Two New Approaches to Feature Selection with Harmony Search

Ren Diao and Qiang Shen

***Abstract*— Many search strategies have been exploited in implementing feature selection, in an effort to identify smaller and better subsets. Such work typically involves the use of heuristics in one form or another. In this paper two novel methods are presented by applying harmony search to feature selection. In particular, it demonstrates the potential of utilising this search mechanism in combination with fuzzy-rough feature evaluation. The resulting techniques are compared with approaches that rely on hill-climbing, genetic algorithms and particle swarm optimisation.**

Key words: Harmony Search; Feature Selection; Meta Heuristics; Fuzzy-rough Sets

I. Introduction

The main aim of feature selection (FS) is to determine a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original features [3]. In real world problems FS is usually necessary due to the abundance of noisy, irrelevant, redundant or misleading features. For instance, by removing these factors, learning from data techniques such as text processing and web content classification can benefit greatly. Given a feature set size n, the task of FS can be seen as a search for an "optimal" feature subset through the competing 2^n candidate subsets. The definition of an optimal subset may vary, depending on the problem at hand. Although an exhaustive method may be used for this purpose, it is impractical for most datasets. Usually FS algorithms involve heuristic or random search strategies in an attempt to avoid this prohibitive complexity. However, the degree of optimality of the final feature subset is often reduced.

Harmony search (HS) [5] is an meta-heuristic algorithm. It mimics the improvisation process of music players. HS algorithm has been very successful in a wide variety of optimisation problems, presenting several advantages with respect to traditional optimisation techniques. It imposes only limited mathematical requirements and is not sensitive to the initial value settings. The HS algorithm generates a new potential solution vector, after considering all existing vectors. The basic HS algorithm has been improved by introducing methods to tune parameters dynamically [8].

Rough set theory (RST) [10] has been used successfully as a selection tool to discover data dependencies and reduce the number of attributes contained in a dataset by purely structural methods [1]. Given a dataset with discretised attribute values, by the use of rough sets it is possible to find a subset (termed *reduct*) of the original attributes that are the most informative; all other attributes can be removed from the dataset with minimal information loss. However, it is most often the case that the values of attributes may be both crisp and real-valued, and this is where traditional rough set theory encounters a problem. It is not possible in the theory to say whether two attribute values are similar and to what extent they are the same; for example, two close values may only differ as a result of noise, but in the standard RST-based approach they are considered to be as different as two values of a different order of magnitude. Dataset discretisation must therefore take place before reduction methods based on crisp rough sets can be applied. This is often still inadequate, however, as the degrees of membership of values to discretised values are not considered at all. Also, the discretisation may result in information loss.

In order to combat this, extensions of rough sets based on fuzzy-rough sets [4], tolerance relations [14], and massive discretisation [15] (also known as roughfication) have been developed. The focus of this paper is the data reduction method [6], [7] based on fuzzy-rough sets. Fuzzy-rough sets encapsulate the related but distinct concepts of vagueness (for fuzzy sets [17]) and indiscernibility (for rough sets), both of which occur as a result of uncertainty in knowledge. The fuzzy-rough set-based approach considers the extent to which fuzzified values are similar. Previously, an incremental hill-climbing algorithm was employed to discover the best feature subset. However, this often led to the discovery of non-optimal feature subsets, both in terms of the resulting dependency measure and the subset size. With an aim to further improve the performance of fuzzy-rough set based FS technique, this paper presents two novel approaches to FS by exploiting the advantages offered by HS.

The rest of this paper is structured as follows. Section II introduces the main concepts of harmony search. Section III describes the theory of fuzzy-rough set data reduction. Section IV explains how feature selection problem can be modelled as an optimisation problem solvable by harmony search, and details the approaches developed to tackle the problem, with application to feature selection in general, and fuzzy-rough feature selection in particular. Section V shows the experimentation carried out on real-world problem cases and presents the results along with discussions. Section VI concludes the paper and proposes further work in the area.

II. The Principles of Harmony Search

Harmony search mimics the improvisation process of musicians, during which, each musician plays a note for

Ren Diao and Qiang Shen are with the Department of Computer Science, Aberystwyth University, UK (email: {rrd09, qqs}@aber.ac.uk).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

finding a best harmony all together. When applied to optimisation problems, the musicians represent the decision variables of the cost function, and HS acts as a meta heuristic algorithm which attempts to find a solution vector that optimises this function. In the process, each decision variable (musician) generates a value (note) for finding a global optimum (best harmony). The harmony search algorithm has a novel stochastic derivative (for discrete variable) based on musician's experience, rather than gradient (for continuous variable) in differential calculus.

