Comparison of Evolutionary Algorithms for Timetable Scheduling

Anuj Singh Koli[1], Taikhoom Rajodwala[2], Amit Mittal[3]

[1,2,3]*Department of Computer Engineering, Institute of Engineering and Technology, Devi Ahilya Vishwavidyalaya, Indore (M.P.)*

---***---

***Abstract*— Timetable scheduling is a ubiquitous problem, faced by all kinds of educational institutes around the world. Generally, timetables are created manually by faculties or admins but the resultant time tables aren't optimized. If an evolutionary algorithm-based approach is used then faster and optimal results could be achieved. This paper will discuss various evolutionary algorithms that can be used to solve the university timetable scheduling problem. Scheduling problems come under the category of NP-hard optimization problems, which means there is no linear polynomial time solution available to solve these. That's why evolutionary algorithms are perfectly suited for such tasks. In this paper, we are mainly going to discuss Tabu Search, Particle Swarm Optimization, Classical Genetic Algorithm, Memetic Algorithms and (1+1) Evolutionary Algorithm. Furthermore, we have presented a discussion on relevant work done by various researchers, and a brief description of how evolutionary algorithms can be used for timetabling problems. Finally results obtained from various independent research, based on specific use cases they are applicable for, are combined and compared.

***Keywords*—** Constraint Satisfaction, Evolutionary Algorithms, University Timetable Scheduling, Guided Operators, Metaheuristic Search, Tabu Search, Particle Swarm Optimization, Genetic Algorithm, Memetic Algorithm, (1+1) Genetic Algorithm

I. Introduction

Timetable scheduling is a complex problem and belongs to a class of NP-hard computational problems. Which means that it isn't possible to generate a solution in polynomial time but it is possible to identify a correct solution. Additionally, timetable scheduling is a type of constraint satisfaction problem where one needs to identify the valid combination of events and resources based on various constraints. For example, one needs to identify a valid combination of a classroom, faculty, time, and a batch; while ensuring constraints like a given classroom and given batch is free at that particular time.

Generally, timetable scheduling is done manually at universities because there isn't any generic solution that can satisfy different kinds of requirements of various universities. Often, institutions have to compromise with suboptimal timetables that either don't satisfy all required constraints or utilise resources inefficiently. However, recent developments in evolutionary computation have shown some promising results when used in conjunction with human intelligence, it provides us with a close to optimal timetable. Evolutionary computation refers to optimisation algorithms that are inspired by nature and biological evolution. Various algorithms like Tabu Search, Particle Swarm Optimisation, Genetic Algorithm, Memetic Algorithm and (1+1) Evolutionary Algorithm can be used to generate a feasible timetable for universities. Additionally, each of the aforementioned algorithms is suitable for optimisation of a discrete class of constraints and one can choose which algorithm to apply based on their particular use case.

The evolutionary algorithms have some common characteristics. For instance, initially, they all start with a group of possible solutions known as swarm or population. Second, they all use mutation operators (random or guided) to modify active candidates according to the current best candidate and in turn, search the local neighbourhood of a given solution. Lastly, they all apply group intelligence to identify global optimum. Moreover, a common trend is seen among various advancements in different evolutionary algorithms where initially researchers worked on randomised operators with no contextual knowledge, but then they experimented with guided operators which used contextual knowledge to identify the next best solution and found them to be superior to their randomised counterparts.

We have worked on developing the simplest form of GA, i.e. (1 + 1) Genetic Algorithm [18] for solving timetabling problems and verified the results obtained from various studies under consideration. In the (1+1) version of GA, we selected the new best solution between a parent and a child. (1+1) GA converges at a faster rate because we store domain knowledge in local memory and use this knowledge to avoid moving to sub-optimal points in search space. Also, both unguided and guided mutations are used by us and finally, results are compared.

This research paper is aimed to help the future researcher to identify the most suitable algorithm based on their problem requirement. In the upcoming sections, we will discuss various classes of timetable scheduling problems and appropriate algorithms for those classes.

This paper is organised as follows: Section II describes the problem statement under scrutiny, Section III talks about related work done by other academic professionals and literature survey, Section IV deals with describing various extensions of evolutionary algorithms, comparison and use cases, and the results are presented in Section V. Section VI concludes our research and an acknowledgement is made in this section.

