Optimization: A Comparative Study of Genetic and Tabu Search Algorithms

Bajeh, A. O.
Dept. of Computer Science
University of Ilorin
Ilorin, Nigeria

Abolarinwa, K. O.
Dept. of Computer Science
University of Ilorin
Ilorin, Nigeria

ABSTRACT

Examination timetabling problem like all scheduling problems are NP-hard problems in which the complexity and time needed to solve the problem increase with the problem size. This paper aims to compare Genetic Algorithm and Tabu Search approaches to solve this kind of problem. Both algorithms were tested with regard to the quality of generated timetables and the speed with which the timetables are generated using collected test data. The test shows that though both algorithms are capable of handling the examination timetabling problem, the Tabu Search approach can produce better timetables than Genetic Algorithm, even at a greater speed.

General Terms

Algorithm, Scheduling, Optimization

Keywords

Chromosome, Examination timetabling, Genetic Algorithm, Tabu Search, Generation

1. INTRODUCTION

Timetabling is a part of the large field of scheduling problems. The scheduling problems are essentially the problems that deal with the effective distribution of resource, because resources are usually limited.

Wren in [1] describes timetabling as:"the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives".

This study focuses on examination timetabling problem which is a kind of timetabling in which the problem is to assign a set of examinations to rooms and timeslots.

Like all scheduling problems, examination timetabling problems are NP-hard, which makes it difficult to find timetables that satisfy all user requirements.

The quality of the time table is measured by how well it satisfies the constraints. The constraints we consider for this problem are classed into two, they are: Hard and Soft constraints.

Hard constraints are those constraints that are important to the problem at hand. For this problem, they define a

Hard constraints are those constraints that are important to the problem at hand. For this problem, they define a feasible timetable. Soft constraints are then used to improve the quality of the timetable to satisfy the users of the timetables.

Different methods have already been proposed for solving examination timetabling problems. These methods come from areas such as: Operation Research, Artificial Intelligence and Computational Intelligence. [10]

This paper attempts to compare the performance of two algorithms: Genetic Algorithm and Tabu Search Algorithm, with respect to their ability to solve the examination timetabling problem. This comparison is to be done with the following criteria:

- The quality of the timetables generated.
- The time used in generating the timetables...

2. RELATED WORK

Several works have tried to compare different algorithms for solving NP-hard problems.

Garg [2] proposed a method based on Memetic Algorithm and Tabu Search for the cryptanalysis of Simplified Data Encryption Standard (SDES).This problem could be formulated as an NP-hard problem. These algorithms were also compared and analyzed with respect to their performance for the cryptanalysis on simplified data encryption standard. The results obtained from the test indicate that Memetic Algorithm is more a powerful technique for the handling of the cryptanalysis of SDES.

Verma et al [3] presented a cryptanalysis method based on Genetic Algorithm and Tabu Search to break a Mono-Alphabetic Substitution Cipher in Adhoc networks. They also compared and analyzed the performance of these algorithms in automated attacks on Mono-alphabetic Substitution Cipher. The work showed that Tabu Search is more powerful than genetic Algorithm for the attack on Mono-alphabetic Substitution Cipher.

Hou et al [4] compared three meta-heuristics: Simulated Annealing, Genetic Algorithm and Tabu Search. This was done by applying each algorithm to the docking procedure between inhibitors and protein which tends to be a sophisticated optimization problem. From the comparison made, it was found that Tabu search outperforms the other two algorithms.

Merz and Freisleben [5] applied Memetic Algorithm, Tabu Search and Ant Colonies to the Quadratic Assignment problem. This work demonstrated that Memetic Algorithm is more suitable for the Quadratic assignment problem.

Wilke and Ostler [6], using real world school time tabling problem, compared the performance of four algorithms, namely Tabu Search, Simulated Annealing, Genetic Algorithm and Branch & Bound. They recommended Simulated Annealing to generate the school timetable as it gives the best trade-off between execution time and quality of results. This was followed by Tabu Search, Genetic Algorithm and Branch & Bound in that order.

Stutzle et al [12] compared the performances of some nature-inspired algorithms on the Travelling Salesman Problem. The algorithms compared include: Genetic Algorithm using DPX-crossover as proposed by Merz and Freisleben, Repair based Genetic Algorithm, Ant Colony Optimization and Iterated Local Search Algorithm. They concluded that the Iterated Local search Algorithm performed better than the other algorithms.