A. Key Concepts

The key concepts of HS algorithm are musicians, notes, harmonies and harmony memory. In most optimisation problems solvable by HS, the musicians are the decision variables of the function being optimised. The notes played by the musicians are the values each decision variable can take. The harmony contains the notes played by all musicians, or a solution vector containing the values for each decision attribute. The harmony memory contains harmonies played by the musicians, or a storage place for solution vectors.

A more concrete representation of harmony memory is a two dimensional matrix, where the rows contain harmonies (solution vectors) and the number of rows are predefined and bounded by the harmony memory size. Each column is dedicated to one musician, and the entire column stores all the notes played by him in all harmonies, referred to as the note domain for each musician in this paper.

B. Iteration Steps and Algorithm Illustration

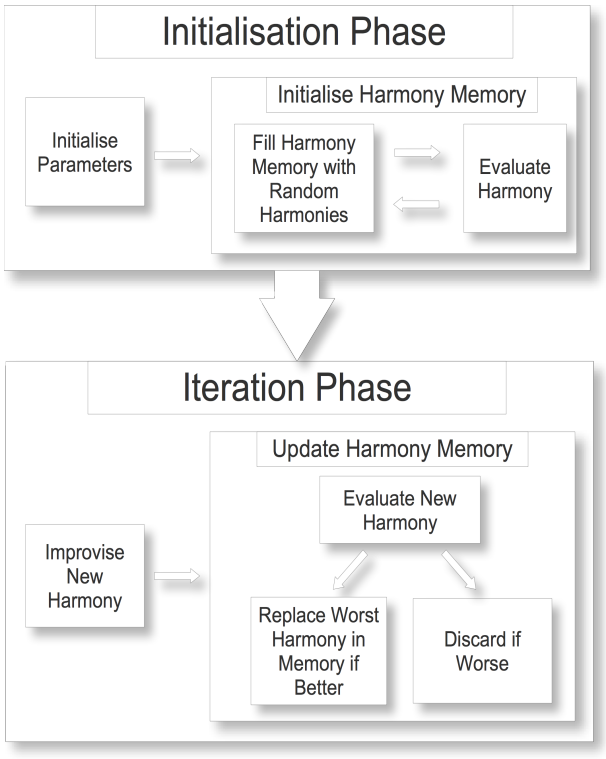

Fig. 1. Harmony Search Illustrated

Harmony search can be divided into two core phases, initialisation and iteration, as shown in Fig. 1. A simple example problem taken from [5] is used for a better illustration:

Minimise

$$(a-2)^2 + (b-3)^4 + (c-1)^2 + 3 \tag{1}$$

where

$$a, b, c \in \{1, 2, 3, 4, 5\}$$

1) Initialisation:

a) Initialise Problem Domain: The parameters of HS are assigned according to the problem, including: size of harmony memory, number of musicians, max iteration, and optionally, the harmony memory considering rate (HMCR), and the pitch adjustment rate (PAR). In the example, the number of musicians is 3, each corresponds to the decision attributes a, b and c. The harmony memory size is 3, and the objective function is the function to be minimised, the lower the better.

b) Initialise Harmony Memory: Harmony memory is filled with randomly generated solution vectors. In the example problem, 3 randomly generated solution vectors are {2,2,1}, {1,3,4} and {5,3,3}.

2) Iteration:

a) Improvise New Harmony: A new value is chosen randomly by each musician out of their note domain, and together forms a new harmony. In the example, musician a may randomly choose 1 out of {2,1,5}, b chooses 2 out of {2,3,3} and c chooses 3, forming a new harmony {1,2,3}.

There are two factors which affect the note choice of a musician, HMCR and PAR. HMCR, range from 0 to 1, is the rate of choosing one value from the historical notes stored in the harmony memory, while $(1 - HMCR)$ is the rate of randomly selecting one value from all possible range of values. If HMCR is set low, the musicians will constantly explore other areas of the solution space, and a higher HMCR will restrict the musicians to historical choices. The PAR parameter causes the musicians to select a neighbouring value based on the following formula $a + (random * bw)$, where bw is an arbitrary distance bandwidth, while $(1 - PAR)$ is the probability of using the chosen value without further alteration. The pitch adjustment is applied after a note is chosen by the musician, either from the HM or all possible value range.

