Evolution of Voronoi-based Fuzzy Controllers

Carlos Kavka, Marc Schoenauer

▶ To cite this version:

Carlos Kavka, Marc Schoenauer. Evolution of Voronoi-based Fuzzy Controllers. PPSN'04, Prof. Xin Yao, 2004, Birmingham, pp.541-550. inria-00000028

HAL Id: inria-00000028

https://hal.inria.fr/inria-00000028

Submitted on 14 May 2005

Evolution of Voronoi-based Fuzzy Controllers

Carlos Kavka[1] and Marc Schoenauer[2]

[1] LIDIC, Departamento de Informática, Universidad Nacional de San Luis
D5700HHW, San Luis, Argentina – `ckavka@unsl.edu.ar`
[2] Équipe TAO, INRIA Futurs – LRI, Université de Paris Sud
91405, Orsay Cedex, France – `Marc.Schoenauer@inria.fr`

Abstract. A fuzzy controller is usually designed by formulating the knowledge of a human expert into a set of linguistic variables and fuzzy rules. One of the most successful methods to automate the fuzzy controllers development process are evolutionary algorithms. In this work, we propose a so-called "approximative" representation for fuzzy systems, where the antecedent of the rules are determined by a multivariate membership function defined in terms of Voronoi regions. Such representation guarantees the ϵ-completeness property and provides a synergistic relation between the rules. An evolutionary algorithm based on this representation can evolve all the components of the fuzzy system, and due to the properties of the representation, the algorithm (1) can benefit from the use of geometric genetic operators, (2) does not need genetic repair algorithms, (3) guarantees the completeness property and (4) can implement previous knowledge in a simple way by using adaptive a priori rules. The proposed representation is evaluated on an obstacle avoidance problem with a simulated mobile robot.

1 Introduction

One of the most successful areas of application of fuzzy logic is control, where fuzzy controllers have proved to be very effective in the context of controlling complex ill defined processes [7]. A fuzzy controller is usually designed by formulating the knowledge of a human expert into a set of linguistic variables and fuzzy rules [4]. However, there is still no systematic way to perform this process. A large number of methods to automate this process and to evaluate and fine tune the obtained fuzzy controllers have been proposed in the literature, with methods based on reinforcement learning, neural networks and evolutionary algorithms being the most successful ones (see [8, 11] and references therein).

Defining a fuzzy rule amounts to select the membership functions for the input variables, and the corresponding values for the outputs of the rule. An important issue is to ensure that the whole search space is covered by the set of fuzzy rules. A way to overcome this problem is to use the so-called grid representation, i.e. to define fuzzy sets from intervals of the values of the input variables. However, this requires a large number of parameters (the interval values), especially as the dimension of the input space increases.

On the other hand, partitions of an n-dimensional space can be easily evolved using Voronoi diagrams [12], and a complete set of fuzzy rules can hence be defined by attaching a linear function to each subset of a such Voronoi partition: this is the basic idea of the *Fuzzy Voronoi* representation for fuzzy systems that is proposed in this work.

The paper is organized as follows: section 2 introduces the FV representation for fuzzy systems and its evolution, and discusses some interesting properties like the ϵ-completeness, and the way a priori rules can be added by the user while their application domain is evolved by the evolution. Section 3 validates the proposed approach with some experimental results on a simple problem of Evolutionary Robotics. Finally, section 4 draws some quick conclusions.

2 The Fuzzy Voronoi Representation

This section introduces the basic concepts of computational geometry used to construct the FV representation, and describes how these concepts are used to represent Takagi-Sugeno fuzzy systems, before discussing the main properties of this representation.

