AUTOMATIC TIME TABLE GENERATION USING GENETIC ALGORITHM

Mrs.G.Maneesha[1], T.Deepika[2], S.BhanuSri[3],N.RaviKumar[4] ,P.SivaNagamani[5].

[1] Assistant Professor, Dept. of Computer Science & Engineering, Dhanekula Institute of Engineering & Technology, Ganguru, Vijayawada, Andhra Pradesh, India.
[2,3,4,5] Bachelor of Technology, Dept. of Computer Science & Engineering, Dhanekula Institute of Engineering & Technology, Ganguru, Vijayawada, Andhra Pradesh, India.

Correspondence should be addressed to Mrs. G.Maneesha maneesha.gudapati@gmail.com,deepusmiley987@gmail.com sykambhanusri@gmail.com

ABSTRACT- Timetable generation is a difficult task but it is very important in educational institutions. Generating time table manually occurs conflits. This automatic timetable generation software generates the timetables automatically by taking the inputs like the number of subjects, teachers, the workload of a teacher, semester, and priority of subjects. In our project, we are going to use algorithms like genetic, heuristic, resource scheduling to reduce the difficulties in generating the timetable.

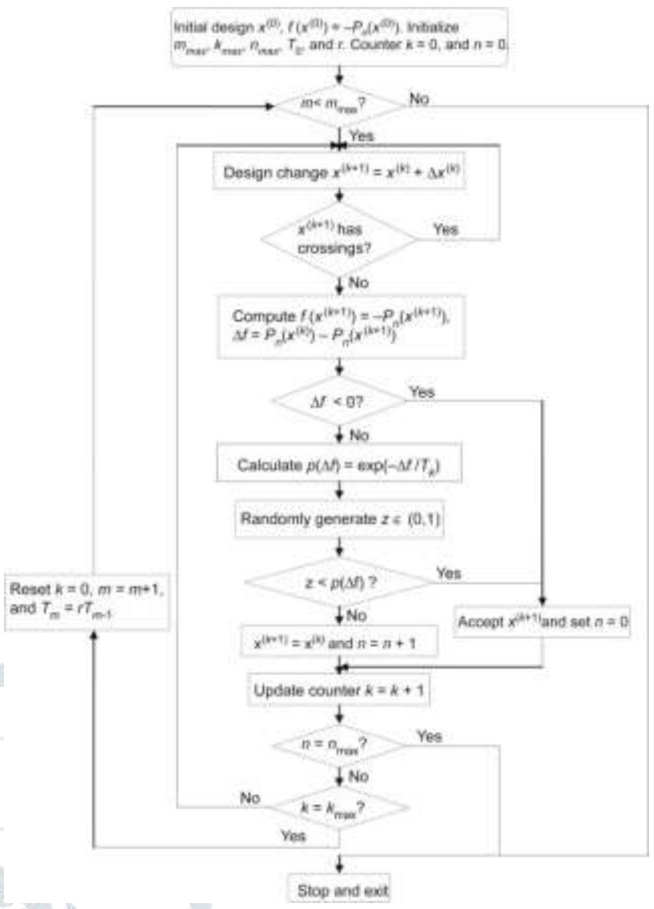
KEYWORDS-Genetic Algorithm, Resource Scheduling

I.INTRODUCTION

Although most of the university administrative work has been computerized, due to the difficulties involved, the schedule is still done manually. Manual programming requires a lot of time and effort. A schedule consists of assigning a certain resource to objects placed in time and space so that they fulfill a set of ideal objectives. The subject of the university class schedule forces us to find some spaces and classrooms to comply with the limitations imposed on courses, teachers, classrooms, etc. This problem is a combinatorial optimization problem, in which the calculation time increases exponentially with the increase in the number of variables. In the last ten years, various methods have been adopted to solve the problem of setting timetables for schools and universities. In our article, this problem is formulated as a constraint satisfaction problem, and we discuss various methods that can handle hard and soft constraints. Under no circumstances may the strict restrictions be violated. For example, two classes cannot be assigned to a teacher at the same time, a student cannot participate in two classes at the same time, a room cannot be opened for more than one class at the same time, and so on. Soft restraints are necessary, but not critical. For example, a timetable should be established so that a group of students does not have to come to the university for classes. It will help to manage all the periods automatically and also will be helpful for faculty to get a timetable on their phone by using the application. The Maximum and minimum workload for faculty for a day, week, and month will be specified for the efficient generation of the timetable

