Efficiency Improvement of DC* through a Genetic Guidance

Ciro Castiello
and Corrado Mencar
Dept. of Informatics
University of Bari "A. Moro"
ciro.castiello@uniba.it
corrado.mencar@uniba.it

Marco Lucarelli
SITAEL S.p.A.
Mola di Bari, Bari, Italy
marco.lucarelli@sitael.com

Franz Rothlauf
Information Systems & Business Administration
Johannes Gutenberg-Universität Mainz
rothlauf@uni-mainz.de

***Abstract*—DC* is a method for generating interpretable fuzzy information granules from pre-classified data. It is based on the subsequent application of LVQ1 for data compression and an *ad-hoc* procedure based on A* to represent data with the minimum number of fuzzy information granules satisfying some interpretability constraints. While being efficient in tackling several problems, the A* procedure included in DC* may happen to require a long computation time because the A* algorithm has exponential time complexity in the worst case. In this paper, we approach the problem of driving the search process of A* by suggesting a close-to-optimal solution that is produced through a Genetic Algorithm (GA). Experimental evaluations show that, by driving the A* algorithm embodied in DC* with a GA solution, the time required to perform data granulation can be reduced by at least 45% and up to 99%.**

I. Introduction

Information granulation is a crucial task for data analysis, since information granules highlight high-level properties and relationships about an universe of discourse, while hiding useless low-level details pertinent to single data [1]. Once formed, information granules help to understand hidden relationships among data and to capture the perceptive character of concepts in the way they are conceived by human beings. In this sense, fuzzy set theory stands as a convenient modeling framework for representing and processing these pieces of information. and fuzzy granules can be easily understood by humans, especially when they are described in natural language [2], [3]. When employed for describing a dataset, fuzzy information granules are often defined in terms of one-dimensional fuzzy sets defined on each input feature. This involves the granulation of each feature into a fuzzy partition, which should satisfy a number of interpretability constraints and criteria [4], [5]. In particular, the fuzzy sets for each partition should be as few as possible, provided that they describe data in an accurate way.

Interpretability and accuracy are conflicting requirements in fuzzy modeling. The ultimate aim should be to exploit all possible degrees of freedom in a fuzzy model to adapt to data while satisfying interpretability constraints. A possible approach in this direction is to profit of variable granularity of the fuzzy sets involved in each feature. This possibility is rarely exploited by the available techniques for interpretable data granulation, which usually generate uniform partitions for each feature.

DC* is a method for generating interpretable fuzzy partitions with variable granularity. Each fuzzy set of the partition is defined to conform to available data and, at the same time, to satisfy a number of interpretability constraints. The number of fuzzy sets for each partition is optimized so as to be as small as possible (possibly, just one fuzzy set is identified for a feature, which is eventually removed). To carry out data granulation, DC* requires the user only to specify an upper bound for the number of information granules to be derived from data.

DC* [6], [7] is specifically designed for pre-classified data and performs a two-step clustering process firstly involving the LVQ1 algorithm and then a search procedure based on A*. While being efficient in tackling several problems, the A* procedure included in DC* may happen to require a long computation time because the A* algorithm has exponential time complexity in the worst case. Several extensions could be implemented in order to improve the average execution time of A* [8], but many of them could lead to miss the optimality of the solution that is guaranteed by A*. In this paper, we approach the problem of driving the A* search process by suggesting a sub-optimal solution that is deemed close to an optimal solution. The A* algorithm operates as usual but, in case of candidate solutions sharing the same priority level, the one which is closest to the suggested solution is explored first. In this way, the optimality property of A* is preserved while speeding up the search process.

The sub-optimal solution used for driving A* is produced through a Genetic Algorithm (GA) implemented to solve the same search problem by means of a fast converging evolutionary process. In this way, a solution can be quickly found, though without any guarantee of optimality. Nevertheless, because of the peculiar optimization process performed by the GA, the resulting solution is usually close to optimal. Experimental evaluations show that, by driving the A* algorithm embodied in DC* with a GA solution, the time required to perform data granulation can be reduced by at least 45% up to 99%.

