Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 409 – 416

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

T-GEN: A Tabu Search based Genetic Algorithm for the Automatic Playlist Generation Problem

Sneha Antony*, Jayarajan J N

Department of Computer Science and Engineering, Rajagiri School of Engineering & Technology, Kochi- 682 039, Kerala, India

Abstract

Genetic algorithm is a promising technique for generating automatically, sequences of music that satisfy arbitrary user criteria. But this has a disadvantage of selecting clones of the individuals as parents for the next generation. The outcome being poor diversity, affects the final output as genetic diversity serves as an avenue for population to adapt to the steadily changing environment. The proposed method T-GEN, incorporates tabu search in the selection procedure. This enhances diversity and thereby improves the output by giving fitter individuals in the final generation. We describe the concrete implementation of the method which is supported by experimental analysis.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

Keywords: Genetic algorithm; Tabu search; Playlist generation; music; diversity.

1. Introduction

Today, internet has emerged to become the most important and popular medium of music distribution. This has brought along with it, demands for fast and reliable ways to organize, manage, access, and discover music online. In recent years, digitalisation of music has made it a mere necessity to have music services to manage the huge collection of songs in the database[23]. The Automatic Playlist Generation Problem is to generate a playlist that

* Corresponding author. Tel.: +91 471 2440450.
E-mail address: antony.sneha@yahoo.com.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.038

satisfy user constraints, from the songs in the music database. To date, many applications that have been designed to generate playlists have risen to popularity[6]. Each of the applications is based on a variety of approaches.

The Local Search based approach[15] has the disadvantage of being stuck in the local optimum[24]. Simulated Annealing[7] has a mechanism to escape from the local optimum, but it is slow and expensive to compute[25]. The Similarity based approaches[9-12] is not consistent among different human cultures/religions and can be context independent at times[6]. The disadvantage of the Markov Chain based approach[16-18] is that the assumption on which they are based are too strong. The user's choice of next track to listen to may or may not depend on the previous track heard[26]. Genetic Algorithm[1] is an efficient algorithm to solve optimisation problems which can be represented by chromosomal encodings and has multiples solutions. Genetic Algorithm opts for the best fit solutions in each generation. So as explained in Section 3, there is a possibility for clones of the best solution to get generated repeatedly. This repetition and selection of best solutions reduces diversity. The near optimum results of genetic algorithm can be improved if the diversity can be enhanced. This paper proposes an approach that strives to meet this goal by incorporating tabu search into genetic algorithm.

The paper is organized as follows: Section 1 contains the introduction. Section 2 has a brief overview of Genetic Algorithm followed by the problem statement in Section 3. Section 4 elaborates on the proposed approach, T-GEN. Section 5 shows the experimental analysis. Section 6 concludes the paper.

2. Genetic Algorithm

Genetic algorithm is a heuristic search and optimisation technique that mimics the process of natural evolution. The mechanism is simulated through three operators: selection, crossover and mutation[19-22]. Genetic algorithm derives its inspiration from nature wherein even the smallest creatures are able to adapt to changing environments through the process of evolution. Applying the nature's process of evolution to the computer systems has been a topic much researched upon. Genetic algorithms are self learning algorithms and these can be applied only to problems whose solution can have a chromosomal encoding. To apply the genetic operators of selection, crossover and mutation, an initial population is randomly generated. Genetic algorithm is an iterated process and the population in each iteration is called a generation. The fitness of each individual in a generation is evaluated and the algorithm stochastically opts for the best fit in each generation. Crossover and mutation are applied on the selected individuals to form a new generation. This new generation is then used for the next iteration. The algorithm terminates when the preset maximum number of generations is reached or when the threshold of fitness level is attained[8]. The genetic operators are described below.

2.1. Selection

The selection process is used to stochastically select the best fit from the population in each generation. The fitness of a solution is problem dependent and is a measure of how good each solution is. In the playlist generation problem, the fitness of a playlist is the number of constraints satisfied by it.

2.2. Crossover

The crossover operator is applied after the selection operator and is used to create new solutions from the existing solutions available in the mating pool. This operator exchanges the gene information between two solutions in the mating pool, to generate a new solution. This new solution will have features (gene characteristics) of both its parents. The process is continued until a new population of appropriate size is obtained.

2.3. Mutation

Mutation is the process of introducing new features in to the solution strings of the population pool after crossover, to maintain diversity in the population. Mutation randomly selects a gene from the chromosome of the individual and switches its bits.