II. PROBLEM DESCRIPTION (HARD AND SOFT CONSTRAINTS)

The Timetabling problem (TTP) is a special case of the Job Shop Scheduling problem (JSP) and like JSP, we need to assign a resource to an event. In the case of TTP, we have to assign a combination of [faculty, student, course, room] (resource) to a time slot (event). The total number of events is a product of the number of days and number of working hours. As the number of resources and events increases, their total number of possible combinations rise exponentially and a combinatorial explosion takes place.

The timetable scheduling problem is an optimisation problem and we try to optimise two kinds of constraints, namely hard and soft constraints. Hard constraints are those constraints that can't be violated and violation of which means that solution is infeasible and incorrect. On the other hand, soft constraints are those constraints that could be violated and violation of which means that though the solution is feasible, still it is a sub-optimal and better solution than the current solution exists.

Sample hard constraints:

i. No two professors can attend the same room or the same batch at a time.

ii. No two batches are assigned the same room or the same professor at a time.

iii. No room can host multiple classes at a time.

iv. No batch or professor should be assigned to multiple rooms at a time.

v. Every day must have a break after three hours of continuous classes.

vi. Lectures can only be conducted in the type of classroom allowed based on lecture type.

vii. Each combination of batch and subject must be assigned respective minimum lectures.

viii. Assigned room should have enough capacity to accommodate all students.

Sample Soft constraints:

i. A professor should be assigned classes based on his preference of time.

ii. There shouldn't be more than 2 lectures of the same subject on a given day.

iii. There should be a minimum of 3 lectures and a maximum of 8 lectures in a day.

iv. There should be a minimum ideal time between lectures for both students and faculties.

v. Rooms should be used efficiently and a minimum number of rooms should be used.

vi. Lecture should be uniformly distributed among allowed days.

vii. Each faculty should teach at least 3 days a week.

viii. Schedule for Faculty and students shouldn't be hectic and should have only limited lectures in a day.

ix. Lectures and Lab of a subject must be assigned on the same day.

x. Longer classes must be assigned at the start of the day.

III. RELATED WORKS

The idea of solving timetable scheduling problems with the help of algorithms was first introduced by Werra in 1985[1]. Werra suggested dealing with timetabling problems using graphs and networks. After that, a large number of researchers had shown interest in solving timetabling problems using various algorithms and this led to the formation of PATAT (Practice and Theory of Automated Timetabling) [2] in 1995. Moreover, the International Competition of Timetabling was established with the help of PATAT in 2002 (ITC, 2002). Recently ITC 2019 was conducted with the aim of stimulating research in the domain of complex timetable scheduling problems. Since 1985, various Meta-heuristics methods like Simulated annealing [3], Tabu search [4], Particle Swarm Optimisation, Genetic Algorithms, Memetic Algorithms are proposed to solve the timetabling problem.

A. Tabu Search

Daniel Costa in 1994 proposed basic Tabu Search to compute an operational timetable. To further improve results Zhipeng & Jin-Kao, in 2010[11], used tabu search to deal with the timetabling problem. They have broken down tabu search into three steps: initialization, intensification and diversification. In this case, intensification is used to search in local search space while the final diversification step is used to avoid getting stuck in a local optimum. On the other hand, some researchers tried working with a hybrid approach. For example, Sadaf Naseem Jat & Shengxiang Yang [12] worked on a hybrid Genetic Algorithm and Tabu Search approach, where they used Genetic Algorithm in the first step to generate an initial solution and in the second step Tabu Search is used to further improve the optimality of solution obtained in the first step.

B. *Particle Swarm Optimization*

The particle swarm optimization (PSO) algorithm, proposed by Kennedy and Eberhart [16], is a metaheuristic algorithm based on the concept of swarm intelligence. It is of great importance noting that dealing with PSO has some advantages when compared with other optimization algorithms as it has fewer parameters to adjust. In the early 1990s, several studies regarding the social behaviour of animal groups were developed. These studies showed that some animals belonging to a certain group, that is, birds and fishes, can share information among their group, and such capability confers these animals a great survival advantage. Inspired by these works, Kennedy and Eberhart [16] proposed in 1995 the PSO algorithm, a metaheuristic algorithm that is appropriate to optimize nonlinear continuous functions. After this algorithm was proposed, many modified versions were also proposed such as by Yuhui and Gireesha [17], which involved the application of weight to the main equation of the algorithm to make it more biased and to converge to an optimum point more quickly. The idea behind this weight is to divert the search for global optimum by the particles according to the problem to be solved, that is, instead of using the generalised approach for all problems, it tends to introduce a bias to the equation with weight value specific to that problem.

