Playlist Generation Using Reactive Tabu Search

Sneha Antony and J.N. Jayarajan

Department of Computer Science & Engineering
Rajagiri School of Engineering & Technology
Kochi, India
antony.sneha@yahoo.com, jayarajanjn@rajagiritech.ac.in

Abstract. This paper proposes a reactive tabu search algorithm to solve the automatic playlist generation problem effectively. The conventional tabu search, though produces promising results, has the disadvantage of the tabu size being fixed. The tabu size is fixed by the user and it determines the capacity of the memory. A preset parameter does not yield appropriate results in a heuristic algorithm and so there must be mechanisms to make the tabu list size variable in order to get optimum results. The reactive tabu search, has mechanisms to make tabu size variable and thereby avoids often repeated solutions. This proposed method for the playlist generation problem yields better results than the conventional tabu search. The concrete implementation of the method has been described, which is supported by experimental analysis.

Keywords: Reactive tabu search, prohibition period, Tabu search, playlist generation, music.

1 Introduction

A revolution has taken place in the way we as a society listen to and consume music. The evolution in the distribution of music from having to hear it at the performance place to being able to hear it at one click today, has happened because of the emergence of the internet. This has brought about with it demands for fast and reliable ways to discover music according to the user's preference online. The problem here is, with the million song collection available online, how can a user discover or select a few songs suitable for him, in the most efficient manner. The Automatic Playlist Generation Problem is to generate a playlist that satisfy user constraints, from the songs in the music database [5].

The playlist generation problem is a combinatorial optimization problem. An exhaustive search in this case may not be feasible so a near optimum result that can be computed with minimal cost is desired. The popular meta-heuristic algorithms for the combinatorial optimization problems include ant colony optimization, particle swarm optimization, simulated annealing, genetic algorithms and tabu search. A study done in [4], points out the effectiveness of tabu search when compared to other meta heuristics. In [9], the specific strategies of tabu search is compared and contrasted with strategies based on evolutionary algorithms. It shows how genetic algorithm procedures can be improved by alternate

S.C. Satapathy et al. (eds.), *Emerging ICT for Bridging the Future – Volume 2*,
Advances in Intelligent Systems and Computing 338, DOI: 10.1007/978-3-319-13731-5_12

frameworks like tabu search that use adaptive memory structures. Hence, tabu search has emerged as a promising technique to solve combinatorial optimisation problems like playlist generation.

A tabu search based method to solve this problem is given in [1]. But the problem with the approach is that the internal parameter of tabu size has to be preset manually by the user. It is not appropriate to fix parameter value used in a meta heuristic algorithm. There must be mechanisms to change the value according to different problems and more importantly according to different runs of the same program. This paper proposes a method to solve this problem by using reactive tabu search. This work is more distinguished as it uses adaptive tabu tenure. A reactive tabu search brings in the element of machine learning whereby the size of the tabu tenure is automated, by monitoring its behaviour in each iteration of a run of the program. The search process is monitored, and the algorithm responds to the formation of cycles by adapting the tabu list size.

The paper is organised as follows: Section 2 gives a brief overview on the working of tabu search. Section 3 mentions the problem with manually setting the value of tabu tenure. This is explained with the help of graphs. Section 4 elaborates on the proposed approach with its experimental results in section 5. Section 6 concludes the paper.

2 Tabu Search

Tabu search is a meta-heuristic search method employing local search methods used for mathematical optimization [10]. 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 same solutions are not selected again. 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 [1]. In [1], the automatic playlist generation has been solved using tabu search. The results of this approach, given its benefits over evolutionary methods, are surely promising. The steps of tabu search are as follows:

1. Generate a solution randomly and set it as the $Best_{global}$.
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, $Best_{admissible}$ and insert it into tabu list.
5. Update tabu list, by popping 1st In value if the size exeeds the limit.
6. If $Best_{admissible} > Best_{global}$, then set $Best_{global} = Best_{admissible}$
7. Check whether stopping criterion is satisfied, if so, get final solution, else repeat the process with the $Best_{admissible}$ solution selected as the current solution.

The two main concepts in tabu search are that of intensification and diversification [7]. Intensification is the process by which the algorithm thoroughly checks the portions of the search space that seem promising, in order to make sure that the best solutions in these areas are found. Diversification is the process by which the algorithm prevents being stuck in a local optimum by forcing the search into previously unexplored areas of the search space.