In the next Section, DC* is briefly outlined, while the

978-1-5090-6034-4/17/$31.00 ©2017 IEEE

genetic guidance method is described in more detail in Section III. Experimental results are discussed in Section IV.

II. Model

DC* is the acronym for Double Clustering with A*: it is a method to be applied in classification problems, which produces a partition of the multi-dimensional space hosting the samples of some given dataset. In particular, DC* is able to produce fuzzy partitions characterized by variable granularity. The obtained fuzzy granules are useful to derive an interpretable knowledge base that can be readily expressed in form of natural language.

To achieve this aim, DC* is organized in subsequent steps:

1) a data compression step, generating a number of prototypes from data;
2) a one-dimensional clustering step, producing on each data dimension the prototype projections together with their suitable clustering, in order to provide the structure of the fuzzy partition;
3) a fuzzy granulation step, addressing the task to identify the fuzzy information granules to be expressed in linguistic form.

DC* has been thoroughly presented in [6], [7]. In order to supply the reader with the basic elements to understand the working principle of DC*, we describe in the following paragraphs the main stages of the overall clustering process.

A. Data compression

We assume the availability of a dataset $\mathbf{D} \subset \mathbb{R}^n$ composed by n_D samples descibed by n features and properly tagged with some specific class labels. The aim of the data compression step consists in generating a number n_P of prototypes in order to provide a synthetic description of $\mathbf{D}$. Particularly, every prototype should be representative of some other data, determining clusters of samples characterized by the same class labels. The number of prototypes to be generated is a user-defined value, being also the single parameter requested to set-up the run of DC*. This value has a crucial role in the overall process leading to the compilation of the knowledge base, since it represents the upper bound for the number of fuzzy rules included in the final predictive model. To put in practice the data compression step, the LVQ1 algorithm [9] is exploited, which proves to be useful in analyzing the dataset contents and deriving a suitable collection of well-separated labeled prototypes.

B. One-dimensional clustering

The n_P prototypes, generated from data compression, are projected on each dimension $d = 1, 2, \ldots, n$ of the feature space. In this way, n sequences of prototype projections are identified: $P_d = [p_d^1, \ldots, p_d^{n_P}]$ (with $d = 1, \ldots, n$), and each sequence includes an ordered list of points laying on each axis of the multidimensional space ($p_d^i \leq p_d^{i+1}$). Each projected point is associated with a class label, corresponding to the label characterizing the prototype from which the point is derived. The one-dimensional clustering operates simultaneously along every dimension of the space to identify groups of projected points. Specifically, an ordered set of *cuts* is identified for each dimension, which are defined by midpoints of any couple of elements (p_d^i, p_d^{i+1}) corresponding to different class labels. All the points laying between two adjacent cuts (or between a cut and the data boundary) are grouped in a single cluster. Thus, any subset S_d of cuts corresponds to a clustering of projections on dimension d.

When considering the cut sequences drawn on every dimension, a configuration of cut sequences $\sigma = (S_1, S_2, \ldots, S_n)$ is assembled, which splits the multidimensional space into a number of hyper-boxes. We are interested in assessing such a space granulation in terms of the prototypes contained in every hyper-box: *pure* hyper-boxes are those containing prototypes with equal class label. (Hyper-boxes with no prototypes within are also pure.) On the other hand, we are also willing to deal with a reduced number of cuts, so that the overall space granulation may be coarse enough to make data description as simple as possible. Therefore, the aim of this clustering step is to seek for an *optimal* configuration of cut sequences σ^* producing pure hyper-boxes with the minimum possible number of cuts[1].