The average fitness of the population generally increases after applying the genetic operators. This is because only the best individuals from the population are allowed to breed. A small number of less fit solutions are also included to bring about diversity. Theses operators are applied repeatedly on the generation till the termination condition is reached. The termination condition can be a limit on the number of generations or a set threshold of fitness value for the solution.

3. Problem Statement

The principle of genetic algorithm is to select the best and to discard the rest. The fitter individuals have a higher chance of surviving to the next generation and these are the ones selected to breed new solutions in the crossover process. As this process continues for several generations, clones of the fitter individuals tend to get formed. This is because the crossover process tends to select the best fit repeatedly, leading to lesser variety of gene characteristics in the population. On running the algorithm over several generations we see that the population contains clones of the same playlists. This definitely reduces diversity and hence the output.

Diversity is the backbone of evolution. Genetic diversity is fundamental to species survival, and to the continued evolution of new species[2-5]. A 2007 study conducted by the National Science Foundation found that genetic diversity and biodiversity are dependent upon each other—that diversity within a species is necessary to maintain diversity among species, and vice versa. The proposed method T-GEN solves this issue by incorporating tabu search into the genetic algorithm.

4. Proposed Method: T-GEN

In order to introduce more diversity into the population and to decrease the formation of clones in the population, T-GEN incorporates tabu search into the crossover process of genetic algorithm. The underlying assumption behind our approach is that enhancing the diversity will surely lead to fitter individuals (solutions). Before we dwell into T-GEN we will start with a brief description on tabu search.

4.1. Tabu Search

Tabu search is a meta-heuristic search method employing local search methods used for mathematical optimization[13]. Local search techniques randomly generate a single solution and then jump from the solution to its neighbouring solution in the hope of finding a better solution. Neighbouring solutions are generated by making small changes to the solution in hand. Tabu search enhances the performance of the local search techniques by using memory structures that keeps track of the visited solutions. If a solution has been previously visited within a specific short-term period, it is marked as ‘‘tabu’’ so that the process does not repeat itself. Tabu search avoids being stuck at local optimum because of the memory structures and as a result, this becomes an advantageous technique for solving combinatorial optimization problems[14]. The steps of tabu search are as follows:

1. Generate a solution randomly.
2. Evaluate its fitness and insert it into tabu list.
3. Generate neighbourhood of the solution and evaluate fitness of each.
4. Select the best admissible solution and insert it into tabu list.
5. Update tabu list.
6. Check whether stopping criterion is satisfied, if so, get final solution, else repeat the process with the best admissible solution selected.

4.2. System Architecture

The system architecture is shown in Fig 1. It is composed of 4 modules: User Interface module, User Constraint module, Music database and the Playlist Generation module.

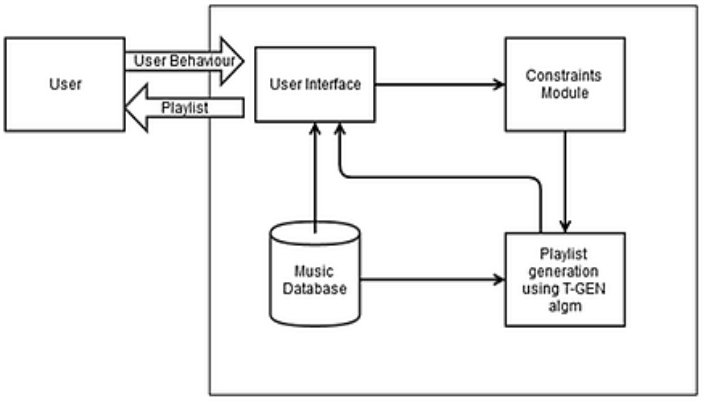

Fig. 1. System Architecture for the Playlist Generation System

- *User Interface module*: The user interface gets user behaviour from the user and dispatches commands to the algorithm and displays results of the algorithm.
- *Music Database module*: The music database is the online or offline collection of songs of the user.
- *User Constraints module*: This module contains all the user constraints. In the proposed system all the user constraints have been preset by the user depending upon his taste. These are the user behaviour given to the User Interface module.
- *Playlist Generation module*: This module is the where the core algorithm works. It uses the Tabu Search based Genetic Algorithm(T-GEN) to solve the automatic playlist generation problem. It uses the constraints from the Constraint Module and songs represented by song Ids from the Music Database to generate the final playlist(represented by song Ids) and delivers it to the User Interface. The UI fetches the songs from the music database and outputs the result to the user.

