Computer Assisted Mechanics and Engineering Sciences, **12**: 207–221, 2005.

Influence of clustering pre-processing on genetically generated fuzzy knowledge bases

Aleksander Przybylo, Sofiane Achiche, Marek Balazinski, Luc Baron
École Polytechnique de Montréal, Mechanical Engineering Department
C.P. 6079, Succ. Centre-Ville, Montréal (Québec) H3C 3A7, Canada

(Received March 31, 2005)

Automatic knowledge base generation using techniques such as genetic algorithms tend to be highly dependent on the quality and size of the learning data. First of all, large data sets can lead to unnecessary time loss, when smaller data sets could describe the problem as well. Second of all, the presence of noise and outliers can cause the learning algorithm to degenerate. Clustering techniques allow compressing and filtering the data, thus making the generation of fuzzy knowledge bases faster and more accurate. Different clustering algorithms are compared and the validation of the results through a theoretical 3D surface, shows that when compressing the data to 5% of its original size, clustering algorithms accelerate the learning process by up to 94%. Moreover, when the learning data contains noise and/or a large amount of outliers, clustering algorithms can make the results more stable and improve the fitness of the obtained FKBs.

1. Introduction

Machine learning applications such as Genetic Algorithms (GA) [7] allow automatic building of knowledge bases on phenomena without any theoretical knowledge nor expert assistance, based only on a set of learning data. The quality of the obtained knowledge bases is highly dependant on that of the data set that is presented to the learning algorithm. Hence, learning data sets containing a large amount of noise and/or outliers will result in a loss of fitness of the knowledge base, since the learning algorithm will try to model the noise/outliers instead of the real phenomenon. Moreover, lack of knowledge about the phenomenon often leads to "over-representation", meaning collecting a very large data set when a much smaller one would be sufficient. Such large data sets will inevitably cause a considerable time loss which can be problematic for applications that need a fast an accurate response.

This paper presents a study of the robustness in terms of Gaussian noise and outlier probability of Real Binary-Like Coded Genetic Algorithm (RBCGA) used to generate Fuzzy Knowledge Bases (FKB) and proposes a technique based on clustering algorithms to improve the fitness of the obtained FKBs and reduce the time needed to perform the learning process. First, short presentations of the Fuzzy Decision Support System (FDSS) and RBCGAs describe the main aspects and specificities of these techniques and the software that was used. A description of the clustering techniques and validation results follow in the next two sections.

2. Fuzzy decision support system

A rule-based approach to decision making using fuzzy logic techniques may consider imprecise vague language as a set of rules linking a finite number of conclusions. The knowledge base of such systems consists of two components: a linguistic terms base and a fuzzy rules base [3]. The former is divided into two parts: the fuzzy premises (or inputs) and the fuzzy conclusions (or outputs).

2.1. Theoretical specificities

In this paper we consider FKBs of N multiple inputs and one single output (MISO). Moreover, we consider only general overlapping triangular fuzzy sets on the premises and sharp-symmetric triangular fuzzy sets on the conclusion. The representation of such imprecise knowledge by means of fuzzy linguistic terms makes it possible to carry out quantitative processing in the course of inference that is used for handling uncertain (imprecise) knowledge. This is often called approximate reasoning [21]. This knowledge, expressed by ($k = 1, 2, \ldots, K$) finite heuristic fuzzy rules of the type MISO, may be written in the form:

$$R^k_{\mathrm{MISO}} : \text{ if } x_1 \text{ is } X^k_1 \text{ and } x_2 \text{ is } X^k_2 \text{ and } \ldots \text{and } x_N \text{ is } X^k_N \text{ then } y \text{ is } Y_k, \tag{1}$$

where $\{X^k_i\}^N_{i=1}$ denote values of linguistic variables $\{x_i\}^N_{i=1}$ (conditions) defined in the following universe of discourse $\{X_i\}^N_{i=1}$; and Y^k stands for the value of the independent linguistic variable y (conclusion) in the universe of discourse $\mathbf{Y}$. The global relation aggregating all rules from $k = 1$ to K is given as

$$R = \mathrm{also}^K_{k=1}\left(R^k_{\mathrm{MISO}}\right), \tag{2}$$

where the sentence connective *also* denotes any t- or s-norm (e.g., min ($\wedge$) or max ($\wedge$) operators) or averages. For a given set of fuzzy inputs $\{X'_i\}^N_1$ (or observations), the fuzzy output Y' (or conclusion) may be expressed symbolically as:

$$Y' = \left(X'_1, X'_2, \ldots, X'_N\right) \circ R, \tag{3}$$

where $\circ$ denotes a compositional rule of inference (CRI), e.g., the sup-$\wedge$ or sup-prod (sup-$*$). Alternatively, the CRI of Eq.(3) is easily computed as

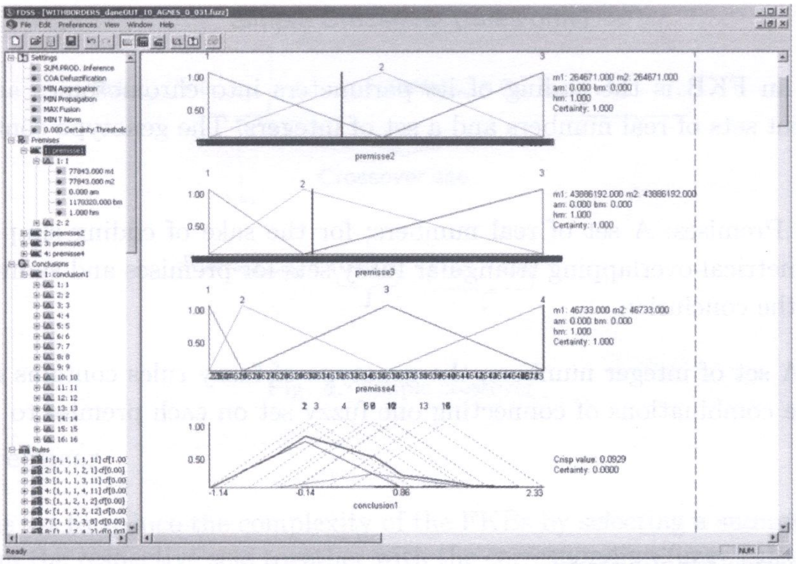
$$Y' = X'_N \circ \ldots \circ \left(X'_2 \circ \left(X'_1 \circ R\right)\right). \tag{4}$$

The CRI mechanisms allow us to obtain different conclusions represented as the membership function Y'. In FDSS Fuzzy-Flou, there are three defuzzification methods: the centre of gravity (COG); the mean of maxima (MOM); and the height method. All the results presented in this paper are obtained using the Σ-sup*-*-* CRI (sum-prod) and COG as defuzzification.

