Author prepared draft.
D. Vincze, Sz. Kovács: Performance Optimization of the Fuzzy Rule Interpolation Method 'FIVE'
Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII),
Vol.15 No.3, 2011, Fuji Technology Press, ISSN 1343-0130, Tokyo, Japan, pp. 313-320.

Performance Optimization of the Fuzzy Rule Interpolation Method "FIVE"

Dávid Vincze and Szilveszter Kovács

Department of Information Technology, University of Miskolc, Miskolc, Hungary
E-mail: {david.vincze, szkovacs}@iit.uni-miskolc.hu

[Received 00 00 2011; accepted 00 00 2011]

Fuzzy Rule Interpolation (FRI) methods are efficient structures for knowledge-representation with relatively few rules. In spite of their good knowledge representation efficiency, their high computational demand makes the FRI methods hardly suitable for embedded real-time applications, for which short reasoning time has a high importance. On the other hand, the fact that currently available devices have increased computational power gives the FRI methods an opportunity to appear in real-time embedded applications. Therefore, the need for a low-computation and low-resource-demand FRI method is emerging. The goal of this paper is to introduce some implementation details of such an FRI method, together with its brief time and space complexity analysis. The paper also gives some hints for further performance optimization possibilities.

Keywords: fuzzy rule interpolation (FRI), FRI FIVE implementation, FRI performance optimization, FRIQ-learning

1. Introduction

In classical fuzzy reasoning methods (e.g., the Zadeh-Mamdani Compositional Rule of Inference (CRI)), it is obvious that having covering (complete) rule bases is a must. A traditional fuzzy-rule-based system requires a complete rule base with all of the possible rules set, even though lots of these rules are unimportant from the standpoint of the actual application. A fuzzy rule base is called sparse or incomplete if an observation that does not hit any of the rules in the rule base may exist. Accordingly, there can be observations for which no conclusion can be reached with traditional fuzzy reasoning techniques. On the other hand, in many embedded control application areas, having no conclusion is an avoidable situation. There are some traditional workarounds for such situations in the literature, e.g., applying the last real conclusion instead of the missing one, but these can have some unpredictable side effects. One real solution for the sparse rule base is the application of Fuzzy Rule Interpolation (FRI) methods. In this case, the derivable rules are intentionally missing from the rule base, as FRI methods are capable of providing reasonable (interpolated) conclusions even if none of the existing rules fire under the current observation. The rule base of an FRI system is not necessarily complete, so it can contain the most significant fuzzy rules alone without risking the chance of having no conclusion for some of the observations. In this case, since there is an efficient knowledge representation, a considerable amount of unnecessary work can be avoided during the rule base creation. On the other hand, most of the FRI methods share the burden of high computational demand, e.g., the task of searching for the two closest rules surrounding the actual observation, or calculating the conclusion at least in some characteristic α-cuts. Moreover, in some methods, the interpretability of the fuzzy conclusion gained is not straightforward [1]. There have been a lot of efforts to rectify the interpretability of the interpolated fuzzy conclusion [2]. In [3], Baranyi et al. give a comprehensive overview of recent FRI methods. Beyond these problems, some of the FRI methods are originally defined for one dimensional input space and need special extension for the multidimensional case (e.g., [4, 5]). In [6], Wong et al. give a comparative overview of the FRI methods capable of multidimensional input space. In [4], Jenei introduces an axiomatic treatment of the FRI methods. In [7], Johanyák introduces an automatic way for sparse fuzzy model identification from sample data. The high computational demand, mainly from the search for the two closest surrounding rules to an arbitrary observation in the multidimensional antecedent space, makes many of these methods hardly suitable for real-time applications. Some FRI methods, (e.g., the method introduced by Jenei et al. in [5], FRIPOC [8], LESFRI [9], and VEIN [10]), eliminate the search for the two closest surrounding rules by taking all the rules into consideration, hence speeding up the reasoning process. On the other hand, keeping the goal of constructing a fuzzy conclusion and not simply speeding up the reasoning process, they still require some additional (or repeated) computational steps for the elements of the level set (or at least some relevant α levels). An application-oriented aspect of the FRI, the low computational and resource demand is emerging in the concept of the FRI method "FIVE" (Fuzzy Interpolation based on Vague Environment (FIVE)). In the following, the implementation details of this method will be studied.

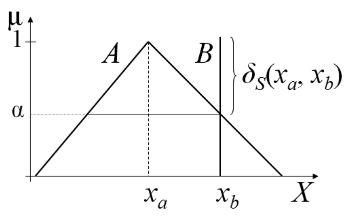


Fig. 1. The α-cuts of $\mu_A(x)$ contain the elements that are $(1-\alpha)$-indistinguishable from x_a, where A is a fuzzy set and B is a singleton fuzzy set at x_b.

2. FRI Based on Vague Environment: FIVE

In the concept of FIVE, an application-oriented aspect, the need for low computational and resource demand has a high importance. The method was originally introduced in [11–13], and [14] to satisfy the speed requirements of direct embedded fuzzy control in which the conclusions of the fuzzy controller are applied directly as control actions in a real-time system.