4.3. Implementation

The playlist generation problem is to automatically generate playlists that satisfies user constraints, from a large music database. A playlist is a set of songs from the music database. Each song is represented by a set of attributes. We have used 10 attributes in this system, namely: title, album, singer, lyricist, genre, mood, language, duration and year. Each song will have values for each of these attributes, and this will uniquely identify a song. In the proposed system, the set of songs in the playlist is represented by their songIds.

The fitness of a solution (playlist) is determined by the number of constraints satisfied by each song in the playlist. For the purpose of comparison between the two approaches, T-GEN and Genetic, 2 types of constraints have been used: Include and Exclude. The Include constraint mentions the attributes of songs which must be included in the playlist and the Exclude constraint mentions the attributes that must be excluded. The fitness is calculated by incrementing the value for each Include constraint attribute satisfied by each song in the playlist and by incrementing the value once if the Exclude constraint attribute is not present in any of the songs in the particular playlist. For a playlist P, and set of constraints C, the fitness function F(P) can be given as in (1).

$$F(P) = \sum_{C_j \in C} satisfy(P, C_j) \tag{1}$$

where,

$$satisfy(P, C_j) = \begin{cases} 1, & \text{if P satisfies } C_j \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The Tabu Search based Genetic Algorithm uses the basic structure of Genetic algorithm. The crossover function of genetic algorithm is incorporated with tabu search so as to bring in more diversity to the population. The prototypical algorithm of the tabu search based crossover function is given below.

```
Function tabu_crossover()
{
        TabuList ← ϕ
        count ← 0
        while(count<= maxcount)
        {
                Select 2 parents from the mating pool, P1 & P2
                If P1 ϵ TabuList
                {
                        Generate neighbours of P1
                        Select best fit of neighbours, Nbest
                        P1 = Nbest
                }
                If P2 ϵ TabuList
                {
                        Generate neighbours of P2
                        Select best fit of neighbours, Nbest
                        P2 = Nbest
                }
                If TabuList = maxsize
                        Pop First two elements
                Insert P1 and P2 into TabuList
                Perform single point crossover to generate offspring O1 & O2
        }
}
```

Algorithm 1. Algorithm of Function tabu_crossover()

The initial population is randomly generated and a Roulette wheel selection approach is used to select individuals in the mating pool. Parents are selected from the mating pool to generate offspring using the modified crossover function. The crossover probability of 0.7 is used for the process. Then single bit mutation with a probability of 0.1 is carried out on the population formed after crossover. For the mutation process, a point is randomly selected in the playlist and the particular songId is changed to another randomly generated number which indicates another song in the music database. The newly formed population acts as the next generation and the same process is carried out till the termination condition is reached. The functioning of the crossover process in T-GEN is as follows:

- The parents selected initially from the mating pool are put into the tabu list.
- A single point crossover is applied to the parents to generate offspring.
- Now on selecting parents again from the mating pool, we check whether these are the same as previously selected by checking the tabu list elements.
- If either of the parents are found to be the same, neighbours of the particular parents are generated using simple move operations like addition, subtraction and replacement.

- The best fit amongst the neighbours is selected to act as the parent. The newly selected parents are also added to the tabu list.
- If the tabu list exceeds its preset limit, the first in element is popped out.

5. Experimental Analysis

The tabu search based genetic algorithm, T-GEN, was tested along with the standard implementation of the genetic algorithm over the same set of inputs. T-GEN outperformed the standard genetic algorithm in all cases. For the experimental analysis, a music database with 5000 songs and a playlist size of 5 has been used. The number of constraints preset by the user is set to 20. Both the algorithms, T-GEN and genetic were made to run in the same program for 100 generations, using the same randomly created initial population. The results of the experiment are as shown in Fig.2. The program was made to run 25 times and the fitness of the final output (playlist) is plotted in each case. As seen from the graph, the proposed method satisfies all 20constraints in most of the cases. This proves that by including a controlled diversity factor, the results can be improved.