Arostegui Jr. M.A et al [13] also compared the relative performance of Tabu search, Simulated Annealling and Genetic Algorithm on Facility Location Problem (FLP) on various types of FLP under time-limited, solution-limited, and unrestricted conditions. They submitted that the performance of Tabu search was better in all cases while Simulated Annealing and Genetic Algorithm where more partial to the problem type and the criterion used.

Houck et al [14] worked on problem characteristics that make one algorithm more efficient than the other by considering two kinds problems: Location-Allocation and the Quadratic Assignment Problem. The result of tabu search algorithm and genetic algorithm were then compared. They submitted that tabu search performs more efficiently than genetic algorithm in the quadratic assignment problem, while GA is more efficient for the location-allocation problem.

Analyzed, the results obtained by the various works shows that various problems can be solved by the various algorithms, once they can be formulated as NP-hard problems. Also, the computation time and quality of solution differ from one algorithm to another.

3. PROBLEM DESCRIPTION

As stated earlier, examination timetabling is a problem in which the task is to assign a set of examinations to rooms and time slots, while satisfying some constraints.

In this particular implementation, we consider the case of the examination timetable for the Faculty of Information and Communication Sciences in the University of Ilorin. This faculty consists of 5 departments with a total of 74 examinations to be taken in a period of 12 days. Each day consist of 3 slots, namely: morning, afternoon, and evening. These examinations can be scheduled in any of 3 venues, with capacities of: 100, 150, and 250. The table below shows the timeslot allocation for the examination.

Table 1. Timeslot allocation for examination

Period / Day	Morning	Afternoon	Evening
Day1	1	2	3
Day2	4	5	6
Day3	7	8	9
Day4	10	11	12
Day5	13	14	15
Day6	16	17	18
Day N	N*3-2	N*3-1	N*3

The constraints guiding the examination timetabling we are considering are:

Hard constraints:

- No student can sit more than one examination at a time.
- The number of students for a particular exam must not exceed the available space.

Soft constraint:

- No student should have two exams consecutively.

4. THE ALGORITHM

4.1 Genetic Algorithm

The originators of Genetic Algorithm (GA) were John Holland and De Jong. In their respective books titled “Adaptation in natural and artificial systems” and

"Adaptation of the behavior of a class of genetic adaptive systems," both published in 1975. (Davis, 1991)[7].

Genetic Algorithms are metaheuristic methods based on Darwin's theory of evolution that aims to find solutions to NP-hard problems. The basic idea of Genetic Algorithms is to first generate an initial population randomly which consist of individual solution to the problem called *Chromosomes,* and then evolve this population after a number of iterations called *Generations*. During each generation, each chromosome is evaluated, using some measure of fitness. To create the next generation, new chromosomes, called *offspring,* are formed by either merging two chromosomes from current generation using a crossover operator or modifying a chromosome using a mutation operator. A new generation is formed by selection, according to the fitness values, some of the parents and offspring, and rejecting others so as to keep the population size constant. Fitter chromosomes have higher probabilities of being selected. After several generations, the algorithms converge to the best chromosome, which hopefully represents the optimum or suboptimal solution to the problem. [8]

Pseudocode for GA

```
Step 1  Generate initial population.
Step 2  Evaluate population.
Step 3  Apply Crossover to create offspring.
Step 4  Apply Mutation to offspring.
Step 5  Select parents and offspring to form the new population for the next generation.
Step 6  If termination condition is met finish, otherwise go to Step 2.
```

In general, a GA has five basic components:

(i) A genetic representation of potential solutions to the problem.

(ii) A way to create a population (an initial set of potential solutions).

(iii) An evaluation function rating solutions in terms of their fitness.

(iv) Genetic operators that alter the genetic composition of offspring (crossover, mutation, selection, etc.).

(v) Parameter values that genetic algorithm uses (population size, probabilities of applying genetic operators, etc.) [8].

In this study we use direct encoding whereby each chromosome represents a candidate solution. In this representation, the chromosome is a list of numbers whose length is the number of courses to be scheduled (say *e*); each element of the list is the number of timeslots available (say between *1*and *t*). The interpretation of such chromosome is that if *n*th number in the list is t, then exam n is scheduled to occur at time *t*. For example, if we have a chromosome [3, 8, 9, 4, 12, 6, 15], then it means that exam 1 takes place timeslot 3; exam 2 takes place at timeslot 8, etc.

