Predicting Shrutis in Harmonium using Temporal Mining

Mr. Yogesh P. Pingle
Department of Information Technology, Mumbai University
Vidyavardhini's College of Engineering and Technology, Vasai (W), Thane
yogesh.pingle@vcet.edu.in

ABSTRACT

Although being invented in Paris by Alexander Debian in 1840, the Harmonium has become an inseparable part of our Hindustani Classical Music. When a Singer sings in the company of the Harmonium that is the Shruti Harmonium, it not only fills colors in his performance but also wins the hearts of the Listeners. Thus our main objective over here is to replace the Harmonium Player with the software.

This Paper focuses on the objective to develop a software that will not only accompany the singer with the normal notes but also will predict the singer's next note. The need to develop this software is because the current software's available only passively repeat the notes, lacking intelligence which will be taken care in our proposed software. As we know there are various Shruti's defined in our Hindustani Classical Music. These Shruti's will be fed into the database before hand which will contain the name of the raga , timing of the raga according to the kaal chakra , frequency of each raga and other details. So when the singer will sing, the software will listen to the singer for a definite period of time, it will map it to the database, identify the Shruti and predict the next note of the singer accordingly. Not only this but the software will keep playing the notes even if the singer takes a break for continuity and binding the audience to the performance.

Though the currently available Software's can play a variety of Musical instruments but The Harmonium is an exception. We wish to develop this software by using the 22

Shruti's and their corresponding frequencies which have been researched by Dr. Vidyadhar Oke. We will be using the Sound Card Hardware for mapping the singer's voice to the 22 Shruti database. The Temporal Mining algorithms are used to develop the proposed software.

Algorithm used for implementing the proposed software are as follows: Algorithm for Mining Maximum Length-Repeating Patterns (MLRP) in the Music database, which is used for extraction of MLRP's occurring in the music according to the data stored in the database. The next algorithm that we will use is mining transposed motifs (music pattern) algorithm that is used to evaluate the frequently occurring patterns; it is based upon the Sequence pattern mining algorithm. The Sequence pattern mining algorithm discovers all the frequent sub- sequences among a large database of sequences D or in a unique large sequence. The next algorithm we will be using is the algorithm for speech recognition and language processing.

Keywords - Shruti Harmonium, 22 Shruti's, Hindustani Raga's, Harmonium Note Frequencies, Clustering, Sound Card, Mapping

I. INTRODUCTION

A majority of us enjoy music. It may be through listening songs, singing or humming tunes, or simply by playing some musical instrument. Any means gives equal enjoyment and happiness. However, singing and playing an instrument are serious acts. One needs to be thoroughly proficient with it to enjoy himself as also to entertain, impress and enthrall others. Thus, a professional is expected to practice regularly, to polish and sharpen his skills and also to improve himself. A singers practice is however incomplete without the support of a musical instrument. Thus, practicing in the company of a harmonium or a tabla or a sitar is considered the right way of practicing. In this fast moving world of today, an accomplice is not always available for all riyaz sittings of a singer. Singers have to do away on their own on many occasions. Thus, we felt the need of bailing out such singing artists with our unique discovery.

Our instrument assists a singer in his practicing sessions and fills in the place of a personal harmonium player who may not be available at that point of time. The instrument is an

'ELECTRONIC HARMONIUM PLAYER' which records the singer's frequency of singing, his range, his style, his breaks, and other complimentary factors of his singing and simply strikes the right harmonium cords. The specialty of the instrument is its real-time acting prowess. The harmonium notes are delivered in synchronization with the singers vocals. The breathers are also taken into consideration and filled in effectively if expected. Using the player a singer can have practice sessions as per his convenience as per his timings and that too without the availability of a live harmonium player to support and assist him.

II. WORKING

The 'ELECTRONIC HARMONIUM PLAYER' works magically. The player consists of a mike or a micro phone attached to it as an input machine. The singer is expected to sing into this micro phone. Along with transporting the singer's voice clearly to the audience, the microphone will also send in the frequency of the singer's voice, his sung notes and his throw of voice towards our next unit.

