A Fuzzy System for Image Pixel Classification and its Genetic Optimization

Peter Bauer, Ulrich Bodenhofer, Erich Peter Klement
Fuzzy Logic Laboratorium Linz–Hagenberg
Institut für Mathematik, Johannes Kepler Universität
A-4040 Linz, Austria
email: ip@flll.uni-linz.ac.at

Abstract

In this paper a fuzzy method for a certain kind of image pixel classification is introduced. It is the most important result of the development of an inspection system for a silk–screen printing process. The algorithm computes a fuzzy segmentation of a given image into four different types of areas which are to be checked by applying different criteria. The second part of the text deals with the optimization of the parameters involved. For this purpose some variants of genetic algorithms are tried out and compared with other probabilistic optimization methods.

1 The Basic Idea

As anticipated above, we have to decide for each pixel of an image to which kind of area it belongs. The following four types were specified by experts of the collaborating company. For certain reasons, which can be explained with the special principles of the silk-screen printing process, it is sufficient to consider only these types:

Homogeneous area: uniformly colored area

Edge area: pixels within or close to visually significant edges

Raster: area which looks rather homogeneous from a certain distance, but which is actually obtained by printing small raster dots of two or even more colors

Aquarelle: rastered area with high chaotic deviations (e.g. small high-contrasted details in picture prints)

The magnifications in figure 1 show how theses areas typically look like. Of course, transitions between two or more of these areas are possible, hence a fuzzy model is recommendable.

First of all we should define more precisely what we are talking about.

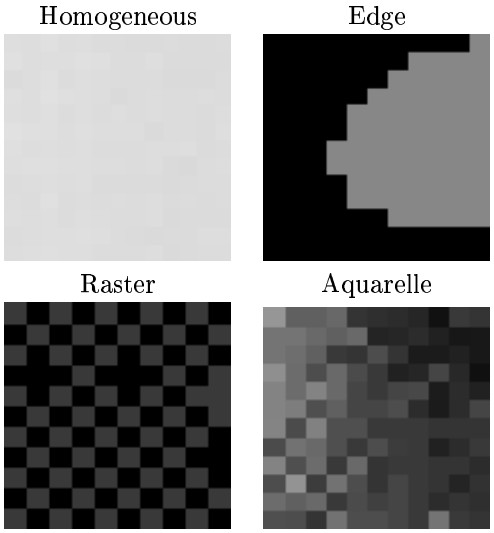

Figure 1: Magnifications of typical representatives of the four types

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

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

with 3-dimensional entries $(u_r(i,j), u_g(i,j), u_b(i,j)) \in \{0,\ldots,255\}^3$ is called a 24-bit color image of size $N \times M$. A coordinate pair (i,j) is called a pixel and the values $(u_r(i,j), u_g(i,j), u_b(i,j))$ are called the gray-values of pixel (i,j).

It is near at hand to use something like the variance or an other measure for deviations to distinguish between areas which show only low deviations, such as homogeneous areas and rasters, and areas with high deviations, such as edge areas or aquarelles. On the contrary, it is intuitively clear that such a measure can never be used to separate edge areas from aquarelles, because any geometrical information is neglected. Experiments have shown that well-known standard edge detectors, like the Laplacian and Mexican Hat, cannot distinguish sufficiently if deviations are chaotic or anisotropic. Another possibility we also took into con-

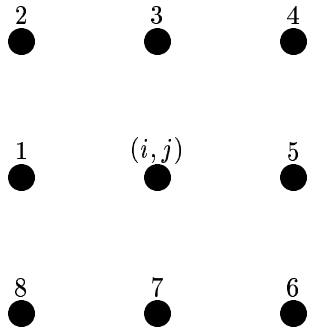

Figure 2: Enumeration of the neighborhood of a pixel

sideration was to use wavelet transforms. Since the size of the image is approximately 1 Megabyte and the segmentation has to be done in at most three seconds, it is obvious that such highly advanced methods would require too much time. Finally we found a fairly good alternative which is based on the discrepancy norm. This approach uses only, as filter masks like the Laplacian or the Mexican Hat also do, the closest neighborhood of a pixel. Figure 2 shows how the neighbors are enumerated for our algorithm. For an arbitrary but fixed pixel (i, j) we can define the enumeration mapping l with the following table:

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

(2)

If we plot one color extraction with respect to this enumeration, i.e $(u_x(l(k)))_{k\in\{1,\ldots,8\}}$ where $x \in \{r, g, b\}$, we typically get curves like those ones shown in figure 3. From these sketches it can be seen easily that a measure for the deviations can be used to distinguish between homogeneous areas, rasters, and the other two types. On the other hand, the most eyecatching difference between aquarelles and edge areas is that edge areas show long connected peaks while aquarelles typically show chaotic, mostly narrow peaks. So a method which judges the shape of the peaks should be used in order to separate edge areas from aquarelles. A simple but effective method for this purpose is the so-called discrepancy norm.