C. *Genetic and Memetic Algorithm*

A classical genetic algorithm-based approach was used by Esraa A. Abdelhalim in 2006 [5] to generate an optimal solution for Alexandria University. Here Esraa emphasised on maximizing resource utilization and found that timetabling problem is closely related with space optimisation problem as different approaches that are used in solving the space allocation problems can also be used for solving the timetabling problems. Moving forward, some researchers further tried to improve the behaviour of the classical genetic algorithm by improving various components of the genetic algorithm. For instance, Vinayak Sapru in 2010[6] used guided mutation in place of random mutation and found that by using guided mutation, the algorithm converges to a feasible solution in all cases and converges much faster. Similarly, Antariksha Bhaduri in 2009[10] used the memetic algorithm to further improve the approach of finding an optimal solution. Antariksha implemented Genetic Artificial Immune Network (GAIN) and found that convergence rate became much slower for GA when it located a sub-optimal result. However, GAIN continued to converge at an almost similar rate until the valid solution was located.

On the contrary, in classical genetic algorithms, each of the candidate solutions or chromosomes is a passive entity, that is, operators work in random fashion rather than focusing on the problem at hand. While in hybrid evolutionary algorithms or memetic algorithms candidate solutions work as an active entity, that is, operators are directed/guided and capable of performing local refinements. Therefore to convert a GA to a Memetic Algorithm (MA), one needs to modify its basic operators like mutation, selection, and crossover. For instance, Burke, E. & Newall [13] dissected basic mutation operator into light and heavy mutation operator and combined hill climbing to create a memetic algorithm for university exam timetabling. Similarly, Özcan, E. and A. Alkan [14] implemented violation directed mutation and violation directed hierarchical hill-climbing method and showed some promising results. According to Pablo Moscato [15], MA is close to a form of population-based hybrid genetic algorithm (GA) coupled with an individual learning procedure capable of performing local refinements.

Finally, the representation of a chromosome in the genetic algorithm also plays a significant role. A chromosome can be represented in one dimension as a binary string [7] or floating-point representation [8], or in two dimensions as a matrix [5], or in three dimensions as a cube [9].

IV. PROPOSED SOLUTIONS

A. *Tabu Search*

Tabu search is an enhanced form of local search. In local search, we look into the immediate neighbourhood of the current solution in hope for a better solution. However, because of this, we have a risk of running into a local optimum. To deal with this tabu search was invented. Tabu search avoids visiting a non-optimal solution by maintaining a tabu list that consists of forbidden nodes. This tabu list is used for future reference to verify if a given candidate solution is already visited and found to be non-optimal. Moreover, tabu search uses the concept of intensification and diversification where a non-optimal move is made to get out of local optima if no better move is available.

Tabu search could also be used to create a university timetable. In tabu search, we first create a suboptimal timetable based on some heuristics. Then we create all neighbouring solutions, modifying the current solution with a single mutation. From this solution set, we select the best solution so far S^b. The S^b is verified against the tabu list, and if it is present in the tabu list then we go to the second-best solution, else we compare the S^b with the current solution. If S^b is better, we replace the current with S^b. At last, we check if the desired solution/optimal timetable is achieved, otherwise we repeat the process and keep on exploring the search space.