The main idea of FIVE is based on the fact that most of the control applications serve crisp observations and require crisp conclusions from the controller. Adopting the idea of the Vague Environment (VE) [15], FIVE can handle the antecedent and consequent fuzzy partitions of the fuzzy rule base by scaling functions [15] and therefore turn the fuzzy interpolation into crisp interpolation. The idea of a VE is based on the indistinguishability of elements. In VE, the fuzzy membership function $\mu_A(x)$ indicates the level of similarity of x to a specific element x_a which is a representative or prototypical element of the fuzzy set $\mu_A(x)$, or equivalently the degree to which x is indistinguishable from x_a (see, e.g., on **Fig. 1**) [15]. Two values in a VE are ε-indistinguishable if their distance is less than or equal to ε. The distances in a VE are weighted distances Eq. (1). The weighting factor or function is called the scaling function (factor) [15]:

$$\delta_s(x_a, x_b) = \left| \int_{x_b}^{x_a} s(x)\,dx \right| \leq \varepsilon \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \qquad (1)$$

where $\delta_s(x_a, x_b)$ is the scaled distance of the values x_a, x_b and $s(x)$ is the scaling function on X.

If the VE of a fuzzy partition (the scaling function or at least the approximate scaling function [11, 14]) exists, the member sets of the fuzzy partition can be described by points in that VE (see, for example, exact scaling function s in **Fig. 2** and an approximate scaling function in **Fig. 3**).

Having the VE concept and the scaling-function-based similarity calculation, any crisp interpolation, extrapolation, or regression method can be adapted very simply for FRI [11, 14]. Because of its simple multidimensional applicability, in FIVE, the Shepard-operator-based interpolation (first introduced in [16]) is adapted (see, e.g., **Fig. 4**).

To be more precise, if there are singleton rule consequents (c_k), the fuzzy rules R_k have the following form:

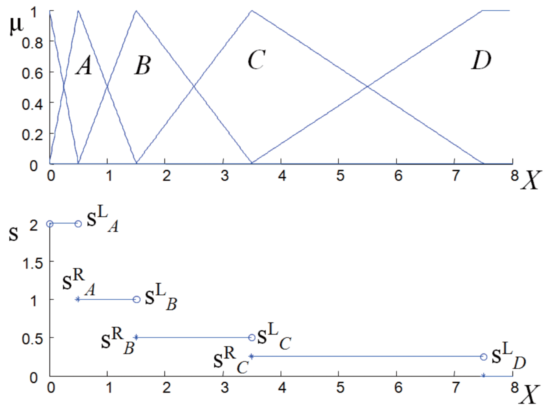


Fig. 2. A Ruspini fuzzy partition (fuzzy sets $A-D$) and its scaling function $s(x)$ on the universe of discourse X.

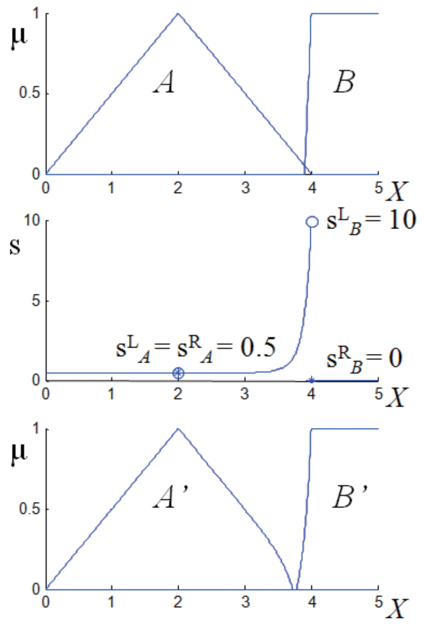


Fig. 3. A fuzzy partition (A, B), its approximate scaling function $s(x)$ [11, 14], and the corresponding approximated partition (A', B') on the universe of discourse X.

$$\textbf{If } x_1 = A_{k,1} \textbf{ And } x_2 = A_{k,2} \textbf{ And} \ldots \textbf{ And } x_m = A_{k,m}$$
$$\textbf{Then } y = c_k \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \qquad (2)$$

Adapting the VE concept and the scaling-function-based similarity calculation to the Shepard-operator-based interpolation, the conclusion of the interpolative fuzzy reasoning can be obtained as [14]:

$$y(\mathbf{x}) = \begin{cases} c_k & \text{if } \mathbf{x} = \mathbf{a}_k \text{ for some } k, \\ \dfrac{\sum_{k=1}^{r} c_k \Big/ \delta_{s,k}^{\lambda}}{\sum_{k=1}^{r} 1 \Big/ \delta_{s,k}^{\lambda}} & \text{otherwise.} \end{cases}$$
$$. \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \qquad (3)$$