2 The discrepancy norm $\|.\|_D$

Definition 2

$$\begin{aligned} \|.\|_D : \quad & \mathbf{R}^n \longrightarrow \mathbf{R}^+ \\ & (x_1, \ldots, x_n) \mapsto \max_{1\le\alpha\le\beta\le n} \left| \sum_{i=\alpha}^{\beta} x_i \right| \end{aligned} \tag{3}$$

It can be proven easily that this mapping is a norm, a proof is provided e.g. in [Bodenhofer, 1996].

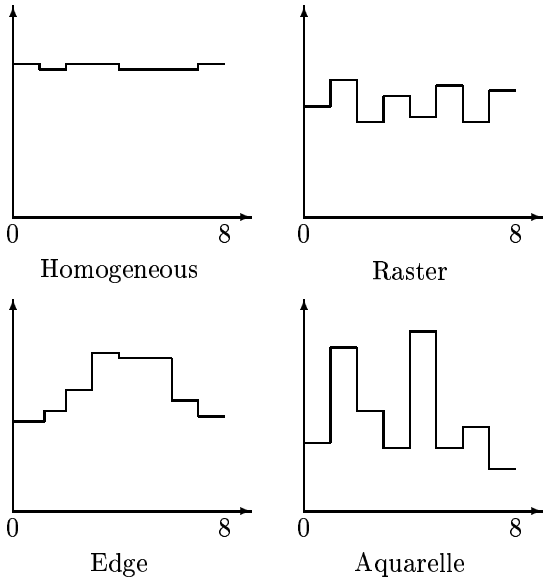

Figure 3: Typical gray-value curves of the form $(u_x(l(k)))_{k\in\{1,\ldots,8\}}$

In measure theory the discrepancy between two measures μ and ν on $\mathbf{R}$ is defined by $\mathcal{D}(\mu,\nu) := \max_{a\le b} |\mu([a,b]) - \nu([a,b])|$. If we have two discrete measures $\overline{\mu}$ and $\overline{\nu}$ on the set $\{1,\ldots,n\}$, where $\overline{\mu}(i) =: x_i$ and $\overline{\nu}(i) =: y_i$, then $\mathcal{D}(\overline{\mu},\overline{\nu})$ and $\|\vec{x}-\vec{y}\|_D$ are equal. Thus we will call $\|.\|_D$ discrepancy norm in $\mathbf{R}^n$.

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

Theorem 3

$$\|\vec{x}\|_D = \max_{1\le\beta\le n} X_\beta - \min_{1\le\alpha\le n} X_\alpha, \tag{4}$$

where the values

$$X_j := \sum_{i=1}^{j} x_i$$

denote the partial sums.

Proof: If we assign 0 to x_0 and x_{n+1} we can conclude

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

■

3 The Fuzzy System

For each pixel (i,j) we consider the nearest eight neighbors enumerated as described in section 1. Then we can use

$$\begin{aligned} v(i,j) := & \sum_{k=1}^{8} \left(u_r(l(k)) - \overline{r}\right)^2 \\ & + \sum_{k=1}^{8} \left(u_g(l(k)) - \overline{g}\right)^2 \\ & + \sum_{k=1}^{8} \left(u_b(l(k)) - \overline{b}\right)^2 \end{aligned} \tag{5}$$

as a measure for the size of the deviations in the neighborhood of (i,j) and

$$\begin{aligned} e(i,j) := & \| u_r(l(.)) - (\overline{r},\ldots,\overline{r}) \|_D \\ & + \| u_g(l(.)) - (\overline{g},\ldots,\overline{g}) \|_D \\ & + \| u_b(l(.)) - (\overline{b},\ldots,\overline{b}) \|_D \end{aligned} \tag{6}$$

as a measure whether the pixel is part of or lying adjacent to a visually significant edge, where $\overline{r}$, $\overline{g}$ and $\overline{b}$ denote the mean values

$$\begin{aligned} \overline{r} &:= \tfrac{1}{8} \sum_{k=1}^{8} u_r(l(k)) \\ \overline{g} &:= \tfrac{1}{8} \sum_{k=1}^{8} u_g(l(k)) \\ \overline{b} &:= \tfrac{1}{8} \sum_{k=1}^{8} u_b(l(k)). \end{aligned}$$