2.2. FDSS Learning paradigm

In general, FDSS requires a knowledge base in order to support the decision-making process of endusers. The FKB can be created manually by a human expert or automatically learned from a set of sampled data. In this paper the automatic learning process of the FDSS knowledge base is automatic. The learning process is aimed at producing knowledge bases that are manageable by either a human expert or a computer. The FKBs must accurately reproduce the set of learned data, while interpolating or extrapolating fair conclusions in other situations. A minimalist approach is implemented through an automatic reduction of fuzzy rules and sets on the premises, whenever the approximation error is not penalized too greatly by this reduction.

Figure 1 shows a screen printout of the FDSS Fuzzy-Flou software used as a validation tool for the genetically generated FKBs. It was developed at Ecole Polytechnique de Montréal (Canada) and the Silesian University of Technology (Poland). For more information on the Fuzzy-Flou software, please refer to [3].



FDSS - [WITHBORDERS_daneOUT_10_AGNES_0_031.fuzz]

File Edit Preferences View Window Help

Settings
- SUM.PROD. Inference
- COA Defuzzification
- MIN Aggregation
- MIN Propagation
- MAX Fusion
- MIN T Norm
- 0.000 Certainty Threshold

Premises
- 1: premisse1
  - 1: 1
    - 77843.000 m1
    - 77843.000 m2
    - 0.000 am
    - 1170320.000 bm
    - 1.000 hm
  - 2: 2
- 2: premisse2
- 3: premisse3
- 4: premisse4

Conclusions
- 1: conclusion1
  - 1: 1
  - 2: 2
  - 3: 3
  - 4: 4
  - 5: 5
  - 6: 6
  - 7: 7
  - 8: 8
  - 9: 9
  - 10: 10
  - 11: 11
  - 12: 12
  - 13: 13
  - 14: 14
  - 15: 15
  - 16: 16

Rules
- 1: [1, 1, 1, 1, 11] cf[1.00
- 2: [1, 1, 1, 2, 1] cf[0.00]
- 3: [1, 1, 1, 3, 11] cf[0.00
- 4: [1, 1, 1, 4, 11] cf[0.00
- 5: [1, 1, 2, 1, 2] cf[0.00]
- 6: [1, 1, 2, 2, 12] cf[0.00
- 7: [1, 1, 2, 3, 8] cf[0.00]

m1: 264671.000 m2: 264671.000
am: 0.000 bm: 0.000
hm: 1.000
Certainty: 1.000

m1: 43886192.000 m2: 43886192.000
am: 0.000 bm: 0.000
hm: 1.000
Certainty: 1.000

m1: 46733.000 m2: 46733.000
am: 0.000 bm: 0.000
hm: 1.000
Certainty: 1.000

Crisp value: 0.0929
Certainty: 0.0000

Ready

Fig. 1. Graphic interface of the FDSS Fuzzy-Flou

3. Automatic generation of FKBs

GAs are powerful stochastic optimization techniques based on the analogy of the mechanics of biological genetics and imitate the Darwinian survival of the fittest approach [7]. As shown in Fig. 2, each individual of a population is a potential FDSS Fuzzy-Flou knowledge base. Figure 2 presents the encoding/decoding scheme as well as the four basic operations, i.e.: reproduction, mutation, evaluation and natural selection, of the developed GA learning software. The method uses iterative improvement of individuals at each generation to converge toward multiple optima simultaneously. When the number of unknown parameters increases, GA exhibits only a polynomial increase of the complexity [6, 16], while the other optimization techniques show an exponential increase. The RBCGA developed by the authors [1] is a combination of a real coded genetic algorithm (RCGA) and a binary coded genetic algorithm (BCGA).

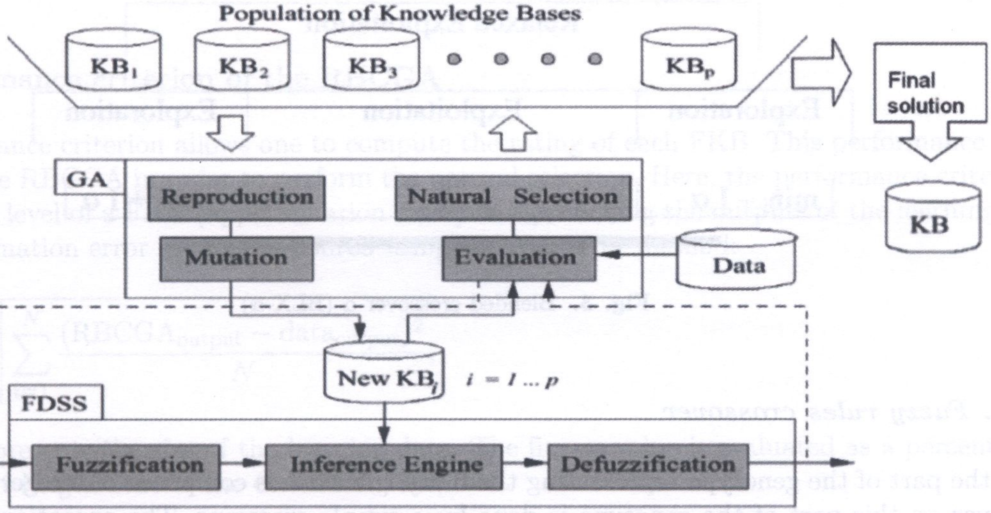


Fig. 2. Genetic learning paradigm

3.1. Coding

The *genotype* of an FKB is the coding of its parameters into chromosomes and corresponds to several independent sets of real numbers and a set of integers. The genotype contains the following items:

- Input/Output Premises: A set of real numbers; for the sake of coding simplicity, we consider only non-symmetrical-overlapping triangular fuzzy sets for premises and symmetrical triangular fuzzy sets for the conclusion.

- Fuzzy Rules: A set of integer numbers; the genotype of fuzzy rules contains information about all the possible combinations of connecting one fuzzy set on each premise to a fuzzy set on the conclusion.

3.2. Multi-crossover mechanisms

The evolution of a population of FKBs at each generation is achieved by the reproduction of the "best" individuals, based on their abilities to survive natural selection. Reproduction is performed by crossover of the genotype of the parents to obtain the genotype of an offspring, using a multi-crossover which is composed of a premises/conclusion crossover and a fuzzy rules crossover. These mechanisms are governed by the initiating probability p_c.