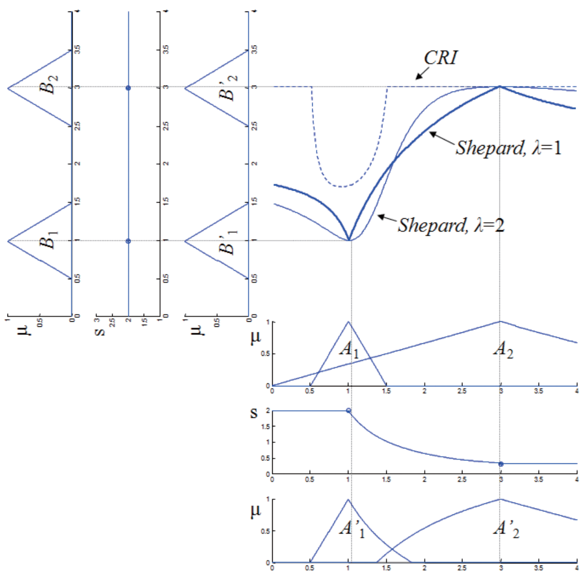


Fig. 4. Interpolation of two fuzzy rules ($\mathrm{R}_i : A_i \rightarrow B_i$), using the Shepard-operator-based FIVE ("*Shepard*" (3)), and for comparison, the min-max CRI with "center of gravity" defuzzification ("*CRI*" on figure).

where $\lambda > 0$ and $\delta_{s,k}$ are scaled distances:

$$\delta_{s,k} = \delta_s(\mathbf{a}_k, \mathbf{x}) = \left[\sum_{i=1}^{m} \left(\int_{\mathrm{a}_{k,i}}^{\mathrm{x}_i} s_{\mathrm{X}_i}(\mathrm{x}_i)\, d\mathrm{x}_i \right)^2 \right]^{\frac{1}{2}} . \quad (4)$$

and s_{X_i} is the i^{th} scaling function of the m-dimensional antecedent universe, $\mathbf{x}$ is the m-dimensional crisp observation, and $\mathbf{a}_k$ are the cores of the m-dimensional fuzzy rule antecedents $\boldsymbol{A}_k$.

The code of the FIVE FRI together with other FRI methods is freely available as a MATLAB FRI Toolbox [17], and it can be downloaded from [18].

3. Implementation of FIVE

In the following, the MATLAB implementation of FIVE (according to the implementation available in [18]) will be discussed in more detail.

The implemented functions can be divided into three main groups: *preprocessing*, *conclusion generation,* and *visualization* groups. The preprocessing group is responsible for calculating the lookup tables of the vague environments, based on the scaling functions. The conclusion generation group has the task of calculating the conclusions of the rule bases, based on the supplied observations. The visualization group supplies methods for fuzzy set reconstruction and application debugging, and also for knowledge base visualization.

The preprocessing group has two main functions. The first is the calculation of the scaling function of a fuzzy partition:

$$SCF = FIVEGScFunc\,(U, PSC) \quad . \; . \; . \; . \; . \quad (5)$$

where U is a vector of some given points from the universe of discourse, PSC is the set of coordinates and values describing relevant points of the scaling function, and SCF is a vector of the generated scaling function values (lookup table) in the positions defined in U. Scaling functions are defined by a set of left and right scaling values of some reference points (PSC in Eq. (5)). For example, the scaling function on **Fig. 3** is defined as $[(2, 0.5, 0.5), (4, 10, 0)]$.

The second function calculates the lookup table values of the scaled (vague) distances according to Eq. (1):

$$VE = FIVEGVagEnv\,(U, SCF) \quad . \; . \; . \; . \; . \; . \quad (6)$$

where VE is a vector of the pre-calculated scaled distances (lookup table) for the values defined in U.

According to Eqs. (3) and (4), the conclusion is calculated by the two functions based on the scaled distances of the observation and the rule antecedents. The first calculates the scaled distance D, based on the lookup table VE:

$$D = FIVEVagDist\,(U, VE, P1, P2) \quad . \; . \; . \; . \quad (7)$$

where $P1$ and $P2$ are arbitrary points inside the universe of discourse of U.

The second calculates the Euclidean scaled distance of the observation and the rule antecedents according to Eq. (4):

$$RD = FIVERuleDist\,(U, VE, R, X) \quad . \; . \; . \; . \quad (8)$$

where array R contains the rules of the rule base, X is the vector of the actual observation, and y is the calculated consequent. Rows in R are the vector of rules composed from the position number of the predefined linguistic terms of antecedents in their partition. The last element of the rule vector is the constant consequent value (c_k in Eqs. (2) and (3)).

The final conclusion y is then obtained by the following function (according to Eq. (3)):

$$y = FIVEVagConcl\,(U, VE, R, X, L) \quad . \; . \; . \; . \quad (9)$$

where L is the power factor of the Shepard interpolation (see λ in Eq. (3)).

From the viewpoint of the conclusion generation functions, the block diagram of FIVEVagConcl is summarized in **Fig. 5**.

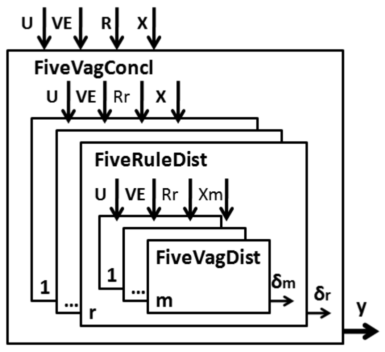


Fig. 5. Block diagram of the *FIVEVagConcl* function.