Given the above example, with $HMCR = 0.9$, and $PAR = 0.1$, musician a will choose from HM {2,1,5} with a probability of 90%, and choose out of all possible values {1,2,3,4,5} with a probability of 10%, after making a choice, say, 3 out of all possible values, the musician will choose a neighbouring value with 10% probability, and the value 4, is then chosen in the end.

b) Update Harmony Memory: If the new harmony is better than the worst harmony in the harmony memory, judged by the objective function, the new harmony is then included in harmony memory and the existing worst harmony is removed. The new harmony {1,2,3} has the evaluation score of 9, making it better than the worst harmony in the

memory {1,3,4} which has a score of 16, therefore the harmony {1,3,4} is removed from memory, replaced with {1,2,3}. If {1,2,3} had a larger score than 16, it would be the one being discarded.

The algorithm continues to iterate until the maximum number of iterations has been reached. In the example, if the musicians later choose {2,3,1}, which is very likely as those numbers are already in the note domains, the problem will be solved with a minimal score of 3.

C. A Probabilistic View of Harmony Search

In order to demonstrate the convergence capability of harmony search, consider the harmony memory with the following parameters: the size of the HM (the number of harmonies in HM) = M, the number of instruments (variables) = N, the number of possible notes (values) of an instrument = L, the number of optimal note (value) of instrument i in the HM = H_i, $HMCR = H_r$, and the optimal harmony (optimal vector) = (x,y,z). The probability of finding the optimal harmony, Pr(H) is

$$Pr(H) = \subseteq_{i=1}^{N} \left[Hr \frac{H_i}{M} + (1 - Hr) \frac{1}{L} \right] \quad (2)$$

where the pitch adjusting rate is not considered because it is an optional operator.

Initially, the HM is filled with random harmonies. If there is not any optimal note of all instruments in the HM,

$$H_1 = H_2 = ... = H_N = 0$$

and

$$Pr(H) = \left[(1 - Hr) \frac{1}{L} \right]$$

This means that the probability Pr(H) is very low. However, if the schema of optimal harmony such as (*,y,z), (x,*,z), (x,y,*) have better evaluation than others, the number of optimal notes of instrument i in the HM, H_i will be increased iteration by iteration. Consequently, the probability of finding the optimal harmony, Pr(H) will be increased.

III. Fuzzy-Rough Data Reduction

The rough set selection process described in [1] can only operate effectively with datasets containing discrete values. However, most datasets contain real-valued features and so it is necessary to perform a discretisation step beforehand. This is typically implemented by standard fuzzification techniques. As membership degrees of feature values to fuzzy sets are not exploited in the process of reduct search, important information has been lost. By employing *fuzzy-rough* sets, it is possible to use this information to better guide feature selection.

A fuzzy-rough set is defined by two fuzzy sets, fuzzy lower and upper approximations, obtained by extending the corresponding crisp rough set notions. In the crisp case, elements that belong to the lower approximation (i.e. have a membership of 1) are said to belong to the approximated set with absolute certainty. In the fuzzy-rough case, elements may have a membership in the range [0,1], allowing greater flexibility in handling uncertainty.

Fuzzy-Rough Feature Selection [6], [7] (FRFS) is concerned with the reduction of information or decision systems through the use of fuzzy-rough sets. Let $I = (\mathbb{U}, \mathbb{A})$ be an information system, where $\mathbb{U}$ is a non-empty set of finite objects (the universe) and $\mathbb{A}$ is a non-empty finite set of attributes such that $a : \mathbb{U} \rightarrow V_a$ for every $a \in \mathbb{A}$. V_a is the set of values that attribute a may take. For decision systems, $\mathbb{A} = \{\mathbb{C} \cup \mathbb{D}\}$ where $\mathbb{C}$ is the set of input features and $\mathbb{D}$ is the set of decision features.

FRQUICKREDUCT($\mathbb{C}$,$\mathbb{D}$).
$\mathbb{C}$, the set of all conditional features;
$\mathbb{D}$, the set of decision features.