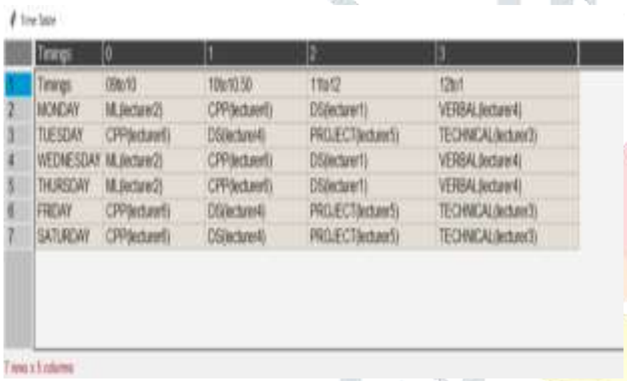
II.ALGORITHM

Stochastic search algorithms are designed for problems with inherent random noise or deterministic problems solved by injected randomness. In structural optimization, these are problems with uncertainties of design variables or those where adding random perturbation to deterministic design variables is the method to perform the search (Leng, 2015). The search favors design with better performance. An important feature of stochastic search algorithms is that they can carry out a broad search of the design space and thus avoid local optima. Also, stochastic search algorithms do not require gradients to guide the search, making them a good fit for discrete problems. However, there is no necessary condition for an optimum solution and the algorithm must run multiple times to make sure the attained solutions are robust. To handle constraints, penalties can also be applied to designs that violate constraints. For constraints that are difficult to be formulated explicitly, a true/false check is straightforward to implement. Randomly perturbed designs are checked against constraints, and only those passing the check will enter the stage of performance evaluation. Stochastic search can be applied to one design or a population of them (Leng, 2015), using for example SA or GA, respectively. Arora (2004) depicts the logic of SA and GA for the convenience of application. A monograph devoted to stochastic search and optimization (Spall, 2003) provides further details on a broad scope, including mathematical theory, algorithm design, and applications in simulation and control. SA is a mimic of the natural process of annealing in metallurgy (Kirkpatrick et al., 1983). The algorithm performs iteratively; the code generates a new candidate design by randomly perturbing the variables of the current elite design that performs best. A unique feature of SA is that its "hill-climbing" property allows inferior designs to be accepted in place of elite ones to expand the search space and prevent the algorithm from becoming trapped in a low-quality local minimum (Leng, 2015). The probability that a suboptimal design is accepted is a function of the magnitude of performance loss and a user-selected parameter. This parameter is tightened as the optimization progresses, reducing the probability of accepting suboptimal designs (Leng, 2015). Two influential parameters of the algorithm are initial "temperature" T0 and the rate at which this temperature is reduced, referred to as the "cooling rate" r (Leng, 2015). The reduction of T occurs when a certain number (k max) of qualified designs has been evaluated. SA terminates after the temperature has been reduced m max times. The product of k max and n max is the maximum number of objective function evaluations, commonly used as an indicator of algorithm efficiency (Leng, 2015). Convergence is said to occur if the elite design does not change over a large number of iterations (n max). See Spall (2003) and Arora (2004) for more detailed discussions of SA.