Table 1. Measured run times of the main FIVE functions.

Name of function	Number of calls	Execution time	% of time
FIVEVagConcl	1200000	79.970 s	6%
FIVERuleDist	1200000	561.985 s	43%
FIVEVagDist	9315028	658.764 s	51%

In ordinary applications, the two preprocessing functions (Eqs. (5) and (6)) and the conclusion generation function Eq. (9) are called directly. These three functions dramatically simplify the usage of the FIVE FRI for the potential application programmer.

4. Optimization of FIVE

An analysis of the implementation of the FIVE method shows that the most frequently used function is the *FIVEVagDist* function. Profiling the code of a simple application example which uses FIVE also proves that the most frequently called and the most resource hungry function is *FIVEVagDist* (see benchmark results in **Table 1**).

The *FIVEVagDist* function is responsible for calculating the scaled distance of two points based on pre-calculated lookup table values Eq. (7).

The first step of fetching values from a lookup table is the alignment of the two requested point coordinates to the pre-calculated positions of the universe of discourse (stored in U).

The original implementation of FIVE [18] is capable of handling arbitrary positions of the universes (see **Fig. 6A**).

Experience shows that in most cases FIVE based applications (e.g., [19–21]) use a fixed resolution universe description (see **Fig. 6B**) with points stored in an array in ascending order. Therefore, it seems to be possible to simplify the function by restricting the universe description from the arbitrary to the fixed resolution without having a high impact on its practical applicability.

In the original implementation [18] for aligning (see **Fig. 7**), the input points (one point for a rule in the rule base and one point for the current observation) to the pre-calculated positions of the lookup table, the method first calculates all the possible distances. Then it searches through for the nearest pre-defined universe positions i for each input point:

$$i_k = minindex\left(\left|U_{k_{1...n}} - P_k\right|\right) \quad \ldots\ldots\ldots (10)$$

where P_k is the input point, $U_{k_{1...n}}$ is the n element vector of the pre-calculated positions in the k^{th} dimension, *minindex* is the function calculating the index of the minimal element in an array, and i_k is the index of the alignment position of P_k.

In Eq. (10), the subtraction must be executed for all the pre-calculated positions; then, the smallest element should be found to gather the index of the nearest pre-calculated position in the array U.

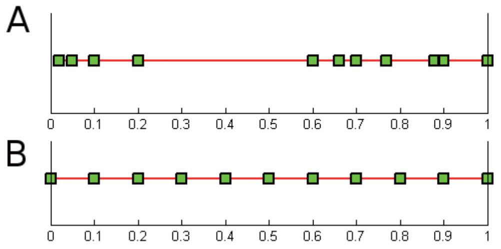


Fig. 6. Lookup table positions representing the universe of discourse in an antecedent dimension.
A: arbitrary universe, B: universe with a fixed resolution.

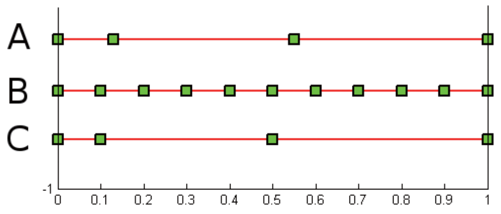


Fig. 7. Lookup table positions representing the universe of discourse in an antecedent dimension.
A: input points, B: universe with a fixed resolution,
C: input points aligned to the positions of the universe.

The number of calculations could be reduced by fixed resolution universe description (see **Fig. 6B**), where the pre-defined universe points are stored in an array in ascending order. In this case, it is possible to directly calculate the alignment position in one step:

$$i_k = round\left(n_k \cdot \frac{P_k - U_k}{U_{k_n} - U_{k_1}}\right) \quad \ldots\ldots\ldots (11)$$

where n_k is the number of the pre-calculated positions, U_{k_1} is the first pre-calculated position, U_{k_n} is the last pre-calculated position in the k^{th} dimension, and *round*( ) is a function which round towards the nearest integer.

The suggested modification makes the actual input point alignment process time complexity independent of the size of the lookup table that represents the universe of discourse. Therefore, the performance gain depends on the size of the applied universe; the larger the universe description table, the larger the performance gain.

The uniform time complexity of the original *FIVEVagDist* function can be approximated as:

$$m \cdot (9 \cdot n + 16) \quad \ldots\ldots\ldots (12)$$

where n is the average size of the lookup tables representing the m-dimensional antecedent universes.

The uniform time complexity of the optimized *FIVEVagDist* function is independent of the number of antecedent dimensions, and in usual cases ($n >> 1$) are much better than in the original version:

$$m \cdot 57. \quad \ldots\ldots\ldots (13)$$

The space complexity is the same for both versions of the function, as it is dominated by the size of the pre-calculated lookup table:

$$m \cdot n. \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (14)$$

In Landau notation, the time complexity of the original function is $O(mn)$, while in the modified function it is only $O(m)$. The space complexity is $O(mn)$ in both cases.