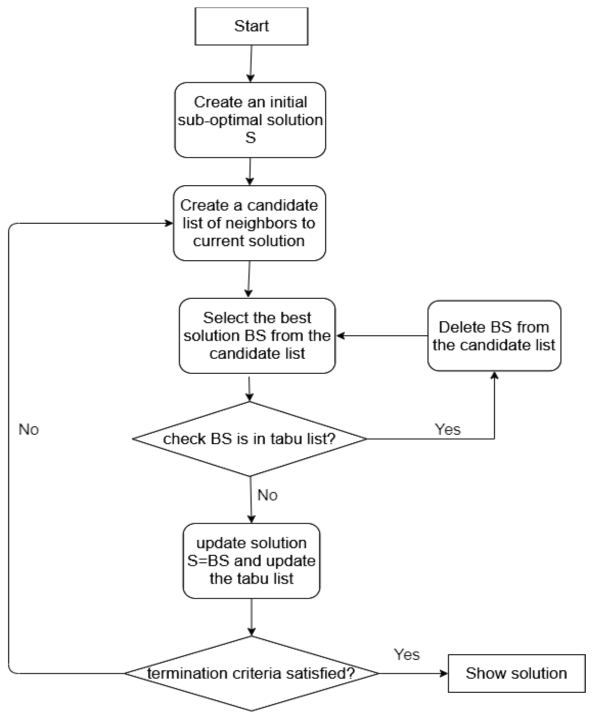
Start
Create an initial sub-optimal solution S
Create a candidate list of neighbors to current solution
Select the best solution BS from the candidate list
Delete BS from the candidate list
check BS is in tabu list?
Yes
No
No
update solution S=BS and update the tabu list
termination criteria satisfied?
Yes
Show solution

Fig. 1 Tabu Search Flow Diagram

B. *Particle Swarm Optimization*

The Particle Swarm Optimization (PSO) algorithm is inspired by the concept of swarm intelligence, often seen in animal groups. The PSO uses a swarm of particles to find a solution in search space, each particle looks into its local neighbourhood for a good solution. While also keeping track of global optima or best solution found so far, by communicating with the other particles. In each generation, a particle updates its position, with respect to its previous position and position of the other particles. The value by which any particle changes the position is defined by three terms. The first term is a product between parameter w (inertia weight) and particle's previous velocity, which is the reason it denotes a particles' previous motion into the current one. The individual cognition term, which is the second term, is calculated by means of the difference between the particle's own best position and its current position. One may notice that the idea behind this term is that as the particle gets more distant from the best position, the difference between best position and current position must increase; therefore, this term increases, attracting the particle to its best own position. Finally, the third term is social learning. Because of it, all particles in the swarm can share the information of the best point achieved regardless of which particle had found it. Its format is just like the second term and is defined by the difference between global best position and particle's current position.

One of the use cases of particle swarm optimization is the university timetable scheduling problem. Here we start by generating an initial population of potential solutions/particles (swarm), where each particle corresponds to a randomly generated timetable. In each generation, two kinds of mutation are performed: in the first mutation for each particle, a class is swapped from the local best solution, meanwhile, in the second mutation, a class is randomly selected from global best and swapped with a class in the current timetable. This process is repeated until further mutations don't result in any improvement of particle fitness or maximum iterations are completed.

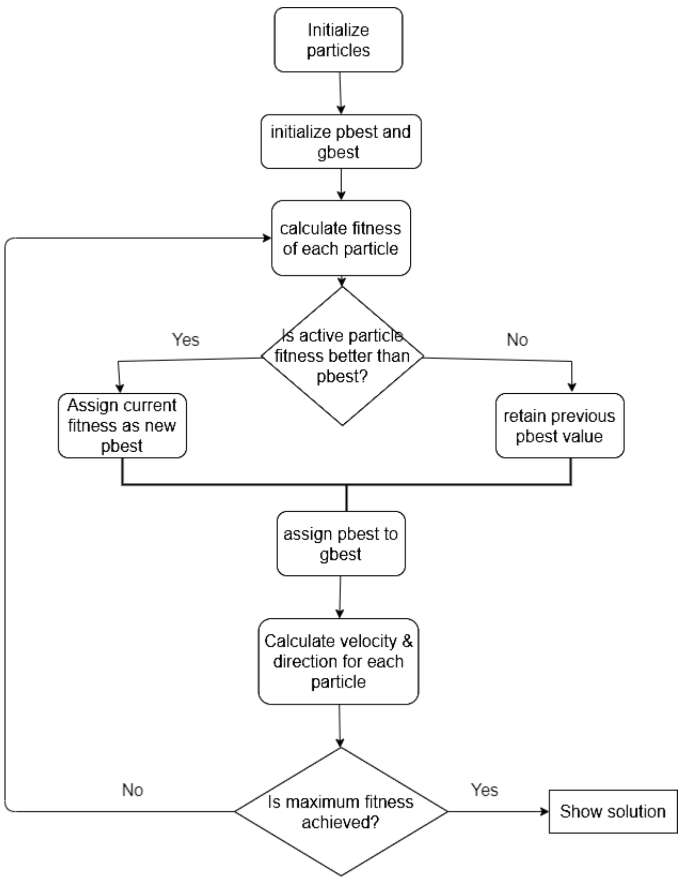
Initialize particles
initialize pbest and gbest
calculate fitness of each particle
Is active particle fitness better than pbest?
Yes
No
Assign current fitness as new pbest
retain previous pbest value
assign pbest to gbest
Calculate velocity & direction for each particle
Is maximum fitness achieved?
No
Yes
Show solution