(1) $R \leftarrow \{\}$, $\gamma'_{best} \leftarrow 0$, $\gamma'_{prev} \leftarrow 0$
(2) **do**
(3) $T \leftarrow R$
(4) $\gamma'_{prev} \leftarrow \gamma'_{best}$
(5) $\forall x \in (\mathbb{C} - R)$
(6) **if** $\gamma'_{R \cup \{x\}}(\mathbb{D}) > \gamma'_T(\mathbb{D})$
(7) $T \leftarrow R \cup \{x\}$
(8) $\gamma'_{best} \leftarrow \gamma'_T(\mathbb{D})$
(9) $R \leftarrow T$
(10) **until** $\gamma'_{best} == \gamma'_{prev}$
(11) **return** R

Fig. 2. The fuzzy-rough QUICKREDUCT algorithm

A fuzzy-rough QUICKREDUCT algorithm, based on the crisp version [1], has been developed as given in Fig. 2. It employs the fuzzy-rough dependency function γ' to choose which features to add to the current reduct candidate. The algorithm terminates when the addition of any remaining feature does not increase the dependency. As with the original algorithm, for a dimensionality of n, the worst case dataset will result in $(n^2 + n)/2$ evaluations of the dependency function. However, as fuzzy-rough set-based feature selection is used for dimensionality reduction prior to any involvement of the system which will employ those features belonging to the resultant reduct, this operation has no negative impact upon the run-time efficiency of the system.

IV. Harmony Search for Feature Selection

The aim of this work is to develop a harmony search [5] based, stand alone, reusable search strategy that can find optimal feature subsets according to a wide range of subset evaluation methods.

As explained in section 2, harmony search is best suited to solve problems with a fixed set of decision attributes and an objective function to optimise. However, feature selection is a problem with variable sized solutions, there are no obvious direct ways of applying harmony search.

Therefore we need to map each key concepts of HS into elements in FS. There are obvious analogies such as: each feature subset can be seen as a harmony, and the objective

function can be substituted by a subset evaluation method, for example, the fuzzy-rough dependency measure. When applied to FRFS, the aim is to find minimal reducts, therefore the fitness of a solution will be judged by two factors, 1) the fuzzy-rough dependency score, and 2) the size of the reduct.

A. Horizontal Approach

1) Mapping Key Components
The horizontal approach maps musicians onto the available features to be selected. The note domain of each musician is then a binary value, indicating whether or not the corresponding feature is included in the harmony. And the actual harmony can be represented as a series of bits. For example, as shown in Fig. 3, a harmony {0,1,1,0,0,0} will translate into feature subset {B,C}.

Fig. 3. Harmony Translation in Horizontal Approach

2) Iteration Steps
The initialisation step involves filling the harmony memory with randomly generated harmonies, i.e. randomly generated bit sets. In order to improvise a new harmony, each musician randomly selects a value out of their note domain. Together, such selected values form the new bit set. This set is then translated back to a feature subset and evaluated. If the evaluation score is higher than any of the feature subsets in the harmony memory, it replaces the worst subset; otherwise, the new bit set is discarded. The process iterates until max iteration is reached.
3) Harmony Memory Considering Rate
In this approach, HMCR has little practical impact because the available notes for each musician are very limited. The most significant use of it is in terms of flipping the bit value, which includes a previously unselected feature, or vice versa. Hence, in this application, this parameter is simply implemented as the bit flip rate.
4) Other Parameter Settings
The three tuneable parameters are harmony memory size, bit flip rate and maximum number of iterations. The harmony memory size is a sensitive parameter, in most cases it is set between half of the total number of features to the total number of features, leaving less than half of the features outside the harmony memory. A large harmony memory will give each musician more notes to choose from when improvising a new harmony. However, it will require a longer initialisation in order to fill up the harmony memory and hence, may lead to slower convergence.

B. Vertical Approach

1) Mapping Key Components
This approach tackles the problem of FS from a different viewpoint. It treats musicians as independent experts, and each musician can vote for one feature to be included in the feature subset when improvising a new harmony. The harmony is then the combined vote from all musicians, indicating which features are being nominated. The entire pool of original features forms the range of notes available to the musicians. Multiple musicians are allowed to choose the same attribute, and they may opt to choose no attribute at all. For example, as illustrated in Fig. 4, the harmony {A,-,B,B,C,-} will translate into feature subset {A,B,C}, the top row are labels for musicians.