2.1 Domain Partition

The domain partition strategy is based on Voronoi diagrams. A Voronoi diagram induces a subdivision of the space based on a set of points called *sites*. Formally [2], a Voronoi diagram of a set of p points $\mathcal{P} = \{P_1, \ldots, P_p\}$ is the
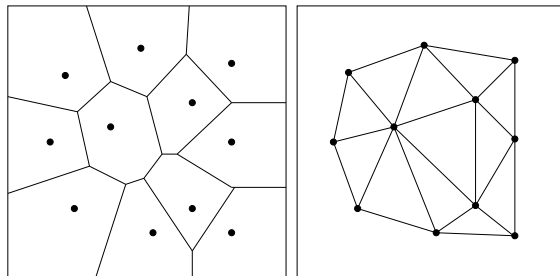
Fig. 1. An example of a Voronoi diagram (left) and the corresponding Delaunay triangulation (right) for a set of points in $\mathbb{R}^2$

subdivision of the plane into p cells, one for each site in $\mathcal{P}$, with the property that a point M lies in the cell corresponding to a site P_i if and only if the distance between M and P_i is smaller than the distance between P and all other P_j ($j \neq i$). Formally, the Voronoi cell defined by the site P_i is defined as: $\mathcal{V}(P_i) = \{Q \mid dist(Q, P_i) \leq dist(Q, P_j) \ \forall \ i \neq j\}$, where $dist(x, y)$ is the Euclidean distance between the points x and y. A related concept is the so called Delaunay

triangulation $\mathcal{T}$, defined as the maximal planar subdivision (i.e. a subdivision such that no edge connecting two vertices can be added to S without destroying its planarity) whose vertex set is $\mathcal{P}$ and such that the circumcircle of any triangle in T does not contain any point of $\mathcal{P}$ in its interior. Figure 1 illustrates an example of a Voronoi diagram and its corresponding Delaunay triangulation in $\mathbb{R}^2$. Note that these definitions can be straightforwardly extended to $\mathbb{R}^n$, with $n \geq 2$ – all details can be found in [2].

2.2 The FV representation for Takagi-Sugeno fuzzy systems

A general Takagi-Sugeno fuzzy system has l input variables $x_1, x_2, \ldots, x_l$ and m output variables $v_1, v_2, \ldots, v_m$ [1]. Rule R_k of such a fuzzy system has the following form:

$$
\begin{aligned}
if \ x \ is \ S_k \ \ then \ \ &v_1^k = a_{10}^k + a_{11}^k x_1 + \ldots + a_{1l}^k x_l \\
&and \ \ldots \ and \\
&v_m^k = a_{m0}^k + a_{m1}^k x_1 + \ldots + a_{ml}^k x_l
\end{aligned}
\tag{1}
$$

where S_k is a fuzzy set, $x = (x_1, x_2, \ldots, x_l)$ the input vector, and a_{j0}^k and a_{ji}^k ($1 \leq i \leq l$, $1 \leq j \leq m$) are the real valued parameters defining the outputs as a linear combination of the inputs.

The membership value of the input vector x to the fuzzy set S_k can be defined in different ways. For instance, all input variables can be fuzzified independently: assume x_i is fuzzified by $p_i \geq 1$ fuzzy sets A_{ij} ($1 \leq i \leq l$, $1 \leq j \leq p_i$) with membership functions μ_{ik_i} ($k_i = 1, 2, \ldots, p_i$). Then the left-hand side of Equation (1) becomes $if \ x_1 \ is \ A_1^k \ and \ \ldots \ and \ x_l \ is \ A_l^k$, where each A_i^k is one of the A_{ij}, the antecedent fuzzy sets (or linguistic labels) associated to the input variable x_i.