The next unit is that of a hardware sound card. The sound card will take in the frequencies of each and every note of the

singer. The sound card will have various port numbers. From the sound card the frequencies will be transferred to a program which will read the frequencies. After reading and taking in the frequencies, the singers frequencies read through the sound card ports will be mapped with the harmonium frequencies. For this the already existing information about the 'Hindustani ragas', the ' 22 Shruti's', the 'harmonium note frequencies' will be taken into consideration. These will already be present in the computer, in the form of noted, updated and well maintained tables. Thus, the program, using this information will map the singer's frequency with the harmonium frequency and pass it to the next hardware. Using this hardware, the actual harmonium notes shall be played and those will be real time and in synch with the singer's notes. The singers tendency of taking 'tans' and 'murkhis' will also be well supported by this entire process. At the step where the sound card comes into picture, a separate program is also included which stores the singer's frequencies in a separate table and in turn in the database. This stored matter can be later used to play harmonium notes in the backdrop again when the singer takes a break and is not singing. Such is the importance of the electronic ghost players.

III. BLOCK DIAGRAM OF PROPOSED SYSTEM

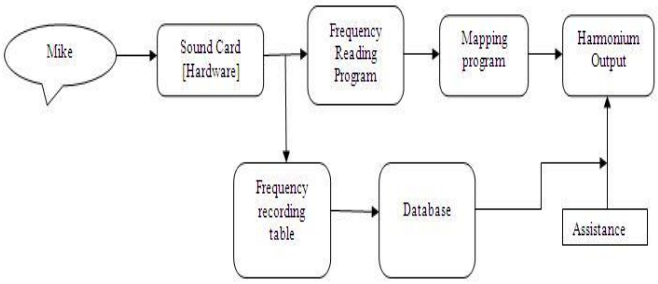

Fig. 1 Proposed System

DATABASE

We will be storing the data in the database in the form of tables. These are split up into 2 categories:
1) Permanent Tables
2) Temporary Tables

The Permanent tables will be storing the 22 Shruti's, Raga's along with their characteristics in separate tables for reference. While the temporary tables will again be divided into two: Temporary Header and Temporary details which will be storing the singers session for a definite period of time and also will be used for predicting the singer's notes when he/she takes a break.

1) Permanent Tables:

Table storing the exact frequencies of the 22 Shruti's in 12 scales:

