Rule based fuzzy models and identification

L. T. Kóczy

Institute of Information Technology and Electrical Engineering,
Széchenyi István University, Győr, Hungary

&

Department of Telecommunication and Media informatics,
Budapest University of Technology and Economics, Hungary

Outline

Classic rule based fuzzy models

1. Rule interpolation
2. Hierarchical fuzzy rule bases
3. Interpolation of hierarchical rule bases

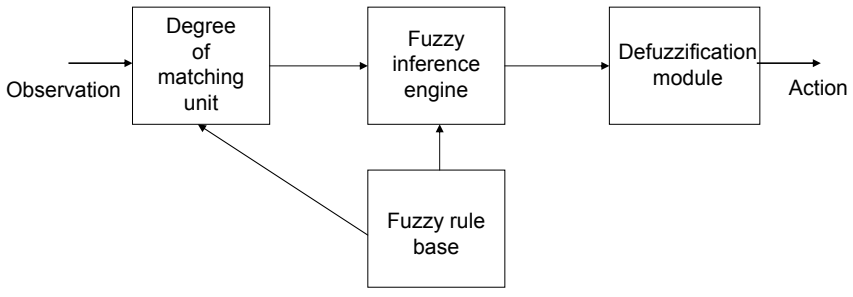
Model Identification

4. Model identification by clustering (flat systems)
5. MI by clustering (hierarchical systems)
6. MI by bacterial memetic algorithm

Conclusions

Classic rule based fuzzy models

General scheme of a fuzzy system



Linguistic variables and their representations

- *Linguistic variable* (*linguistic term*) defined by L. A. Zadeh:

  "By a linguistic variable we mean a variable whose values are words or sentences in a natural or artificial language. For example, *Age* is a linguistic variable if its values are linguistic rather than numerical, i.e., *young*, *not young*, *very young*, *quite young*, *old*, *not very old* and *not very young*, etc., rather than 20, 21, 22, 23, ..."

Linguistic rules: **IF** x = A **THEN** y = B

A is the rule antecedent, B is the rule consequent

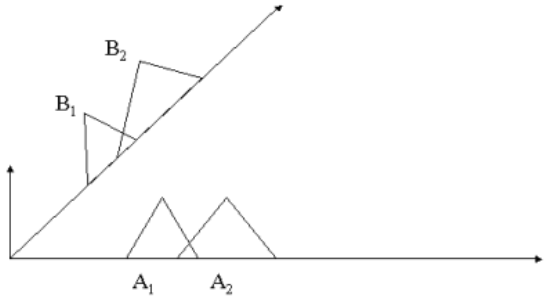
Example: „**IF** traffic is heavy in this direction **THEN** keep the green light longer"

If x = A **then** y = B "*fuzzy point*" A×B

If x = A_i **then** y = B_i i = 1,...,r "*fuzzy graph*"

Fuzzy rule = fuzzy relation (R_i)

Fuzzy rule base = fuzzy relation (R), is the union (s-norm) of the fuzzy rule relations R_i:

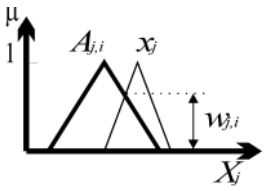


Fuzzy rule base relation R containing two fuzzy rules $A_1 \rightarrow B_1$, $A_2 \rightarrow B_2$ (R_1, R_2)

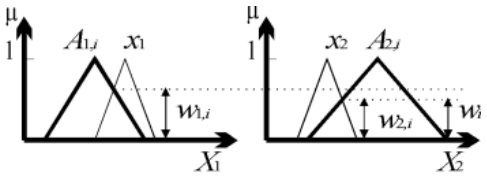
Fuzzy inference mechanism (Mamdani)

- **If** $x_1 = A_{1,i}$ **and** $x_2 = A_{2,i}$ **and**...**and** $x_n = A_{n,i}$ **then** $y = B_i$

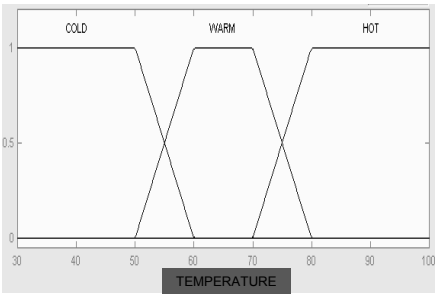
The weighting factor w_{ji} characterizes, how far the input x_j corresponds to the rule antecedent fuzzy set $A_{j,i}$ in one dimension

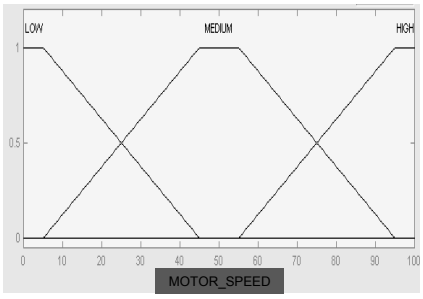


The weighting factor w_i characterizes, how far the input x fulfils to the antecedents of the rule R_i.



Fuzzy systems: an example

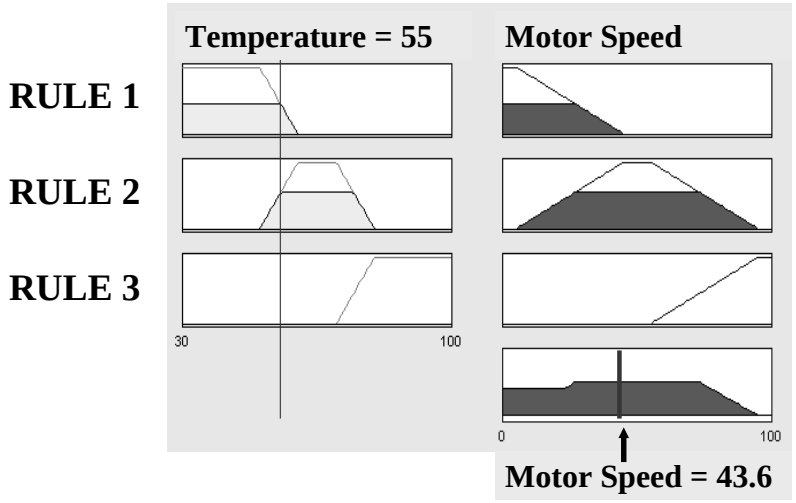




Fuzzy systems operate on fuzzy rules:

IF *temperature* is COLD **THEN** *motor_speed* is LOW
IF *temperature* is WARM **THEN** *motor_speed* is MEDIUM
IF *temperature* is HOT **THEN** *motor_speed* is HIGH

Inference mechanism (Mamdani)



1. Rule interpolation

The Curse of Dimensionality in Fuzzy Control

If there are k input state variables, and in each there are (max) T terms, the number of rules covering the space densely is

$$r = T^k$$

How to decrease this expression?

1. Decrease T
   Sparse rule bases, rule interpolation (Kóczy and Hirota, 1990)
2. Decrease k
   Hierarchical structured rule bases (Sugeno, 1991)
3. Decrease both T and k
   Interpolation of hierarchical rule bases (Kóczy and Hirota, 1993)

Decreasing T

SYMBOLIC EXPERT CONROL

↓

FUZZY CRI/ MAMDANI CONTROL

↓

FUZZY INTERPOLATIVE CONTROL

Decreasing *T*

RIPE

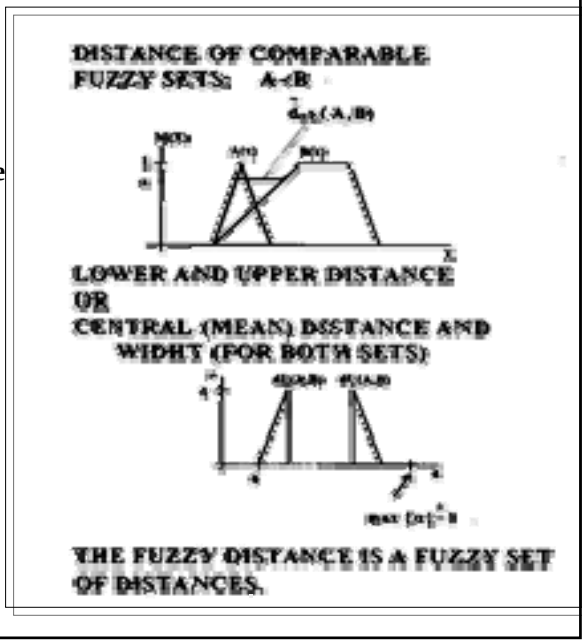
UNRIPE

???

Fuzzy Distance

Fuzzy distance of comparable fuzzy sets:

for all $\alpha \in [0,1]$ - the pairwise distances between the two extrema of these fuzzy sets (the "lower" and the "upper fuzzy distance" of the two α-cuts)



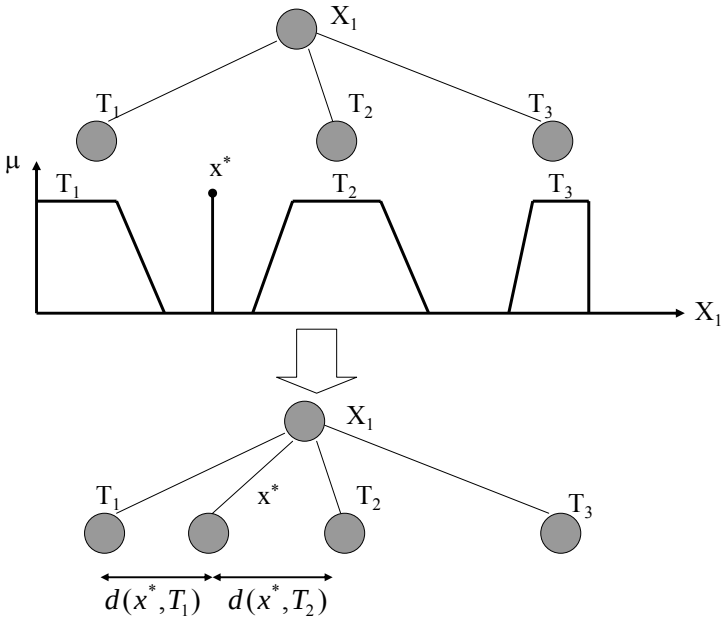
Fundamental equation of linear interpolation and its solution for B*