The FV representation on the other hand considers joint fuzzy sets defined from a Voronoi diagram $\mathcal{P} = \{P_1, \ldots, P_p\}$. There are as many rules as Voronoi sites, and fuzzy set S_k is defined as by its multivariate membership function μ_k that takes its maximum value 1 at site P_k, and decreases linearly to reach value 0 at the centers of all neighbor Voronoi sites. An example of such a joint fuzzy set is shown in figure 2-a for $n = 2$. Formally, the membership value of the input vector x to the joint fuzzy set S_k is defined by:

$$
\mu_{S_k}(x) = \begin{cases} l_C(x) \ x \in \mathcal{V}(P_k) \\ 0 \quad \ \ \ \text{elsewhere.} \end{cases}
\tag{2}
$$

where $C = P_k$ is the Voronoi site defining S_k and the Voronoi cell $\mathcal{V}(P_k)$, and $l_C(x)$ is the barycentric coordinate of x in the simplex $T_C(x)$ of the Delaunay triangulation of $\mathcal{P}$ that has C as a vertex and to which x belongs. Figure 2-b shows an example of the Voronoi diagram and the associated Delaunay triangulation. On Figure 2-c, the barycentric coordinate $l_C(x)$ corresponds to the (normalized) gray area (volume if $n > 2$) of the sub-simplex formed by x and vertices of simplex $T_C(x)$ but C.

Note that a very large triangle containing all points in the domain is defined in such a way that there are no open Voronoi regions in the input domain.

(a)   (b)   (c)
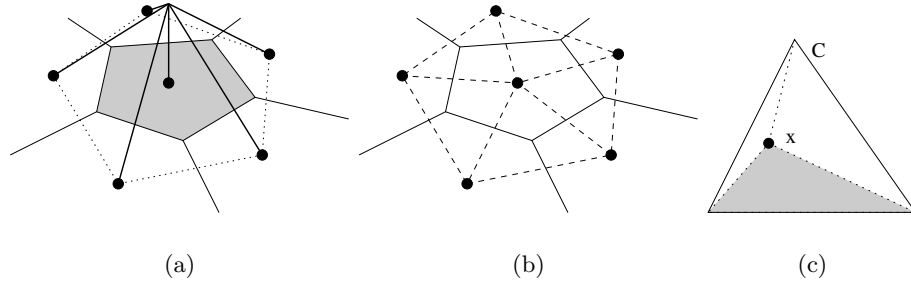
Fig. 2. An example of a (a) joint fuzzy set for a single Voronoi region for $n = 2$, where the membership value is represented in the z-axis, and a (b) Voronoi diagram (solid line) and its corresponding Delaunay triangulation (dotted line) for $n = 2$. The graphic (c) shows an example of the membership computation for $n = 2$. The outer triangle corresponds to the simplex defined by the Delaunay triangulation to which x belongs. The membership value corresponds to the area of the shadowed triangle. Note that the value of the area is 1 when x is equal to C and it goes down linearly to 0 on the side of the triangle opposite to C

Representation: the FV representation is hence defined by a (variable length) Voronoi diagram $\mathcal{P} = \{P_1, \ldots, P_p\}$ (each P_i is defined by n coordinates), and for each $k \in [1, \ldots, p]$, the set of coefficients $a_{ij}^k, i \in [1, \ldots, l], j \in [1, \ldots, m]$ defining the value of the outputs.

2.3 Evaluation of a FV Takagi-Sugeno fuzzy system

In order to evaluate the output of such a FV Takagi-Sugeno fuzzy system at point x, the Delaunay triangulation of the set $\mathcal{P}$ has to be computed. Then, the membership functions corresponding to all Voronoi cells that intersect the simplex $T(x)$ to which x belongs have to be computed. Finally, the value of the output variable v_j is computed by summing up the values v_j^k of each activated rule, weighted by their corresponding membership:

$$v_j = \sum_{S_k/T(x) \wedge S_k \neq \emptyset} \mu_{S_k}(x) v_j^k(x). \tag{3}$$

2.4 Evolution of FV systems

There are two possible approaches for optimizing a FV system: either freeze the number of rules (Voronoi sites), and use any parametric optimization algorithm – but the power of this representation would somehow fade away if the number of sites has to be fixed in advance – or use the flexibility of Evolutionary Algorithms to evolve variable-length genotypes, letting evolution adjust the granularity of the fuzzy rules, i.e. the number and location of the Voronoi sites.