Fig. 2 Particle Swarm Optimization Flow Diagram

C. *Genetic Algorithm*

Genetic algorithm is derived from the Darwinian theory of evolution. In GA we create an initial population of random individuals and solutions. Then we calculate fitness for each of the individuals in the current population and after that, we select the fittest individual among the population using a selection operator based on the resultant fitness value. Candidates with the highest fitness will be directly moved to the next generation. While candidates with high yet not the best fitness value will be used by crossover operators to generate new individuals. Then a mutation operator is used to induce random changes to the population, to ensure diversity. Finally, candidates obtained after applying all previous operators will become part of the next generation. We will check this new population if desired fitness value is achieved, if not then the process will be repeated until the desired optimal solution is achieved.

As explained, this process can be used to generate an optimal university timetable. Here we first create an initial population consisting of random infeasible timetables. We calculate the fitness value of each candidate using a fitness function, which measures the number of collisions among classes. From this pool, we select timetables with least collision of classes or highest fitness value (that is, no. of constraint violation), chosen candidates are termed as elitist individuals and directly become part of the next generation. Moving on, we use selection schemes like tournament selection to randomly select a few fit individuals that were left out in initial selection. With the help of crossover operators, we swap classes between two times tables to generate a new group of individuals to create a new population. Then we apply the mutation operator with a low probability. Mutation operators randomly swap classes within a timetable to ensure genetic diversity and avoid getting stuck into local optima. Finally, we calculate fitness value or collisions among the new generation and if a feasible timetable is achieved then the process could be stopped, else the cycle is repeated, starting from crossover until a feasible solution is not found. The feasible solution here refers to a timetable that satisfies all kinds of constraints.

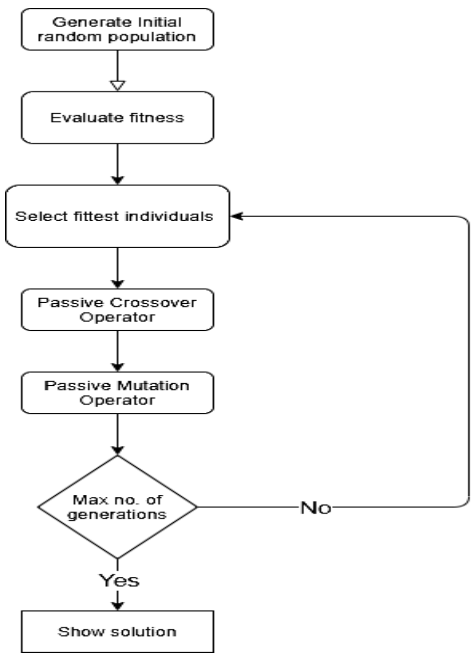
Generate Initial random population
Evaluate fitness
Select fittest individuals
Passive Crossover Operator
Passive Mutation Operator
Max no. of generations
No
Yes
Show solution

Fig. 1 Genetic Algorithm Flow Diagram

D. Memetic Algorithm

Memetic algorithms are extensions of genetic algorithms. It's a hybrid algorithm that combines GA with local search or heuristic search like tabu search, to avoid premature convergence. Implementation of MA is similar to GA, however, unlike GA, MA uses advanced guided operators. Like GA, we first create an initial population, then the fitness of all individuals is calculated, and fittest individuals are separated and moved forward. Then the selection operator is used to select candidates for the crossover, but the rest of the process differs from this point forward. Crossover operator in MA is guided and only swaps best subsections of chromosomes that guarantee the highest fitness after a crossover operation. Similarly, the mutation operator is also guided and before performing a mutation, checks if the resulting mutation leads to greater fitness; if the not different mutation is tried. After that, a local search is used to find a better solution in the neighbourhood of the current solution. If a candidate with higher fitness is identified then the given candidate is replaced with this newly found candidate. In the end, we apply fitness function on the newly generated population. If the population achieves the highest or lowest fitness value based on what kind of fitness function we are trying to achieve, that is, maximising or minimising function, then we stop the execution, else we reiterate until an expected solution is found.