$$R_1 = A_1 \to B_2, \qquad R_2 = A_2 \to B_2,$$

where $A_1 \prec A^* \prec A_2$, $B_1 \prec B_2$

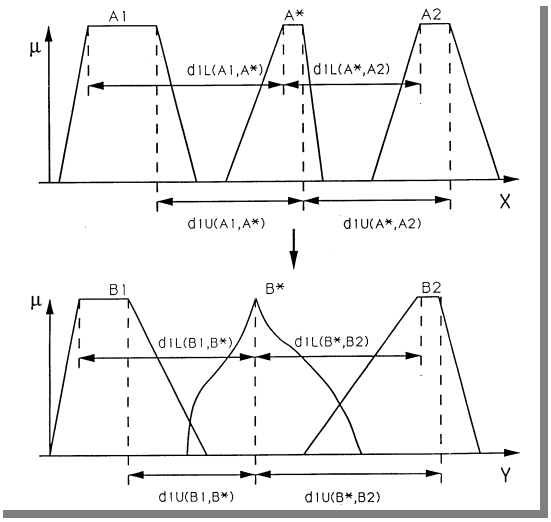
$$D(A^*, A_1) : D(A^*, A_2) = D(B^*, B_1) : D(B^*, B_2)$$

$$\inf\{B_\alpha^*\} = \frac{\frac{\inf\{B_{1\alpha}\}}{d_{\alpha L}(A_{1\alpha}, A_\alpha^*)} + \frac{\inf\{B_{2\alpha}\}}{d_{\alpha L}(A_{2\alpha}, A_\alpha^*)}}{\frac{1}{d_{\alpha L}(A_{1\alpha}, A_\alpha^*)} + \frac{1}{d_{\alpha L}(A_{2\alpha}, A_\alpha^*)}} \qquad \sup\{B_\alpha^*\} = \frac{\frac{\sup\{B_{1\alpha}\}}{d_{\alpha U}(A_{1\alpha}, A_\alpha^*)} + \frac{\sup\{B_{2\alpha}\}}{d_{\alpha U}(A_{2\alpha}, A_\alpha^*)}}{\frac{1}{d_{\alpha U}(A_{1\alpha}, A_\alpha^*)} + \frac{1}{d_{\alpha U}(A_{2\alpha}, A_\alpha^*)}}$$

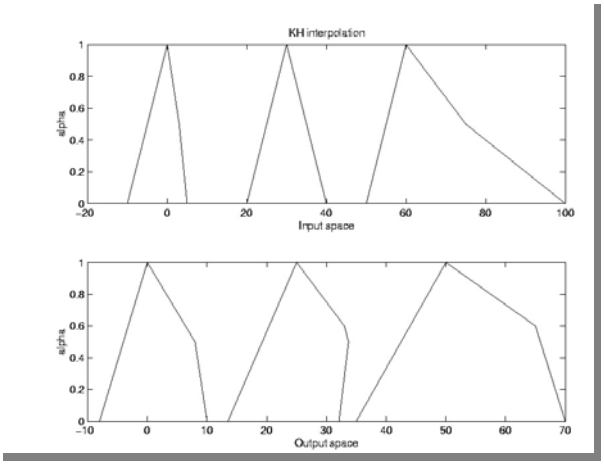


Fundamental Equation of Fuzzy Interpolation

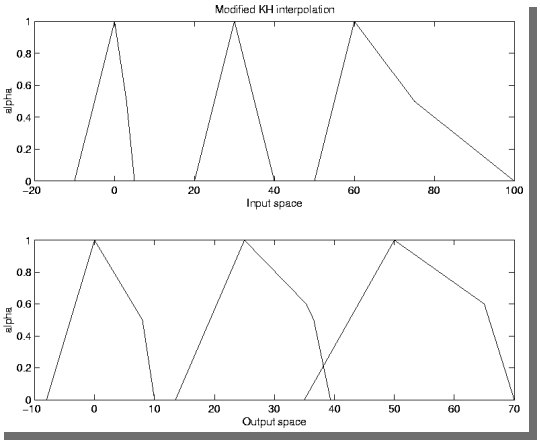
The result for the linear interpolation method



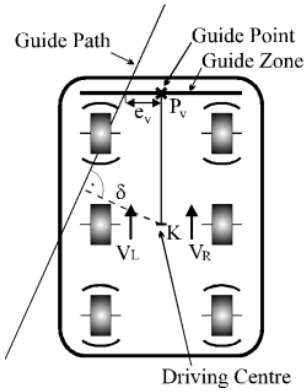
An example for the case when the result for the linear interpolation method is not a fuzzy set



Modified interpolation method for the example above



Path tracking of a guide path controlled automated guided vehicle (AGV)



e_v: the distance between the guide path and the guide point
δ: the estimated momentary path tracking error

V_L is the contour speed of the left wheel
V_R is the contour speed of the right wheel

Speed: $V_a=(V_L+V_R)/2$
Steering: $V_d=V_L-V_R$

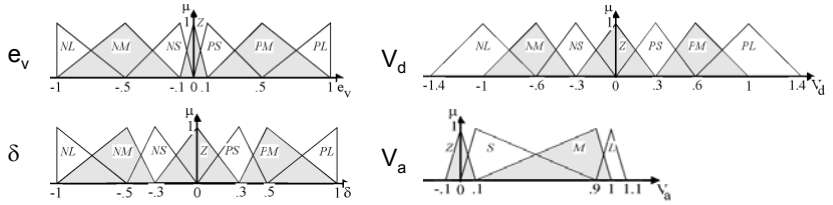
Rules describing the steering (V_d):
If $e_v = A_{1,i}$ **and** $\delta = A_{2,i}$ **then** $V_d = B_i$

Rules describing the speed (V_a):
If $e_v = A_{1,i}$ **and** $\delta = A_{2,i}$ **then** $V_d = B_i$

An example rule:
If the distance between the guide path and the guide point (e_v) is *Positive Middle* **and** estimated path tracking error (δ) is *Negative Middle* **then** the steering (V_d) is *Zero* **and** the speed (V_a) is *Middle*

Path tracking of a guide path controlled automated guided vehicle (AGV) (cont.)

- Fuzzy partitions for the linguistic variables:



- Rule bases:

V_d

δ / e_v	NL	NM	Z	PM	PL
NL				NL	
NM	PL		PS	PS	NL
Z		PL		NL	
PM	PL	NS	NS		NL
PL		PL			

V_a

δ / e_v	NL	NM	Z	PM	PL
NL					Z
NM					
Z	S		L		S
PM					
PL	Z				

2. Hierarchical fuzzy rule bases

The Curse of Dimensionality in Fuzzy Control

If there are k input state variables, and in each there are (max) T terms, the number of rules covering the space densely is

$$r = T^k$$

How to decrease this expression?

1. Decrease T
   Sparse rule bases, rule interpolation (Kóczy and Hirota, 1990)
2. Decrease k
   Hierarchical structured rule bases (Sugeno, 1991)
3. Decrease both T and k
   Interpolation of hierarchical rule bases (Kóczy and Hirota, 1993)

Decreasing k effectively

”Divide and conquer” algorithms/ Sugeno’s helicopter

Hierarchically structured rule bases with locally reduced variable sets

State space**:** $X = X_1 \times X_2 \times \ldots \times X_k$

Partitioned subspace: $Z_0 = X_1 \times X_2 \times \ldots \times X_{k_0}, k_0 < k$

In Z_0:

$$\prod = \{D_1, D_2, \ldots, D_n\}, \quad \bigcup_{i=1}^{n} D_i = Z_0$$

In each D_i a sub-rule base R_i is defined, reduction works if in each sub-rule base the input variable space X_i is a sub-space of

$$X / Z_0 = X_{k_0+1} \times X_{k_0+2} \times \ldots \times X_k$$

Decreasing *k* effectively (cont'd)

Pessimistic case:

$$\forall i : X_i = X / Z_0$$

Similarly bad case:

$$\exists i : X_i = X / Z_0$$

(In the latter case there is at least one domain where the total number of variables to be taken into consideration is *k*)

Hierarchical fuzzy rule bases

In these cases complexity is still $O(T^k)$, as the size of R_0 is $O(T^{k_1})$ and each R_i, $i > 0$, is of order $O(T^{k-k_1})$, so $O(T^{k_1}) \times O(T^{k-k_1}) = O(T^k)$

In some concrete applications:

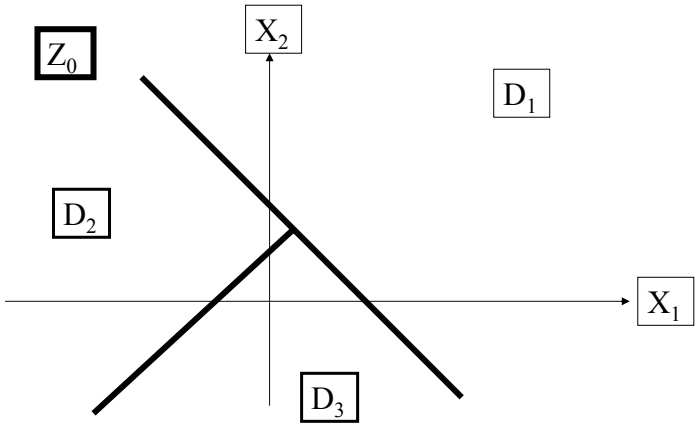
In each D_i a proper subset of $\{X_{k_1+1}, \ldots, X_k\}$ can be found so that each R_i contains only $k_i < k - k_0$ input variables $\Rightarrow$ if

$$\max_{i=1}^{n} \{k_i\} = K < k - k_0,$$

then the resulting complexity will be, $O(T^{k_0+K}) < O(T^k)$

Often it is impossible to find Π so that $k_i < k - k_0$, $i=1,\ldots,n$ because such a partition does not exist.