In order to derive σ^*, we employ the A* algorithm which proves to be effective in dealing with this kind of search process, but is characterized by an exponential computational complexity. This represent a major drawback, since the search must be performed over several cut sequences (nodes), starting from the initial node σ_0 with no cuts, corresponding to the multidimensional space considered as a whole, and proceeding with the analysis of successive nodes by adding a single cut to the previous configuration at each step. Since multiple choices can be made for drawing new cuts, each node gives rise to a queue of candidate subsequent nodes. Therefore, the search is oriented by a cost function which evaluates each node composing the current queue during the process. The cost function takes into account both the total number of cuts in each node and the hints coming from a heuristic properly designed for the problem at hand. In particular, the heuristic exploits class information: starting from a given node, the aim is to identify the minimal number of necessary (possibly, not sufficient) cuts to be added to the current node in order to achieve a solution. In this way, the nodes in the queue are ranked according to the cost function evaluation; as a tie-breaker in case of equal cost values, further priority evaluations can be considered (which favor the production of a final fuzzy partition characterized by coarser information granules and fewer input features).

C. Fuzzy granulation and knowledge base compilation

As a result of the previously described clustering process, an optimal configuration of cut sequences $\sigma^* = (S_1^*, S_2^*, \ldots, S_n^*)$ is derived. The fuzzy granulation step is devoted to exploit this piece of information to produce a granulation of the

[1]It may be the case that several optimal configurations exist: the search is satisfied by anyone of them.

multidimensional space in terms of fuzzy sets. To this aim, we adopted a specific technique called Variable Fuzziness (VF) which operates over each dimension $d = 1, \ldots, n$ on the basis of the cuts included in corresponding sequence S_d^* [10]. The VF technique is characterized by a number of key points:

- the d-th feature is excluded if $S_d^* = \emptyset$;
- the fuzzy partitions derived on every dimension are Strong Fuzzy Partitions (SFPs);
- the fuzzy sets defining each SFP are trapezoidal in their shape.

At the end of the fuzzy granulation stage, a predictive model is built up on the basis of the hyper-boxes produced by σ^* and the fuzzy sets defined by the VF technique.

Particularly, each dimension d is associated with a linguistic variable x_d, and each sequence of trapezoidal fuzzy sets defined on d is associated with a set L_d of linguistic terms. In this way, each hyper-box corresponds to an information granule described in terms of linguistic variables and linguistic terms addressing a single class c (uniqueness of class is due to the occurrence of pure hyper-boxes only in the partition produced by σ^*). Such a piece of information is expressed in form of a IF-THEN rule:

$$\text{IF } x_1 \text{ is } A_1 \text{ AND } \cdots \text{ AND } x_n \text{ is } A_n \text{ THEN } c$$

where each linguistic variable x_d is related to one of the linguistic terms included in the corresponding set L_d (i.e., $A_d \in L_d$), and the AND connective is implemented by a specific t-norm.The ensemble of hyper-boxes produced by σ^* is described by several IF-THEN rules which altogether compose the fuzzy rule base of the predictive model generated from data.

III. A GENETIC GUIDE FOR DC*

As previously observed, computational complexity is a crucial issue for the overall evaluation of DC*. In this section we are going to discuss how the efficiency of DC* can be enhanced by resorting to a properly designed Genetic Algorithm (GA). The role of the GA is to compensate for the possible lack of information of the heuristic in discriminating among the different nodes to be evaluated. It is quite common, in fact, that several nodes are equally assessed in terms of cost values during the search process: this greatly expands the front of investigation explored by the A* algorithm. The idea, therefore, is to supply the search mechanism with a suitable guidance in order to accelerate the convergence of DC* to an optimal solution.