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 done in a rather simple way: We have to compute the degrees of membership to which the pixel belongs to the four types of areas. Hence the output of the fuzzy system is a vector $t(i,j) = (t_{\mathsf{H}}(i,j), t_{\mathsf{E}}(i,j), t_{\mathsf{R}}(i,j), t_{\mathsf{A}}(i,j))$ with $t_{\mathsf{H}}, t_{\mathsf{E}}, t_{\mathsf{R}}, t_{\mathsf{A}} \in [0,1]$. Since the parameterization of the fuzzy systems is independent from the coordinates in our case we just write v and e for the two inputs $v(i,j)$ and $e(i,j)$ which are treated as linguistic variables in the following. Experiments have shown that $[0,600]$ and $[0,200]$ are appropriate universes of discourse for v and e, respectively. We used simple fuzzy partitions for the fuzzy decomposition of the input space. Their typical shape can be seen in figure 5.

Five rules, which cover all the possible cases, complete the fuzzy system:

IF	v **is** low			**THEN**	$t = \mathsf{H}$
IF	v **is** med	**AND**	e **is** high	**THEN**	$t = \mathsf{E}$
IF	v **is** high	**AND**	e **is** high	**THEN**	$t = \mathsf{E}$
IF	v **is** med	**AND**	e **is** low	**THEN**	$t = \mathsf{R}$
IF	v **is** high	**AND**	e **is** low	**THEN**	$t = \mathsf{A}$

Experimental results: Since it is not possible to reproduce the results with a black and white print, results will be shown in the lecture on color transparencies. The algorithm does not even take three seconds for an image with approximately 200000 pixels on a workstation with a 150 MHz CPU.

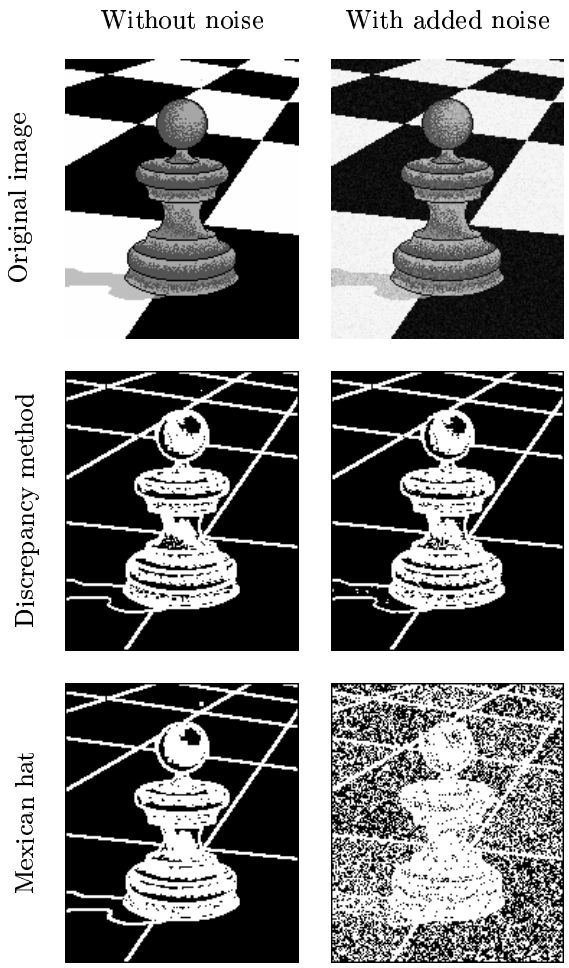

Figure 4: Comparison between e and a standard 3×3 filter mask

4 Genetic Optimization

As apparent from figure 5, the partitions depend on the six parameters v_1, v_2, v_3, v_4, e_1, and e_2. An interesting question is, of course, how to choose these values properly. In order to optimize them, we need an objective criterion for judging the quality of the decision. Unfortunately, the specification of the four types is given in a verbal, vague form, which cannot be formalized mathematically. However, it can be decided by a human whether the result of the segmentation algorithm for given parameters matches his own understanding of the four areas. So we implemented a little painting program with pencils, rubbers, edge detection- and filling algorithms which can be used to make a segmentation by hand. This handmade segmentation can then be used as a reference.