Table 1

Shruti no	C (Sapthet 1)	C# (Kali 1)	D (Sapthet 2)	E flat (Kali 2)	E (Sapthet 3)	F (Sapthet 4)	F# (Kali 3)	G (Sapthet 5)	A flat (Kali 4)	A (Sapthet 6)	B flat (Kali 5)	B (Sapthet 7)
1	261.6255526	277.1826192	293.6647570	311.1269738	329.6275481	345.2282240	369.9944168	391.9954218	415.3046954	440	466.1637543	493.8833055
2	275.6219814	292.0113189	309.3752171	327.7715266	347.2619406	367.9112153	389.7883367	412.9863807	437.5226419	463.5390945	491.1325663	520.3060473
3	279.0672562	295.6614633	313.2424073	331.8587719	351.6027179	372.5101055	394.6607111	418.1284604	442.5916749	469.3333332	497.2413481	526.8088631
4	290.6950883	307.9806877	326.2941741	345.5965372	366.2328309	388.0313596	411.1049071	435.6804753	461.4496611	488.8888884	517.9597373	548.7592259
5	294.3287466	311.8304463	330.3729513	350.3173462	370.8309913	392.3817516	415.2437185	440.9546013	467.2197818	494.9999995	524.4342340	555.6197193
6	310.0747288	328.5127334	348.3471189	368.7403795	390.6696867	413.9001168	438.8119008	464.8873778	492.2125716	521.4814808	552.4933852	585.3491773
7	313.9506629	332.6191425	352.3977379	373.3923580	395.5380672	419.0738682	443.9932996	470.3943175	498.3656338	527.9999993	559.3955153	592.6599673
8	327.0319403	346.4782733	367.3803455	388.9387164	412.0344343	436.5352791	462.4930201	489.9942688	519.1306682	549.9999989	582.7047338	617.3541319
9	331.1198396	350.8092517	371.5694573	393.7703754	417.1848648	441.9919701	468.2741828	496.1152174	525.6200040	556.8749989	589.9935126	625.0710585
10	348.8340695	369.5758247	391.5533084	414.9359541	439.5033965	465.6376309	493.3258879	522.6605745	553.7395925	586.6666653	621.5516839	658.5110739
11	353.1944954	374.1955353	396.4472410	420.3214136	444.9971889	471.4881013	499.4924615	529.1958317	560.6613374	593.9999986	629.3210799	666.7426522
12	367.9109325	389.7880673	412.3663632	437.5223196	463.5387382	491.1021885	520.3046471	551.2455743	584.0222261	618.7499981	655.5427911	694.5233977
13	372.5098191	394.6604077	418.1281390	442.3913344	469.3329724	497.2409659	526.8084851	558.1341150	591.3225039	626.4843731	663.7370750	703.2049402
14	392.4383278	415.7739273	440.4971340	466.5304591	494.4413285	523.8423342	554.9976233	587.9951457	622.5570439	659.9999978	699.2466436	740.8249573
15	413.4329707	438.0159758	464.3628241	491.5574382	520.8929136	551.8668210	584.6823330	619.4459689	656.2839606	695.3086394	736.6538465	780.4575679
16	418.6008829	443.4921893	469.3635094	497.3031562	527.4040790	558.7651563	591.9910647	627.1526885	664.4875101	703.9999974	745.8520196	790.2132875
17	436.0429860	461.9710299	488.4412595	516.5449540	546.3792445	582.0470374	616.6573586	653.3257168	692.1744892	733.3333301	776.9396333	823.1388435
18	441.4931183	467.7456678	495.5592752	525.2267560	556.2464850	589.3226234	624.3659786	661.4522883	700.8266704	742.4999967	786.6513493	833.4290751
19	465.1120916	492.7690984	522.3705766	553.1165174	586.0045272	620.8501730	657.7678490	696.8807643	738.2194549	782.2222185	828.7355755	878.0147633
20	470.9299927	498.9287121	528.5965500	560.2285501	593.3293838	628.6108002	665.9894471	705.1617729	747.5484481	791.9999962	839.0947712	888.9899475
21	490.5479088	519.7174381	550.5214163	583.3533727	618.0316494	654.8029165	693.7395278	734.9514027	778.4962996	824.9999955	874.0670823	926.0311947
22	496.4797576	526.2138757	557.5041841	590.5551111	625.7772990	662.9879529	702.4[illegible]27[illegible]9	744.[illegible]7808236	788.4300034	835.3124955	884.9827559	937.6355845
1	523.2511025	554.3652351	587.3295106	622.2539440	659.2550925	698.4564440	739.9888293	783.9908851	830.6093859	879.9999949	932.3275226	987.7666073

Format of table storing the Shruti's information:

Table 2

Shruti_id	Shruti_name	Frequency (Hz)	Distance

Format of table storing the Raga's information:

Table 3

Raga_id	Raga_name	Aaroh	Avroh	Pakad

Format of table storing the Singer's information:

Table 4

Singer_id	Singer Name	Frequency notation

2) Temporary Tables:

Format of table storing the Temporary Header's information:

Table5

Temp_id	Start_time	End_time	Shruti_id	Raga_id

Format of table storing the Temporary Detail's information:

Table 6

Temp_id	Shruti_id			Time_details

IV. ALGORITHM

CLUSTERING PROBLEM

N1 N2

Clustering[4] is a formal study of algorithms and methods for classifying objects without category labels. A cluster is a set of objects that are alike, and objects from different clusters are not like. The set of n objects X={X1, X2…., Xn} is to be clustered. Each $X \in Rp$ is an attribute vector consisting of p real measurements describing the object. The objects are to be clustered into non overlapping groups C = {C1, C2… Ck} (C is known as a clustering), where k is the

number of clusters, C1UC2U….UCk = X, $\neq \varphi$ *i C* and $\cap = \varphi$ *i j C C* for i ≠ j. The objects within each group should be more similar to each other than to objects in any other group, and the value

of k may be unknown. If k is known, the problem is referred to as the k-clustering problem. Many methods described in the literature assume that k is given by the user , these methods search for k clusters according to a predefined criterion. Doing so, the number of ways of sorting N objects into k clusters is given by Liu [5]: Thus, there are a large number of possible partitions even

for moderate N and k (e.g. NW (25, 5) ≈ 2.5×1015), and the complete enumeration of every possible partition is simply not possible [10]. In other words, it is not easy to find the best partitioning even assuming that k is known. Indeed, this is rarely the case in practice. A usual approach is to run a clustering algorithm several times and, based on the obtained results; choose the value for k that provides the most natural clustering. This strategy assumes domain knowledge and usually has the disadvantage of searching for the best solution in a small subset of the search space. Consequently, these methods have, in general, low probabilities of success.