To achieve this aim, we launch the GA prior to the execution of the A* algorithm. The GA is intended to produce a particular point in the search space, namely a configuration of cut sequences σ^{GA}, standing as a steering reference during the subsequent search process. In order to explain in which sense a node can be useful to pilot the A* investigation, we should recall that any node σ is visited only moving from some previous nodes including the same cuts of σ except for one. Therefore, any search path determines a chain of correlated nodes, that are those which specialize the overall space partition by adding

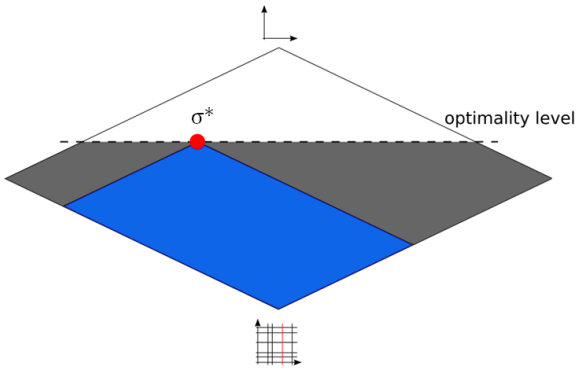

Figure 1. An example of search space with the location of the optimal solution σ^*.

one additional cut at each depth level of search. The GA can prove to be useful if σ^{GA} belongs to a search path where σ^* is also included (possibly at some previous depth level). In other words, the inclusion $S_d^{\sigma^*} \subseteq S_d^{\sigma^{GA}}$ should be attained for any d. As a trivial example, the final node corresponding to a configuration including every possible cuts belongs to the search path including σ^* (as well as any other node).

In Figure 1 a search space is represented moving from the initial node σ_0 on the top, down to the final node. The location of the optimal configuration σ^* determines the optimal depth level of search: the colored area under σ^* is the place for all the subsequent nodes that can be obtained by further specializing the optimal solution. To provide a significant guidance to the search process, σ^{GA} should find its placement within that area. Moreover, the steering contribute of σ^{GA} would be enhanced if it were placed closer to σ^*. We measure the dissimilarity between two nodes $\sigma^{'}, \sigma^{''}$ on the basis of the Jaccard similarity index:

$$\delta(\sigma', \sigma'') = 1 - \frac{\sum_{i=1}^{n} |S_i' \cap S_i''|}{\sum_{i=1}^{n} |S_i' \cup S_i''|} \in [0, 1], \qquad (1)$$

thus assuming that closeness between a couple of nodes increases with the number of cuts those nodes are sharing. In this way, once the sub-optimal solution provided by the GA is available, the previous distance measure can be applied to each node visited during the search process, to evaluate its distance from σ^{GA}: nodes characterized by a reduced distance value will be preferred. However, in order to ensure the optimality of the overall search process, such a priority ranking is applied after the node evaluation in terms of cost function (described in the previous section).

Concerning the implementation of the GA, the solutions are binary encoded. Each gene in the chromosome stands for a cut and the "0", "1" values indicate the absence and the presence of that cut in a given sequence, respectively. Therefore, if we consider a specific dimension d, it is possible to write down a binary vector $\mathbf{v}_d$ whose elements correspond to all the cuts that can be drawn on d: $\mathbf{v}_d = (v_{1d}, v_{2d}, \ldots, v_{\tau_d d})$, being τ_d the total number of possible cuts insisting on d. Obviously the sequence of cuts S_d, which is part of a generic configuration

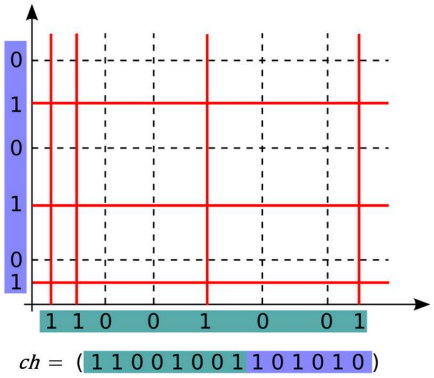

Figure 2. A sample configuration of cut sequences (in a bi-dimensional space) and its chromosomic representation.

of cut sequences $\sigma = (S_1, \ldots, S_n)$, can be easily extrapolated from $\mathbf{v}_d$ by considering only the cuts corresponding to the elements $v_{kd} = 1$ (for $k = 1, \ldots, \tau_d$). The populations evolving through the generations of the GA are composed by individuals that are full configurations of cut sequences, therefore each generic individual's chromosome must include all dimensions:

$$ch = (v_{11}, \ldots, v_{\tau_1 1}, v_{12}, \ldots, v_{\tau_2 2}, \ldots, v_{1n}, \ldots, v_{\tau_n n}) \quad (2)$$

In Figure 2 we give an example of how the different cuts are encoded in a chromosome.