In typical cases in which the universes consist of about one thousand reference points ([19, 22]), the performance gain in time complexity is significant. (See the application example section for measurement results in a benchmark problem.)

4.1. Other Possible Optimizations

Analysis of the implemented *FIVERuleDist* function produces further enhancement of the overall performance. By default, FIVE stores antecedents and consequents in the same array (the rule base itself), which is convenient from the viewpoint of the programmer. Although it is convenient for the programmer, it means that extra resources are consumed when the array is splitting if only the antecedents are required. This is the case with the *FIVERuleDist* function. In the original implementation, *FIVERuleDist* truncates the corresponding arrays of the universes, the vague environments, and the rule bases so that they contain only the data of the antecedents. These truncations are performed for every rule in the rule base. The rule bases, universes, and vague environments do not change while the application is running, so it is possible to perform these truncations in advance and cache the result (antecedent caching). Then, the truncated antecedent arrays have to be passed as parameters to the corresponding FIVE functions. This means a change in the application programming interface, so existing applications have to be modified slightly to follow this new syntax. Creating the aforementioned truncated array (antecedent cache) will consume more memory of course, but this extra usage of memory is insignificant compared to the gain in speed.

The original *FIVERuleDist* function has the time complexity of

$$m \cdot (4 \cdot n + 8) \quad \ldots\ldots\ldots\ldots \quad (15)$$

and space complexity of

$$m + 4 \cdot n - 3 \quad \ldots\ldots\ldots\ldots \quad (16)$$

while the optimized version has time complexity

$$m \cdot (4 \cdot n + 2) \quad \ldots\ldots\ldots\ldots \quad (17)$$

and space complexity of

$$m + 2 \cdot n + 2 \quad \ldots\ldots\ldots\ldots \quad (18)$$

In Landau notation, both versions have the time complexity of $O(mn)$. The high difference in the benchmark execution time (FIVE with antecedent truncation in **Table 2** and **Fig. 8**) is mainly inherited from the high expense of the matrix truncation function. In the applied uniform time complexity calculation, it is hidden as a single calculation step.

Table 2. Measured run times [sec] of the main FIVE functions.

Name of function	execution time	Speed-up
Unmodified FIVE	1120.74 s	1 x
FIVE with fixed resolution universes	717.19 s	≈ 1.6x
FIVE with antecedent caching	645.07 s	≈ 1.7x
FIVE with conclusion lookup tables	931.97 s	≈ 1.2x
FIVE with all modifications	149.44 s	≈ 7.5x

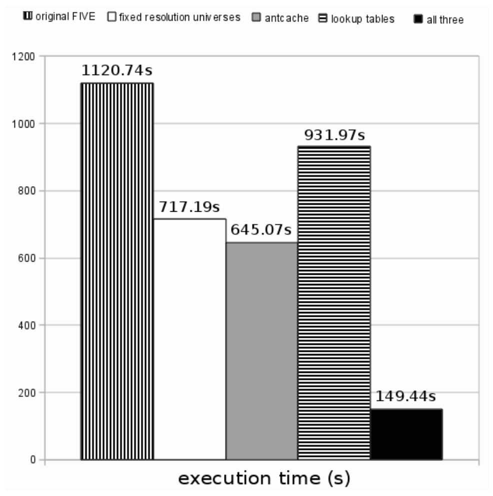


Fig. 8. Execution times of the vehicle navigation application benchmark, from left to right: unmodified FIVE, FIVE with fixed resolution universes, FIVE with antecedent caching, FIVE with conclusion lookup tables, and FIVE with all the modifications.

Another efficient method of achieving performance improvement in small embedded applications is the precalculation of the consequents for all the possible lookup table positions of the antecedent universe of discourse. If the lookup table resolutions of the universes are low, the rule bases with more than one antecedent can also use this approach. The space complexity should be investigated carefully because the number of required precalculated elements grows exponentially with the number of antecedent dimensions: $O(n^m)$. For example, with a typical resolution of a 1000-element lookup table and roughly 100,000 repeated evaluations, this simple method can yield a considerable speed gain (the costly FIVE calculations only have to be performed 1000 times instead of roughly 100,000 times during the evaluations).

For swift application startup, caching the generated universes with their calculated vague environments and rule bases on storage is also a plausible solution.

In the age of multi-core CPU systems, it is straightforward to study parallelization possibilities in FIVE. There are two locations in the implementation where the advantages of parallel execution can be exploited. One is in the *FIVERuleDist* function (see **Fig. 5**). The iterations of the main *for* loop of the function are data independent, because in each iteration the weight of a different rule (independent data sets) is calculated even if the same code is executed.

The next candidate for the parallelization is the *FIVE-VagDist* function (see **Fig. 5**). In this case, the observation and the rule antecedent vague distances can be calculated independently for each antecedent dimension.

For embedded real-time applications, the additional computational cost of parallelization, data distribution, and data collection also have to be taken into consideration, as the level of available parallelization (the number of the fuzzy rules in case of the *FIVERuleDist* and the number of antecedent dimensions in case of the *FIVE-VagDist*) are not so high.