3.2.1. Premises/conclusion crossover

The mechanism used is called blending crossover α (BLX-α) [19], where α determines the exploitation/exploration level of the offspring (see Fig. 3). The parameter α is set to 1.0 for the first third of the generations (exploration) to 0.5 for the second third (relaxed exploitation) and finally to 0.1 for the last third of the evolution (exploitation). These changes in exploitation/exploration balance, help avoiding premature convergence [2].

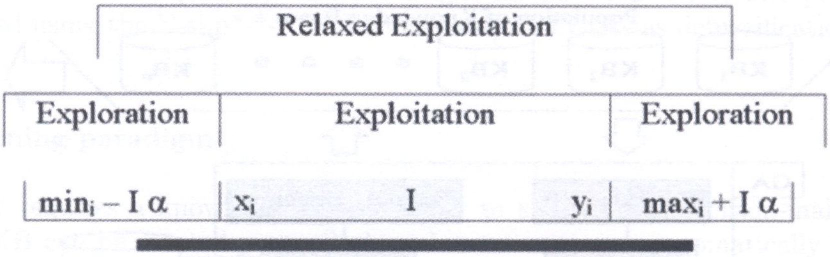


Fig. 3. Blended crossover α (BLX-α)

3.2.2. Fuzzy rules crossover

Since the part of the genotype representing the fuzzy rule base is composed of integer numbers, the crossover on this part of the genotype is done by a simple crossover. The operation is performed by exchanging the end part of the sets (containing the fuzzy rules) of the parents at a randomly selected crossover site, as shown in Fig. 4.

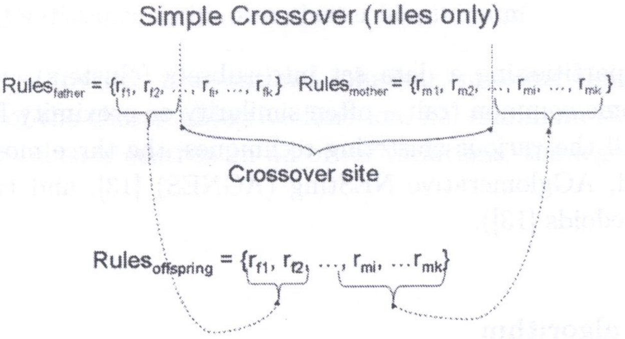

Simple Crossover (rules only)
Rules$_{father}$ = {r$_{f1}$, r$_{f2}$, ..., r$_{fi}$, ..., r$_{fk}$}
Rules$_{mother}$ = {r$_{m1}$, r$_{m2}$, ..., r$_{mi}$, ..., r$_{mk}$}
Crossover site
Rules$_{offspring}$ = {r$_{f1}$, r$_{f2}$, ..., r$_{mi}$, ...r$_{mk}$}

Fig. 4. Simple crossover

3.3. Fuzzy set reducer

This mechanism is set to reduce the complexity of the FKBs by selecting a summit on each premise and erasing it from the respective sets together with the corresponding fuzzy rules. This mechanism is governed by the initiating probability p_r.

3.4. Mutation

Mutation is a random alteration of a new member's genotype in the population. Mutation makes it possible to seek completely new solutions without any control on the direction, as opposed to gradient-based optimization techniques. The probability p_m governs the occurrence of this mechanism. In this paper, uniform mutation [11], similar to BLX-α with $\alpha = 0$, is applied.

3.5. Natural selection

Natural selection is performed on the population by keeping the "most promising" individuals, based on their fitness. The first generation begins with P FKBs and the same number is generated by crossover and mutation. To keep the population constant, we apply natural selection on the $2P$ FKBs by ordering them according to the performance criterion and keeping the P first FKBs. In this paper the population size is set to 100 and the number of generations is set to 500.

3.6. Performance criterion of the RBCGA

The performance criterion allows one to compute the rating of each FKB. This performance rating is used by the RBCGA in order to perform the natural selection. Here, the performance criterion is the accuracy level of a FKB (approximation error) in reproducing the outputs of the learning data. The approximation error Δ_{rms} is measured using the *rms* error method:

$$\Delta_{\mathrm{rms}} = \sqrt{\sum_{i=1}^{N} \frac{(\mathrm{RBCGA}_{\mathrm{output}} - \mathrm{data}_{\mathrm{output}})^2}{N}}, \tag{5}$$

where, N represents the size of the learning data. The fitness value is evaluated as a percentage of L, the output length base of the conclusion, i.e.,

$$\phi_{\mathrm{rms}} = \frac{L - \Delta_{\mathrm{rms}}}{L} \times 100. \tag{6}$$

4. Clustering

"Clustering consists of partitioning a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait – often similarity or proximity for some defined distance measure" [20]. Among all the various clustering techniques, the three most popular were selected: one hierarchical method, AGglomerative NESting (AGNES) [13], and two partitioning methods (K-Means [17] and K-Medoids [13]).

4.1. The hierarchical algorithm

Since most of the hierarchical methods are relatively time consuming, only the simplest one, AGNES, has been tested. It starts with each object in its own cluster and merges the clusters closest to each other, thus creating larger clusters. A step by step description of the algorithm follows.

- Place each object in its own cluster;
- Calculate the distances between each pair of clusters;
- Merge the pair of clusters with the smallest distance;
- Repeat steps 2 and 3 until the desired number of clusters is attained.

The AGNES algorithm is deterministic. Its execution time depends only on the size of the data and the number of clusters desired. It does not depend on the way the data is ordered. On the other hand, it suffers from several handicaps, mostly due to its rigidity. The most important one is the fact that once two clusters are combined, that combination cannot be undone. Clusters cannot be separated and objects cannot be swapped between clusters. This isn't the case of the partitioning algorithms.

4.2. The partitioning algorithms

The partitioning algorithms start with an arbitrary distribution in the desired number of clusters and then examine its neighbourhood (similar distributions with one object swapped between two clusters). The Best Possible (BP) versions search for the best similar distribution whereas the First Better (FB) versions perform the swap as soon as a better one is found. Finally, the algorithms stop when the distribution is better than its entire neighbourhood. The two partitioning methods, K-Means and K-Medoids, differ only by the central measure of the clusters. Their step by step description is as follows:

- Create as many clusters as needed;
- Place about as many objects in each cluster;
- Compute the centroids/medoids of each cluster;
- For each object search for clusters with centroids/medoids closer to the object than its actual cluster's centroid/medoid;
- Perform the swap when:
  - ○ the closest cluster is found (BP);
  - ○ a closer cluster is found (FB);

- Stop when each object's distance to its own cluster is minimum.