Another alternative involves optimizing k according to numeric criteria. In this case, k is unknown and the number of ways of grouping N instances into k clusters, considering S different scenarios (each one resulting from a different k),is [5]:Σs *NW*(*N*, *k*) (2) The problem of finding an optimal solution to the partition of N data into k clusters is NP-complete and, provided that the number of distinct partitions of N instances into k clusters increases approximately as kN/k!, attempting to find a globally optimum solution is usually not computationally feasible. This difficulty has stimulated the search for efficient approximation algorithms. Furthermore

traditional clustering algorithms search a relatively small subset of the solution space (these subsets are defined by the number of clusters, the clustering criteria, and the clustering method). Consequently the probability of success of these methods is small. Algorithms such as single-linkage are deterministic and will repeatedly and the same solution for a given data set, whereas algorithms such as k-means conduct a local search starting from an initial partition. In each case, the solution may be a local optimum, which is not necessarily the global solution. This is exacerbated when the solution space is very large. Clearly, we need an algorithm with the potential to search large solution spaces effectively. The genetic algorithms have been widely employed for optimization problems in several domains. Their success lies in their ability to span a large subset of the search space.

GENETIC CLUSTERING ALGORITHM

We proposed a Genetic algorithm to the problem of k clustering, where the required number of clusters is known. Various adaptations are used to enable the GA to cluster and

to enhance their performance. Further the Genetic Clustering Algorithms are tested on databases, which are benchmarks for data mining applications or heuristics are added to enable the GAs to cope with a larger number of objects. Genetic algorithm for the clustering problem fall into the following areas representation, fitness function, operators and parameter values.

A. Representation

Genetic representations for clustering or grouping problems are based on underlying scheme. The scheme represents the objects with gene values, and the position of these genes signifies how the objects are divided amongst the clusters. The use of simple encoding scheme causes problems of redundant codification and context insensitivity. This has led researchers to devise complicated representations and specialized operators for clustering problems. The cluster label based on n bit encoding is simple compared to parameterization of prototype location. In such a representation many genotype translate to a unique phenotype. The notion of cluster labels built into the representation makes little intuitive sense. Such representations have spawned off a set of pre treatment methodologies to make the representations suitable for genetic operators. Let us consider a dataset formed by N instances. Then, a genotype is an integer vector of (N+1) positions. Each position corresponds to an instance, i.e., the 'I'th position (gene) represents the i-th instance, whereas the last gene represents the number of clusters (k). Thus, each gene has a value over the alphabet {1,2,3,y,k}. For instance, in a dataset composed of 20 instances, a possible genotype is: Genotype: 1123245125432533424 5 In this case, three instances {1,2,8} form the cluster whose label is 1. The cluster whose label is 2 has 5 instances {3,5,9,13,18}, and so on. Standard genetic operators are usually not suitable for clustering problems for several reasons. First, the encoding scheme presented above is naturally redundant, i.e., the encoding is one-to-many. In fact, there is k! different genotypes that represent the same solution. Thus, the size of the search space the genetic algorithm has to search is much larger than the original space of solutions. This augmented space may reduce the efficiency of the genetic algorithm. In addition, the redundant encoding also causes the undesirable effect of casting context dependent information out of context under the standard crossover, i.e., equal parents can originate different offspring. For this reason, the development of genetic operators specially designed to clustering problems has been investigated.

B. Fitness Function

Objective functions used for traditional clustering algorithms can act as fitness functions for Genetic Clustering algorithms. However if the optimal clustering corresponds to the minimal objective function value, we will need to transform the

objective function value, since GAs work to maximize their fitness values. In addition fitness values in a GA need to be positive if we are using fitness proportional selection.