Fitness function measures quality of the chromosomes. As the number of the constraints that a chromosome satisfied increases, so does the chromosome's quality. So the evaluation of the chromosomes and the selection of the constraints are also vital in GAs.
In this study we use the following to calculate the fitness value:

$$f(c)= P(1)\,R(1) + P(2)\,R(2) + P(3)\,R(3)$$

Let P(1), P(2) and P(3) represent the value of the penalty for each constraint. R(1), R(2) and R(3) represent the number of times the candidate timetable violates the restrictions 1, 2 and 3 respectively. In the computer program used, the weights can be determined by the user. In this way, a user can decide how much importance each constraint has.

The initialization procedure is another important issue in all genetic algorithms because it should create a random initial population which spread in the whole search space. Diversity of initial population gives algorithm the opportunity to search the whole space of possible solutions and not to stick with the local optima. [8.] In our study, the chromosomes used for initial population are generated randomly.

4.2 Tabu Search

Fred Glover proposed in 1986 a new approach, which he called Tabu Search, to allow Local Search (LS) methods to overcome local optima. The basic principle of TS is to pursue LS whenever it encounters a local optimum by allowing non-improving moves; moving back to previously visited solutions is prevented by the use of memories, called tabu lists, that record the recent history of the search, a key idea that can be linked to Artificial Intelligence concepts.[11]

It starts from a random initial solution and successively moves to one of the neighbors of the current solution. The difference of tabu search from other metaheuristic approaches is based on the notion of tabu list, which is a special short term memory. That is composed of previously visited solutions that include prohibited moves. In fact, short term memory stores only some of the attributes of solutions instead of whole solution. So it gives no permission to revisited solutions and then avoids cycling and being stuck in local optima. During the local search only those moves that are not tabu will be examined if the tabu move does not satisfy the predefined aspiration criteria. These aspiration criteria are used because the attributes in the tabu list may also be shared by unvisited good quality solutions. A common aspiration criterion is better fitness, i.e. the tabu status of a move in the tabu list is overridden if the move produces a better solution. [9]

Pseudocode for Tabu Search

```
Step 1     Generate initial solution x.
Step 2     Initialize the Tabu List.
Step 3     While set of candidate solutions X' is not complete.
Step 3.1   Generate candidate solution x' from current solution x
Step 3.2   Add x' to X' only if x' is not tabu or if at least one Aspiration Criterion is satisfied.
Step 4     Select the best candidate solution x* in X'.
Step 5     If fitness(x*) > fitness(x) then x = x*.
Step 6     Update Tabu List and Aspiration Criteria
Step 7     If termination condition met finish,
           otherwise go to Step 3.
```

In this implementation, we used the same solution representation as that of the GA. A group of 100 timetables are first generated and sorted according to their objective function, which is also similar to the fitness function in GA. The best timetable is then picked and tested by tabu restriction and aspiration criteria. Then this timetable is randomly mutated a number of times to generate its neighbors, which forms a new set solutions for the next iteration.

5. EXPERIMENTS AND RESULTS

The test on both examination schedulers was run on a Laptop with the following configurations: Pentium(R) Dual-Core 2.0 GHz, 2.0 GB RAM, Windows Vista Home Basic Operating System (SP1)

This test was conducted with the following parameters for the GA and TS examination timetable:

Parameters for GA:

- Number of days for Examination=12
- Maximum evolution=(See table below)
- Population size=100
- Weight of Class size error=10
- Weight of group clash error=10
- Weight of Consecutive exam error=5

Parameters for TS:

- Number of days for Examination=12
- Maximum iteration=(See table below)
- Neighborhood Size=100
- Weight of Class size error=10
- Weight of group clash error=10
- Weight of Consecutive exam error=5

In the first experiment, we aimed to analyze the quality of the timetables generated by both algorithms. This was done by comparing the fitness function of the generated timetable and varying the maximum number of iteration (Maximum evolution for GA). This gives the result in the table below.

Table 2: Fitness of Generated Timetable at different given maximum iteration.

Maximum Iteration	Genetic Algorithm (Fitness)	Tabu Search (Fitness)
100	7260	1035
200	7430	810
300	7295	730
400	7530	635
500	7475	705
600	7105	765
700	7835	650
800	7430	620
900	6185	560
1000	7270	575
Average	5083	708.5

The second experiment aimed to compare the two algorithms based on the speed of execution. This was done by comparing the time taken to generate the timetable with the above maximum iteration and fitness. This gives the table below:

Table 3: Time taken to generate timetable with the above fitness.