3 Problem Statement

The major task when applying a heuristic algorithm to a problem is the representation of the problem and the fitness function. Setting these two items will determine the bridge between the algorithm context and the original problem. Determining the internal parameters of the heuristic algorithm will depend on both the representation taken for the particular problem and the fitness function. Due to the element of randomness involved, setting the right parameter values can only be done on a trial and error basis. Even then, getting them right is not possible because these values tend to change from problem to problem and even with each run of the same problem. In case of tabu search, the tabu list size is kept fixed. It is this tabu tenure that determines the occurrences of cycles when searching for the solution. The tabu tenure or prohibition period determines the time for which a previously selected solution is prevented from being selected again. Setting the prohibition period low will hinder the diversification process of tabu search, whereby getting stuck in a local optimum. Setting the value high, will hinder the intensification process whereby missing out on spotting the best solution. Therefore setting the value must strike a balance between the intensification and diversification elements [11].

Previous works on the heuristic algorithms for playlist generation have been implemented by presetting the parameter values based on trial and error method. Fig. 1 shows the output of the playlist generation problem solved using tabu search when using a prohibition period of 3 and 5 respectively. The program was made to run for 15 times and the fitness of the output was plotted. The fitness of a solution (playlist) is the number of constraints satisfied by it. The number of constraints given by user is 20. Both the programs were made to run with the same initial solution in parallel for 50 iterations in each run. In figure 1 it can be seen that while at some runs a prohibition of 5 gives better output, at some points it is tabu search with prohibition period 3 that gives better output. In the 2nd run, prohibition period of 5 gives output with fitness value 19, whereas prohibition period of 3 gives only 17. In case of the 6th run, prohibition period 5 gives 16 whereas prohibition period 3 gives 19. This shows that presetting values for a meta heuristic algorithms is not appropriate. There must be mechanisms to automate this process and dynamically generate the prohibition period in each iteration.

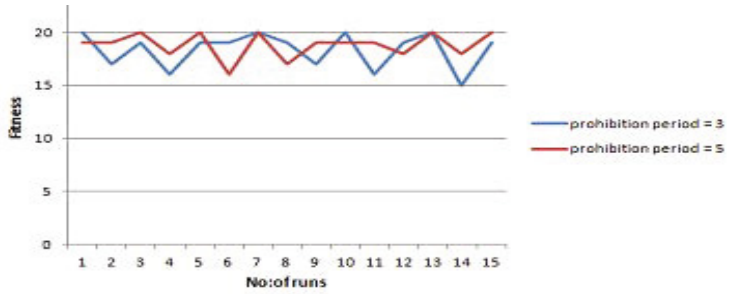

Fig. 1. Graph showing comparison between prohibition period of 3 and 5

Globally there are two methods for setting parameter values: parameter tuning and parameter controlling. Parameter tuning is the age old method in which the value is set before the run of the program. Parameter control is a alternative to parameter tuning as it starts with an initial value which gets changed during the run of the program [8]. The proposed method of reactive tabu search for the automatic playlist generation problem advocates the use of parameter control.

4 The Proposed Method : RTS

Reactive Search Optimisation uses machine learning concepts on local-search heuristics. The Reactive Search framework introduces feedback schemes in heuristics for combinatorial optimisation problems [2][12]. The conventional meta heuristics like genetic algorithm, simulated annealing, tabu search etc, require manual adjustment of search parameters in advance for efficient search. The disadvantage of parameter tuning is corrected by parameter control in reactive search optimisations. Reactive Tabu Search (RTS) is an extension of tabu search and forms the basis for reactive search optimisations. RTS explicitly monitors the search and reacts to the formation of cycles by adapting the prohibition period [6].

RTS uses a feedback-based tuning mechanism of the tabu list size(prohibition period) and strikes a balance between the intensification and diversification mechanisms. Initially the prohibition period is fixed. When a solution previously selected is encountered, the prohibition period increases. If a repetition does not occur for a specific short period of time, the prohibition size is decreased.[3] In order to avoid the long search cycles, RTS uses the escape procedure which is a mechanism for diversification.