C. Genetic Operators

The operators pass genetic information between subsequent generations of the population. As a result, operators need to be matched with or designed for the representation, so that the offspring are valid and inherit characteristics from their parents. Operators used for genetic clustering or grouping include some of the selection, crossover and mutation methods.

1) Selection

Chromosomes are selected for reproduction based on their relative fitness. Thus the representation is not a factor when choosing an appropriate selection operator, but the fitness function is. If all fitness values are positive, and the maximum fitness value corresponds to the optimal clustering, then fitness proportional selection may be appropriate. Otherwise, a ranking selection method may be used. In proposed Genetic Clustering Algorithm, the genotypes corresponding to each generation are selected according to the roulette wheel strategy, which does not admit negative objective function values. For this reason, a constant equal to one is summed up to the objective function before the selection procedure takes place. The highest fitness genotype is always copied into the succeeding generation.

2) Crossover

The crossover operator is designed to transfer genetic material from one generation to the next. The major concerns with this operator are validity and context insensitivity. It may be necessary to check that offspring produced by a certain operator are valid and reject any invalid chromosomes. The proposed Genetic Clustering Algorithm crossover operator combines clustering solutions coming from different genotypes. It works in the following way. First, two represents k1 clusters, the Genetic Clustering Algorithm randomly chooses $c \in \{1,2,...K1\}$clusters to copy into G2. The unchanged clusters of G2 are maintained and the changed ones have their instances allocated to the corresponding nearest clusters (according to their centroids). In this way, an offspring G3 is obtained. The same procedure is employed to get an offspring G4, but now considering that the changed clusters of G2 are copied into G1.Note that, although the crossover operator produces offspring usually formed by a number of clusters that is neither smaller nor larger than the number of clusters of their parents, this operator is able to increase or decrease the number of clusters.

3) Mutation

Mutation introduces new genetic material into the population. In a clustering context this corresponds to moving an object from one cluster to another. Two operators for mutation are used in the Genetic Clustering Algorithm, the first operator works only on genotypes that encode more than two clusters. It eliminates a randomly chosen cluster, placing its instances into the nearest remaining clusters (according to their centroids).

The second operator divides a randomly selected cluster into two new ones. The

first cluster is formed by the instances closer to the original centroid, whereas the other cluster is formed by those instances closer to the farthest instance from the centroid.

OBJECTIVE FUNCTION

The objective function evaluates the fitness of individual strings. All most all partition evaluation functions provide some kind of measure of inter-cluster isolation and/or intracluster homogeneity. For a good partition, there should be appreciable inter-cluster isolation and intra-cluster homogeneity. The homogeneity within a cluster is calculated by the sum of distances between all pairs of objects with in a cluster. We use an objective function based on the Euclidean distance:

(,) ( ) ( ) 2..... ( )
2, 2 2 2

$$i\ j\ i1\ j1\ i\ j\ in\ jn\ dX\ X = x - x + x - x + + x - x$$

(3)

Where Xi= (xi1, xi2….. xin) and Xj = (xj1,xj2…..,xjn) are two dimensional data objects. The calculation of distances between two instances represents the main computational cost of the Genetic Clustering Algorithm.

V. EXPECTED EXPERIMENTAL RESULTS

Below fig. 2 shows that Shrutis verses frequencies used in scale C (SAFED 1) & C# (Kali 1)

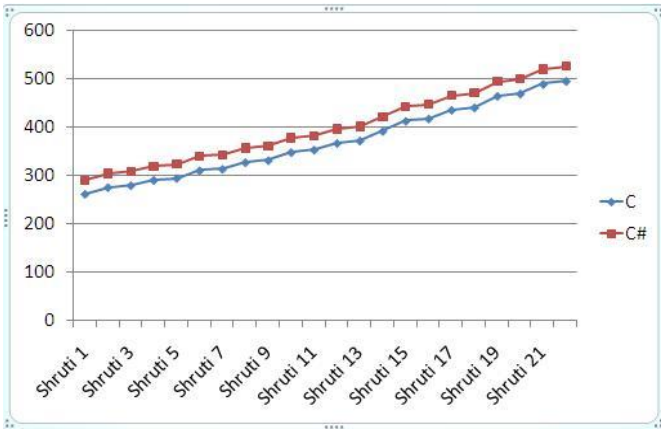

Fig. 2 Shrutis Vs. Frequency w.r.t defined Scale

The following fig. 3 shows the real time performance for Raga Todi played in scale C.