The GA works on the basis of a fitness function designed while keeping in mind the characteristics of the optimal configuration σ^*. Since σ^* must include the minimum number of cuts producing pure hyper-boxes, the fitness function $f(\cdot)$ is composed by two elements. A granularity measure $g(\cdot)$ is computed to express the ratio between the number of cuts involved in the individual's representation and the total number of all the admissible cuts in the space:

$$g(ch) = \frac{\sum_{d=1}^{n} \sum_{k=1}^{\tau_d} v_{kd}}{\sum_{d=1}^{n} \tau_d}.$$

A pureness measure $p(\cdot)$ is computed to express the ratio between the number of pure hyper-boxes involved in the partition produced by the individual and the total number of hyper-boxes:

$$p(ch) = \frac{P(ch)}{H(ch)},$$

being $P(\cdot)$ and $H(\cdot)$ a couple of supporting functions: given a specific configuration of cut sequences, they measure the number of pure hyper-boxes and the total number of hyper-boxes, respectively.

In order to assign greater relevance to the pureness factor, the final form of the fitness function is defined as follows:

$$f(ch) = [1 - g(ch)][p(ch)]^5.$$

Tournament selection is applied to create the reproduction pool, the number of generation is set to 20, and the crossover and the mutation rate are set to 0.7 and $1/\sum_{d=1}^{n} \tau_d$, respectively. Both crossover and mutation operators are applied to reproduce individuals from one generation to another. Crossover rate is set to 0.7, while mutation rate is set to $\frac{1}{n_{ch}}$. Finally, the pre-fixed population cardinality is equal to twice the length of the chromosome: $2\sum_{d=1}^{n} \tau_d$.

Table I
DESCRIPTION OF THE DATASETS EMPLOYED IN THE EXPERIMENTAL SESSION.

Dataset	samples	features	classes	abbrev.
Glass Identification	214	9	6	GLASS
Ionosphere	351	33	2	IONO
Iris	150	4	3	IRIS
Statlog-Shuttle	58000	9	7	SHUTTLE
Vertebral Column (3 cl.)	310	6	3	VERT3
Wine	178	13	3	WINE
Wisconsin Breast Cancer	683	9	2	WBC

IV. EXPERIMENTAL RESULTS

In this Section we carry on a thorough evaluation of the DC* method putting a major emphasis on the efficacy of the genetic algorithm at improving the convergence speed of the overall algorithm.

A. Experimental setup

We are interested in verifying if the GA-guided A* approach effectively realizes an improvement of the convergence speed of the A* algorithm in DC*. Such an evaluation has been conducted by comparing the computational behavior of the two approaches (i.e. either with or without genetic guidance) while tackling the same clustering tasks. As an additional term of comparison, it is possible also to refer to the computational performance of an approach which is *totally* based on the employment of the genetic algorithm (similarly to an early approach of ours [11]). We are going to exhibit, in fact, the experimental evidences specifically related to the genetic search producing σ^{GA}, which can be considered as an alternative stand-alone solution for the clustering problem at hand. However, it should be highlighted that in general the genetic search is not able to ensure the production of an optimal solution in terms of the number of cuts.

In addition, we are interested in assessing the performance of the proposed approach in terms of accuracy and interpretability. We compared DC* with Hierarchical Fuzzy Partitioning (HFP) algorithm [12]. For our experiments, we chose datasets from the UCI Machine Learning Repository[2] in order to encompass a number of heterogeneous benchmark classification problems. A synthetic description of the employed datasets is reported in Table I. The datasets have been adopted in their original form, except for Ionosphere (where a feature has been removed since it exhibits a constant value) and Glass Identification (where a class has been removed since it is not represented by any sample).