Fig. 4. Harmony Translation in Vertical Approach

2) Iteration Steps
The harmony memory of size m is initialised by random generation of harmonies. This provides each musician a note domain of m attributes, including identical attributes, and nulls. A new harmony is produced by each musician randomly choosing one attribute from their note domain. The new harmony is then evaluated using the cost function. It is used to replace the worst harmony in the harmony memory if a better score is achieved, or discarded otherwise.
3) Harmony Memory Considering Rates
The vertical approach allows a much greater range of notes for musicians, which enables the use of the HMCR to its full potential. HMCR will indicate whether a musician should pick from their own note domain, or from the entire pool of features. This will greatly increase the chance of escaping from a local optimal. Fig. 5 demonstrates the effect of HMCR, the bottom row is the new harmony being improvised, musician $E4$ has 2 choices in the harmony memory, $-$ for discard, and feature B, but HMCR will force the musician to choose another value, the randomly chosen feature D is therefore included in the new harmony.

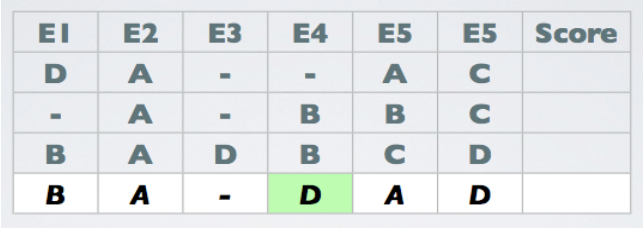

Fig. 5. HMCR Effect in Vertical Approach

4) Parameter Settings
There are four tuneable parameters in this approach: harmony memory size, total iteration, HMCR, and number of musicians. Where compared to the horizontal approach, the extra parameter is the number of musicians, which is set to the number of original features. In this approach, it is set to a number between half of the number of features and the total number of features, similar to the value used for harmony memory size. Having too few musicians will significantly reduce the algorithm's performance, because it assumes that the optimal subset is no greater than the number of musicians. On the other hand, having too many musicians, especially having more musicians than the total number of features, will result in oversized solutions and much longer convergence, because almost all the features will get chosen by random during the initialisation phase and the musicians will have too many choices to choose from when improvising new harmonies. One way to combat this would be to encourage musicians to discard votes, or produce the solution vector using majority voting scheme.

The vertical approach makes good use of the HS principle, giving each musician a much wider range of notes, resulting in a much more diverse harmony memory. A more diverse harmony memory makes good use of HMCR, making it easier for the algorithm to escape from local optimal and to have stronger exploration power. However, it has the downside of introducing an extra tuneable parameter and slightly more complex structure.

V. Experimentation and Discussion

To demonstrate the capability of the two HS based search methods and the utility of fuzzy-rough feature selection (FRFS), experimental comparisons are carried out, involving several other search methods: the greedy hill-climbing search, genetic algorithm (GA) [16] and particle swarm optimisation (PSO) [13], where both GA and PSO are evolutionary algorithms. All methods are implemented and applied as pre-processors within a complex system monitoring application, then tested and compared on 9 real-valued UCI datasets [9]. All methods except hill-climbing require additional parameter settings for their operation, which are given below in Table I.

The parameters for both HS approaches are set to be comparable with others in terms of number of musicians and maximum number of iterations. In order to show the convergence performance of the algorithm.

A. Experimental Results

The search objectives are: 1) finding a fuzzy-rough reduct which has a dependency score of 1, or a subset with the highest possible dependency; 2) size of the reduct should be as small as possible. All tested methods are successful in achieving the first objective, being able to find fuzzy-rough reduct for each dataset.

TABLE I
Parameter Settings

Algorithm	Parameter	Values
Horizontal Harmony Search (HHS)	Memory Size	25
	Max Iteration	500
	Bit Flipping Rate	0.9
Vertical Harmony Search (VHS)	Memory Size	10
	Max Iteration	100
	HMCR	0.8
	# Musicians	10
Genetic Algorithm (GA)	Cross Over Prob	0.6
	Max Generation	100
	Mutation Prob	0.033
	Population Size	20
	Cross Over	0.6
Particle Swarm Optimisation (PSO)	C1	2.0
	C2	2.0
	Max Generation	100
	# Particles	40

As shown in Table II, the horizontal approach found the smallest reducts in 5 out of 9 datasets. However, it did not perform well for the other 4 datasets, finding the largest subsets comparing to other methods. It is interesting to note that these 4 datasets have significantly more features than the others, and that the horizontal approach produced the final answers very late in the search process, without being able to find the optimal answers within the set maximum number of iterations. Further experiments show that this approach can determine improved or optimal result but with a much larger maximum iteration figure, typically over 1000.