Now assume that we have N sample pixels for which the inputs $(v_k, e_k)_{k \in \{1,\ldots,N\}}$ are already computed and that we have a reference classification of these pixels $\tilde{t}(k) = (\tilde{t}_{\mathsf{H}}(k), \tilde{t}_{\mathsf{E}}(k), \tilde{t}_{\mathsf{R}}(k), \tilde{t}_{\mathsf{A}}(k))$ where

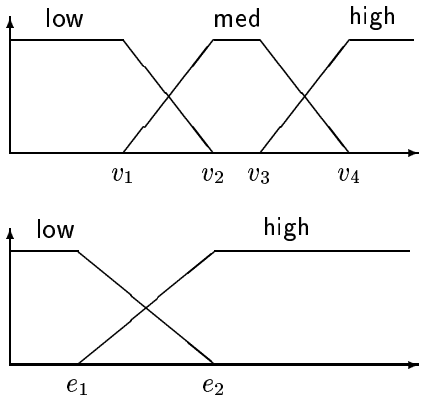

Figure 5: The linguistic variables v and e

$k \in \{1, \ldots, N\}$.[1] Then one possibility to define the performance (fitness) of the fuzzy systems would be

$$\frac{1}{N} \sum_{k=1}^{N} \big(d(t_{\mathsf{H}}(k), \tilde{t}_{\mathsf{H}}(k)) + d(t_{\mathsf{E}}(k), \tilde{t}_{\mathsf{E}}(k)) + d(t_{\mathsf{R}}(k), \tilde{t}_{\mathsf{R}}(k)) + d(t_{\mathsf{A}}(k), \tilde{t}_{\mathsf{A}}(k)) \big), \tag{7}$$

where $t(k) = (t_{\mathsf{H}}(k), t_{\mathsf{E}}(k), t_{\mathsf{R}}(k), t_{\mathsf{A}}(k))$ are the classifications actually obtained by the fuzzy system for the input pairs (v_k, e_k) with respect to the parameters v_1, v_2, v_3, v_4, e_1, and e_2. $d(.,.)$ is an arbitrary metric on $[0, 1]$. 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. In order to keep the amount of computation low, we "simplified" the procedure as follows: Choose a partition $(P_1, \ldots, P_K)$ of the input space and count the number $(n_1, \ldots, n_K)$ of sample points $\{p_1^j, \ldots, p_{n_j}^j\}$ each part contains. Then the desired classification of a certain part can be defined as

$$\tilde{t}_X(P_i) := \frac{1}{n_i} \sum_{j=1}^{n_i} \tilde{t}_X(p_j^i) \quad \text{with } X \in \{\mathsf{H}, \mathsf{E}, \mathsf{R}, \mathsf{A}\}. \tag{8}$$

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

$$f(v_1, \ldots, v_4, e_1, e_2) := \frac{50}{N} \sum_{i=1}^{K} n_i \cdot (2 - (*)) \,. \tag{9}$$

with

$$(*) := \sum_{X \in \{\mathsf{H},\mathsf{E},\mathsf{R},\mathsf{A}\}} \big(\tilde{t}_X(P_i) - t_X(\phi(P_i)) \big)^2 .$$

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 it would take if we used (7).

[1] Since the geometry plays no role if the values v and e are already computed, we can switch to one dimensional indices here, what simplifies the formulas a little bit.

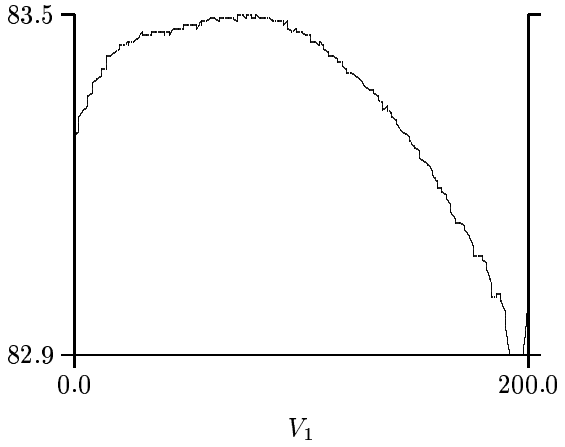

Figure 6: Cross section of a function of type (9)

Remark 4 Note that in (9) 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 has then to be maximized.