Partitioning Z_0



$$\prod = \{D_1, D_2, D_3\}$$

Reducing hierarchical rule base

$$(\neg\exists i : X_i = X / Z_0)$$

$R_0(z_0 \in Z_0)$: **If** z_0 **is** D_1 **then use** R_1

If z_0 **is** D_2 **then use** R_2

…

If z_0 **is** D_n **then use** R_n

$R_1(z_1' \in Z_1 \mid X / Z_0)$:

If z_1' **is** A_{11} **then** y **is** B_{11}

If z_1' **is** A_{12} **then** y **is** B_{12}

…

If z_1' **is** A_{1m_1} **then** y **is** B_{1m_1}

Reducing hierarchical rule base (cont'd)

$$(\neg\exists i : X_i = X / Z_0)$$

$R_2(z_2' \in Z_2 \mid X / Z_0)$: **If** z_2' **is** A_{21} **then** y **is** B_{21}

If z_2' **is** A_{22} **then** y **is** B_{22}

…

If z_2' **is** A_{2m_2} **then** y **is** B_{2m_2}

… … …

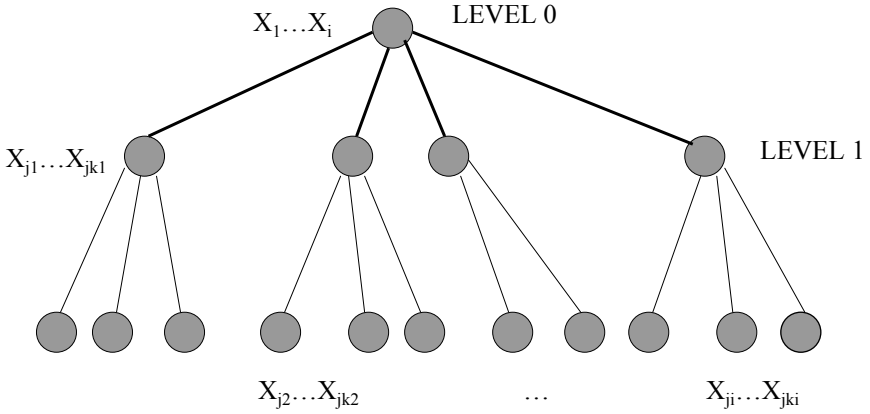
$R_n(z_n' \in Z_n \mid X / Z_0)$: **If** z_n' **is** A_{n1} **then** y **is** B_{n1}

If z_n' **is** A_{n2} **then** y **is** B_{n2}

…

If z_n' **is** A_{nm_n} **then** y **is** B_{nm_n}

HIERARCHICAL DECISION

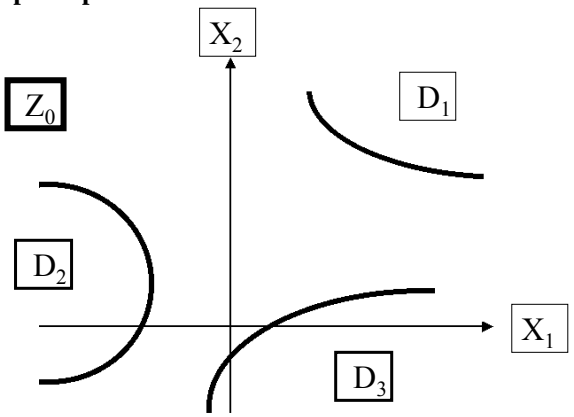


$j_1 \ldots j_{k1}, j_2 \ldots j_{k2}, \ldots, j_i \ldots j_{ki} \in \{k_0 + 1 \ldots k\}$

PROBLEM(i)

Partition Π usually does not exist because it is not possible to separate the areas of influence for the subsets of variables

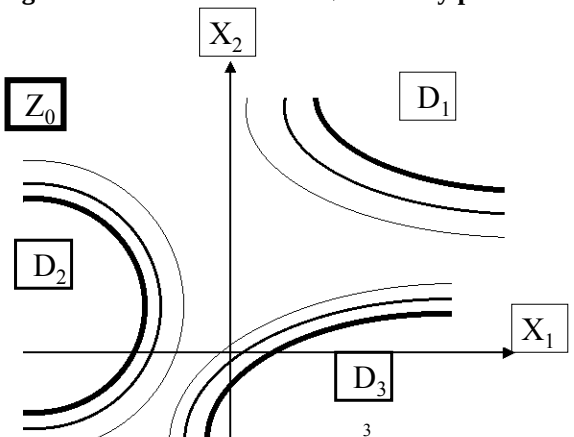
⇨ **"Sparse partition" Θ**



$$\Theta = \{D_1, D_2, D_3\},\ Z_0 - \Theta \neq \varnothing$$

PROBLEM(iia)

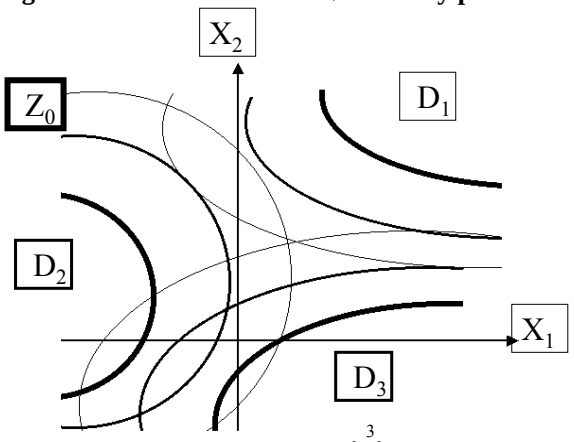
The elements of partition Θ are usually not crisp sets as the influence of each variable subset is fading away gradually when getting farther from the core ⇨ "Fuzzy partition" Θ



$$\Theta = \{D_1, D_2, D_3\},\ Z_0 - \bigcup_{i=1}^{3} \text{supp}\ (D_i) \neq \varnothing$$

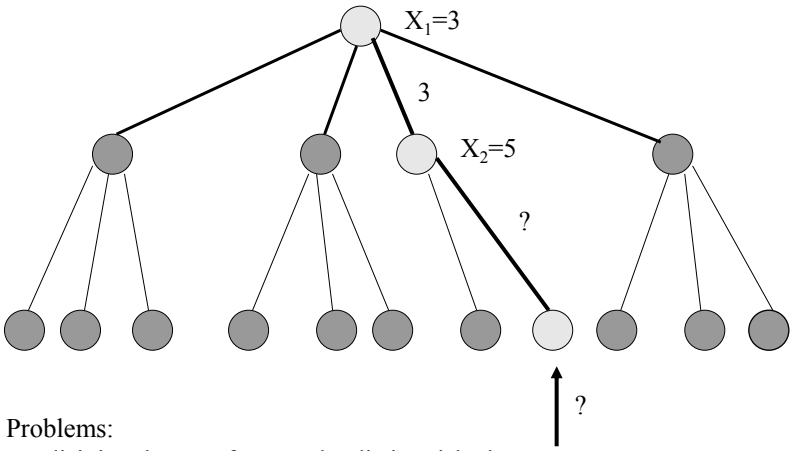
PROBLEM(ii)

The elements of partition Θ are usually not crisp sets as the influence of each variable subset is fading away gradually when getting farther from the core ⟹ "Fuzzy partition" Θ



$$\Theta = \{D_1, D_2, D_3\},\ Z_0 - \bigcup_{i=1}^{3} \text{core}\ (D_i) \neq \varnothing$$

E.G.:



Problems:
No disjoint classes of X_1 can be distinguished
Fuzzy values for X_1 are known
How to resolve this?

3. Interpolation of hierarchical rule bases

The Curse of Dimensionality in Fuzzy Control

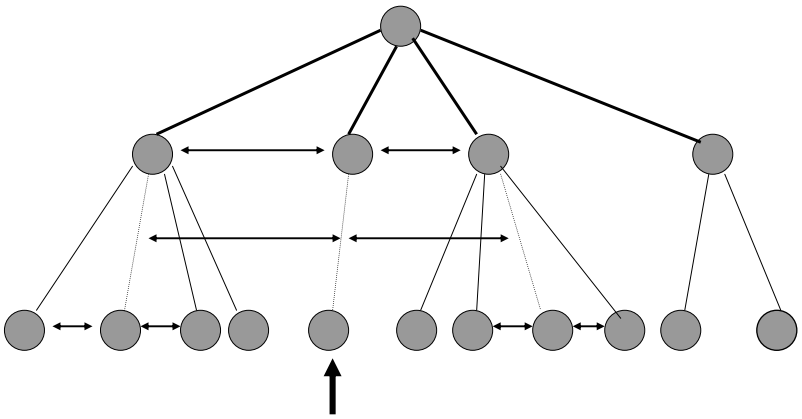
If there are k input state variables, and in each there are (max) T terms, the number of rules covering the space densely is

$$r = T^k$$

How to decrease this expression?

1. Decrease T
   Sparse rule bases, rule interpolation (Kóczy and Hirota, 1990)
2. Decrease k
   Hierarchical structured rule bases (Sugeno, 1991)
3. Decrease both T and k
   Interpolation of hierarchical rule bases (Kóczy and Hirota, 1993)

Graph of hierarchical interpolation



Interpolation of rules + rule bases

Possibly different subsets of variables

Interpolation in hierarchical fuzzy rule bases

Algorithm:

1.Determine the projection A_0^* of the observation A^* to the subspace of the fuzzy partition $\hat{\Pi}$. Find the flanking elements in $\hat{\Pi}$.

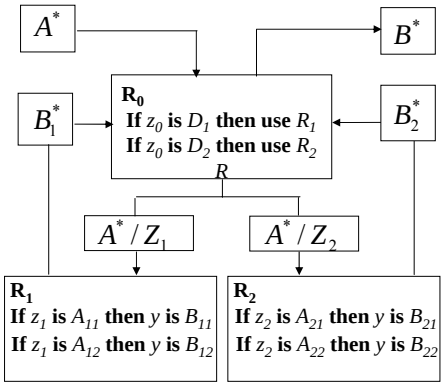
2.Determine the degrees of similarity (reciprocal distances or dissimilarities) for each D_i in $\hat{\Pi}$. These will be w_i.

3.For each R_i, where $w_i \neq 0$, determine A^*_i the projection of A^* to Z_i. Find the flanking elements in each R_i.

4. Determine the sub-conclusions B^* for each sub-rule base R_i.

Replace the sub-rule bases by the sub-conclusions in the meta-rule base R_0 and calculate the final conclusion B^* by applying w_i as degrees of similarity.