The algorithm converges towards near-optimal solutions, meaning that it does not necessarily find the global optimum of the chosen objective function, i.e. minimum sum of distances from the objects to their respective cluster centres. In all of its variations, the algorithms only find a local optimum.

4.3. Central measure

In the case of the partitioning methods two different central measures were tested, medoids and means. The medoids require more heavy computations than the means but they also offer a strongly improved robustness against outliers, since, contrarily to the mean, the distance of an outlier to the rest of the objects has no effect on the value of the medoid.

4.4. Distance/similarity functions

All of the algorithms have been tested with the Euclidean distance function in order to evaluate the similarity between two objects or clusters. In order not to "penalize" variables with small variances, values for each variable can be normalized with mean equal to 0 and variance equal to 1. Also, a much simpler technique is to normalize within the range from 0 to 1. Nevertheless, this approach is very sensitive to outliers.

Other distance functions such as Minkowski [9], Canberra [18] and Harmonically Summed [12] have been implemented but they showed no serious improvement in the obtained results. Also the correlation ranks such as Pearson's [15] and Kendall's [14] have been implemented. They offer a much improved robustness against outliers but the necessity of computing the correlations at each step represents a very important time cost.

4.5. Optimal number of clusters

The Calinski–Harabash pseudo-F (CHpF) [5] (see Eq. (7)) statistic has been implemented and can be used to obtain an estimate of the optimal number of clusters for each individual distribution.

$$\mathrm{CHpF} = \frac{\dfrac{SST - SSE}{k-1}}{\dfrac{SSE}{n-k}}, \tag{7}$$

where: n is the number of objects, k is the number of clusters, SST is the sum of squared distances from the clusters centroids to the overall data centroid, SSE is the sum of squared distances from the objects to their respective clusters' centroids.

One way to obtain the optimal number of clusters is to run a hierarchical algorithm and compute the CHpF statistic at each step. Once the optimal CHpF statistic is obtained, a partitioning method can be run to optimize the distribution at the appropriate level in the hierarchy.

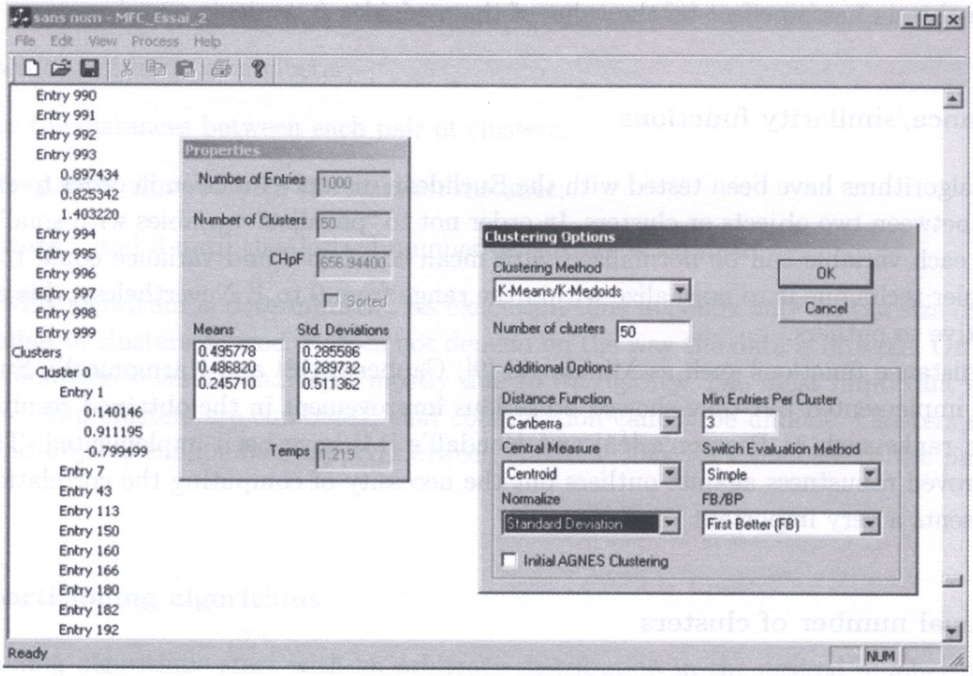
In the tests described in the following section this technique has not been used systematically, since it would have been necessary to perform it for each individual data set and the time required to perform the necessary operations would have been too important. It was only used to roughly estimate a global compromise between the compression ratio and the optimal number of clusters for all the data sets.

5. Validation results

Three different sets of tests have been performed. The first one only shows the time reduction when either one of the three algorithms are used. The second one shows the robustness of the GAs and clustering algorithms in terms of noise and the third one in terms of outliers. In all of these tests a theoretical 3D surface has been used (see Eq. (8)).

$$f(x,y) = 3x^2y - y^3. \tag{8}$$

For the purpose of the studies presented in this section a clustering module was developed by the authors and used to obtain all the results. Figure 5 shows a screen printout of the aforementioned module.



Entry 990
Entry 991
Entry 992
Entry 993
0.897434
0.825342
1.403220
Entry 994
Entry 995
Entry 996
Entry 997
Entry 998
Entry 999
Clusters
Cluster 0
Entry 1
0.140146
0.911195
-0.799499
Entry 7
Entry 43
Entry 113
Entry 150
Entry 160
Entry 166
Entry 180
Entry 182
Entry 192

Number of Entries: 1000
Number of Clusters: 50
CHpF: 656.94400
□ Sorted

Means	Std. Deviations
0.495778	0.285586
0.486820	0.289732
0.245710	0.511362

Temps: 1.219

Clustering Method: K-Means/K-Medoids
Number of clusters: 50
Additional Options
Distance Function: Canberra
Min Entries Per Cluster: 3
Central Measure: Centroid
Switch Evaluation Method: Simple
Normalize: Standard Deviation
FB/BP: First Better (FB)
□ Initial AGNES Clustering

Fig. 5. Graphic interface of the clustering module

Numerous data sets with variable Gaussian noise and outlier probabilities have been generated randomly. Outliers have been generated by replacing a random object with a random value in the interval of two times the total range of the data set. Data sets for each parameter combination have been generated ten times and the clustering and learning processes have been run separately. The final results presented in this paper are means (μ) of each set of ten runs. Standard deviations (σ) are also included in the tables to show the stability (or the lack of it) of the different algorithms.