Additionally, the aforementioned processes can be implemented for university timetabling. Just like classical GA, we will create an initial population of sub-optimal timetables from which we will select new individuals according to their fitness for the next generation. However, for the upcoming steps, guided operators will be used. Consequently, the crossover operator will only swap those classes between different candidates/timetables that will result in higher fitness. Similarly, the mutation operator will also use heuristic knowledge to swap classes in a timetable and only perform mutations that lead to fitter individuals. Then we will use a local search algorithm to generate a better individual or meme, that can encourage a faster convergence. At last, we will check the fitness value of the latest spawn population to decide if iteration should be terminated or the solution needs further refinements.

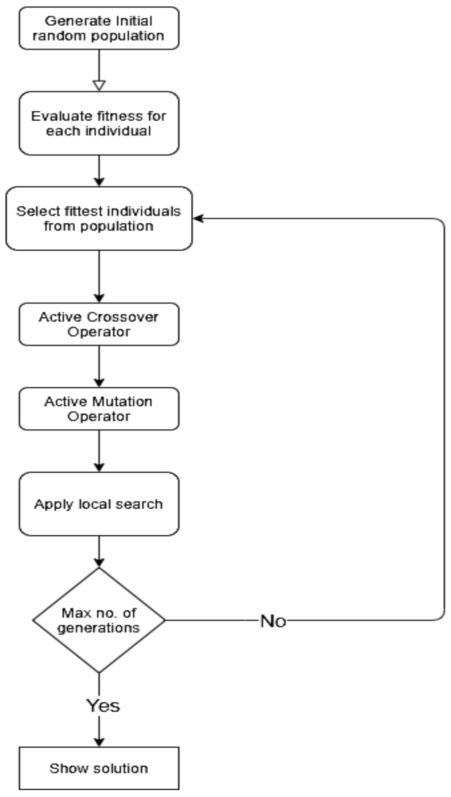
Generate Initial random population
Evaluate fitness for each individual
Select fittest individuals from population
Active Crossover Operator
Active Mutation Operator
Apply local search
Max no. of generations
No
Yes
Show solution

Fig. 4 Memetic Algorithm Flow Diagram

TABLE I
COMPARISON OF ALGORITHMS

Factors	**Tabu Search**	**Particle Swarm Optimization**	**Genetic Algorithm**	**Memetic Algorithm**	**(1+1) Genetic Algorithm**
Average iterations	Large (5000-7000)	Medium (500-1000)	Low (50-100)	Medium (500-1000)	Large (3000-6000)
Initial population size	Large (>500)	Medium (30)	Large (100)	Large (100)	Small (2)
Optimality of final solution	Local optimum	Global optimum	Global optimum	Global optimum	Local optimum
Running time	Lowest	Low	High	Highest	Low
Individual type	Simple	Simple to complex	Simple to Highly complex	Simple to Highly complex	Simple to Highly complex
Biological process Inspired from	Simulated annealing	Swarm intelligence	Natural selection	Natural & cultural evolution	Natural evolution
Factors for next generation	Tabu list or list of infeasible solutions	Previous velocity, particle's current optimum and global optimum	Fitness of new individuals	Fitness of new individuals and tabu list	Fitness of next offspring
Type of problem	Large nonlinear optimization	Nonlinear continuous	Global optimization	Multi-class, multi-objective feature selection	Nonconvex optimization in high dimension and black box scenarios
Constraints optimization level	Hard only	Mostly hard and some soft	Hard and soft	Hard and soft	Mostly hard and some soft

V. RESULTS

Tabu search (TS) algorithm starts with creating an initial feasible timetable using a greedy approach. Thus is best suited for problems where there are a lower number of resources to optimise and fewer number of constraints to satisfy, for example, exam scheduling problems, job shop scheduling, dynamic space allocation, etc. Zhipeng and Jin-Kao in 2010[19], implemented an adaptive tabu search (ATS) algorithm which dynamically optimises various parameters during evaluation. They have tested their algorithm on 4 personal instances and 14 public competition instances of the ITC-2007. They found that for smaller data sets TS performs like a normal local search (LS). However for large data sets, LS outperforms TS and is able to achieve slightly better fitness value. But then they improved TS by creating a hybrid algorithm, where they provided adaptive characteristics to classical TS. They were further able to optimise their algorithm and were able to achieve even better fitness values than LS with ATS.