The evolutionary algorithm is described in details in [12, 6]. The crossover operator is based on geometrical exchange of Voronoi sites between both parents

with respect to a random hyperplane. The mutation operator can either modify the parameters of a particular rule by some standard Gaussian mutation, or add or delete a Voronoi site, i.e. a rule (see next subsection 2.5). Practical details on the algorithms, including all parameters, will be given in section 3. But before experimentally validating the FV representation, next subsection will discuss some of its properties.

2.5 Properties

First of all, the FV representation belongs to the class of approximative representations [1], where each fuzzy rule defines its own fuzzy sets. It also provides continuous output, as most fuzzy systems. However, it also has a number of useful properties, that we shall now discuss in turn.

ϵ-**completeness property**: All FV-based fuzzy systems defined with the FV representation fulfills the ϵ-completeness property at any required level, which establishes that any input must belong to at least one fuzzy set with a membership value not smaller than a threshold value ϵ:

$$\forall x \in U \,\exists A \in \{A_1, \dots, A_n\} \, \mu_A(x) \geqslant \epsilon. \tag{4}$$

For the FV representation, it is clear from the definition of the membership function of equation (2) that this property will hold with $\epsilon = \frac{1}{2}$, as $l_C(x)$ will be above 0.5 if x lies in the Voronoi cell defined by C. This property guarantees an adequate representation for every input point, since there is always a rule that is applied with at least a known value of membership.

No need for genetic repair algorithms: Since it is not possible to define wrong or non complete fuzzy systems, the fuzzy systems produced by applying mutation or crossover operators are always valid control systems.

Adaptive fuzzy rules: The influence on the output of a particular fuzzy rule in the FV representation does not only depend on the rule itself, it also depends on all neighbor rules. The area of application $\mathcal{A}_k$ of a fuzzy rule R_k is defined as the union of all Delaunay regions which contain the point P_k, center of the rule R_k. Formally:

$$\mathcal{A}(R_k) = \bigcup_{P_k \in D_j} D_j \quad D_j \in D = \{D_1, \dots, D_\gamma\}. \tag{5}$$

where P_k is the center of the rule R_k and $D = \{D_1, \dots, D_\gamma\}$ is the Delaunay partition of the set $\mathcal{P} = \{P_1, \dots, P_p\}$. Figure 3 shows an example of the application area of some rules in a regular partition, and illustrates the interdependency of application areas of neighboring rules when some rules are removed or added.

The evolutionary algorithm evolves individuals that represent complete fuzzy systems defined by a set of fuzzy rules that are synergistically related, and not
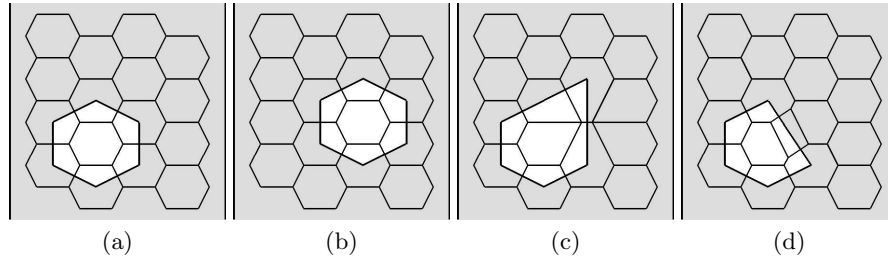
Fig. 3. Diagram (a) shows the application area of a fuzzy rule. Diagram (b) shows the application area of one of its neighbor rule. Diagram (c) shows the application area of the rule of diagram (a) when the rule of diagram (b) is removed, and diagram (d) the application area of the rule of diagram (a) when a rule is added between both rules.

fuzzy systems defined with a set of independent fuzzy rules. The variation operators hence modify the application areas of all fuzzy rules, while still maintaining the required ϵ-completeness level.