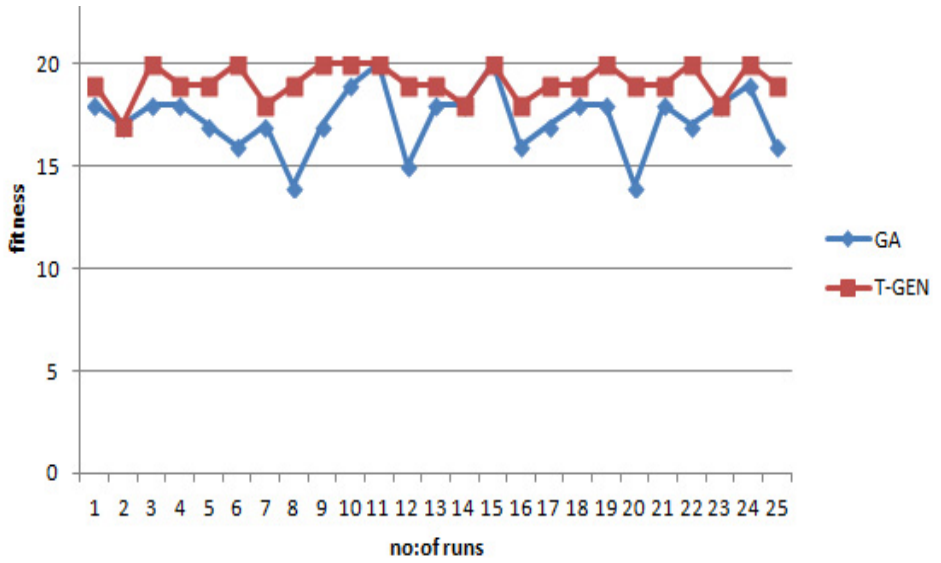

Fig. 2. Graph representing comparative results

To prevent the issue of clones of the solutions being generated, a set operator can be used so as not to set the same solutions repeatedly. Though using the set operator overcomes the issue, it does not give good results as a small number of less fit and a small number of clones of highly fit individuals are necessary to drive the evolution process. Using the set operator, the clones are eliminated completely. T-GEN outperforms it as it does not completely stop clones of best fit individuals from being selected but ensures that these clone selection does not hinder the diversity. In does this by using the memory structure feature of tabu list which prevents previously selected solutions from being selected again for a specific time period.

For the purpose of experiment, the population size has been set to 25. The initial population is randomly generated and in each iteration, the average fitness of the population increases. Fig.3 shows the increase in value of best fit in each generation, on implementing T-GEN. The program was made to run for 100 generations. T-GEN reached the maximum fitness value of 20 in the 68^{th} generation, thereby stopping the program there.

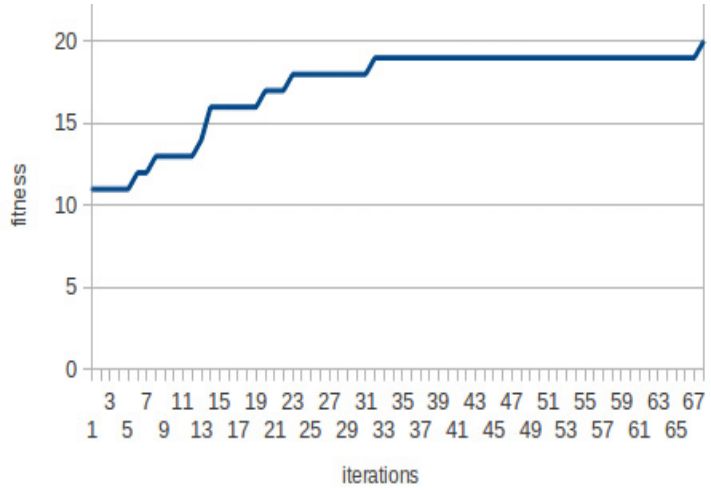

Fig. 3. Graph depicting fitness of best fit in each iteration.

6. Conclusion and future work

Various kinds of promising music services have been proposed to deal with large music collections. This paper proposes a novel approach for automatic playlist generation by combining genetic algorithm and tabu search, and discusses the disadvantage of the existing system that the proposed system overcomes. The new approach, T-GEN, improves the diversity of the population and thereby enhances the output by generating fitter individuals.

As future work more number of constraints can be taken into account. Apart from the constraints preset by the user, data mining techniques can be used to derive constraints automatically depending on listening history of the user.