Example for interpolation in a hierarchical model

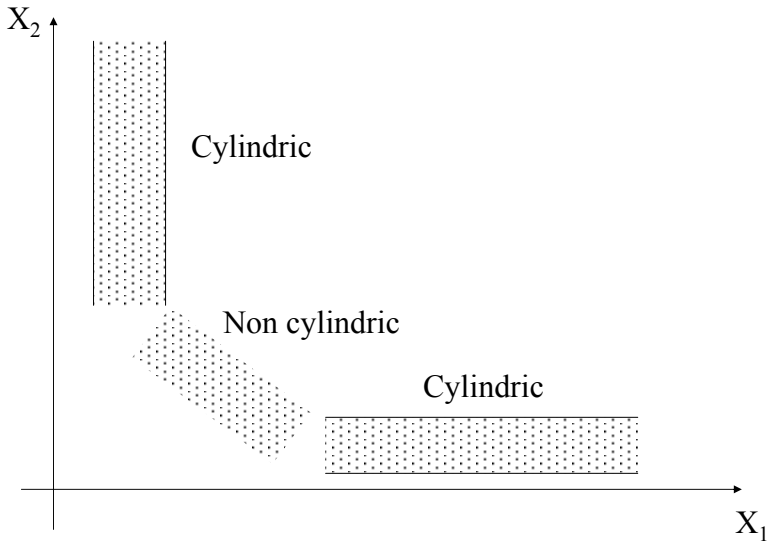


Conclusion

It is possible to apply interpolation in fuzzy rule bases with meta-levels, even if the meta-rule base is sparse.

The proposed method is similar to the approach of Mamdani, compared to the original CRI method of Zadeh, in the sense that every calculation is restricted to the projections of the observation, etc., reducing the computational complexity this way.

How sparse partitions of a subspace of the whole state space can be obtained ?



Model Identification

Motivation

- How can an optimal or at least a quasi-optimal fuzzy rule base for a certain system be found if there is no human expert and the linguistic description of the system is also not available?
- Evolutionary algorithms might be a possible alternative answer
- Clustering algorithms can also be used for rule extraction
- Traditional training algorithms based on first, or second order derivatives of the model's output (possibly applied to Neural Networks) offer a third answer
- Each of these techniques has advantages and disadvantages
- Combining two or more of them might lead to essentially better results

4. Model identification by clustering (flat systems)

Fuzzy Clustering

- Fuzzy rule extraction from data attracts significant amount of interest
- Most of the non-neural approaches rely on fuzzy clustering
- Fuzzy c-Means clustering: predominant
- User Input: Number of clusters

$$J_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}(U_{ik})^m \| x_k - v_i \|^2,\ 1 \le m \le \infty \qquad (1)$$

Where:
J_m(U,V) - SSE for the set of fuzzy clusters represented by the membership matrix U, and the set of cluster centers V
$||x_k - v_i||^2$ - distance between the data x_k and the cluster center v_i

Fuzzy Clustering (contd.)

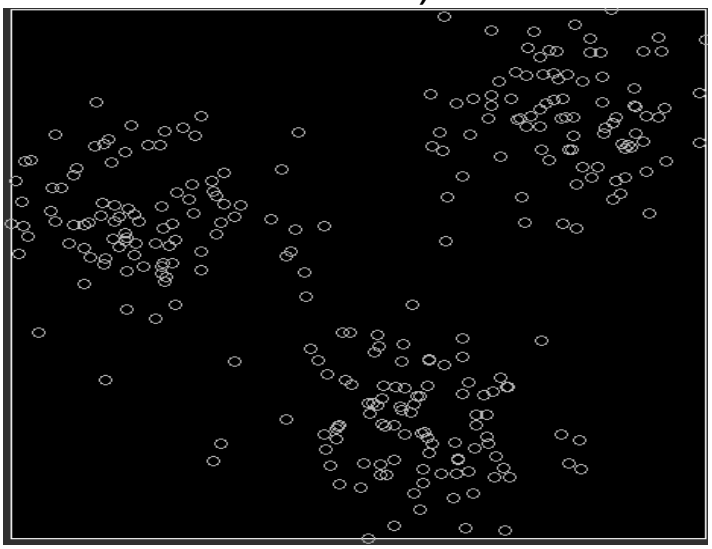
- The necessary conditions for (1) to reach its minimum are:

$$U_{ik} = \left(\sum_{j=1}^{c} \left(\frac{\| x_k - v_i \|}{\| x_k - v_j \|} \right)^{2/(m-1)} \right)^{-1} \qquad \forall i, \forall k \qquad (2)$$

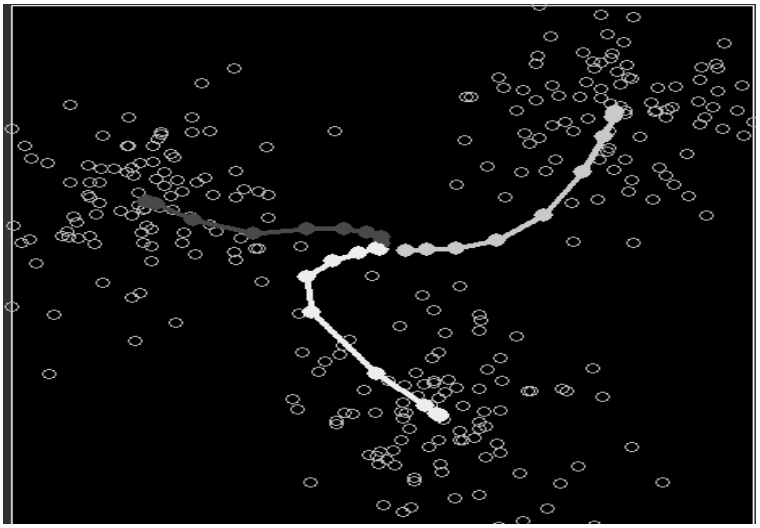
$$v_i = \frac{\sum_{k=1}^{n} (U_{ik})^m x_k}{\sum_{k=1}^{n} (U_{ik})^m} \qquad (3)$$

  - The FCMC algorithm iteratively adjusts a set of cluster centers to minimize (1)

Fuzzy Clustering („good" clusters for FCMC)



Fuzzy Clustering (evolution of cluster centers)



Cluster Validity Problem

- Problem of finding the optimal number of clusters (often by means of a criterion)
  - ♦ Fukuyama and Sugeno Criterion:

$$S(c)=\sum_{k=1}^{n}\sum_{i=1}^{c}(U_{ik})^m(\|x_k-v_i\|^2-\|v_i-\bar{x}\|^2)\quad 2<c<n$$

  - The number of clusters c is determined so that $S(c)$ reaches a local minimum as c increases.

Cluster Validity (contd.)

- Problem 1: Unable to tell when data has only one cluster (e.g. in given projection)
- Problem 2: When clusters are close to one another, $||V_i-X||$ is always small and the criterion falls apart.

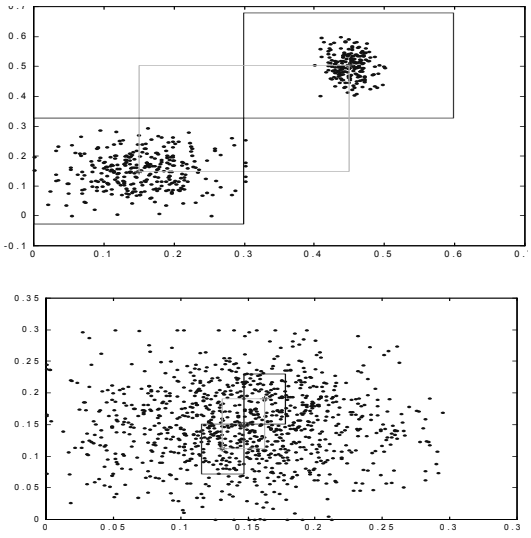
♦ We propose the following criterion for merging

$$P(v)=\sum_{j=1}^{n} e^{-4\left\|(v-x_j)/\frac{(v_i-v_j)}{2}\right\|^2} \quad (5)$$

♦ v_i and v_j should be merged if for new center point

$v_m = (v_i+v_j)/2$, $P(v_m)>v_i$ *or* $P(v_m)>v_j$,

Cluster Validity („good" and „bad" clusters)



5. MI by clustering (hierarchical systems)

Clustering and Feature Selection

- Suppose that input X is clustered into clusters C_i (i = 1, …, N_c) and $U = \{\mu_{ik} \mid i = 1\ldots N_c, k = 1\ldots N\}$ is the fuzzy partition matrix obtained.
- Feature ranking based on the interclass separability is formulated by means of the following fuzzy between-class and within-class scatter (covariance) matrices.

$$Q_b = \sum_{i=1}^{N_c} \sum_{j=1}^{N} \mu_{ij}^m (v_i - \bar{v})(v_i - \bar{v})^T$$

$$Q_w = \sum_{i=1}^{N_c} Q_i$$

$$Q_i = \frac{1}{\sum_{j=1}^{N} \mu_{ij}^m} \sum_{j=1}^{N} \mu_{ij}^m (x_j - v_i)(x_j - v_i)^T$$

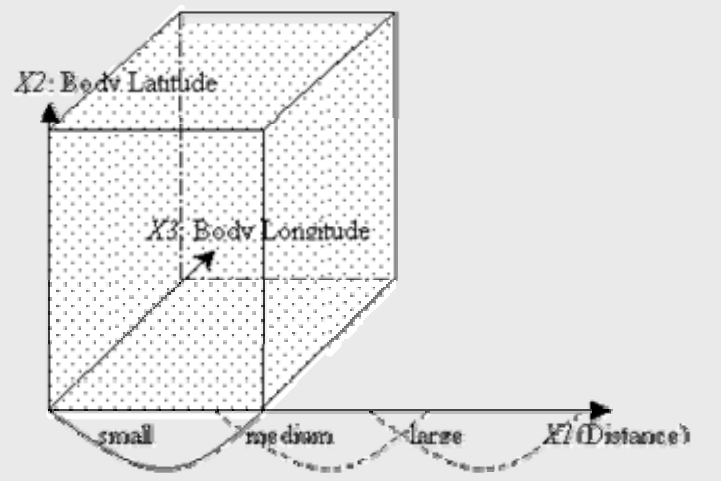
$$\bar{v} = \frac{1}{N_c} \sum_{i=1}^{N_c} v_i$$

Problem of Finding Z_0

Consider the following helicopter control fuzzy rules extracted from the hierarchical fuzzy system:

If *distance* (from obstacle) is *small* then *Hover*
Hover: if (helicopter) *body* rolls *right* then
move *lateral* stick *leftward*
if (helicopter) *body* pitches *forward* then
move *longitudinal* stick *backward*

Problem of Finding Z_0



Geometric representation data in a hierarchical fuzzy system

Problem of Finding Z_0

- Three clusters can be clearly observed in x_1.
- No cluster structure in x_2 and x_3 - data points are evenly distributed across the domains.
- On one hand, we need to rely on the feature selection technique to find a 'good' subspace (Z_0) for the clustering algorithm to be effective
- On the other hand, most of the feature selection criteria assume the existence of the clusters beforehand.