The size of the initial data sets was chosen in order to obtain reasonably pronounced execution times. Data sets of 1000 objects gave palpable results. As for the size of the compressed data sets, it was set to 50 (see Sec. 4.5). The performance of each FKB is obtained by the method described in section 3.6 except that the data$_{\text{output}}$ in Eq. (5) is replaced by the actual theoretical values consisting of a uniformly distributed set of 1000 points. Figures 6 and 7 present 3D and grayscale map representations of the theoretical data set and of the data set obtained for Gaussian noise with 0.1 standard deviation and 20% of outlier probability. The outliers in Fig. 7 have been isolated from the rest of the learning data and represented as crosses in order to highlight their large quantity.

It is noteworthy that:

- the RBCGAs have been run with $p_c = 0.92$, $p_r = 0.08$ and $p_m = 0.02$ (see Secs. 3.2, 3.3 and 3.4);
- the tests have been performed on a Pentium IV 3.00 GHz PC.

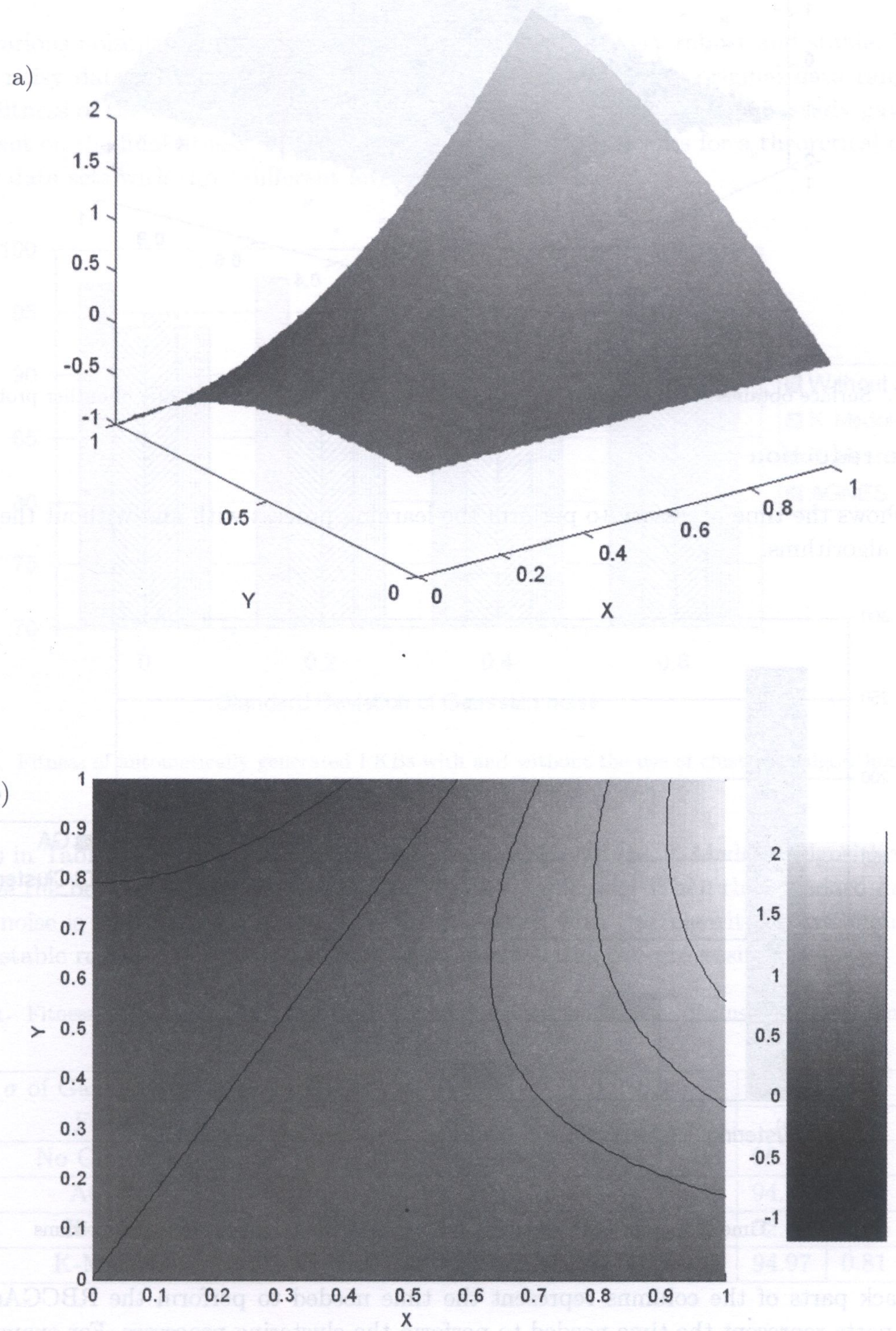


Fig. 6. a) Theoretical surface in 3D form, b) theoretical surface in grayscale map form

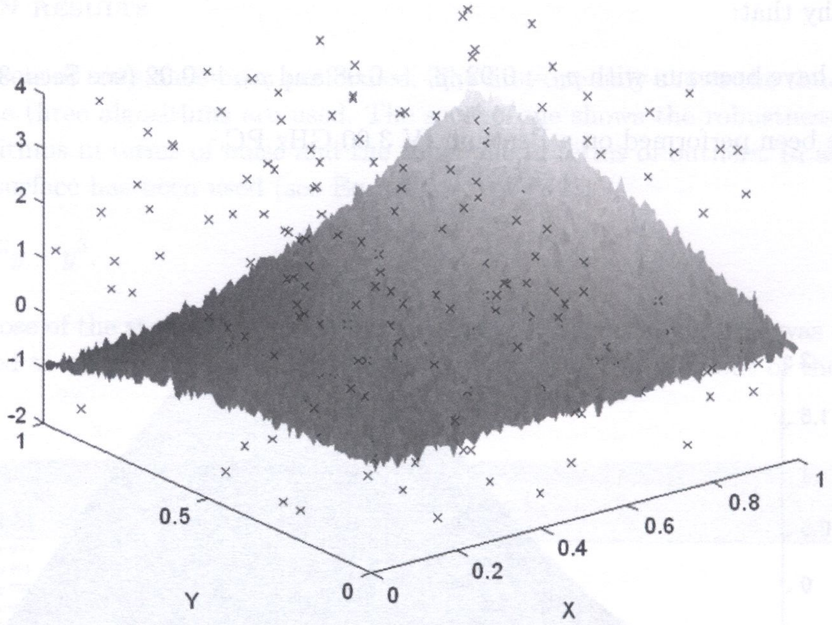


Fig. 7. Surface obtained with Gaussian noise with 0.1 standard deviation and 20% of outlier probability

5.1. Time reduction

Figure 8 shows the time necessary to perform the learning process with and without the aid of the clustering algorithms.