For 6 out of 9 datasets, the vertical approach managed to find reducts of the same size as others. It found the exclusive smallest reduct for the *cleveland* dataset. In the *heart* dataset case, the vertical approach found the largest reduct of size 8, one feature more than all other search methods. Further investigation shows that this approach found exactly the same subset but with one redundant feature. It indicates that although HS can locate the regions where the optimal solution resides very quickly, it may be struggle to find the absolute optimal solution. Where greedy hill-climbing is very good at finding such solutions. Further experiments show that the vertical approach can indeed find the best subset for the *heart* dataset, but only after 165 iterations, where the sub-optimal solution with 8 features was found after just 75 iterations.

The C4.5 [11] classifier learner was employed for the purpose of evaluating the resulting subsets following the FS phase. The classification results in Table III show that the reducts found by the vertical approach give comparable performance as with those using equal or smaller subset sizes obtained by other approaches. In particular, the vertical approach found the exclusive best subset for dataset *glass* which greatly increased the classification performance from the unreduced 67.29% to 69.63%. However, this approach does not seem to work well for the *wine* dataset, that matches classification performance of other methods. The main reason for such performance variation is that FRFS is a filter approach. That is, all search strategies are trying to

TABLE II
REDUCT SIZE

Dataset	# of Instances	# of Features	HHS	VHS	GA	PSO	HC
2-completed	390	39	9	6	7	6	6
3-completed	390	39	9	7	8	7	6
cleveland	297	14	9	7	8	8	8
glass	214	10	8	8	8	8	9
heart	270	14	7	8	7	7	7
ionosphere	230	35	11	7	10	7	8
iris	150	5	4	4	4	4	4
olitos	120	26	6	5	6	5	5
wine	178	14	5	5	5	5	5

TABLE III
CLASSIFICATION RESULT

Dataset	Unreduced	HHS	VHS	GA	PSO	HC
2-completed	83.33	82.31	81.28	81.28	81.54	86.41
3-completed	83.08	73.33	81.54	79.74	82.82	80.51
cleveland	51.85	51.52	50.84	50.84	58.59	50.84
glass	67.29	64.95	69.62	64.95	64.95	67.29
heart	76.67	79.26	79.26	79.26	70.37	79.26
ionosphere	87.83	89.57	86.09	82.61	86.96	85.22
iris	96	96	96	96	96	96
olitos	67.5	60.83	60	60.83	54.17	63.33
wine	94.38	92.13	82.02	90.45	92.13	95.51

optimise the fuzzy rough dependency score as well as subset sizes, not the classification accuracy of each subset. The final classification results are depended on the performance of the fuzzy rough dependency measure.

B. Discussions

The experimental results show that PSO is also very powerful and efficient. It often agrees with the solution found by the vertical approach. Both horizontal and vertical approaches outperform the genetic algorithm. It can be seen that hill-climbing may produce reasonably good results in certain situations, but can still get stuck in local optimal.

In terms of computation performance and robustness, harmony search based approaches are computationally inexpensive themselves, because the algorithm comprises a very simple concept, and the implementation is also straightforward. The actual run time of the entire feature selection process is then determined by two main factors, the max number of iterations, and the efficiency of the subset evaluation method. In the experimental evaluations, the horizontal approach requires longer processing time in order to find comparable quaility subsets. The vertical approach makes better use of the algorithm and can converge much quicker, the actual running time is similar to that of GA and PSO based searches. Empirically, and as may be expected, the larger the number of instances in the dataset, the longer time is needed for computing fuzzy rough dependency measures.

The use of harmony memory in HS offers a major advantage over that of techniques like genetic algorithms, as it maintains a record of the historical data processed by previous iterations. All elements of the memory together contribute to the new harmony, while changes in genetic populations result in the destruction of previous knowledge of the problem. Harmony memory considering rate and pitch adjustment rate also help greatly in escaping from the local best solution. Comparing between the two HS based methods, it is obvious that the vertical approach gives better results with smaller harmony memory and fewer iterations.

In all experiments, there has been no attempt to optimise the parameters for each search method. The same parameter settings are used for easy comparison. It can be expected that the results obtained with optimisation would be even better than those already observed.