Algorithm of Finding Z_0

1. Rank each input dimension by its cylindricity and let F be the set of features ordered (ascending) by their importance.
2. For i = 1 … $|F|$
   a) Construct a hierarchical fuzzy system (algorithm described in Section 3) using the subspace $Z_0 = X_1$ x … x X_i.
   b) Compute ε_i to be the re-substitution error of the fuzzy system constructed during the i^{th} iteration.
   c) If i > 1 and $\varepsilon_i > \varepsilon_{i-1}$, stop

   end for

Algorithm of Hierarchical Clustering

1. Perform fuzzy c-means clustering on the data along the subspace Z_0. The optimal number of clusters, C, within the set of data is determined by means of the FS index
2. The previous step results in a fuzzy partition $\Pi = \{D_1, \ldots D_C\}$. For each component in the partition, a meta rule is formed as:

   If Z_0 is D_i then use R_i
3. From the fuzzy partition Π, a crisp partition of the data points is constructed. That is: for each fuzzy cluster D_i, the corresponding crisp cluster of points is determined as $P_i = \{p \mid \mu_i(p) > \mu_j(p) \ \forall \ j \neq i\}$.
4. Construct the sub-rule bases. For each crisp partition P_i, apply a feature extraction algorithm to eliminate unimportant features. The remaining features (known as true inputs) are then used by a fuzzy rule extraction algorithm to create the fuzzy rule base R_i.

Simulation Result

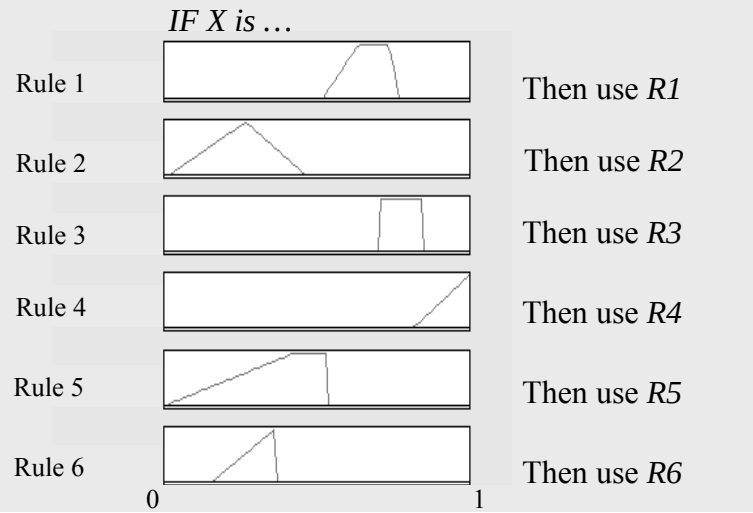
- Building porosity (PH) estimator for real world Petroleum data.
- The well logs available are: Depth, GR (Gamma Ray), RDEV (Deep Resistivity), RMEV (Shallow Resistivity), RXO (Flushed Zone Resistivity), RHOB (Bulk Density), NPHI (Neutron Porosity), PEF (Photoelectric Factor) and DT (Sonic Travel Time). Normalised data [0, 1] is used.
- Altogether 633 rows of data. The same set of data is used for training as well as testing.

Simulation Result (contd.)

- The proposed algorithm produces 6 meta-rules. The input variable 'Depth' was selected by the algorithm to form the subspace Z_0

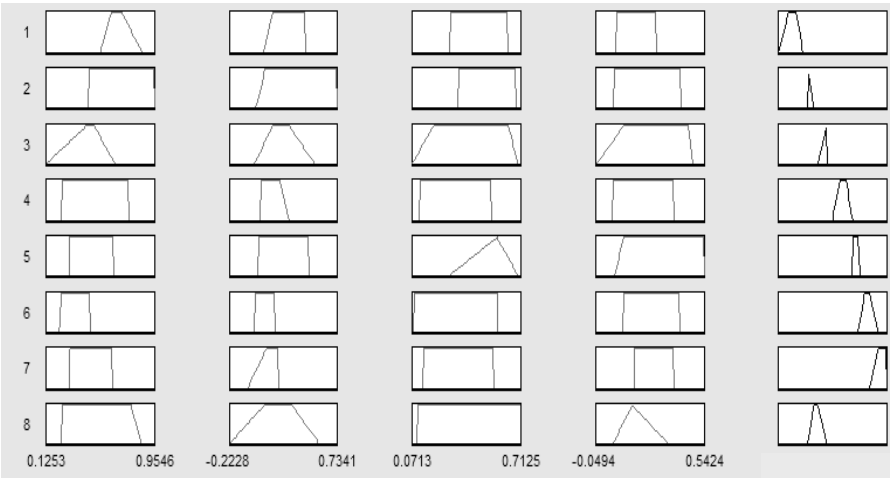
Sub-rulebase	Input variables	# rules
1	GR RMEV RXO PEF	8
2	GR	3
3	GR RXO NPHI	8
4	GR RDEV RMEV RXO	7
5	GR RXO RHOB	6
6	GR RXO	5

Simulation Result (contd.)



Meta rules generated

Simulation Result (contd.)



Sub-rulebase 1

Simulation Result (contd.)

- Error for Hierarchical Fuzzy: 0.0163
- For comparison purposes, a 'flat' fuzzy system was also constructed.
- The feature selection process selected the following true inputs: DEPTH, GR, RMEV, RXO, RHOB, and PEF.
- Altogether 11 fuzzy rules were generated
- Error for Flat Fuzzy : 0.0374
- The low error rate of the hierarchical version can be explained by the fact that it has more rules in total.
- Despite the larger number of rules, the hierarchical version can still be more efficient than the flat version since, unlike tradition fuzzy system, not all the rules are needed to be processed.
- Depending on the system input, many of the fuzzy rules in the sub-rule bases might not need to be processed.

Simulation Result (contd.)

- In terms of the modeling process, the flat fuzzy system has a higher complexity. Using 6 variables, the complexity is $O(T^6)$.
- For the hierarchical fuzzy system, we have $O(T^1) \times O(T^4) + O(T^1) \times O(T^1) + O(T^1) \times O(T^3) + O(T^1) \times O(T^4) + O(T^1) \times O(T^3) + O(T^1) \times O(T^2) = O(T^{1+4}) = O(T^5)$.
- Both versions use the same set of true inputs.
- The hierarchical modeling scheme, however, has successfully used the variable *Depth* to separate the problem domain into multiple sub-domains such that in each sub-domain, only a subset of the remaining variables plays a significant role in influencing the output
- This is the essential point how the complexity has been reduced.

6. MI by bacterial memetic algorithm

Outline of the section

- Motivation
- Background
  - Fuzzy encoding method
  - Bacterial Evolutionary Algorithms
  - The Levenberg-Marquardt method
- Bacterial Memetic Algorithms
- Simulation results
- Conclusions

Motivation

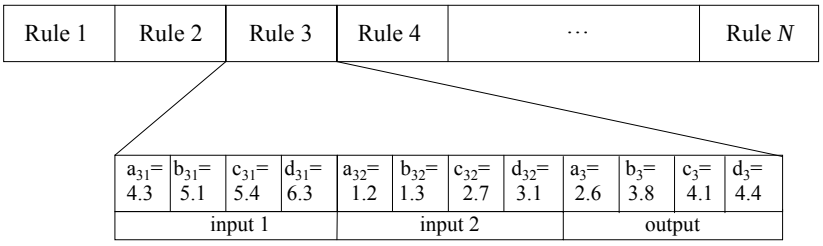
- How can an optimal or at least a quasi-optimal fuzzy rule base for a certain system be found if there is no human expert and the linguistic description of the system is also not available?
- Evolutionary algorithms might be a possible alternative answer
- Clustering algorithms can also be used for rule extraction
- Traditional training algorithms based on first, or second order derivatives of the model's output (possibly applied to Neural Networks) offer a third answer
- Each of these techniques has advantages and disadvantages
- Combining two or more of them might lead to essentially better results

6.1. Bacterial Evolutionary Algorithms

- Nature inspired optimization techniques
- Based on the process of microbial evolution
- Applicable for complex optimization problems
- Individual: one solution of the problem
- Intelligent search strategy to find a *sufficiently good* solution (quasi optimum)
- Faster convergence (conditionally)

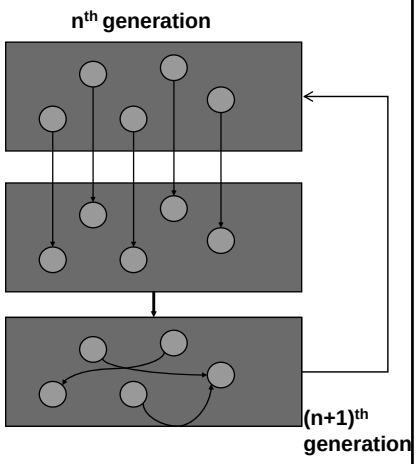
Fuzzy encoding method

- Fuzzy rule system with trapezoidal membership functions
- Trapezoids are described by the four breakpoints:
  - $A_{ij}(a_{ij},b_{ij},c_{ij},d_{ij})$: MF of the j^{th} input in the i^{th} rule
- Each fuzzy system is a bacterium
- Encoding a system with two inputs:

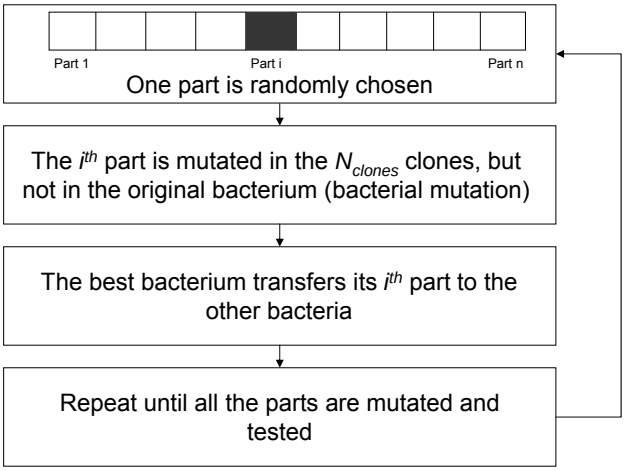


The algorithm

- Generating the initial population randomly
- Bacterial mutation is applied for each bacterium
- Gene transfer is applied in the population
- If a stopping condition is fullfilled then the algorithm stops, otherwise it continues with the bacterial mutation step