Maximum Iteration	Genetic Algorithm (Time)ms	Tabu Search (Time)ms
100	8268	4035
200	16561	7781
300	24847	11745
400	32464	15884
500	39609	19486
600	47752	23721
700	55365	27004
800	64636	33773
900	71947	37399
1000	80188	39974
Average	44163.7	22080.2

6. DISCUSSION OF RESULTS

The overall quality of the timetable is evaluated by the fitness function that adds up violations of all constraints by testing it with all examinations in the faculty. Each constraint has an associated 'weight' or 'penalty' defined during the run time by the user. The results are shown in the tables above. Table 2 compares the quality of the generated timetables by the fitness function of the timetables, it shows that the average fitness of GA and TS is 5083 and 708.5 respectively.

Table 3 shows that the average time of generating the timetables with the above quality is 44163.7Ms and 22080.2ms respectively.

These tests show that TS can produce better solution, with less computing time, than those produced by GA. However, GA can produce several different near optimal solutions at the same time because of its holds the whole the whole generation of chromosomes which may not originate from the same parents.

7. CONCLUSION

It has been seen that both the GA and TS algorithms performed directed evolution on an examination Timetabling problem, and tried to produce timetables void of hard constraint and soft constraints violations. The results obtained from both were very promising.

With the stated objectives, both algorithms have effectively demonstrated the ability to solve complex optimization problems of which examination timetabling is part of. Tabu Search however, produced better results than Genetic Algorithm with respect to quality and speed of generating the timetables.

8. REFERENCE

[1] Wren, A. (1996): Scheduling, Timetabling and Rostering – a special relationship? In Lecture Notes in Computer Science: Practice and Theory of Automated Timetabling, E. Burke and P. Ross, editors. Springer Berlin, Germany, Vol 1153 pp. 46-75.

[2] Garg, P. (2005): A Comparison of Memetic & Tabu Search for the Cryptanalysis of Simplified Data Encryption Standard Algorithm, Journal of Theoretical and Applied Information Technology, Vol IV No. 4, 2005-2008 pp. 360-366.

[3] Verma, A. K., Dave, M. and Joshi, R. C. (2007): Genetic Algorithm and Tabu Search attack on the Mono-Alphabetic Substitution Cipher in Adhoc Networks, Journal of Computer Science(3): pp. 134-137, 2007.

[4] Hou, J. H., Wang, J. M. and Xu, X. J. (1999): A Comparison of Three Heuristic Algorithms for Molecular Docking, Chinese Chemical Letters Vol. 10, No. 7, pp. 615-618, 1999.

[5] Merz, P. and Freisleben, B. (1999): A Comparison of Memetic Algorithms, Tabu Search and Ant Colonies for the Quadratic Assignment Problem, In 1999 Congress on Evolutionary Computation (CEC'99) IEEE Press, Piscataway, NJ, pp. 2063–2070.

[6] Wilke, P. and Ostler, J. (2008): Solving the School Time Tabling Problem using Tabu Search, Simulated Annealing, Genetic and Branch & bound Algorithms. In the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, http://w1.cirrelt.ca/~patat2008/PATAT_7_PROCEEDINGS/ Papers/Wilke-WD2c.pdf, last accessed 19 April, 2011.

[7] Davis, L. (1991): Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold

[8] Gen, M., Cheng, R. and Lin, L. (2008): Network Models and Optimization; Multiobjective Genetic Algorithm Approach. Springer-Verlag London.

[9] Glover, F. (1990): Tabu search, A tutorial Interfaces, 20(4): pp. 74-94, July 1990

[10] Al-Milli N.R (2010): *Hybrid Genetic Algorithms with Great Deluge for Course Timetabling*. International Journal of Computer Science and Network Security, Vol.10 No.4. Page 283-288.

[11]Gendreau M, (2002): *An Introduction to Tabu Search. http://home.ifi.uio.no/infheur/Bakgrunn/Intro_to_TS_Gendreau.htm (Last visited on 18th October, 2011.)*

[12] Stutzle T., grun A., Linke S., Ruttger M.(2000): *A Comparison of Nature Inspired Heuristics on the Travelling Salesman Problem*, Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem

Solving from Nature, volume 1917 of LNCS. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.791 (Last visited 19th of October, 2011).

[13] Arostegui Jr. M.A, Kadipasaoglu S.N, Khumawala B.M (2006): *An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems*. International Journal of Production Economics (2006)
Volume: 103, Issue: 2, Pages: 742-754.

[14]Houck C.R, Joines J.A, Kay M.G (2011): *Characterizing Search Spaces For Tabu Search. Currently under second review in European Journal of Operational Research.*