Figure 6 shows a sketch of such a fitness function, where v_2, v_3, v_4, e_1 and e_2 are kept constant and v_1 is varied between 0 and 200. It can be seen easily that f is continuous but not differentiable and that there are a lot of local maxima. Hence, it is not recommendable to use a conventional optimization method, such as gradient descent or a Newton-like method. Seemingly the one and only way out of this trap was to use a probabilistic method. This requires, first of all, a 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:

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	=	$s_5 + s_6$

If fuzzy sets of a more general shape are used, this coding is unapplicable. Codings for such cases can, for instance, be found in [Takagi and Lee, 1993] or [Shimojima *et al.*, 1995].

In order to compare the performance of various approaches, we 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 was 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 [van Laarhoven and Aarts, 1987] or [Otten and van Ginneken, 1989] for a detailed description)

Raw genetic algorithm with proportional selection, one-point crossing over, and bitwise mutation (see [Goldberg, 1989], [Geyer-Schulz, 1994], or [Holland, 1975]); the size of the population was 20.

Hybrid GA: combination of a genetic algorithm with the hill climbing method

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. So we tried out each one of them 20 times for one certain problem in order to get more information about their behavior. 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 1 shows the results in more detail.

The hill climbing method with a random selection of the initial string converged rather quickly. Unfortunately it always got caught 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 involved parameters, the performance could be improved remarkably.

The raw genetic algorithm looked pretty good from the first run on, 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 50-th generation the best-fitted individual was taken as initial string for a hill climbing method. Although the performance increased, the hybrid method still seemed to be worse than the improved simulated annealing algorithm. The reason that the effects of the modification were not so dramatical 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 25-th generation was used as initial string of the hill climbing method. The result exceeded the expectations by far. The algorithm was, in all cases, nearer

	f_{max}	f_{min}	$\bar{f}$	σ_f	It
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 1: Some results

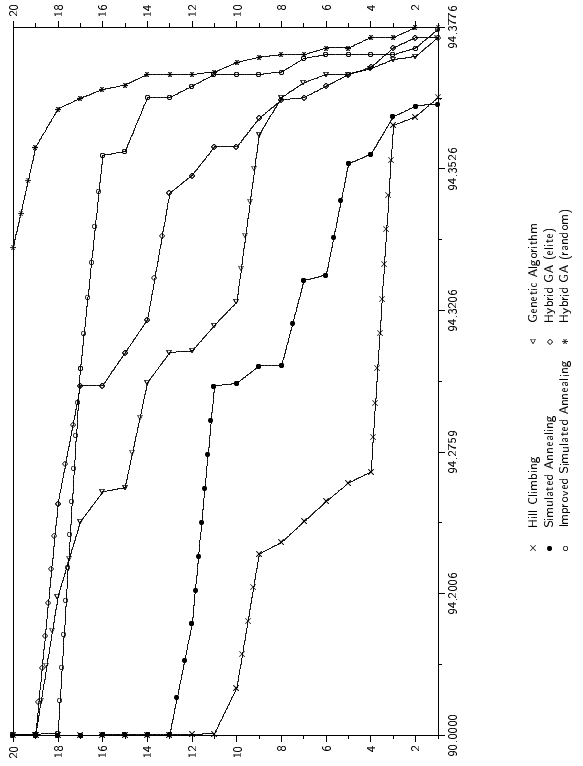

Figure 7: A graphical representation of the results

to the global solution than the improved simulated annealing was (compare with table 1) but, surprisingly, sufficed with less invocations of the fitness function.

Figure 7 shows the results in more detail. 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 brought the best results. The x axis is not a linear scale in this figure. It was transformed in order to make small differences visible.

References

[Bodenhofer, 1996] U. Bodenhofer. Tuning of fuzzy systems using genetic algorithms. Master's thesis, University of Linz, 1996.

[Geyer-Schulz, 1994] A. Geyer-Schulz. *Fuzzy Rule-Based Expert Systems and Genetic Machine Learning.* Springer, Heidelberg, 1994.

[Goldberg, 1989] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989.

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

[Klement and Slany, 1993]
E. P. Klement and W. Slany, editors. *Fuzzy Logic in Artificial Intelligence*, volume 695 of *Lecture Notes in Artificial Intelligence.* Springer, Berlin, Heidelberg, June 1993.

[Otten and van Ginneken, 1989] R. H. J. M. Otten and L. P. P. P. van Ginneken. *The Annealing Algorithm.* Kluwer Academic Publishers, Boston/Dordrecht/London, 1989.

[Shimojima *et al.*, 1995] 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*, 71:295–309, 1995.

[Takagi and Lee, 1993] 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, 1993.

[van Laarhoven and Aarts, 1987] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* Kluwer Academic Publishers, Dordrecht/Boston/London, 1987.