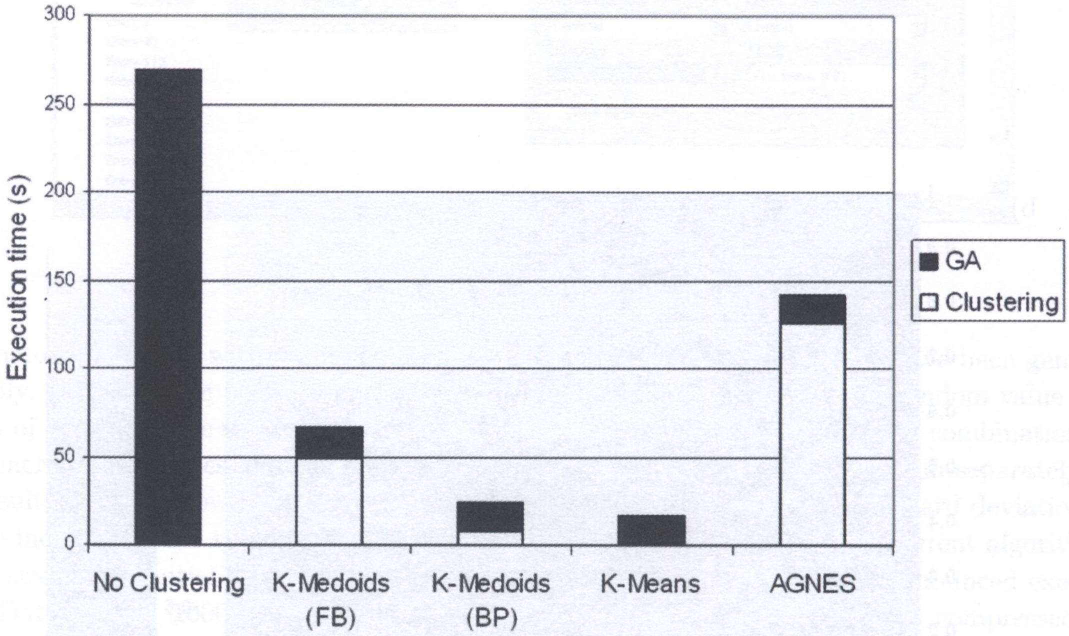


Fig. 8. Time reduction of the learning process with the use of clustering algorithms

The black parts of the columns represent the time needed to perform the RBCGAs, whereas the white parts represent the time needed to perform the clustering processes. For example, in the case of the KMeans algorithm a 94% time gain is obtained. The other algorithms offer time gains of 47% (AGNES), 75% (K-Medoids FB) and 91% (K-Medoids BP). The longer execution time for the FB version over the BP version of the K-Medoids algorithm is explained by the fact that more exploration steps must be performed in order to achieve the final distribution. The results presented in the rest of the paper are obtained with the FB version which in most of the cases gave slightly better results.

5.2. Noise influence

Tests on various noise levels have shown that the RBCGAs are very robust and stable. Even tests with very noisy data with standard deviation equal to double of the original data range lead to over 90% fitness of the obtained FKBs. No clustering algorithms tested in this study gave a global improvement on the final fitness. Figure 9 and Table 1 show the results for a theoretical data set as well as for data sets with three different levels of Gaussian noise.

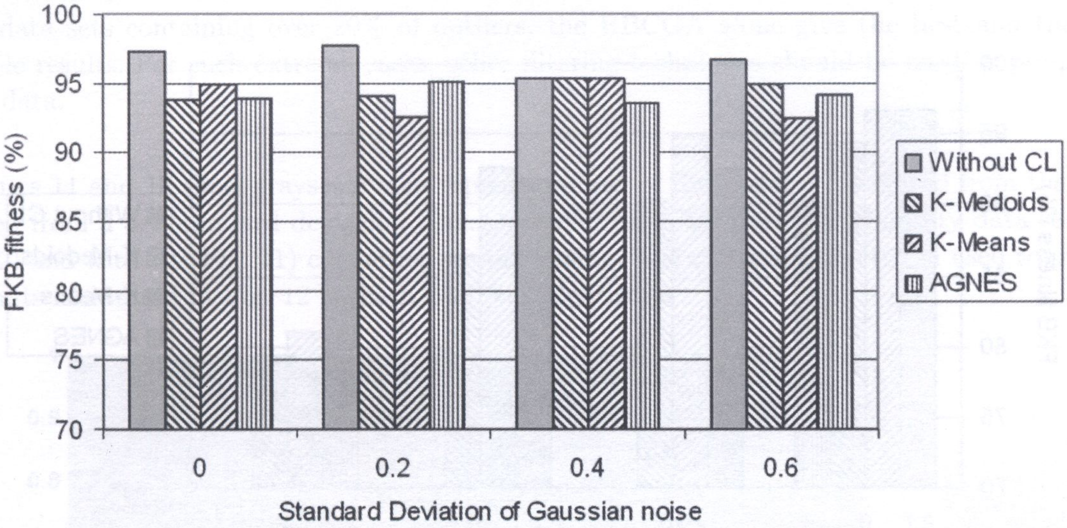


Fig. 9. Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of noise

Results in Table 1 show that in case of noisy data, although the K-Medoids algorithm does not always give the best results fitness-wise, it is the most stable one. When the standard deviation of Gaussian noise is larger than 0.4, data sets pre-processed with this algorithm give slightly less fit but more stable results than those obtained with no clustering pre-processing.

Table 1. Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of noise

σ of Gaussian noise	0.0		0.2		0.4		0.6	
Fitness	μ	σ	μ	σ	μ	σ	μ	σ
No Clustering	97.39	0.31	97.67	0.63	95.31	1.30	96.71	1.46
AGNES	93.92	2.88	95.15	1.76	93.55	2.32	94.25	3.01
K-Means	94.97	1.99	92.51	5.05	95.32	2.17	92.47	3.75
K-Medoids	93.83	1.72	94.05	1.17	95.36	1.31	94.97	0.81

It is noteworthy that the standard deviations for a given parameter combination are influenced not by the lack of consistency of the genetic algorithms but by the fact that each data set is composed of different data samples. Several tests on the same data set actually give close to zero standard deviation on the obtained fitnesses.

5.3. Outliers influence

At Sec. 5.2 it has been shown that the clustering algorithms offer hardly any improvement on the fitness of the obtained FKBs in the case of noisy data. However, in the case of outlier probabilities, the K-Medoids algorithm allows the RBCGA to produce FKBs with approximately 95% fitness for data sets with between 10% and 15% of outlier probabilities. In order to "absorb" the outliers in clusters containing other objects, a minimum number of objects per cluster can be forced in the case of the partitioning algorithms. This avoids outliers forming individual clusters but also makes the algorithm more constrained. Figure 10 and Table 2 present the fitnesses of FKBs obtained from data sets containing various amounts of outliers and 0.1 standard deviation Gaussian noise.