4.2. Application Example

For the first benchmark of the performance, the optimization of a simple vehicle navigation demo application is selected (see details in [22]). The goal of the application is to control an unmanned robot capable of room surveillance, making it cycle through given waypoints within a room with walls and moving obstacles to avoid. When the path of the robot seems to be blocked by an obstacle, the robot is capable of turning around and heading in the opposite direction. This application uses the original FIVE method with four simple rule bases. The universes have fixed resolutions of 1001 elements. One of the rule bases has only one antecedent; another one has only two. Hence, for them, lookup tables of the conclusions can be generated. The other two rule bases have 3 and 10 antecedent dimensions. As a benchmark, the main loop of the vehicle navigation application example is analyzed. The resulting execution times of 10^8 main vehicle navigation cycles are summarized in **Table 2** and **Fig. 8**.

4.3. Optimizing FIVE for FRIQ-Learning

A special application example of the optimized FIVE FRI is the FRI-based Q-learning (FRIQ-learning) introduced in [20] and [23]. The FRIQ-learning is an extension of the traditional Fuzzy Q-learning (FQ-learning) method with the capability of handling sparse fuzzy rule bases. The introduction of FIVE FRI in FQ-learning allows the omission of deducible fuzzy rules from the rule base representing the action-state value function. This reduction also adds the potential for applying FRIQ-learning in higher state dimensions than the original FQ-learning, thanks to the sparse fuzzy rule-base model of the action-state space.

The efficiency of the FRI method has a high impact on FRIQ-learning, as the methodology (similarly to the Q-learning) requires an extremely high number of repeatedly executed action-state value function-model updates (FRI reasoning) in runtime. Because of the similar repeatedly executed FRI operations, the optimization of the applied FIVE FRI, suggested in this paper, could improve FRIQ-learning to a great extent.

From the previously introduced FIVE optimization methods, first of all, the fixed resolution universe description can be applied to speed up FRIQ-learning.

Because of the high dimensionality of the action-state space, a conclusion lookup table is practically unimplementable in this case.

Beyond the FIVE optimization methods introduced previously, because of the specialties of the FRIQ-learning calculations, another optimization method is also approaching. In FRIQ-learning for action-state value model updates and action selection, the values of each possible action of a given state have to be calculated.

According to [23], the action-state values function FRI model has the following rule form (k^{th} rule):

$$\textbf{If } s_1 = S_{k,1} \textbf{ And} \ldots \textbf{And } s_m = S_{k,m} \textbf{ And } a = A_u$$
$$\textbf{Then } \tilde{Q}(s_1, \ldots, s_m, a) = Q_k \quad \ldots\ldots \quad (19)$$

where $S_{k,i}$ is the label of the i^{th} membership function of the m-dimensional state space, A_u is the label of the u^{th} membership function of the one dimensional action space, Q_k is the singleton conclusion, and $\tilde{Q}(s_1, \ldots, s_m, a)$ is the approximated, continuous action-state value function.

In accordance with Eq. (19), the FRI action-state values function model has a multidimensional state and a single action dimension ([20] and [23]). Caching the calculated vague distances of the observed state and the rule antecedents related to the state dimensions of the action-state value FRI rule base can dramatically speed up the calculations. In each action-state value model update and action selection cycle, the vague distances of the actual state and the rule antecedents related to the state dimensions have to be calculated only once.

The performance gain that could be achieved by this modification depends on the number of the antecedent fuzzy term sets in the action universe. The higher the number of the action term sets, the better the performance gain becomes.

For execution time benchmarks, a small application example has been presented for FRIQ-learning in [23].

This application has been extended with the proof-of-concept implementation of the suggested FIVE FRI optimized FRIQ-learning. This benchmark is the well-known cart-pole (reversed pendulum) simulation. The program runs through episodes, where an episode is a single cart-pole simulation run. The goal of the application is to move the cart to the center position while balancing the pole. The action-state value function FRI model Eq. (19) has a four-dimensional state and a single-dimensional action universe with a resolution of 21 distinguishable actions.

The execution times of the benchmark show a significant performance gain over the original version. **Table 3** summarizes the runtimes of the first best-action-selection iteration step at the start of the application.

Table 4 summarizes the running time for the first five episodes of a simulation run from the whole application with the various optimizations disabled or enabled.

In conclusion of the benchmarks, it can be stated that the optimized FIVE FRI-based FRIQ-learning can achieve a tenfold increase in speed over the original FIVE FRIQ-learning implementation.

Table 3. Run times [msec] of a best-action-selection iteration with the various optimizations.

	Time	Speed-up
Original version	$\approx$ 115 ms	1x
States calculated only once	$\approx$ 25ms	$\approx$ 9.5x
States calculated only once and using fixed resolution alignment optimization	$\approx$ 9ms	$\approx$ 12.5x

Table 4. Run times [sec] for the first five episodes of a simulation run with the various optimizations.