4.1 System Architecture

The system architecture is shown in Fig.2. It mainly consists of 4 modules: User Interface module, User Constraint module, Music database and the Playlist Generation module.

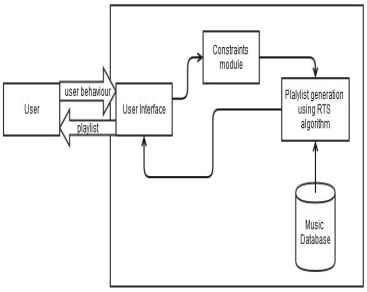

Fig. 2. System Architecture of proposed Playlist Generation System

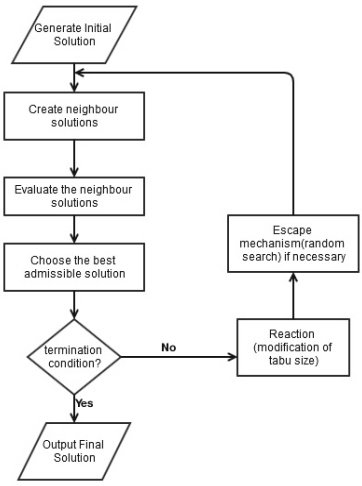

Fig. 3. Flowchart for Reactive Tabu Search

- User Interface module: The user interface gets user behaviour from the user and dispatches commands to the algorithm and displays results of the algorithm. It is this module that interacts with the user.
- Music Database module: The music database is the online or offline collection of songs of the user. It is from this set of about million songs that the playlist is generated.
- 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.
- Playlist Generation module: The playlist generation module is the core algorithm. It uses the Reactive Tabu Search to solve the automatic playlist generation problem.

4.2 Implementation Details

The playlist generation problem is to automatically generate a playlist 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 vector 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,

RTS and Tabu Search, 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 :

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

, where

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

The flowchart of the reactive tabu search algorithm is given in Fig.3. Initially a solution (in this case: a playlist)is generated randomly. The fitness of the solution is evaluated and it is entered into the tabu list. The size of prohibition period is set to be 1 initially. Then neighbours of the solution are generated and evaluated. The best fit among them is selected. The tabu list is updated in a reactive manner. If the solution happens to be the same as previously visited, then the prohibition period is increased or else it is decreased by a percentage. The increase value can be between 1.1 and 1.3. The decrease value can be set between 0.8 and 0.9 [6]. The increase and decrease values are set to be 1.1 and 0.9 in the program. The program terminates when a set no:of iterations is reached or when the maximum fitness value is attained by the playlist.

Two terms that come about in reactive tabu search are: Escape mechanism and Aspiration criterion. Escape mechanism is done so as not to be stuck in a local optimum. The new iteration in this case will start from a random point other than selecting best admissible from the neighbours of the current solution. This is done when the solution selected is found to be in the repeatedly-visited-list. Aspiration is the method by which some tabu moves are disregarded to obtain better local solutions. Generally these are performed when a tabu move leads to an output better than the last best known so far.

Once the best from the neighbourhood has been selected, it is included in the tabu list. The repetition interval of solutions is monitored by maintaining a separate visited list which contains the no:of repetitions and the interval of repetitions of each visited solution. The selected solution is compared with the best known solution so far and then the best is updated accordingly. The process is repeated till the termination condition is reached. The termination condition can be a threshold of fitness level or a limit to the no:of iterations. When the program terminates, it returns the best solution known so far.

5 Experimental Analysis

The Reactive Tabu Search algorithm for playlist generation and Tabu Search were tested on the same set of inputs. The music collection, values for attributes of songs, playlist size, constraint values and even the initial random playlist generated were made sure to be the same for both the programs. Both algorithms were executed in the same program simultaneously. Fig. 4 shows the output obtained on recording values when the program was made to run 20 times. The value of fitness of the returned playlist is plotted for both the algorithms. Reactive tabu search (RTS) is seen to outperform Tabu search (TS). For the experimental setup, 50 constraints were given by the user beforehand. The playlist size of 10 has been used. The program was executed 20 times, each for 100 iterations.