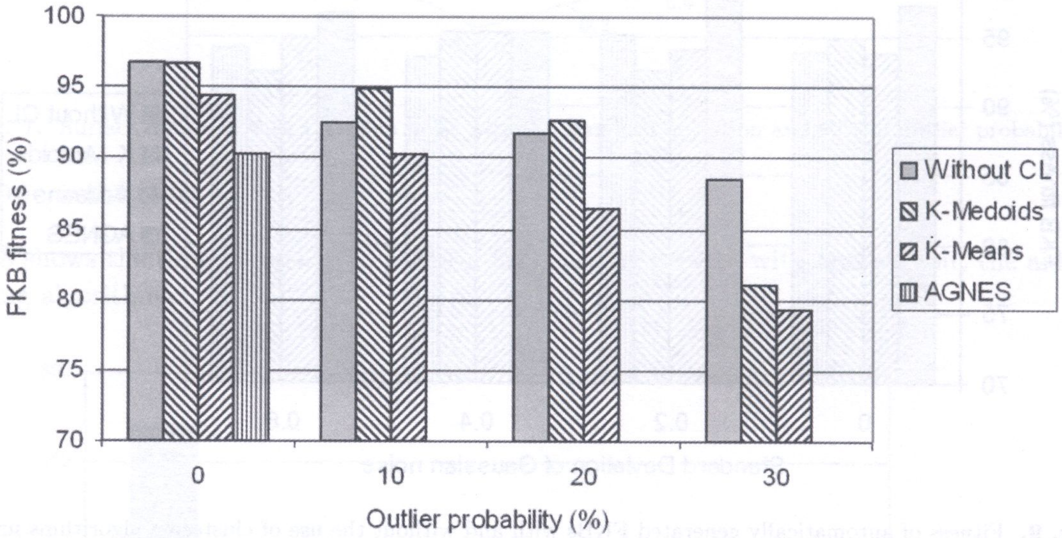


Fig. 10. Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of outliers

Table 2. Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of outliers

Outliers	0%		10%		20%		30%	
Fitness	μ	σ	μ	σ	μ	σ	μ	σ
No Clustering	96.78	0.82	92.48	1.46	91.80	0.69	88.56	1.68
AGNES	90.35	8.21	57.67	3.60	41.40	20.26	42.69	10.34
K-Means	94.40	2.67	90.32	4.78	86.45	2.95	79.30	4.72
K-Medoids	96.59	0.66	94.88	1.11	92.60	0.65	81.07	6.25

Several conclusions can be drawn from Fig. 10 and Table 2:

- without any clustering pre-processing, the fitness declines proportionally to the amount of outliers. It falls below 90% for data sets with between 20% and 30% outlier probabilities;

- the fitness of the FKBs for data sets with up to 20% outlier probability pre-processed with the K-Medoids algorithm is improved by 2.4% for 10% of outliers and by 0.8% for 20% of outliers. It declines abruptly when more than 20% of outliers are present in the learning data, because the algorithm starts forming clusters around outliers exclusively, thus increasing their relative importance;

- until the aforementioned breakdown point of 20% of outliers, the K-Medoids algorithm also gives the most stable results;

- the K-Means algorithm leads to a considerable fitness loss but with a regular behaviour;

- the AGNES algorithm is totally unstable and inapplicable for data sets containing outliers (the columns are not visible in Fig. 10, because the fitness of the generated FKBs falls under 70%);

- for data sets containing over 20% of outliers, the RBCGA alone give the best and the most stable results. For such extreme cases, other filtering techniques should be used to pre-process the data.

Figures 11 and 12 show grayscale map representations of the surfaces generated from the FKBs obtained from a 0.1 standard deviation Gaussian noise and 20% outlier probability data set, with (Fig. 12) and without (Fig. 11) clustering pre-processing. The clustering algorithm used to produce the results illustrated in Fig. 12 is the K-Medoids algorithm.

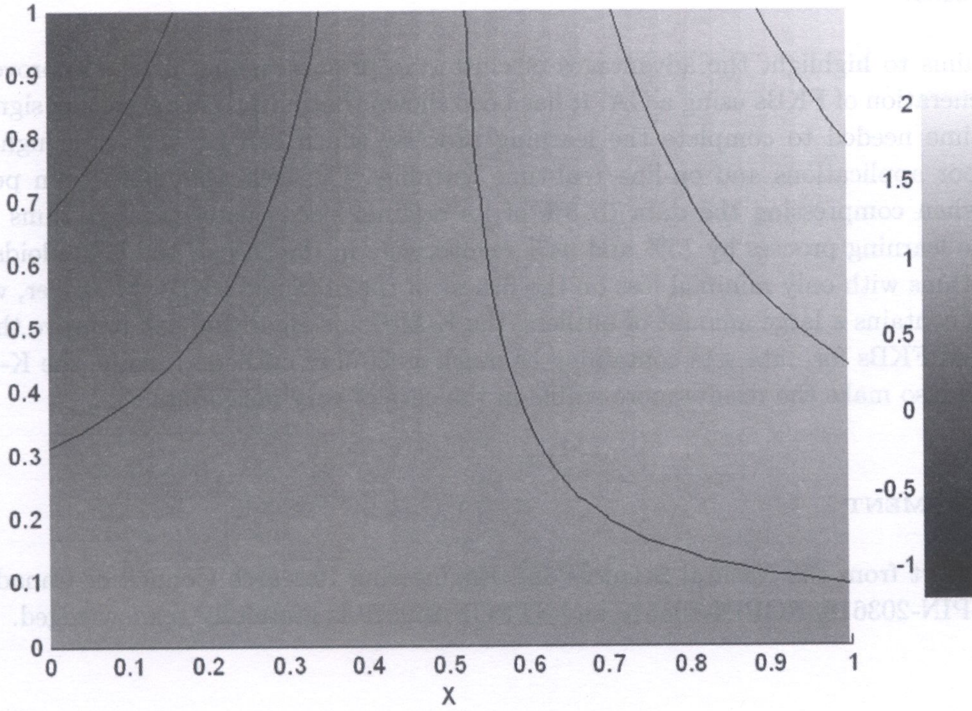


Fig. 11. Surface generated from the FKB obtained without clustering pre-processing from a data set with Gaussian noise with 0.1 standard deviation and 20% of outlier probability

The fitness of the FKB illustrated in Fig. 12 is 1% stronger than that of the FKB illustrated on Fig. 11. These figures can also be compared with Fig. 6 b) in order to see how Fig. 12 approaches the theoretical results more accurately.