Initial design $x^{(0)}$, $f(x^{(0)}) = -P_n(x^{(0)})$. Initialize m_{max}, k_{max}, n_{max}, T_0, and r. Counter $k = 0$, and $n = 0$.
$m < m_{max}$?
No
Yes
Design change $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$
$x^{(k+1)}$ has crossings?
Yes
No
Compute $f(x^{(k+1)}) = -P_n(x^{(k+1)})$, $\Delta f = P_n(x^{(k)}) - P_n(x^{(k+1)})$
$\Delta f < 0$?
Yes
No
Calculate $p(\Delta f) = \exp(-\Delta f / T_k)$
Randomly generate $z \in (0,1)$
$z < p(\Delta f)$?
Yes
No
Reset $k = 0$, $m = m+1$, and $T_m = rT_{m-1}$
Accept $x^{(k+1)}$ and set $n = 0$
$x^{(k+1)} = x^{(k)}$ and $n = n + 1$
Update counter $k = k + 1$
$n = n_{max}$?
Yes
No
No
$k = k_{max}$?
Yes
Stop and exit

GAs are popular stochastic search algorithms based on the idea of Darwin's evolution theory (Holland, 1975; Golberg, 1989). Rather than operating on a single design and its perturbation, as in SA, GA operates on a population of designs (Leng, 2015). The designs are then analyzed and ranked according to their objective function performance (fitness). The generation of a new design population includes a random selection of two designs (parents) and a random exchange of a portion of their properties (reproduction). Occasionally, a design is also randomly perturbed (mutation). This process is repeated until an entirely new population (children) is formed (Leng, 2015). Designs with higher fitness have a higher probability of being selected as parents, and thus the performance of the population as a whole should improve as the optimization progresses (Leng, 2015). Similar to other stochastic search algorithms, GA terminates if either a maximum number of iterations is achieved (k max), or convergence is detected. Convergence is said to occur if the elite design does not change over a large number of iterations (n max) (Leng, 2015). Further development of GA under the general category of evolutionary computation and specific details on its application is available in Spall (2003). A summary of GA in unconstrained optimization of CFS columns by Leng et al. (2011) is shown as a flow chart in Fig. 6.2. As the flow chart in Fig. 6.1, the objective again is to maximize the column's axial capacity. In the formulation, the vector x of a given design is rounded to a user-specified precision and converted into a binary string. This is a straightforward process and facilitates the exchange of information between designs. Parent selection is based on the roulette wheel algorithm, and single-point cross-over is used to exchange information between two parents. To handle the constraint on overlapping, the penalty method is used. If element crossing is detected in a new design, the computed strength is penalized by subtracting a large number. The procedure in the flow chart can be readily applied to other design objectives.

III.RESULTS



Time Table

	Timings	0	1	2	3
1	Timings	09to10	10to10.30	11to12	12to1
2	MONDAY	ML(lecture2)	CPP(lecture6)	DS(lecture1)	VERBAL(lecture4)
3	TUESDAY	CPP(lecture6)	DS(lecture4)	PROJECT(lecture5)	TECHNICAL(lecture3)
4	WEDNESDAY	ML(lecture2)	CPP(lecture6)	DS(lecture1)	VERBAL(lecture4)
5	THURSDAY	ML(lecture2)	CPP(lecture6)	DS(lecture1)	VERBAL(lecture4)
6	FRIDAY	CPP(lecture6)	DS(lecture4)	PROJECT(lecture5)	TECHNICAL(lecture3)
7	SATURDAY	CPP(lecture6)	DS(lecture4)	PROJECT(lecture5)	TECHNICAL(lecture3)

7 rows x 5 columns

IV.CONCLUSION

We have shown that the genetic algorithm method is very effective and useful for the timetable problem. Using the method, we describe and showing the great potential of future orientation schedules is fairer to students. The framework appears to be directly applicable to a wide range of other timing-related issues. For example, the experimental results show that a key aspect of its success is the use of the mutation operator described. The GA framework is successful on many practical "university department size" issues, so we seem to be able to demonstrate the expectation that it may also work well on other issues of a similar size and nature. In other words, there is no reason to suspect that the problem you are testing is particularly easy. Compared with other practical problems, there is still a lot of work to be done to understand how performance adapts to larger and different types of scheduling problems.

V.REFERENCES

[1] Prashanta Kumar, Shreedhar Sanakar, Praveen Kumar P, Syed Muhammad Usman, Vani K A. "Automated Timetable Generator Using Machine Learning" in International Research Journal of Modernization in Engineering Technology and Science.