Particle Swarm Optimization (PSO) algorithm starts with creating an initial infeasible timetable using a random approach. Similar to tabu search, PSO is best suited for low resources and few constraints. As each dimension in which a particle can move corresponds to one of the possible resources, therefore as the number of resources increase, dimension increases and a combinatorial explosion takes place. Nonetheless, because PSO starts with initial random solutions it is better able to optimise soft constraints as compared to tabu search, still it fails to create a feasible solution when a large number of soft and hard constraints need to be satisfied simultaneously. That's why, PSO is best suited for creating timetables for small universities with more hard constraints to work on and less soft constraints to satisfy. Chen, Ruey-Maw & Shih, Hsiao-Fang in 2013[20] created Particle Swarm Optimization with Local Search to address timetabling problems at their university. They have compared conventional PSO with a hybrid solution (that is, combination of PSO and local search). Their findings suggest that though normal PSO is able to find a solution, it often runs into a local optima. Therefore to improve quality of solution and avoid premature convergence local search can be used in conjunction with PSO.

Genetic algorithm (GA) was created to resolve the challenges associated with PSO and TS. Just like PSO and TS, GA starts with a large initial population. However, it applies various operators like mutation, crossover, and selection to find the most feasible solution. With the help of crossover GA is capable of searching large search space as compared to TS. Meanwhile the mutation operator ensures that GA don't get stuck in local optima like PSO. Hence, classical GA is better than standard PSO and TS. On the contrary, GA is slower as compared to previous algorithms and requires more computation power to search a larger search space. That's why GA is better suited for problems that require the most optimal solution and where we have sufficient time to calculate the solution. Generally universities are always aware of their resources like rooms, faculties, courses and usually timetable creation starts months before an actual timetable is needed. Wang Wen-jing in 2018 worked on an improved adaptive genetic algorithm (AGA) for course scheduling. His findings suggest that AGA is able to create better timetables than GA and these timetables along with satisfying all hard constraints also satisfy all soft constraints to a great extent, that is, greater teacher satisfaction, course dispersion, class priority in course scheduling, but AGA is not always superior and takes more time than GA. Finally, Wang was able to identify optimum parameters for GA, that is average generation is around 150 to 250 and average population around 100 to 150.

Memetic algorithms (MA) are a more advanced form of GA. In most cases, MA is the result of combining GA with any other evolutionary or optimisation technique, for instance, MA is a hybrid form of GA and TS. By implying local or tabu search MA converges much faster than GA, meanwhile, providing us with timetables of similar fitness. Moreover, GA as a part of MA ensures that greater width of search space is covered and TS/Hill climbing part ensures that greater depths are searched. Still, MA is computationally more expensive than GA because of the added local search overhead. Burke, Newall and Weare in 2005[22] used the memetic algorithm for exam timetabling. They have used MA on really large datasets obtained from various universities, at the same time, they compared MA results with a Multi start random descent algorithm. According to their observations, MA was able to generate solutions in all cases while random descent was unable to find a solution even when provided with more time and computation power. Also according to Burke, MA uses knowledge from the problem domain for self-improvement of solution. In this way, MAs are more guided as compared to GAs.

VI. CONCLUSION AND FUTURE SCOPE

We have seen implementation of various evolutionary algorithms and how they can be used to solve timetabling problems. Additionally, we have discussed work of other researchers in this domain. From our analysis, we have concluded the following points.

- There is no one fit for all and one has to identify suitable algorithms for their use case by analysing factors like their data size, computation time, number of constraints, and level of accuracy.
- For complex and highly constraint timetabling problems fitness value follows the given order MA > GA > PSO > TS.
- When PSO and TS were mixed with other metaheuristic search methods, they were able to generate results similar to GA and MA.

- The basic difference between all these algorithms is the width and depth of the search space they cover when finding an optimal solution.

All of the comparisons we have made are highly generic and should only be used for basic understanding of algorithms. For deeper understanding and more accurate results one should refer to our referenced research papers in the reference section. Each of the algorithms in study has its own set of pros and cons, however, one can also combine these algorithms to generate new algorithms and further optimise their use case. Likewise, recently many researchers are working with hybrid approaches and have found encouraging results, but still these researches are in their nascent stage and that is why they aren't covered by our study and were left for future study.