VI. CONCLUSION

This paper has presented two novel approaches for feature selection based on a relatively new meta heuristic algorithm, harmony search, to solve feature selection problems. HS has a number of advantages over conventional approaches: fast convergence, simplicity, insensitivity to initial states, and efficiency in finding minimal reduct. Experimental comparative studies show that the horizontal approach can locate good solutions, though optimal reducts may only be found after a large number of iterations. However, the vertical approach

performs very well, and is capable of identifying similar or superior feature subsets for most test datasets, with the resulting classification accuracy being comparable to the state of the art approaches. In almost all aspects, the vertical approach delivered much better result than genetic algorithm and hill-climbing approaches.

The proposed approaches are general, and can be used in conjunction with other filter-based, and even wrapper-based subset evaluation techniques with minimal changes. Owing to the underlying randomised and yet simple nature, the entire solution space of a given problem may be examined by running the HS algorithm in parallel. This will help to reveal a large number of good solutions much quicker than random search or exhaustive search methods. Experimental evaluation of these ideas remains as active research.

Although promising, much can be done to further improve the potential of the present work. In particular, investigations into how the algorithm parameters may be better tuned are necessary. For example, currently, the total number of iterations is predefined, whilst an optimal subset may be found early in the search process or sometimes. It would be useful to develop a better stopping criterion based on the rate of improvement of the emerging solution, and/or the overall quality of the entire harmony memory. Also, a method may be established to dynamically specify the size of the harmony memory, which affects convergence rate and the search perimeters. It can be initialised to a fairly large value, in order to search in wider areas of the search space, but as the process progresses, it can be reduced to narrow down the choices in order to converge to the optimal solution sooner.

Finally, it is worth noting that the HS based approaches are capable of finding many reducts, thanks to its randomised global search nature. However, more investigations are needed in ultilising the pool of reducts to serve as a feature selection ensemble. Working with multiple subsets instead of a single one has indeed become a strong trend in data mining.

REFERENCES

[1] A. Chouchoulas and Q. Shen. Rough set-aided keyword reduction for text categorisation, Applied Artificial Intelligence, vol. 15, no. 9, pp. 843–873, 2001.

[2] C.A.C. Coello, Constraint-handling using an evolutionary multi objective optimization technique, Civil Engineering Environment System, vol. 17, pp. 319–346, 2000.

[3] M. Dash and H. Liu, Feature Selection for Classification, Intelligent Data Analysis, vol. 1, no. 3, pp. 131–156, 1997

[4] D. Dubois and H. Prade, Putting rough sets and fuzzy sets together, Intelligent Decision Support, Kluwer Academic Publishers, Dordrecht, 1992.

[5] Z.W. Geem, J.H. Kim and G.V. Loganathan, A New Heuristic Optimization Algorithm: Harmony Search, Simulation, vol. 76, no. 2, pp. 60–68, 2001.

[6] R. Jensen and Q. Shen. Fuzzy-Rough Sets Assisted Attribute Selection. IEEE Transactions on Fuzzy Systems, vol. 15, no. 1, pp. 73–89, 2007.

[7] R. Jensen and Q. Shen, New Approaches to Fuzzy-Rough Feature Selection, IEEE Transactions on Fuzzy Systems, vol. 17, no. 4, pp. 824–838, 2009.

[8] M. Mahdavi, M. Fesanghary and E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied mathematics and computation, vol. 188 , no. 2, pp. 1567–1579, 2007.

[9] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, UCI Repository of machine learning databases `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[10] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data, Kluwer Academic Publishing, Dordrecht, 1991.

[11] R. Quinlan, Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

[12] A. Vasebi, M. Fesanghary and S.M.T. Bathaeea, Combined heat and power economic dispatch by harmony search algorithm, International Journal on Electric Power, vol. 29, pp. 713–719, 2007.

[13] X. Wang, J. Yang, X. Teng, W. Xia and R. Jensen, Feature selection based on Rough Sets and Particle Swarm Optimization, Pattern Recognition Letters, vol. 28, no. 4, pp. 459–471, 2007.

[14] P. Synak, D. Slezak, Tolerance Based Templates for Information Systems: Foundations and Perspectives, Lecture Notes in Computer Science, vol. 4413, pp. 11–19, 2007.

[15] D. Slezak, J. Wroblewski, Rough Discretization of Gene Expression Data, International Conference on Hybrid Information Technology, vol. 2, pp. 265–267, 2006.

[16] J. Wroblewski, Finding minimal reducts using genetic algorithm, Proccedings of the second annual join conference on infromation science, pp. 186–189, 1995.

[17] L.A. Zadeh, Fuzzy sets, Information and Control, vol. 8, pp. 338–353, 1965.