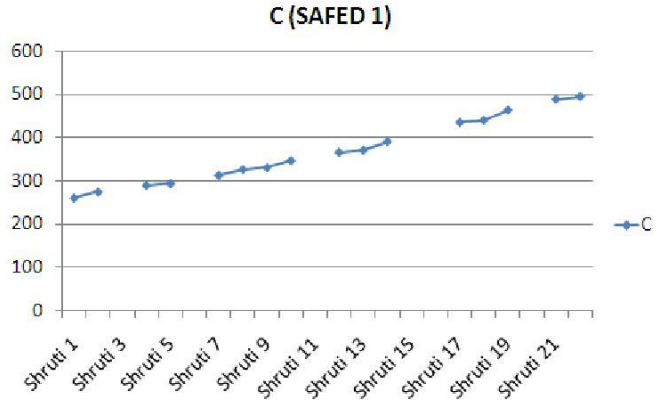

Fig. 3 Shrutis Vs. Frequency in Scale C for Raga Todi

In real time, performance index is calculated. The HARMONIUM PLAYER is expected to solve following problems:

1. Assist a Singer in his practice sessions.
2. Replace the human Harmonium Player.
3. Capture Accurate Frequency of the singer.
4. Play suitable Notes of the Harmonium.
5. Record, Read and Save the Frequencies and Session for future use.
6. Provide Real-Time Assistance.
7. Play all the notes taken by the singer with perfection and in perfect synchronization.
8. Take into consideration the 22 Shruti's, the details of all the Ragas and other Frequency notes of the Singer.
9. Serve two singers efficiently at a time and record details of many others.
10. Play back the singers sung notes during his break times repeatedly.

VI. CONCLUSION

Thus, using this Software, the dependencies and restriction of the singers can be reduced to a great extent.

The singer can practice with his whole and soul. His may even help in producing much more able, well versed and professional singers in the future.

This Software will save the singers time and expenses for the extra professionals as well as will also vanish the urgency that leads problems if the person who plays the Harmonium is unavailable.

It will give the singer, the freedom to practice or riyaz anytime, anywhere and focus alone without being dependent on anyone. We wish to make this Software available at the grass-root level of our nation to preserve our rich culture.

VII. FUTURE SCOPE

This Software can be enhanced and modified according to a person specifically to predict according to the persons way of singing thus creating a way for his own style of music.

Auto-Correct option for the Shruti's so that while learning, any singer can know the right way to sing the Shruti and make no mistakes.

This Software can be developed in the future in such a way that it can be used for a variety of Musical Instruments.

REFERENCES

[1] J.-J. Aucouturier and F. Pachet. Finding songs that sound the same. In Proc. of IEEE Benelux Workshop on Model based Processing and Coding of Audio, pages 1– 8, 2002.

[2] J.-J. Aucouturier and F. Pachet. Tools and architecture for the evaluation of similarity measures: case study of timbre similarity. In Proc. 5th International Conference on Music Information Retrieval, 2004.

[3] D. Bainbridge, S. J. Cunningham, and J. S. Downie.Visual collaging of music in a digital library. In Proc. 5th International Conference on Music Information Retrieval, 2004.

[4] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. In http://www.rawsugar.com/lab, 2004.

[5] A. Berenzweig, D. Ellis, and S. Lawrence. Anchor space for classification and similarity measurement of music. In Proc. IEEE International Conference on Multimedia and Expo, pages I–29–32, 2003.

[6] P. Cano, M. Kaltenbrunner, F. Gouyon, and E. Battle. On the use of FastMap for audio retrieval and browsing. In Proc. 3rd International Conference on Music Information Retrieval, 2002

[7] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery2 (2) (1998), 121–167.

[8] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2001

[9] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In Proc. 10th ACM International Conference on Multimedia, pages 570–579, 2002.

[10] L. Rabiner and B.-H. Juang. Fundamentals of Speech Recognition. Prentice-Hall, 1993