Adaptive a priori rules: In most fuzzy systems, the user can incorporate a priori knowledge by manually defining fuzzy sets and the corresponding fuzzy rules. This process implies that some restriction on the output values and the partition of the input space is introduced in the evolutionary process, but the expected benefit is that the evolutionary process, biased toward hopefully good parts of its search space, will converge faster to better solutions. Similarly, the FV representation allows the definition of a priori rules, i.e. fixed Voronoi sites that will not be modified by evolution. But one big advantages of the FV representation is that the expert does not need to specify the application area of such rules: thanks to the synergistic effect described above, the evolutionary process, by adding rules more or less close to the a priori rules will also tune its domain of application – as will be clear on the experimental results in next section.

3 Experiments

3.1 Obstacle avoidance with a Khepera Robot

Control problems are among the most successful applications of fuzzy systems: we have chosen a simple control problem in Evolutionary Robotics (ER) to validate the FV representation. For its simplicity, because it has been the hardware basis for many experiments in ER [10], and because there exists many good simulation platforms (we have used [9]), the Khepera robot was chosen: it has 8 infrared sensors to measure proximity to objects and levels of ambient light; two independent motors are used to control the speed and direction of the robot.

The obstacle avoidance is one of the simplest problems in ER. Following [10], the fitness of a controller is defined by testing the controller on the (simulated) robot in some given arena during a number r of *epochs*, the robot being

positioned randomly in the arena at start of each epoch. During each epoch, fitness accumulates at each time step proportionally to the robot speed, but is decreased if the robot gets too close to a wall. An epoch stops if the robot hits a wall, or after a predefined number of times steps s. The total fitness is the sum of the fitness obtained during all r epochs. More formally, the fitness (to be maximized) is defined by

$$\text{fitness}(I) = \frac{1}{r} \sum_{i=1}^{r} \sum_{t=1}^{s} d(t) * (1 - a(t)) \qquad (6)$$

where t is the time step, $d(t)$ is the normalized forward speed of the robot (sum of the speed of both motors), and $a(t)$ is the normalized maximum activation of the infrared sensors (i.e. a(t)=1 means that the robot is against a wall).

3.2 Experimental Results

In the results reported here, to keep things simple, but not trivial, the controllers have four inputs and two outputs: the inputs are, respectively, the average of the two left sensors, the two front sensors, the two right sensors and the two back sensors; the outputs are the speeds of the two motors. The precise parameters of the evolutionary algorithm are given in table 1.

| Parameter	value
Dimension of the input space	4
Dimension of the output space	2
Population size	50
Number of generations	300
Selection - replacement	Roulette - Generational
Number of epochs per evaluation	40
Number of time steps per epoch	200
Minimum and maximum size of individuals	10 - 30
Crossover rate (Voronoi)	0.8
Mutation rates (Gaussian - addition - deletion)	0.3 - 0.15 - 0.15