[2]http://archive.ics.uci.edu/ml/

B. Performance evaluation

The first experimental analysis studies the efficiency gain deriving from the adoption of the GA-guided A* approach during the one-dimensional clustering stage. In this sense, our first concern is to compare the computational performances of the original implementation of DC* and the genetic version illustrated in Section III.

In practice, the previously described datasets have been considered and, for each of them, the data compression stage has been carried out. To set up a fair comparison, both the GA-guided approach and the original A* search process have been performed on the basis of the same arrangement of prototypes. In this way, at the end of the clustering stage we obtained, for each classification task, a couple of final solutions (i.e. configurations of sub-sequences of cuts) that are always optimal but may be different with regards to the specific cuts placement. That is due to the different structure of the priority queues considered by the compared approaches: the GA-guided A* approach is inclined in its search towards nodes which are closer to σ^{GA}, while the original A* search tends to privilege cuts that are mainly spaced apart. Additionally, we can consider also the performance of the genetic algorithm only, which ultimately derives the solution coinciding with σ^{GA} (whose optimality, however, is not ensured).

The comparison of the different approaches has been firstly performed on the basis of the computational time (we can refer to the execution time of both the genetic algorithm only and the complete GA-guided A* approach). Moreover, a further comparison can be assessed on the basis of the number of expanded nodes during the exploration of the search space (in this case we obviously refer to the A* search performed on the basis of σ^{GA} identification). The results are summarized in Table II. In order to face the stochastic nature of the genetic computation, the values reported for the GA-guided A* approach are average results of ten different runs of the proposed algorithm on each dataset (the initial prototype configurations have been kept unchanged, so the original A* search is unaffected by such a strategy).

The results reported in Table II show how the adoption of the GA-guided approach is able to produce a relevant gain both in the computational time and in the number of expanded nodes. That lets us maintain that the σ^{GA} identified by the genetic algorithm is an effective way to guide the A* search process. It is noteworthy to highlight also that the final solution derived by the GA-guided A* approach, as expected, is optimal being composed by the same number of cuts included into the final configuration identified by the A* search (such a value is reported in Table II, described as the "final" number of cuts which is common for both the implemented approaches). Moreover, the σ^{GA} always represents a sub-optimal solution, being composed by a greater number of cuts, and its definition is quite stable through the ten runs of the genetic algorithm performed for each dataset (as witnessed by the reported values of standard deviation).

The last row in Table II indicates the dissimilarity $\delta(\sigma^{GA}, \sigma^*)$ between the σ^{GA} and the final optimal solution. The reported values show that the GA guidance is effective in speeding up the A* search even though the final solution is very different from σ^{GA}. An exception is observed in the results related to VERT3 dataset, which is associated also to the best performance gains both in terms of computational time and expanded nodes.

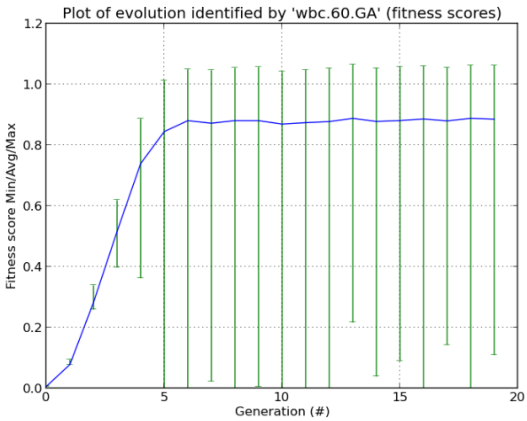

Figure 3. Evolution of the fitness function for the WBC dataset. After six generations, the fitness score converged to the maximum value.

As previously asserted, preliminary experiments with the genetic algorithm indicated that the involved populations are able to reach a quite mature evolution in few generations. We therefore imposed a limit of 20 generations during this session of performance evaluation and it appears to be a suitable choice, as testified by the fitness function trends depicted in Figure 3 for one of the datasets (WBC).