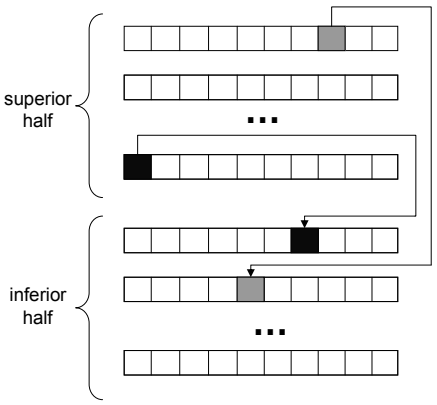
Bacterial mutation



Gene transfer

1. The population is divided into two halves
2. One bacterium is randomly chosen from the superior half (source bacterium) and another from the inferior half (destination bacterium)
3. A part from the source bacterium is chosen and this part can overwrite a part of the destination bacterium

   This cycle is repeated for N_{inf} times (number of "infections")



6.2. The Levenberg-Marquardt method

- Second-order gradient-based training method
- Belongs to the group of 'trust-region' or 'restricted-step' type methods
  - Attempts to define a neighborhood where the quadratic function model agrees with the actual function in some sense. The parameter α controls the radius of neighborhood
- Local search optimizer
- Our assumption: It can be used to improve the evolutionary algorithm, which may find the global optimum with higher precision in this way

Structure of the fuzzy system

Since four parameters describe each membership function, $A_{ij}(a_{ij}, b_{ij}, c_{ij}, d_{ij})$

The relative importance is given by: $\mu_{ij}(x_j) = \frac{x_j - a_{ij}}{b_{ij} - a_{ij}} N_{i,j,1}(x_j) + N_{i,j,2}(x_j) + \frac{d_{ij} - x_j}{d_{ij} - c_{ij}} N_{i,j,3}(x_j)$

where $a_{ij} \le b_{ij} \le c_{ij} \le d_{ij}$

Using the COG defuzzification method the output is: $y(\underline{x}) = \frac{\sum_{i=1}^{R} \int_{y \in \mathrm{supp}\mu_i(y)} y \mu_i(y) dy}{\sum_{i=1}^{R} \int_{y \in \mathrm{supp}\mu_i(y)} \mu_i(y) dy}$

Summarised:

$$y(\underline{x}) = \frac{1}{3} \frac{\sum_{i=1}^{R} (C_i + D_i + E_i)}{\sum_{i=1}^{R} 2w_i(d_i - a_i) + w_i^2(c_i + a_i - d_i - b_i)}$$

$$C_i = 3w_i(d_i^2 - a_i^2)(1 - w_i) \qquad (1)$$

$$D_i = 3w_i^2(c_i d_i - a_i b_i)$$

$$E_i = w_i^3(c_i - d_i + a_i - b_i)(c_i - d_i - a_i + b_i)$$

The Levenberg-Marquardt method in detail

The training algorithm:

- The structure of the fuzzy rule base is pre-determined
- Error minimization criterion: $\Omega = \frac{\left\| \underline{t} - \underline{y} \right\|^2}{2} = \frac{\left\| \underline{e}[k] \right\|^2}{2}$

  t = target o/p, y = model o/p
- Calculate the Jacobian matrix $\underline{\underline{J}}[k] = \left[\frac{\partial y(\underline{x}^{(p)})[k]}{\partial \underline{par}[k]} \right]$
- *par :* parameters of the membership functions (bacterium)
- *k*: iteration variable
- The LM update *s*[*k*] is given as the optimal solution of:

  $$(\underline{\underline{J}}^T[k]\underline{\underline{J}}[k] + \alpha \underline{\underline{I}})\underline{s}[k] = -\underline{\underline{J}}^T[k]\underline{e}[k]$$
- The new *par* can be obtained by *s*[*k*]: $\underline{par}[k+1] = \underline{par}[k] + \underline{s}[k]$

The Levenberg-Marquardt method in detail (contd.)

Jacobian computation:

- Computed on a pattern by pattern basis:

$$\underline{\underline{J}} = \left[\frac{\partial y(\underline{x}^{(p)})}{\partial a_{11}} \quad \frac{\partial y(\underline{x}^{(p)})}{\partial b_{11}} \cdots \frac{\partial y(\underline{x}^{(p)})}{\partial a_{12}} \cdots \frac{\partial y(\underline{x}^{(p)})}{\partial d_1} \cdots \frac{\partial y(\underline{x}^{(p)})}{\partial d_R} \right]$$

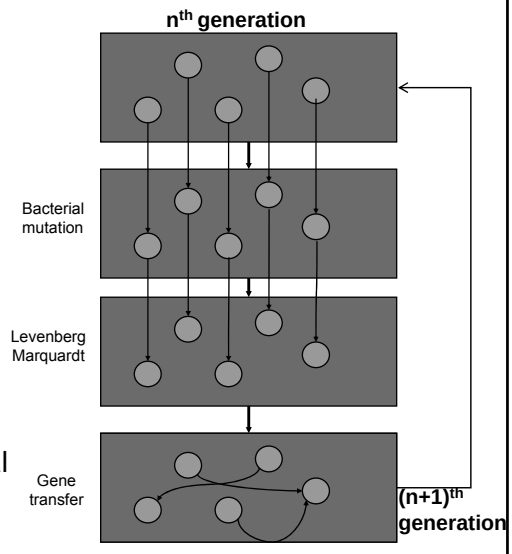
- Applying the chain rule derivatives
- The derivatives of the input and output membership functions as well as of w_i (activation degree of the i^{th} rule) need to be calculated

6.3. The Bacterial Memetic Algorithm

- Evolutionary and local search hybrid methods are usually referred as memetic algorithm
- A new memetic method is proposed: Bacterial Memetic Algorithm
- Idea: combine Bacterial Evolutionary Algorithm doing global search with Levenberg-Marquardt for local search

Bacterial Memetic Algorithm

- Generating the initial population randomly
- Bacterial mutation is applied for each bacterium
- Levenberg-Marquardt method is applied for each bacterium
- Gene transfer is applied in the population
- If a stopping condition is fullfilled then the algorithm stops, otherwise it continues with the bacterial mutation step



Parameters of the algorithm

- N_{gen}: number of generations
- N_{ind}: number of individuals
- N_{clones}: number of alternative clones in the bacterial mutation procedure
- N_{inf}: number of infections in the gene transfer
- N_{iter}: number of iterations in the LM step
- α: regularization parameter in the LM step

Simulation results

- Three examples have been used to show the performance of the algorithm
  - pH problem $pH = -\frac{\log\left(\sqrt{\frac{y^2}{4}+10^{-14}}-\frac{y}{2}\right)+6}{26}, y = 2\times10^{-13}x - 10^{-3}$
  - ICT problem $\theta_2 = \pm tg^{-1}\left(\frac{s}{c}\right) \quad c = \frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2} = \cos(\theta_2) \quad s = \sqrt{1-c^2} = sin(\theta_2)$
  - Six dimensional function $y = x_1 + x_2^{0.5} + x_3x_4 + 2e^{2(x_5-x_6)}$
- Termination criterion $\Omega[k-1]-\Omega[k] < \theta[k] \quad \theta[k] = \tau_f\left(1+\Omega[k]\right)$

$$\left\|\mathbf{par}[k-1]-\mathbf{par}[k]\right\| < \sqrt{\tau_f}\left(1+\left\|\mathbf{par}[k]\right\|\right)$$

$$\left\|\mathbf{g}[k]\right\| \le \sqrt[3]{\tau_f}\left(1+\left|\Omega[k]\right|\right)$$

$$\tau_f = 10^{-4}$$

Simulation results (contd.)

- Parameter values:
  - N_{gen}=20, N_{ind}=10, N_{clones}=8, N_{inf}=4, N_{iter}=10
  - Number of rules is 3
  - Number of training/validation patterns (each):
    - pH:101, ICT: 110, Six-dim.: 200
- Error criteria:

$$MSRE = \frac{1}{N\_pat} \sum_{i=1}^{N\_pat} \frac{(t_i - y_i)^2}{y_i^2}$$

$$MREP = \frac{100}{N\_pat} \sum_{i=1}^{N\_pat} \left| \frac{t_i - y_i}{y_i} \right|.$$

t_i: target output for the i^{th} pattern
y_i: model output for the i^{th} pattern

Simulation results (contd.)

Performance specifications for the Best MSE individuals (pH):

Algorithm	MSE
BEA	4.5e-003
BMA	1.5e-005

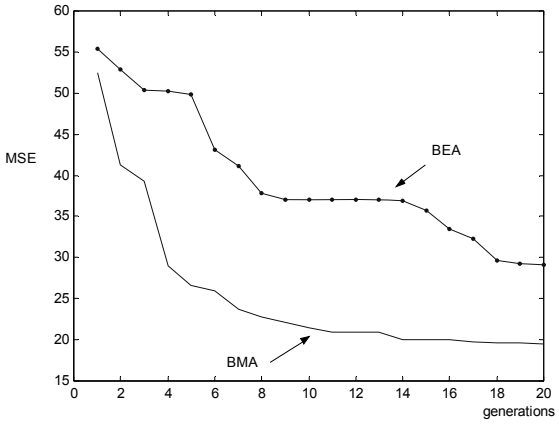
Performance specifications for the Best MSE individuals (ICT):

Algorithm	MSE
BEA	1.2
BMA	1.3e-001

Performance specifications for the Best MSE individuals (6 dimensional function):

Algorithm	MSE
BEA	4.5e+001
BMA	1.5e+001

Simulation results (contd.)



MSE evolution lines for the six dimensional problem

Simulation results (contd.)