References

1. Jia-Lien Hsu and Shuk-Chun Chung. *Constraint based Playlist Generation using Genetic Algorithm*. IEEE 2011.
2. Diaz-Gomez, Pedro A., and Dean F. Hougen. *Empirical Study: Initial Population Diversity and Genetic Algorithm Performance*, In Artificial Intelligence and Pattern Recognition; 2007. pp 334-341.
3. Nsakanda, Aaron Luntala, Wilson L Price, Moustapha Diaby, and Marc Gravel. *Ensuring population diversity in genetic algorithms: A technical note with application to the cell formation problem*. European journal of operational research; 2007. vol 178, no. 2, pp: 634-638.
4. Zhu, Kenny Qili. *A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows*. In Proceedings of Tools with Artificial Intelligence. IEEE 2003.
5. E. K. Burke, S. Gustafson, and G. Kendall, *Diversity in genetic programming: An analysis of measures and correlation with fitness*. IEEE Transactions on Evolutionary Computation; 2004. 8(1) pp:47–62.
6. Sneha Antony, Jayarajan J N. *Survey on Playlist Generation Techniques*. International Journal of Advanced Research in Computer Engineering & Technology(IJARCET); 2014.Volume 3, Issue 2.
7. Steffen Pauws, Wim Verhaegh and Mark Vossen. *Music Playlist Generation using Simulated Annealing*. Elsevier Science; 24 March, 2006.
8. Angus R Simpson, Graeme C Dandy and Laurence J Murphey. *Genetic Algorithms compared to other Optimisation Techniques for Pipe Optimisation, Journal of Water Resouce Planning and Management*. 1994. Volume120, No. 4.
9. Flexer A, Schnitzer D, Gasser M and Widmer G. *Playlist generation using start and end songs*. Proceedings of the international conference on music information retrieval; 2008.
10. Pohle T, Pampalk E and Widmer G. *Generating similarity based playlists using traveling salesman algorithms*. Proceedings of the 8th international conference on digital audio effects (DAFx05); 2005.
11. Pohle T, Knees P, Schedl M, Pampalk E and Widmer G. *Reinventing the wheel: a novel approach to music player interfaces*. IEEE Trans Multimedia; 2007. 9(3). pp :567-575.

12. Dai Y, Ye HW and Gong SJ. *Personalized recommendation algorithm using user demography information*. Proceedings of the knowledge discovery and data mining; 2009. pp 100-103.
13. Fred Glover. *Tabu search-part I*. ORSA Journal on computing ; 1989. 1(3), pp: 190-206.
14. Jia-Lien Hsu and Ya-Chao Lai. *Automatic playlist generation by applying tabu search*. Springer-Verlag Berlin Heidelberg ; January2013.
15. Steffen Pauws, Wim Verhaegh and Mark Vossen. *Fast Generation of Optimal Music Playlists using Local Search*. Proceedings of the 6th International Conference on Music Information Retrieval(ISMIR); 2006.
16. McFee B and Lanckriet G. *The Natural Language of Playlists*. Proceedings of ISMIR; 2011.
17. Chen S, Moore J, Turnbul D and Joachims T. *Playlist Prediction via Metric Embedding*, Proceedings of KDD; 2012.
18. Debora C Correa, Alexandre L.M.Levada and Luciano Costa. *A Graph based method for Playlist Generation*. The 9th International Symposium on Computer Music Modelling and retrieval; 2012.
19. T. Mitchell. *Machine Learning*. The McGraw-Hill Companies; 1997.
20. Genetic algorithm, available at http://en.wikipedia.org/wiki/Genetic Algorithm.
21. R L Haupt and S E Haupt. *Practical Genetic Algorithm*. John Wiley & Sons; 2004.
22. S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer-Berlag New York; 2009.
23. Fields B. *Contextualize your listening: the playlist as recommendation engine*. PhD Thesis, Goldsmiths, University of London; 2011.
24. Lourenço, Helena R, Olivier C Martin, and Thomas Stutzle. *Iterated local search*, arXiv preprint math/0102188; 2001.
25. Lin, Lin, and Chen Fei. *The Simulated Annealing Algorithm Implemented by the MATLAB*. International Journal of Computer Science Issues (IJCSI); 2012. 9(6).
26. Bonnin, Geoffray, and Dietmar Jannach. *A Comparison of Playlist Generation Strategies for Music Recommendation and a New Baseline Scheme*. Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence; 2013.