Table 1. Parameters of the evolutionary algorithm

The performance of the individuals is evaluated on the scenery shown in Figure 4-a, by performing $r = 40$ evaluations from valid random positions of at most $s = 200$ steps each one. The performance of a representative controller found after evolution is shown in the same Figure 4-a, where the little boxes represent the robot displayed every 30 time steps, for a total of 5000 time steps. It can be appreciated that the robot can successfully navigate through the arena, avoiding obstacles, and moving slowly in narrow regions (boxes are more dense). **Generalization abilities**: Figure 4-b shows the performance of the same controller, evaluated in the same conditions but in a different arena from the one it
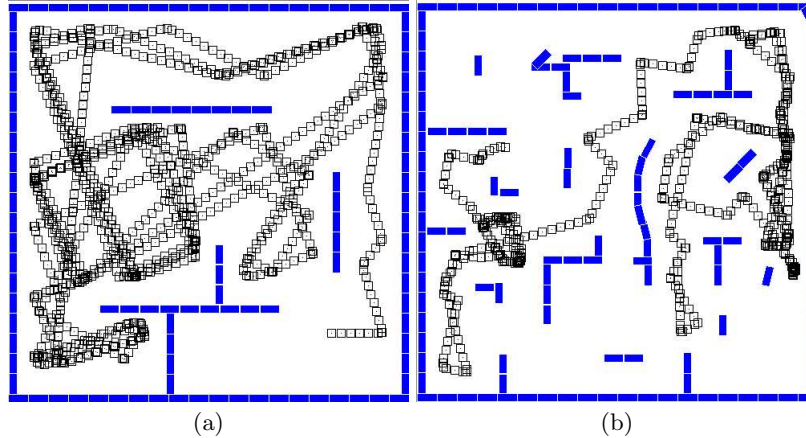
Fig. 4. The graph (a) shows the performance of the best fuzzy controller obtained through evolution in the scenery used for evaluation and the graph (b) shows the performance of the same controller on an unknown environment

had evolved in during evolution. It can be seen that the controller has successfully learned the navigation rules, since it can move and avoid obstacles in an unknown environment. The performance is comparable with the results provided e.g. in [10] and [5], where neuro controllers are trained in a similar arenas.
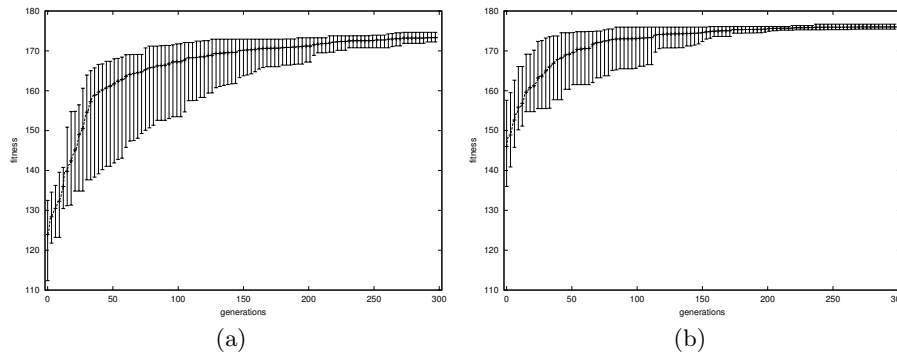

Fig. 5. Fitness vs number of generations, averaged over 5 runs. (a) without a priori rules, and (b) with a priori rules.

A priori rules: Figure 5 shows the usual plot of best fitness (averaged over 5 runs) vs number of generations, together with error bars, when the evolution is performed without (a) and with (b) the a priori rules shown in table 2. The first rule establishes that the robot should go straight ahead (both motors at full speed) when there are no obstacles around (normalized value 0 for all sensors),

point				output v_0					output v_1				
left	*center*	*right*	*back*	a_1	a_2	a_3	a_4	a_5	a_1	a_2	a_3	a_4	a_5
0	0	0	0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1.0	0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

Table 2. A priori rules used during evolution.

and the second rule establishes the same behavior when there are no obstacles in front, but obstacles in all other directions (normalized value 1). Though the final performances reached by the best controllers in the case where a priori rules are used are only slightly better than those in the case without a priori rules, the results in the latter case are obtained more quickly, and, more importantly, are much more robust (smaller error bars). The use of a priori rules makes certainly sense when the simulation is highly costly so the number of runs and the number of generations per run have to be kept small.