The optimal rules obtained by the BEA for the ICT problem:

R_1: if x_1 is [-0.027427, 0.013643, 0.027407, 0.27058] and x_2 is [-0.072697, 0.1739, 0.43876, 0.88877] then y is [1.9286, 1.9644, 2.2663, 2.8419]
R_2: if x_1 is [0.023605, 0.82814, 0.83875, 0.91526] and x_2 is [0.039025, 0.37465, 0.76291, 0.94203] then y is [0.083984, 0.599, 0.83059, 1.6556]
R_3: if x_1 is [0.0055574, 0.14652, 0.57419, 0.57709] and x_2 is [-0.090017, 0.16984, 0.88549, 0.89452] then y is [1.5032, 2.4553, 2.6697, 3.2334]

The optimal rules obtained by the BMA for the ICT problem:

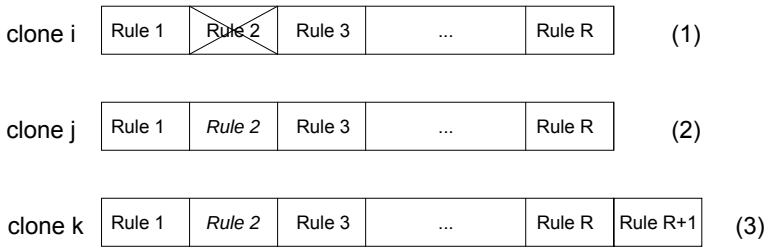
R_1: if x_1 is [-0.05557, 0.096831, 0.096831, 0.61764] and x_2 is [-0.38741, 0.15828, 0.42377, 0.75708] then y is [1.9286, 2.5199, 2.841, 3.6312]
R_2: if x_1 is [-0.13014, 0.49711, 0.79235, 0.99016] and x_2 is [0.30314, 0.72457, 0.76876, 1.114] then y is [0.01816, 0.38369, 1.0447, 1.3301]
R_3: if x_1 is [0.11687, 0.62462, 0.69643, 0.80899] and x_2 is [-0.083077, 0.22048, 0.24413, 0.70602] then y is [1.0631, 1.6966, 1.9513, 2.1227]

Optimal rule base size

- How can the size of the rule base also be optimized?
- Bacteria with different length (different number of rules) are allowed
- The bacterial operators should be improved in order to handle bacteria with different length
- The evaluation criterion must contain the length of the bacterium

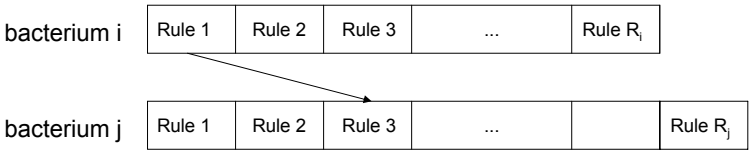
Improved bacterial mutation

- During the mutation of the i^{th} part in the clones, three different operations can happen
  - rule removal (1)
  - rule replacement (2)
  - rule addition (3)

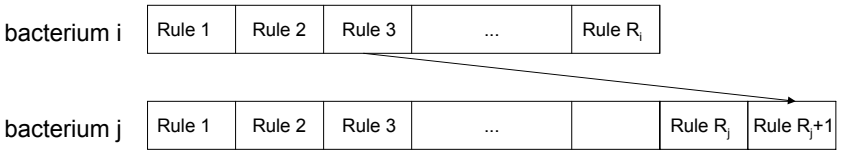


Improved gene transfer

Rule replacement operation



Rule addition operation



Simulation results

- Evaluation criterion:

  BIC = m*ln(MSE)+n*ln(m)

  m: number of patterns

  n: number of rules

  MSE: Mean Square Error

  $$MSE = \frac{1}{m}\sum_{i=1}^{m}(t_i - y_i)^2$$

  t_i: target output for the i^{th} pattern

  y_i: model output for the i^{th} pattern

- 10 sessions of 20 generations were executed
- population size: 10, number of clones: 8, number of infections: 4. Levenberg-Marquardt step runs for 10 iterations in every generation and the maximum allowed bacterium length is 10

Mean Values

Mean values using BEA and BMA

Specification	pH		ICT		Six-dimensional	
	BEA	**BMA**	**BEA**	**BMA**	**BEA**	**BMA**
MSE	6.28×10^{-3}	2.3×10^{-6}	8.69×10^{-1}	3.67×10^{-2}	3.21	1.29
MSRE	4.06×10^{4}	1.02×10^{1}	5.63×10^{13}	1.80×10^{13}	6.20×10^{-2}	3.07×10^{-2}
MREP	2.05×10^{3}	2.69×10^{1}	1.77×10^{8}	1.08×10^{8}	1.86×10^{1}	1.21×10^{1}
MSEv	7.76×10^{-3}	2.08×10^{-6}	1.30	1.12×10^{-2}	4.29	2.22
MSREv	1.63×10^{5}	4.18×10^{1}	1.92×10^{-1}	1.62×10^{-3}	6.50×10^{-2}	3.37×10^{-2}
MREPv	4.12×10^{3}	5.46×10^{1}	2.56×10^{1}	3.04	1.91×10^{1}	1.32×10^{1}
#rules	4.90	7.40	2.20	5.00	6.50	7.30

Best Values

Best values using BEA and BMA

Specification	pH		ICT		Six-dimensional	
	BEA	**BMA**	**BEA**	**BMA**	**BEA**	**BMA**
MSE	2.73×10^{-3}	4.7×10^{-7}	8.42×10^{-1}	1.04×10^{-2}	2.07	4.10×10^{-1}
MSRE	3.71×10^{4}	3.63	5.32×10^{13}	7.09×10^{12}	4.78×10^{-2}	9.56×10^{-3}
MREP	1.97×10^{3}	1.92×10^{1}	2.03×10^{8}	6.39×10^{7}	1.59×10^{1}	7.06
MSEv	5.12×10^{-3}	6.07×10^{-7}	1.26	2.6×10^{-3}	2.66	9.9×10^{-1}
MSREv	1.48×10^{5}	1.53×10^{1}	1.87×10^{-1}	3.74×10^{-4}	4.28×10^{-2}	1.5×10^{-2}
MREPv	3.96×10^{3}	3.93×10^{1}	2.34×10^{1}	1.48	1.47×10^{1}	9.28
#rules	9	8	2	5	7	10

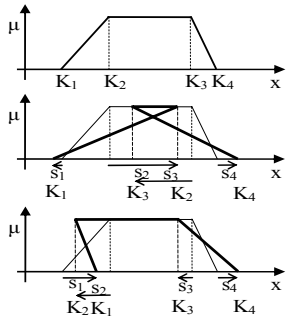
Knot order violation handling methods

- In BMA for fuzzy rule base identification the LM method has to be modified
- When applying update vector *s* to parameter vector *p* that contains the breakpoints of a trapezoidal shaped fuzzy rule, some knots of trapezoid can be randomly swapped
- Abnormal trapezoid is unacceptable as a component of a fuzzy rule
- The proper sequence of knots must be maintained
- In BMA correction is made by an update vector reduction factor when KOV is detected

$$g = \frac{K_{i+1}[k-1] - K_i[k-1]}{2\left(s_i[k] - s_{i+1}[k]\right)},\ 0 < g < 1$$

$$K'_i[k] := K_i[k-1] + g \cdot s_i[k]$$

$$K'_{i+1}[k] := K_{i+1}[k-1] + g \cdot s_{i+1}[k]$$

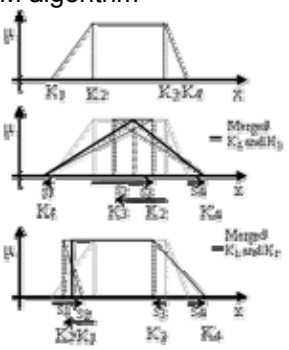


Knot order violation handling method Merge

- In case of KOV the rate of shift of both of the two breakpoints that have been computed by the LM method has to be applied as much as it can be done as far as the violation of the order is not yet occurring
  - Can be applied *after* the update part of LM algorithm
- ***Merge*** the two violating knots into a single one being half-way between the original knots (it results into a triangle or a trapezoid with a vertical edge)

$$K'_i := K'_{i+1} := \left(K_i + K_{i+1}\right)/2$$

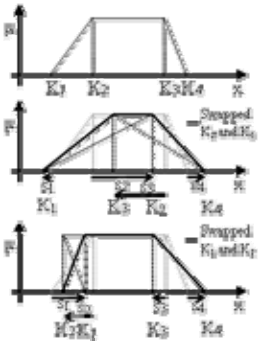
- Possible formation of vertical edges
- Leads to 0-s appearing in Jacobian matrix
- This could decrease speed of convergence of further iterations



Knot order violation handling method Swap

- In case of KOV the rate of shift of both of the two breakpoints that have been computed by the LM method has to be applied as much as it can be done
  - However without formation of trapezoids with vertical edges
  - Can be applied *after* the update part of LM algorithm

- ***Swap*** the two violating knots
  - Formation of abnormal trapezoids or trapezoids with vertical edges can always be avoided
- Swap (K_i, K_{i+1}) $K'_i := K_{i+1}$, $K'_{i+1} := K_i$
- Attempts to preserve the location of the new point generated by the update as much as it is possible without a violation of the sequence
- Avoids the formation of vertical edges



Simulations

- Three different transcendental functions
  - 100-200 patterns (10, 20, 30 runs)

- 1 input variable function
  $N_{Patterns}$ = 100, N_{Ind} = 7,
  N_{Fuzzy_rules} = 3, N_{Clones} = 7, N_{Inf} = 3

$$y = f_1(x) = \sin(x) \cdot e^{\cos(2.7 \cdot x)}$$
$$x \in [0...3\pi]$$

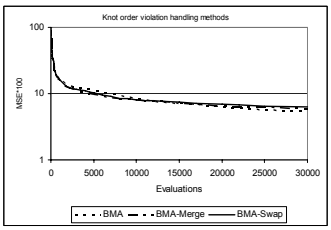
- 2 input variables function
  $N_{Patterns}$ = 200, N_{Ind} = 10,
  N_{Fuzzy_rules} = 3, N_{Clones} = 4, N_{Inf} = 10

$$y = f_2(\underline{x}) = \sin^5(0.5 \cdot x_1) \cdot \cos(0.7 \cdot x_2)$$
$$x_1 \in [0...3\pi],\ x_2 \in [0...3\pi]$$

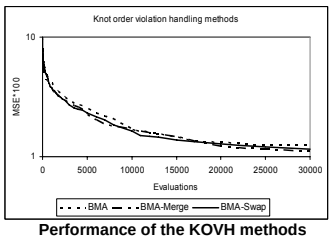
- 6 input variables function
  $N_{Patterns}$ = 200, N_{Ind} = 10,
  N_{Fuzzy_rules} = 5, N_{Clones} = 4, N_{Inf} = 10

$$y = f_3(\underline{x}) = x_1 + x_2^{0.5} + x_3 \cdot x_4 + 2 \cdot e^{2 \cdot (x_5 - x_6)}$$
$$x_1 \in [1...5],\ x_2 \in [1...5],\ x_3 \in [0...4],$$
$$x_4 \in [0...0.6],\ x_5 \in [0...1],\ x_6 \in [0...1.2]$$

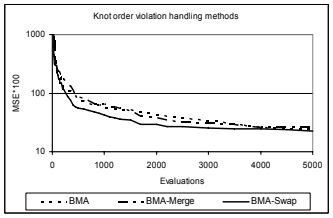
Simulation results
Merge and Swap



Performance of the KOVH methods
1 input variable, 3 fuzzy rules, 30 runs avg.



