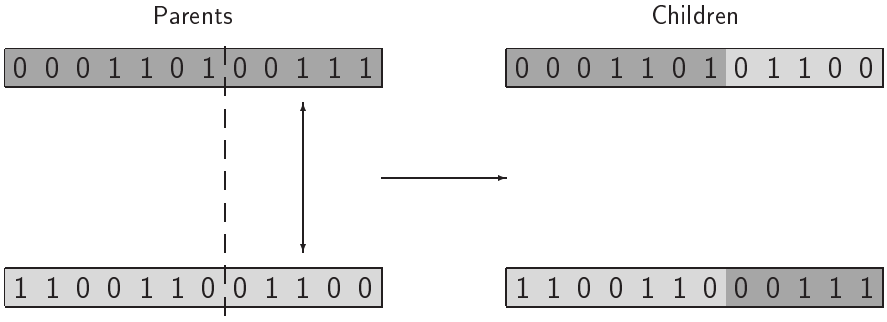

Fig. 7. One-point crossing over of binary strings.

Roulette wheel selection, more specifically, selects an individual with a probability proportional to its normalized fitness (i.e., the sum of fitness values of all individuals is scaled such that their sum is 1). This type of selection is, of course, only applicable if all fitness values are greater than zero, which is guaranteed in this application.

Algorithm 2.

$x := \mathsf{Random}[0,1]$;
$i := 1$

WHILE $i < m$ & $x < \sum_{j=1}^{i} f(b_{j,t}) / \sum_{j=1}^{m} f(b_{j,t})$ **DO**
  $i := i + 1$;

select $b_{i,t}$;

In order to compare the performance of the GAs with other well-known probabilistic optimization methods, we additionally considered the following methods:

Hill climbing: always moves to the best-fitted neighbor of the current string until a local maximum is reached; the initial string is generated randomly.

Simulated annealing: powerful, often used probabilistic method which is based on the imitation of the solidification of a crystal under slowly decreasing temperature (see [7,9] for a detailed description)

Each one of these methods requires only a few binary operations in each step. Most of the time is consumed by the evaluation of the fitness function. So, it is near at hand to take the number of evaluations as a measure for the speed of the algorithms.

Results

All these algorithms are probabilistic methods, therefore their results are not well-determined, they can differ randomly within certain boundaries. In order to get more information about their average behavior, we tried each one of them 20 times for one certain problem. For the given problem we found out that the maximal degree of matching between the reference classification and the classification actually obtained by the fuzzy system was 94.3776% . Table 2 shows the results in more detail, where $f_{\max}$ is the fitness of the best and $f_{\min}$ is the fitness of the worst solution; $\bar{f}$ denotes the average fitness of the 20 solutions, σ_f denotes the standard deviation of the fitness values of the 20 solutions, and # stands for the average number of evaluations of the fitness function which was necessary until the stopping criterion (i.e. that a local maximum is reached in case of hill climbing or that no further improvement can be observed for 100 iterations in case of simulated annealing or genetic algorithms) was fulfilled.

The hill climbing method with a random selection of the initial string converged rather quickly. Unfortunately, it was always trapped in a local maximum, but never reached the global solution (at least in these 20 trials).

The simulated annealing algorithm showed similar behavior at the very beginning. After tuning the parameters involved, the performance improved remarkably.

The raw genetic algorithm was implemented with a population size of 20; p_{C} was set to 0.15, p_{M} was 0.005. It behaved pretty well from the beginning, but it seemed inferior to the improved simulated annealing.

Next, we tried a hybrid GA, where we kept the genetic operations and parameters of the raw GA, but every 50th generation the best-fitted individual was taken as initial string for a hill climbing method. Although the performance increased slightly, the hybrid method still seemed to be worse than the improved simulated annealing algorithm. The reason, that the effects of this modification were not so dramatic, might be that the probability is rather high that the best individual is already a local maximum. So we modified the procedure again. This time, a *randomly chosen individual* of every 25th generation was used as initial solution of the hill climbing method. The result exceeded the expectations by far. The algorithm was, in all cases, nearer to the global solution than the improved simulated annealing was (compare with Table 2), but, surprisingly, sufficed with less invocations of the fitness function.

	$f_{\mathbf{max}}$	$f_{\mathbf{min}}$	$\bar{f}$	σ_f	#
Hill Climbing	94.3659	89.6629	93.5536	1.106	862
Simulated Annealing	94.3648	89.6625	93.5639	1.390	1510
Improved Simulated Annealing	94.3773	93.7056	94.2697	0.229	21968
GA	94.3760	93.5927	94.2485	0.218	9910
Hybrid GA (elite)	94.3760	93.6299	94.2775	0.207	7460
Hybrid GA (random)	94.3776	94.3362	94.3693	0.009	18631

Table 2. Some results

Figure 8 shows a graphical visualization of the results. Each line in this graph corresponds to one algorithm. The curve shows, for a given fitness value x, how many of the 20 different solutions had a fitness higher or equal to x. It can be seen easily from this graph that the hybrid GA with random selection led to the best results. Note that the x-axis is not a linear scale in this figure. It was transformed in order to make small differences visible.