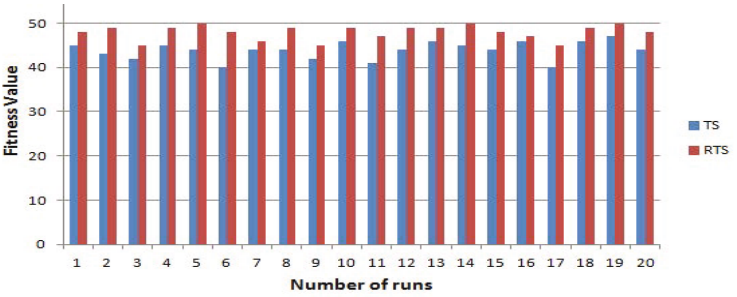

Fig. 4. Graph showing comparative results for TS and RTS

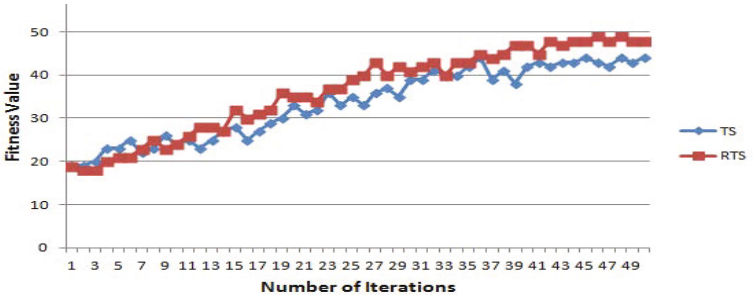

Fig. 5. Graph showing comparative results in progress of a run for TS and RTS

Fig.5 shows the progress in selection of solutions in each iteration of a single run. In this case also both the algorithms are executed with the same set of inputs. The best admissible solution selected from the neighbourhood of the current solution is plotted in each iteration. The program is made to run for 50 iterations. From the graph we see that, the progress of RTS is greater than that of TS. RTS gives an output with fitness 49, whereas TS gives 44.

6 Conclusion

The automatic playlist generation problem being a combinatorial optimisation problem, can be solved best using heuristic or meta-heuristic algorithms, like genetic algorithms, simulated annealing, ant colony optimisation, particle swarm optimisation or tabu search. Tabu search has been shown to give much promising results compared to others in solving combinatorial optimisation problems. But presetting the prohibition period for tabu search will limit the output to local optimum due to repeated cycles caused by inappropriate prohibition period. Reactive tabu search brings in an element of machine learning and changes the value of prohibition period dynamically when the program is executed. The results obtained are proof for the goodness of the algorithm.

References

1. Hsu, J.-L., Lai, Y.-C.: Automatic playlist generation by applying tabu search. International Journal of Machine Learning and Cybernetics 5(4), 553–568 (2014)
2. Battiti, R., Tecchiolli, G.: The Reactive Tabu Search. ORSA Journal of Computing 6(2), 126–140 (1994)
3. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intellengent Optimisation. University of Trento, Department of Information and Communication Technology, Italy, Technical Report DIT-07-049 (2007)
4. Pirim, H., Bayraktar, E., Eksioglu, B.: Tabu Search: a comparative study. In: Local Search Techniques: Focus on Tabu Search, p. 278 (2008) ISBN: 978-3-902613-34-9
5. Fields, B.: Contextualize your listening: the playlist as recommendation engine. PhD Thesis, Goldsmiths, University of London (2011)
6. Brownlee, J.: Reactive Tabu Search. In: Clever Algorithms: Nature-Inspired Programming Recipes, pp. 79–86 (2011)
7. Gendreau, M.: An introduction to tabu search. International Series in Operational Reasearch and Management Science, vol. 57, pp. 37–54. Springer US (2003)
8. Eiben, E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
9. Glover, F., Laguna, M.: Tabu Search. In: Handbook of Combinatorial Optimization, pp. 621–757. Kluwer Academic Publishers (1998)
10. Glover, F.: Tabu search-part I. ORSA Journal on Computing 1(3), 190–206 (1989)
11. Devarenne, I., Mabed, M.H., Caminada, A.: Adaptive tabu tenure computation in local search. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 1–12. Springer, Heidelberg (2008)
12. Battiti, R., Tecchiolli, G.: Training Neural Nets with the Reactive Tabu Search. IEEE Transactions on Neural Networks 6(5), 1185–1200 (1995)