[2] V. Abhinaya, K. Sahithi, K. Akaanksha "Online Application of Automatic Time-Table Generator" in International Research Journal of Engineering and Technology (IRJET).

[3] Akshay puttaswamy, H M Arshad Ali Khan, Chandan S.V, Parkavi.A " A Study On Automatic Timetable Generator" In International Journal Of Science And Innovative Engineering & Technology.

[4] Saritha M, Pranav Kiran Vaze, Pradeep, Mahesh N R "Automatic Time Table Generator" in nternational Journal of Advanced Research in Computer Science and Software Engineering.

[5] Deeksha C S, A Kavya Reddy, Nagambika A, Akash Castelino, K Panimozhi "Automatic Timetable Generation System" in JETIR.

[6] Shraddha Shinde, Saraswati Gurav, Sneha karme "Automatic Timetable Generation using Genetic Algorithm" in International Journal of Scientific & Engineering Research.

[7] AnujaChowdhary "Time Table Generation System" .Vol.3 Issue.2, February- 2014.

[8] Dipti Srinivasan Tian Hou Seow Jian Xin Xu "Automated timetable generation using multiple context reasoning for university models", 2002.

[9] Anirudha Nanda "An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach". International Journal of Machine Learning and Computing.

[10] Bagul, M. R., Pushkar R. Patil, S. C., & Nagare, S. N. (October 2015). A Novel Approach for Automatic Timetable Generation. International Journal of Computer Applications.

[11] Chowdhary, A., Priyanka Kakde, S. D., & Rupal Rushiya, D. G. (February 2015). Timetable Generation System. International Journal of Computer Science and Mobile Computing.

[12] S, D. C., A Kavya Reddy, N. A., & K

Panimozhi, U. S. (April 2015). Automatic Timetable Generation System. JETIR.

[13] Rathod, P. P., Kamlesh K. Lodhiya, M. P., Student CSE, p. S., & S. C. (2016). Automatic Timetable Generator. International Journal of Research in Science & Engineering.

[14] Mittal, D., & Hiral Doshi, M. S. (February 2015). Automatic Timetable Generation using Genetic Algorithm. International Journal of Advanced Research in Computer and Communication Engineering.

[15] M.Nandhini, And S.Kanmani, "Implementation Of Class Timetabling Using Multi_Agents", (2009).

[16] Anirudha Nanda, Manisha P. Pai, and Abhijeet Gole, “An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach”.

[17] Asif Ansari, and Prof Sachin Bojewar, “Genetic Algorithm to Generate the Automatic Time-Table – An Over View”, (2014).

[18] Deeksha C S, A Kavya Reddy, Nagambika A, Akash Castelino, and K Panimozhi, “Automatic Timetable Generation System” (2015).

[19] Prof Er. Shabina Sayed, Ansari Ahmed, Ansari Aamir, and Ansari Zaeem, “Automated Timetable Generator” (2015).

[20] Sandeep Singh Rawat, Lakshmi Rajamani,"A timetable prediction for technical education system using Genetic Algorithm", Journal of Theoretical and Applied Information Technology.

[21] Rushil Raghavjee and Nelishia Pillay “An Application of Genetic Algorithms to the School Timetabling Problem”, School of Information Systems and Technology, Pietermaritzburg Campus University of KwaZulu-Natal.

[22] A.Scharef,“A survey of Automated Timetabling-Artificial Intelligence Review”, 1999.

[23] S.C. Chu and H.L. Fang," Genetic Algorithms vs. Tabu Search in timetable Scheduling", IEEE, 1999, Electronics & Communication dept., University of South Australia, Australia.

[24] Alberto colorni, Marco Dorigo, Vittoria Maniezzo, A Genetic Algorithm To Solve The Timetable Problem, Centre for Emergent Computing.

[25] J.J Grefenstette, editor. Proceedings of the First International Conference on Genetic Algorithms and their Applications. Practice and Theory of Automated Timetabling VI Proceedings of The 6th International Conference.