(a)   (b)
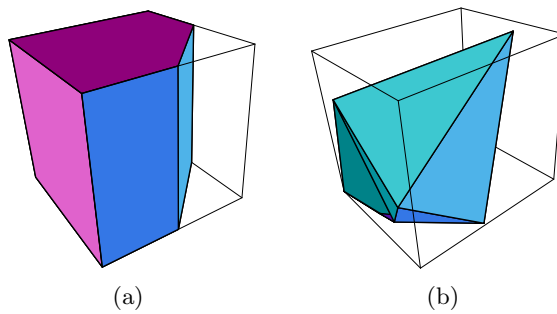
Fig. 6. The (a) original area of application of the first a priori rule and the (b) modified area of application of the same rule at the end of the evolution

The areas of application of the first rule (projected in their first three coordinates) are plotted on Figure 6: (a) shows the initial application area, before evolution started, while (b) is the application of the very same rule after evolution. The algorithm did indeed adjust this domain of application – the user only had to specify the expected behavior at a single point in the input space. Table 3

point				output v_0					output v_1				
left	*center*	*right*	*back*	a_1	a_2	a_3	a_4	a_5	a_1	a_2	a_3	a_4	a_5
0.61	0.98	0.97	0.92	-0.51	-0.04	0.01	0.17	-0.66	0.9	-0.14	0.66	0.48	-0.82
0.82	0.54	0.11	0.67	0.65	-0.46	-0.11	3.31	-0.12	-0.14	-0.31	-0.11	3.54	-0.46

Table 3. An example of rules obtained after evolution.

shows two rules obtained through evolution. The first rule is applied when the only possibility is to move to the left and the second rule when the best option is to move to the right. The normalized outputs produced by the first rule in the center point are 0.01 and 1.0, producing a fast turn to the left. The outputs of the second rule are 0.75 and 0.3, producing a slower turn to the right.

4 Conclusions

A representation for fuzzy controllers based on Voronoi diagrams has been proposed, that can represent fuzzy systems with synergistic rules, fulfilling the ϵ-completeness property and providing a simple way to introduce a priori knowledge. The geometric interpretation of the rules allows the use of geometric genetic operators that proved to be useful also in other contexts. The representation and the algorithms have been validated on a mobile robot obstacle avoidance problem run on a simulator in a four and eight (not shown) dimensions input space, and also on the inverted cart pole system (not shown). Future work include experiments on a real mobile robot, and comparisons with more classical grid representations for Takagi-Sugeno fuzzy systems, and the impact of using the so-called *Symbolic Controllers* approach [3].

References

1. R. Babuška. Fuzzy modeling: Principles, methods and applications. In C. Bonivento, C. Fantuzzi, and R. Rovatti, editors, *Fuzzy Logic Control: Advances in Methodology*, pages 187–220. World Scientific, Singapore, 1998.
2. M. de Berg, M. van Kreveld, M. OVermars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer Verlag, 1998.
3. N. Godzik, M. Schoenauer, and M. Sebag. Evolving symbolic controllers. In G. Raidl et al., editor, *Applications of Evolutionary Computing*, LNCS 2611, 2003.
4. F. Hoffmann. Evolutionary algorithms for fuzzy control system design. *Proceedings of the IEEE, Special Issue on Industrial Innovations using Soft Computing*, 2001.
5. M. Hülse, B. Lara, F. Pasemann, and U. Steinmetz. Evolving neuro-modules and their interfaces to control autonomous robots. *Lecture Notes in Computer Science*, 2130, 2001.
6. C. Kavka and M. Schoenauer. Voronoi diagrams based function identification. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2003.
7. C. Lee. Fuzzy logic in control systems: Fuzzy logic controller - part i. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–418, March/April 1990.
8. P. McQuesten. *Cultural Enhancement of Neuroevolution*. PhD thesis, The University of Texas at Austin, August 2002.
9. O. Michel. Kephera simulator package version 2.0.
10. S. Nolfi and D. Floreano. *Evolutionary Robotics, The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Books, 2000.
11. A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing*, 1(4):180–197, 1997.
12. M. Schoenauer, F. Jouve, and L. Kallel. Identification of mechanical inclusions. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*. Springer Verlag, 1997.