Performance of the KOVH methods
2 input variables, 3 fuzzy rules, 20 runs avg.



Performance of the KOVH methods
6 input variables, 5 fuzzy rules, 10 runs avg.

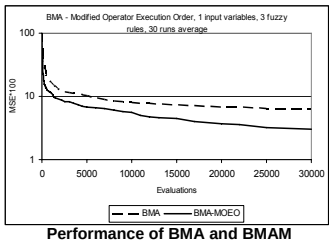
BMA with Modified Operator Execution Order (BMAM)

- BMA integrates BEA and LM as follows:
  1. *Bacterial Mutation operation* for each individual
  2. *Levenberg-Marquardt* method for each individual
  3. *Gene Transfer operation* for a partial population
- LM method is nested into BEA
- Local search is done for every global search cycle
- BMAM uses the *Levenberg-Marquardt* method more efficiently
  - Instead of applying the LM cycle after the *bacterial mutation* as a separate step, we propose the execution of several LM cycles during the *bacterial mutation* after *each mutational step*
- Simulations confirmed improved performance

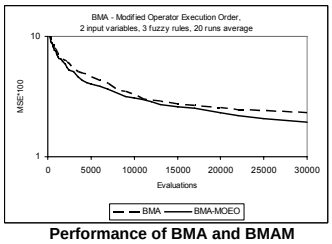
BMA with Modified Operator Execution Order 2

- In the mutational cycle it is possible to gain a rule base that has an instantaneous fitness value that is worse than the one in the previous or the concurrent rule bases
- Potentially better than those, it is located in such a region of search space which has a better local optimum
- LM iterations after each bacterial mutational step, test step is able to choose potentially valued clones that could be lost otherwise
- After *each mutational step* of *every single bacterial mutation iteration* several LM iterations are done (3-5)
- It is enough to run just 3 to 5 of LM iterations per mutation to improve the performance of the whole algorithm
- *"BMA with the modified operator execution order" (BMAM)*
- Works with the same operators like the original BMA, but it uses them in another manner and in another order

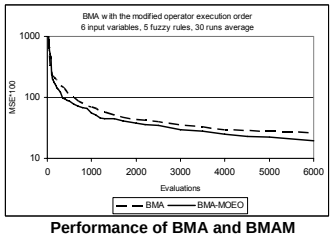
Simulation results
BMA with the Modified Operator Execution Order



Performance of BMA and BMAM
1 input variables, 3 fuzzy rules, 30 runs avg.



Performance of BMA and BMAM
2 input variables, 3 fuzzy rules, 20 runs avg.



Performance of BMA and BMAM
6 input variables, 5 fuzzy rules, 30 runs avg.

Conclusions 1

- Three novel methods were presented to improve the Bacterial Memetic Algorithm used for fuzzy rule base extraction
- First two methods apply knot order violation handling
  - They offer drop-in replacements for the method used in the original BMA
  - They are simpler and easier to be integrated
  - They can improve the performance of the BMA up to 50 percent in the simulated cases (reduced SSE by 34 percent, *swap*)
  - We recommend using the *swap* method as a simple and powerful replacement especially in case of more complex fuzzy rule base

Conclusions 2

- The third method is a structural modification of the BMA in which the order of the operators is modified
  - This method can improve the performance of the BMA up to 100 percent in the simulated cases (reduced SSE by 50 percent)
  - It performed better in case of less complex fuzzy rule base
- We expected that the combination of the *swap* method and the *BMA with the modified operator execution order* will be beneficial
  - The first one improves the performance more in the complex case
  - The other performs better in the less complex case

Modified Bacterial Memetic Algorithm (MBMA)

- Combining the KOVHM Swap and BMAM
  - Benefits of both methods can be utilized
  - The modified algorithm:

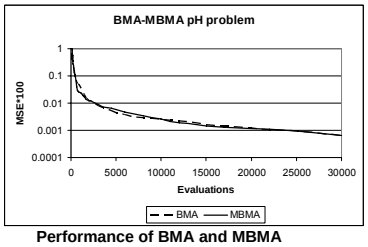
- Create the initial population: N_{Ind} individuals are randomly created and evaluated.
- Apply the ***Modified Bacterial Mutation*** *operation* for each individual:
  - Each individual is selected one by one
  - N_{Clones} copies of the selected individual are created ("clones")
  - Choose the same part or parts randomly from the clones and mutate it (except one single clone that remains unchanged during this mutation cycle)
  - ***Run some Levenberg-Marquardt iterations (3–5)***
    - ***Use method Swap for handling the knot order violations after each LM update***
  - ***Select the best clone and transfer its all parts to the other clones***
  - Repeat the part choosing-mutation-*LM*-selection-transfer cycle until all the parts are mutated, *improved* and tested
  - The best individual is remaining in the population, all other clones are deleted
  - This process is repeated until all the individuals have gone through the ***modified bacterial mutation***
- Apply the *Levenberg-Marquardt* method to each individual (e.g. 10 iterations per individual per generation)
  - ***Use method Swap for handling the knot order violations after each LM update***
- Apply the *gene transfer operation* N_{Inf} times per generation:
  - Sort the population according to the fitness values and divide it in two halves
  - Choose one individual (the "source chromosome") from the superior half and another one (the "destination chromosome") from the inferior half
  - Transfer a part from the source chromosome to the destination chromosome (select the part randomly or by a predefined criterion)
  - Repeat the steps above N_{Inf} times (N_{Inf} is the number of "infections" to occur in one generation.)
- Repeat the procedure above from the ***modified bacterial mutation*** step until a certain termination criterion is satisfied (e.g. maximum number of generations)

Simulations

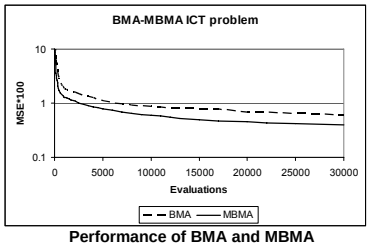
- **pH problem**
  - The aim of this example is to approximate the inverse of a titration-like curve. This type of non-linearity relates the pH (a measure of the activity of the hydrogen ions in a solution) with the concentration (x) of chemical substances.
- **Inverse Coordinate Transformation Problem (ICT)**
  - This example illustrates an inverse kinematic transformation between 2 Cartesian coordinates and one of the angles of a two-links manipulator.
- **Six variable non-polynomial function**
  - This example is widely used as a target function and the output is given by the following expression:

$$y = f_3(\underline{x}) = x_1 + x_2^{0.5} + x_3 \cdot x_4 + 2 \cdot e^{2 \cdot (x_5 - x_6)}$$
$$x_1 \in [1...5],\ x_2 \in [1...5],\ x_3 \in [0...4],$$
$$x_4 \in [0...0.6],\ x_5 \in [0...1],\ x_6 \in [0...1.2]$$

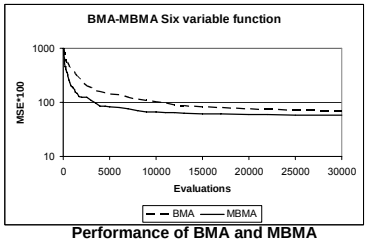
Simulation results MBMA



Performance of BMA and MBMA pH problem, 30 runs avg.



Performance of BMA and MBMA ICT problem, 30 runs avg



Performance of BMA and MBMA Six variable non-polyn. function, 30 runs avg.

Uniform Complexity

- The most expensive part of the algorithm is the *Levenberg-Marquardt* step
- *Moore-Penrose pseudoinverse* calculation is needed
- The uniform complexity of the accurate calculation of the *Moore-Penrose pseudoinverse* is $O(p \times q \times \min(p,q))$
  - p and q are the number of rows and columns of the matrix
- The number of the parameters of a fuzzy rule base ($N_{parameters}$) is less than the number of patterns in the training data set ($N_{patterns}$)
- The uniform complexity of the algorithm is $O(N^2_{parameters} \times N_{patterns})$
- Depends on the amount of data linearly

Conclusions for the BMA

- The Bacterial Memetic Algorithm was proposed
- This approach gives essentially better results as the Bacterial Evolutionary Algorithm
  - In terms of the optimization criterion
  - In the sense of other generalized criteria
- By the bacterial operators local optima can be avoided
- Using the Levenberg-Marquardt method locally best fitting can be approximated
- Thus, global optimum can be found with larger accuracy
- With the improved bacterial operators the number of rules is not needed to predefine

Conclusions

- Complexity reduction is possible by using projections, interpolation of rules and hierarchy, eventually by the combination of these three
- It is essential to find automatic model identification methods for complexity reduced models
- Clustering based approaches compete with evolutionary/ neural/ gradient techniques
- The latter are very promising but at present no implementation for hierarchical models exist (difficulty with expressing the partial derivatives necessary for the Jacobian matrix in LM)
- Further research is needed

Acknowledgments

- K. Hirota, Tokyo Inst. Of Technology: Interpolative (and) hierarchical systems
- A. E. Ruano, C. Cabrita, Univ. of Algarve (Faro): Levenberg-Marquardt algorithm
- T. D. Gedeon, A. Chong, Murdoch Univ. (Perth): Modeling by clustering
- J. Botzheim, Budapest:
  Bacterial Evolutionary and Memetic Algorithm