	Time	Speed-up
Original version	$\approx$ 76.1s	1x
Fixed resolution universes	$\approx$ 24.6s	$\approx$ 3.1x
States calculated only once	$\approx$ 23.5s	$\approx$ 3.2x
States calculated only once and using fixed resolution alignment optimization	$\approx$ 7.9s	$\approx$ 9.6x

5. Conclusion

The main conclusion of the paper is that data structural simplification and run-time caching methods can achieve a relevant speed-up in computational time of FRI applications.

The data structural simplification speeds up lookup table data fetching while caching techniques can speed up repetitive calculations, such as rule evaluations in a single reasoning cycle (see speed-up results in **Table 2**). The gain of caching can be more relevant if the FRI reasoning core is embedded into repetitive calculations, as in FRIQ-learning (see speed-up results in **Table 4**).

This performance gain could be a relevant improvement for embedded FRI real-time applications, for which rapid reasoning is the essential question of the FRI applicability.

Another important conclusion of the paper is the relatively small impact of conclusion lookup tables on the FRI reasoning speed performance (**Table 2** and **Fig. 8**). The speed performance gain that can be achieved is negligible against the exponential space complexity loss of the conclusion lookup tables. (Antecedent universes lookup tables has a space complexity of $O(mn)$, while that of conclusion lookup tables have $O(n^m)$.)

The main reason for the low reasoning speed gain of the conclusion lookup tables is inherited from the effective FRI sparse-fuzzy-rule-based knowledge representation format. This is a special feature of the FRI knowledge representation, i.e., the size of the rule base is relatively small, containing the relevant rules only. For classical fuzzy reasoning methods in which the exponentially-large, complete rule base is required, conclusion lookup tables can achieve a better speed-up factor.

Acknowledgements

This research was partly supported by the Hungarian National Scientific Research Fund grant number OTKA K77809.

References:

[1] L. T. Kóczy and Sz. Kovács, “On the preservation of the convexity and piecewise linearity in linear fuzzy rule interpolation,” Tokyo Inst. Technol., Yokohama, Japan, Tech. Rep. TR 93-94/402, LIFE Chair Fuzzy Theory, 1993.

[2] D. Tikk and P. Baranyi, “Comprehensive analysis of a new fuzzy rule interpolation method,” In IEEE Trans. Fuzzy Syst., Vol.8, No.3, pp. 281-296, June, 2000.

[3] P. Baranyi, L. T. Kóczy, and T. D. Gedeon, “A Generalized Concept for Fuzzy Rule Interpolation,” IEEE Trans. on Fuzzy Systems, Vol.12, No.6, pp. 820-837, 2004.

[4] S. Jenei, “Interpolating and extrapolating fuzzy quantities revisited – an axiomatic approach,” Soft Comput., Vol.5, pp. 179-193, 2001.

[5] S. Jenei, E. P. Klement, and R. Konzel, “Interpolation and extrapolation of fuzzy quantities – The multiple-dimensional case,” Soft Comput., Vol.6, pp. 258-270, 2002.

[6] K. W. Wong, D. Tikk, T. D. Gedeon, and L. T. Kóczy, “Fuzzy Rule Interpolation for Multidimensional Input Spaces With Applications,” IEEE Trans. on Fuzzy Systems, Vol.13, No.6, pp. 809-819, December, 2005.

[7] Zs. Cs. Johanyák, “Sparse fuzzy model identification Matlab toolbox – RuleMaker toolbox,” Proc. of IEEE 6th Int. Conf. on Computational Cybernetics ICCC 2008, Stara Lesná, Slovakia, pp. 69-74, 2008.

[8] Zs. Cs. Johanyák and Sz. Kovács, “Fuzzy Rule Interpolation Based on Polar Cuts, Computational Intelligence,” Theory and Applications, Springer Berlin Heidelberg, pp. 499-511, 2006.

[9] Zs. Cs. Johanyák and Sz. Kovács, “Fuzzy Rule Interpolation by the Least Squares Method,” 7th Int. Symposium of Hungarian Researchers on Computational Intelligence (HUCI 2006), Budapest, pp. 495-506, November 24-25, 2006.

[10] Zs. Cs. Johanyák and Sz. Kovács, “Vague Environment-based Two-step Fuzzy Rule Interpolation Method,” 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics (SAMI 2007), Poprad, Slovakia, pp. 189-200, January 25-26, 2007.

[11] Sz. Kovács, “New Aspects of Interpolative Reasoning,” Proc. of the 6th. Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain, pp. 477-482, 1996.

[12] Sz. Kovács, and L.T. Kóczy, “Approximate Fuzzy Reasoning Based on Interpolation in the Vague Environment of the Fuzzy Rule base as a Practical Alternative of the Classical CRI,” Proc. of the 7th Int. Fuzzy Systems Association World Congress, Prague, Czech Republic, pp. 144-149, 1997.

[13] Sz. Kovács and L.T. Kóczy, “The use of the concept of vague environment in approximate fuzzy reasoning,” Fuzzy Set Theory and Applications, Tatra Mountains Mathematical Publications, Mathematical Institute Slovak Academy of Sciences, Bratislava, Slovak Republic, Vol.12, pp. 169-181, 1997.