C. Comparative evaluation

When we turn to consider the experimental session purely devoted to the analysis of the obtained results, our objective is to evaluate the final outcomes of the DC* method in terms of both accuracy and interpretability. To this end, we compared the results of DC* with HFP which is assumed to be the most affine method for generating interpretable fuzzy partitions from data. In practice, we used the HFP implementation provided by the FisPro tool [13]. The obtained results are reported in Table III. Since HFP returns a Pareto front of models, we selected the model with best accuracy (HFP-A) and the model with the smallest number of rules (HFP-I). For each model, we recorded the accuracy, the number of features and the number of rules averaged over a stratified 10-fold cross-validation.

We observe that, in most cases, DC* can be ranked between the two selections of HFP in terms of both accuracy and average number of rules (the only exception being GLASS and VERT3, where HFP seems to fail in providing a model with satisfactory accuracy). The table shows that, on the average, DC* favors interpretability because the number of rules and involved features is closer to HFP-I than to HFP-A. This is

Table II
PERFORMANCE COMPARISON OF THE GA-GUIDED A* AND THE ORIGINAL A* APPROACHES IN TERMS OF COMPUTATIONAL TIME AND EXPANDED NODES.

		dataset (# prototypes)						
		GLASS (18)	IONO (20)	IRIS (42)	SHUTTLE (21)	VERT3 (24)	WINE (40)	WBC (60)
Computational time (in sec.)	A*	212.69	89.25	86.45	971.68	11,236.43	17,387.67	462.62
	GA-guided	30.24	48.21	24.13	116.43	23.78	2,256.82	149.84
	(GA only)	(6.93)	(27.32)	(5.07)	(8.45)	(5.72)	(27.69)	(21.11)
	performance gain	85.78%	45.98%	72.09%	88.02%	99.79%	87.02%	67.61%
Expanded nodes	A*	65,366	20,571	40,452	128,777	381,063	493,839	81,507
	GA-guided	10,261.9	4,719.7	13,751.7	24,817.7	134,33.8	76,334	24,081.7
	performance gain	84.3%	77.06%	66%	80.73%	96.47%	84.54%	70.45%
# cuts	σ^{GA} (stdev)	6.5 (0.85)	3 (0.47)	5.5 (0.85)	5.8 (0.42)	5.4 (0.70)	4.6 (0.84)	4.2 (0.92)
	final	4	2	4	4	4	3	3
solution distance	$\delta(\sigma^{GA}, \sigma^*)$ (stdev)	0.73 (0.12)	0.71 (0.23)	0.35 (0.22)	0.70 (0.21)	0.32 (0.18)	0.83 (0.09)	0.57 (0.40)

Table III
COMPARATIVE EVALUATION OF DC* AND HFP. FOR EACH DATASET, STRATIFIED 10-FOLD CV HAS BEEN EXECUTED. MEAN∓STD IS REPORTED.

Dataset	Method	err(%)	#features	#rules
GLASS	DC*	43.3±13.7	3.7±0.5	8.4±1.1
	HFP-A	51.0±15.52	5.6±0.9	15.4±5.3
	HFP-I	57.0±11.4	3.7±0.5	2.0±0
IONO	DC*	25.0±9.5	1.1±0.3	2.2±0.6
	HFP-A	16.0±6.2	5.0±3.0	300.0±637.0
	HFP-I	31.0±11.4	1.0±0	2.0±0
IRIS	DC*	8.0±5.8	1.5±0.5	3.2±0.6
	HFP-A	5.33±7.2	3.0±0	8.5±0.8
	HFP-I	33.0±14.0	1.0±0	2.0±0
SHUTTLE	DC*	19.6±2.7	4.0±0	10.0±1.2
	HFP-A	3.7±1.5	22.8±0.5	81.5±22.8
	HFP-I	21.4±0.4	1.0±0	2.0±0
VERT3	DC*	32.3±6.9	3.0±0.5	7.5±1.2
	HFP-A	43.0±8.6	2.1±0.3	4.6±1.3
	HFP-I	47.0±9.2	1.0±0	2.0±0
WBC	DC*	12.6±5.1	1.0±0	2.0±0
	HFP-A	4.9±2.1	24.9±0.8	34.7±18.0
	HFP-I	16.5±6.7	1.0±0	2.0±0
WINE	DC*	23.3±11.1	2.1±0.3	4.1±0.7
	HFP-A	7.2±3.7	5.4±1.1	39.6±20.3
	HFP-I	32.0±5.4	1.0±0	2.0±0