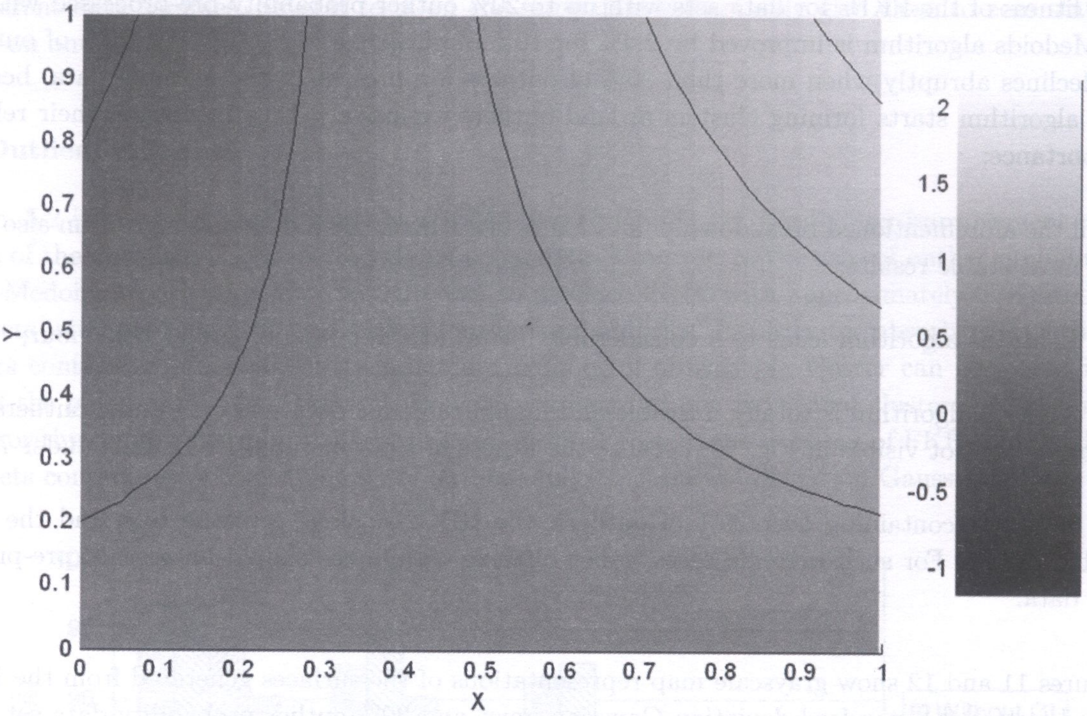


Fig. 12. Surface generated from the FKB obtained with clustering pre-processing from a data set with Gaussian noise with 0.1 standard deviation and 20% of outlier probability

6. Conclusion

This paper aims to highlight the advantages of clustering pre-processing for the improvement of automatic generation of FKBs using a GA. It has been shown that initial clustering can significantly reduce the time needed to complete the learning process, which can be of a very high interest in factory floor applications and on-line/real-time learning. The tests that have been performed show that, when compressing the data to 5% of its original size, clustering algorithms allow to accelerate the learning process by 75% and 94% respectively in the case of the K-Medoids and K-Means algorithms with only minimal loss on the fitness of the obtained FKBs. Moreover, when the learning data contains a large amount of outliers, the K-Medoids algorithm can improve the fitness of the obtained FKBs for data sets containing as much as 20% of outliers. Finally, the K-Medoids algorithm can also make the results more stable in the case of very noisy data sets.

Acknowledgment

Financial support from the Natural Sciences and Engineering Research Council of Canada under grants of RGPIN-203618, RGPIN-105518 and STPGP-269579 is gratefully acknowledged.

References

[1] S. Achiche, M. Balazinski, L. Baron. Real/binary-like coded genetic algorithm to automatically generate fuzzy knowledge bases. *The 4-th International Conference on Control and Automation*, June 2003.

[2] S. Achiche, M. Balazinski, L. Baron. Multi-combinative strategy to avoid premature convergence in genetically-generated fuzzy knowledge Bases. *Journal of Theoretical and Applied Mechanics*, **42**(3): 417–444, 2004.

[3] M. Balazinski, M. Bellerose, E. Czogala. Application of fuzzy logic techniques to the selection of cutting parameters in machining processes. *International Journal for Fuzzy Sets and Systems*, **61**: 307–317, 1993.

[4] L. Baron., S. Achiche, M. Balazinski. Fuzzy decisions system knowledge base generation using a genetic algorithm. *International Journal of Approximate Reasoning*, pp. 25–148, 2001.

[5] T. Calinski, J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, **3**: 1–27, 1974.

[6] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**: 182–200, 2000.

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

[8] J. Han, M. Kambler. *Data Mining Concepts and Techniques.* Morgan Kaufmann Publishers, 2001.

[9] H. Hancock. *Development of the Minkowski Geometry of Numbers.* the Macmillan Company, New York, 1939.

[10] J.A. Hartigan. *Clustering Algorthms.* 1975.

[11] F. Herrera, M. Lozano. *Gradual distributed real-coded genetic algorithms.* IEEE Transactions on Evolutionary Computation, **4**, pp. 43–63, 2000.

[12] M. De Hoon, S. Imoto, S. Miyano. *The C Clustering Library.* Human Genome Center, University of Tokyo, 2004.

[13] L. Kauffman, P.J. Roussewuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons, 1990.

[14] M. G. Kendall. *Rank Correlation Methods.* Griffin, 1962.

[15] E. L. Lehmann, H. J. M. D'abrera. *Nonparametric: Statistical Methods Based on Ranks.* Holden-Day, 1975.

[16] F. G. Lobo, D. E. Goldberg, M. Pelikan. Time complexity of genetic algorithms exponentially scaled problems. *GECCO 2000: Proceedings of the Genetic and Evolutionary Computation Conference,* pp. 151–158, 2000.

[17] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* **1**: 281–297, 1967.

[18] K. V. Mardia. J. T. Kent, J. M. Bibby, *Multivariate Analysis.* Academic Press, London, 1979.

[19] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs.* Springer, New York, 1992.

[20] *Wikipedia, the Free Encyclopedia.* http://www.wikipedia.org/. Consulted on January 27-th 2005.

[21] L.A. Zadeh. *Outline of new approach to the analysis of complex systems and decisions processes.* IEEE Transactions of Systems, Man and Cybernetics, **3**: 28–44, 1973.