[14] Sz. Kovács, “Extending the Fuzzy Rule Interpolation “FIVE” by Fuzzy Observation,” Advances in Soft Computing, Computational Intelligence, Theory and Applications, Bernd Reusch (Ed.), Springer Germany, pp. 485-497, 2006.

[15] F. Klawonn, “Fuzzy Sets and Vague Environments,” Fuzzy Sets and Systems, Vol.66, pp. 207-221, 1994.

[16] D. Shepard, “A two dimensional interpolation function for irregularly spaced data,” Proc. 23rd ACM Int. Conf., pp. 517-524, 1968.

[17] Zs. Cs. Johanyák, D. Tikk, Sz. Kovács, and K. W. Wong, “Fuzzy Rule Interpolation Matlab Toolbox – FRI Toolbox,” Proc. of the IEEE World Congress on Computational Intelligence (WCCI’06), 15th Int. Conf. on Fuzzy Systems (FUZZ-IEEE’06), July 16-21, Vancouver, BC, Canada, Omnipress., pp. 1427-1433, 2006.

[18] The FRI Toolbox. http://fri.gamf.hu/

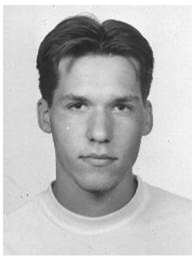
[19] Sz. Kovács and L.T. Kóczy, “Application of the Approximate Fuzzy Reasoning Based on Interpolation in the Vague Environment of the Fuzzy Rulebase in the Fuzzy Logic Controlled Path Tracking Strategy of Differential Steered AGVs”, Computational Intelligence – Theory and Applications, Lecture Notes in Computer Science, Vol.1226, Springer, pp. 456-467, Germany, 1997.

[20] D. Vincze and Sz. Kovács, “Fuzzy Rule Interpolation-based Q-learning,” SACI 2009, 5th Int. Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, pp. 55-59, May 28-29, 2009.

[21] Sz. Kovács, “Interpolative Fuzzy Reasoning in Behaviour-based Control,” Advances in Soft Computing, Vol.2, Computational Intelligence, Theory and Applications, Bernd Reusch (Ed.), Springer, Germany, pp. 159-170, 2005.

[22] D. Vincze and Sz. Kovács, "Behaviour Based Control with Fuzzy Automaton in Vehicle Navigation," Production Systems and Information Engineering, Vol.5, University of Miskolc, Hungary, pp. 151-166, 2009.

[23] D. Vincze and Sz. Kovács, "Incremental Rule Base Creation with Fuzzy Rule Interpolation-Based Q-Learning," I. J. Rudas et al. (Eds.), Computational Intelligence in Engneering, Studies in Computational Intelligence, Vol.313, Springer-Verlag, Berlin Heilderberg, pp. 191-203, 2010.



Name:
Dávid Vincze

Affiliation:
Ph.D. Student, Department of Information Technology, University of Miskolc

Address:
Miskolc-Egyetemváros, H-3515, Miskolc, Hungary

Brief Biographical History:
2008- Department of Information Technology, University of Miskolc
2003- Computer Centre, University of Miskolc

Main Works:
• D. Vincze and Sz. Kovács, "Incremental Rule Base Creation with Fuzzy Rule Interpolation-Based Q-Learning," Computational Intelligence in Engneering, Studies in Computational Intelligence, Vol.313, Springer-Verlag, Berlin Heilderberg, pp. 191-203, 2010.

Membership in Academic Societies:
• Hungarian Fuzzy Association

Name:
Szilveszter Kovács

Affiliation:
Associate Professor, Department of Information Technology, University of Miskolc

Address:
Miskolc-Egyetemváros, H-3515, Miskolc, Hungary

Brief Biographical History:
1989 Received M.Sc. in Electrical Engineering from Technical University of Budapest, Hungary
1993 Received M.Sc. in Computer Engineering from Technical University of Budapest, Hungary
1998 Received Ph.D. in Engineering from University of Miskolc, Hungary
1989-1998 Network Manager, Computer Centre, University of Miskolc, Hungary
1998-2005 Senior Lecturer, Department of Information Technology, University of Miskolc, Hungary
1999-2001 Research Fellow, Gifu Prefectural Research Institute of Manufacturing Information Technology, Japan
2005-2010 Senior Research Fellow, Institute of Information Technologies, Kecskemét College, Hungary
2009-2010 Research Fellow, Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Slovakia
2009-2010 Head of the Department, Department of Automation, University of Miskolc, Hungary
2005-Associate Professor, Department of Information Technology, University of Miskolc, Hungary

Main Works:
• fuzzy systems, fuzzy rule interpolation and its embedded applications, behaviour-based control and reinforcement learning

Membership in Academic Societies:
• EURO (The Association of European Operational Research Societies) Working Group on Fuzzy Sets (EUROFUSE)
• Integrated Intelligent Systems, Japanese-Hungarian Joint Laboratory (IISL)
• John von Neumann Computer Society
• Hungarian Fuzzy Association