especially evident for datasets where HFP-A exhibits a very high accuracy but at the cost of a prohibitive number of rules.

V. CONCLUSION

Experimental results show that driving A* with solution computed by a GA is a winning strategy that significantly reduces the search time for data granulation with DC*. The approach is free of any encumbrance which may occur as a side-effect: even in the worst case the genetic guide does not slow down the search process of A*. The required overhead is due to the execution of GA, which however is a fast converging algorithm. In fact, there is no need to achieve a globally optimal solution since an "acceptable" sub-optimal solution is effective enough for driving the A* algorithm. The result is a fast generation of interpretable fuzzy information granules with adaptive granularity level, which can be used to generate fuzzy rule-based classifiers exhibiting a good tradeoff between accuracy and interpretability.

REFERENCES

[1] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy sets and systems*, vol. 90, no. 2, pp. 111–127, 1997.

[2] A. Bargiela and W. Pedrycz, *Human-centric information processing through granular modelling*, vol. 182 of *Studies in Computational Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[3] J. M. Alonso, C. Castiello, and C. Mencar, "Interpretability of Fuzzy Systems: Current Research Trends and Prospects," in *Springer Handbook of Computational Intelligence* (J. Kacprzyk and W. Pedrycz, eds.), Springer Berlin / Heidelberg, 2015.

[4] S. Zhou and J. Gan, "Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling," *Fuzzy Sets and Systems*, vol. 159, pp. 3091–3131, Dec. 2008.

[5] M. J. Gacto, R. Alcala, and F. Herrera, "Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures," *Information Sciences*, vol. 181, pp. 4340–4360, Mar. 2011.

[6] C. Mencar, M. Lucarelli, C. Castiello, and A. M. Fanelli, "Designing Strong Fuzzy Partitions from data with DC*," *Mathware & Soft Computing*, vol. 21, no. 1, pp. 15–19, 2014.

[7] M. Lucarelli, C. Castiello, A. M. Fanelli, and C. Mencar, "Interpretable knowledge discovery from data with DC*," in *Proc. 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)*, pp. 815–822, Atlantis Press, 2015.

[8] S. Edelkamp and S. Schrödl, *Heuristic Search: Theory and Applications*. Morgan Kaufmann, 2011.

[9] T. Kohonen, *Self-organizing maps*, vol. 30 of *Information Sciences*. Springer Verlag, 2001.

[10] C. Mencar, M. Lucarelli, C. Castiello, and A. M. Fanelli, "Design of Strong Fuzzy Partitions from Cuts," in *Proceedings of the 8th conference of the European Society for Fuzzy Logic and Technology*, Advances in Intelligent Systems Research, (Paris, France), pp. 424–431, Atlantis Press, 2013.

[11] C. Mencar, A. Consiglio, and A. M. Fanelli, "DCγ : Interpretable Granulation of Data through GA-based Double Clustering," in *2007 IEEE International Fuzzy Systems Conference*, pp. 1–6, Ieee, June 2007.

[12] S. Guillaume and B. Charnomordic, "Generating an Interpretable Family of Fuzzy Partitions From Data," *IEEE Transactions on Fuzzy Systems*, vol. 12, pp. 324–335, June 2004.

[13] S. Guillaume and B. Charnomordic, "Learning interpretable fuzzy inference systems with FisPro," *Information Sciences*, vol. 181, pp. 4409–4427, Oct. 2011.