REFERENCES

[1] Werra, D.D., 1985. An introduction to timetabling. Eur. J. Operat. Res., 19: 151-162. DOI: 10.1016/0377-2217(85)90167-5

[2] "International series of conferences on the practice and theory of automated timetabling (patat),"https://patatconference.org/#:~:text=PATAT%20Conferences,of%20computer%2Daided%20timetable%20generation.

[3] D. Zhang, Y. Liu, R. MHallah, and S. Leung, "A simulated annealing with a new neighbourhood structure based algorithm for high school timetabling problems," European Journal of Operational Research, vol. 203, no. 3, pp. 550–558, 1999.

[4] G. White, B. Xie, and S. Zonjic, "Using tabu search with longer-term memory and relaxation to create examination timetables," European Journal of Operational Research, vol. 153, no. 1, pp. 80–91, 2004

[5] Abdelhalim, Esraa & El Khayat, Ghada. (2016). A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA). Alexandria Engineering Journal. 55. 10.1016/j.aej.2016.02.017.

[6] V. Sapru, K. Reddy and B. Sivaselvan, "Time table scheduling using Genetic Algorithms employing guided mutation," 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, 2010, pp. 1-4, doi: 10.1109/ICCIC.2010.5705788.

[7] J. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[8] Z. Michalewicz, Genetic Algorithms + Data Structures=Evolution Programs. Springer-Verlag, New York, 1994.

[9] B. Sigl, M. Golub and V. Mornar, "Solving timetable scheduling problem using genetic algorithms," Proceedings of the 25th International Conference on Information Technology Interfaces, 2003. ITI 2003., Cavtat, Croatia, 2003, pp. 519-524, doi: 10.1109/ITI.2003.1225396.

[10] Bhaduri, Antariksha. (2009). University Time Table Scheduling Using Genetic Artificial Immune Network. ARTCom 2009 - International Conference on Advances in Recent Technologies in Communication and Computing. 289-292. 10.1109/ARTCom.2009.117.

[11] Z. Lu¨ , J. Hao, Adaptive tabu search for course timetabling, Eur. J. Oper. Res. 200 (2010) 235–244

[12] Jat, S.N., Yang, S. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. *J Sched* 14, 617–637 (2011). https://doi.org/10.1007/s10951-010-0202-0

[13] Burke, E. & Newall, J. & Weare, R.. (2006). A memetic algorithm for university exam timetabling. 10.1007/3-540-61794-9_63.

[14] Özcan, E. and A. Alkan. "A Memetic Algorithm for Solving a Timetabling Problem: An Incremental Strategy." (2007).

[15] Moscato, P. (1989). "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms". Caltech Concurrent Computation Program (report 826).

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[17] Shi, Yuhui & Obaiahnahatti, B.Gireesha. (1998). A Modified Particle Swarm Optimizer. Proceedings of the IEEE Conference on Evolutionary Computation, ICEC. 6. 69 - 73. 10.1109/ICEC.1998.699146.

[18] S. Droste, Th. Jansen, I. Wegener, A rigorous complexity analysis of the (1+1) Evolutionary Algorithm for linear functions with Boolean inputs, in: Proc. IEEE Internat. Conf. on Evolutionary Computation ICEC '98, IEEE Press, Piscataway, NJ, 1998, pp. 499–504.

[19] Zhipeng Lü, Jin-Kao Hao, Adaptive Tabu Search for course timetabling, European Journal of Operational Research, Volume 200, Issue 1, 2010, Pages 235-244, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2008.12.007. (http://www.sciencedirect.com/science/article/pii/S0377221708010394)

[20] Chen, Ruey-Maw & Shih, Hsiao-Fang. (2013). Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search. Algorithms. 6. 227-244. 10.3390/a6020227.

[21] Wen-jing, Wang. (2018). Improved Adaptive Genetic Algorithm for Course Scheduling in Colleges and Universities. International Journal of Emerging Technologies in Learning (iJET). 13. 29. 10.3991/ijet.v13i06.8442.

[22] Burke E.K., Landa Silva J.D. (2005) The Design of Memetic Algorithms for Scheduling and Timetabling Problems. In: Hart W.E., Smith J.E., Krasnogor N. (eds) Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing, vol 166. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-32363-5_13