6 Conclusion

In the first part of this paper, we demonstrated the synergy which lies in the combination of fuzzy systems with, more or less, conventional methods. This combination is in particular suitable for designing specific algorithms for time-critical problems. This specifity, however, often results in a loss of universality.

In the second part, we showed the suitability of genetic algorithms for finding the optimal parameters of a fuzzy system, especially if the analytical properties of the objective function are bad. Moreover, hybridization has been discovered as an enormous potential for improvements of genetic algorithms.

Acknowledgement

Ulrich Bodenhofer is working in the framework of the *Kplus Competence Center Program* which is funded by the Austrian Government, the Provincial Government of Upper Austria, and the Chamber of Commerce of Upper Austria.

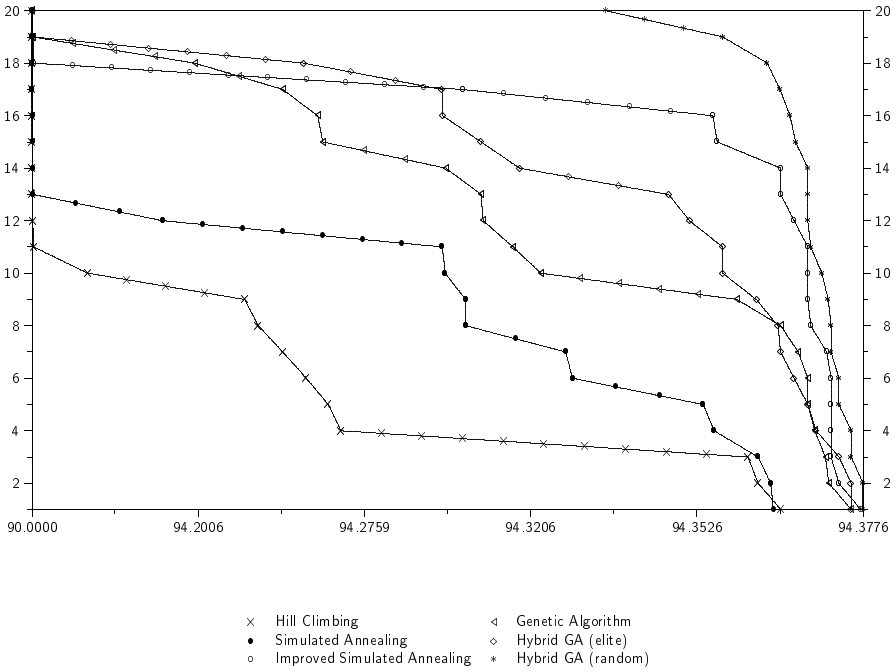

Fig. 8. A graphical representation of the results.

References

1. P. Bauer, U. Bodenhofer, and E. P. Klement. A fuzzy method for pixel classification and its application to print inspection. In *Proc. IPMU'96*, volume 3, pages 1301–1305, 1996.
2. J. C. Bezdek and S. K. Pal, editors. *Fuzzy Models for Pattern Recognition.* IEEE Press, New York, 1992.
3. I. Daubechies. Orthonormal bases of wavelets with finite support-connection with discrete filters. In J. M. Combes, A. Grossmann, and P. Tchamitchian, editors, *Wavelets.* Springer, Berlin, 1989.
4. A. Geyer-Schulz. *Fuzzy Rule-Based Expert Systems and Genetic Machine Learning*, volume 3 of *Studies in Fuzziness.* Physica Verlag, Heidelberg, 1995.
5. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.
6. J. H. Holland. *Adaptation in Natural and Artificial Systems.* The MIT Press, Cambridge, MA, first MIT Press edition, 1992. First edition: University of Michigan Press, 1975.
7. P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* Kluwer Academic Publishers, Dordrecht, 1987.
8. H. Neunzert and B. Wetton. Pattern recognition using measure space metrics. Technical Report 28, Universität Kaiserslautern, Fachbereich Mathematik, November 1987.

9. R. H. J. M. Otten and L. P. P. P. van Ginneken. *The Annealing Algorithm.* Kluwer Academic Publishers, Boston, 1989.
10. A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume II. Academic Press, San Diego, CA, second edition, 1982.
11. E. H. Ruspini. A new approach to clustering. *Inf. Control*, 15:22–32, 1969.
12. K. Shimojima, T. Fukuda, and Y. Hasegawa. Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm. *Fuzzy Sets and Systems*, 71(3):295–309, 1995.
13. H. G. Stark. Multiscale analysis, wavelets, and texture quality. Technical Report 41, Universität Kaiserslautern, Fachbereich Mathematik, January 1990.
14. H. Takagi and M. Lee. Neural networks and genetic algorithms to auto design of fuzzy systems. In E. P. Klement and W. Slany, editors, *Lecture Notes in Artificial Intelligence*, volume 695, pages 68–79. Springer, Berlin, 1993.
15. H. Weyl. Über die Gleichverteilung von Zahlen mod. Eins. *Math. Ann.*, 77:313–352, 1916.