Trans Genetic Coloring Approach for Timetabling Problem

Mina G. Asham
Computer Engineering Department , Cairo University, Giza

Moataz M. Soliman
Computer Engineering Department , Cairo University, Giza

Rabie A. Ramadan
Computer Engineering Department , Cairo University, Giza

ABSTRACT
Timetable problem is well known problem and is extensively studied in the literature. There are many variations of the problem based on the required hard and soft constraints to be satisfied. One variation of the problem is the exam schedule which is similar to the course schedule with different constraints. In this paper, we propose new solution for course and exam schedule problems base d on University Credit Hour System (CHS) requirements. Our solution utilizes Graph Coloring (GC) and Genetic Algorithms (GA) as a hybrid solution. The test cases used in this paper show the tradeoff between the running time of the proposed algorithm and its fitness performance compared to GA and GC algorithms.

General Terms
Time table schedule, Genetic Algorithms, Graph Coloring.

Keywords
Time table schedule, Genetic Algorithms, Graph Coloring, Genetic Coloring.

1. INTRODUCTION
Timetabling problem [15][11][8][41][48] is one of the well-known NP-Hard problems that attracted many of the re-searchers due to its importance in automating the scheduling process. The problem is a collection of set of courses, available rooms, students, professors, and set of constraints. These constraints are usually classified into two classes which are hard and soft constraints. Hard constraints mean that such constraints must be satisfied while soft constraints are recommended to be satisfied but it is not a must. Based on the problem description, there are many variations of the problem due to the required set of rules. In fact, we can classify the timetabling problems into three main classes which are school, course and examination timetable. Due to the large number of constraints, the course schedule is considered the hardest problem that might generate unfeasible solutions [15].

Due to the importance of the timetabling problem, there are many proposed solutions. For instance, some of the solutions focus on heuristic algorithms which try to find good approximate [43], [31], [18], [24], [12], [10], [13], [9], [49], [34],[33], [36], [35], [38], [39], [37], [27], [5], [2], [42], [46], [20], [29]. Some other solutions involve Genetic Algorithms (GA)[7],[19], [26], [6], [3], [44], [30], Tabu Search [25][16], [21], [17], Simulated Annealing [47], [23], [28], and Scatter Search [40] methods. In addition, graph coloring is utilized as one of the solutions [4]. Moreover, recently, Fuzzy Genetic heuristic is used to solve the timetabling problem[1]. For more details, the readers are referred to good survey papers at [45], [32], [22].

Automatic examination timetables were the subject of much research as well. The problem investigates allocating number of exams to a limited number of time periods such that none of the required constraints are violated. Again, the problem involves hard and soft constraints; however, satisfying the hard constraints leads to feasible solution to the exam timetabling. Such constraints are usually different from university to another. Some examples on the hard constraints are:

- Cannot schedule a student to two different exams at the same time.
- No unscheduled exam(s) exist at the end of the timetabling process;
- Based on room capacities, all students must be scheduled.

Examples on soft constraints are:

- Exam A should be scheduled before exam B.
- Avoid consecutive scheduling of two exams for the same student.

Practically, each university has different hard and soft constraints as well as different evolution methods to the quality of the feasible timetable. However, in most of the cases, the measure of timetable quality is calculated based upon the number of satisfied soft constraints. In this paper, we investigate different solutions to both course and exam timetabling problems. The paper proposes a hybrid solution based on graph coloring and genetic algorithms. The new solution is compared to standalone graph coloring and genetic algorithms solutions.

The paper organization is as follows; the problem and its constraints is described in section 2 ; section 3 discusses graph coloring and genetic algorithms solutions; in section 4, our proposed algorithm is presented; section 5 shows the performance evaluation to the proposed solution; finally, the paper is concluded in section 6.

2. PROBLEM DESCRIPTION
The problem that we discuss here is generated due to a new Credit Hour System (CHS) at the School of Engineering at Cairo University. The system requires automatic timetabling for course and exam schedule. By investigating both problems, we tend to develop one solution that can solve both problems with small tuning. Similar to other timetabling problems, the school allows students to register to any course provided that finishing

the pre-course requirements. Therefore, the problem comes from the limited resources at each department at the school of Engineering. Also, the minimum number of students per class is 15 students. The following are the hard constraints that are considered:

1. A teacher can teach only one lecture at a time.
2. A student can attend only one lecture at a time.
3. A room can host only one lecture at a time.
4. Each teacher must have all his/her courses scheduled.
5. Each student must have all his/her courses scheduled.

On the other hand, Soft constraints are these constraints that may be violated; however, it's not preferable to do so as every soft constraint violated will decrease the efficiency of the produced timetable. Examples for these constraints are:

1. A teacher should not teach more than 6 hours a day.
2. A student should not study for more than 8 hours a day.
3. There shouldn't be gaps in the activity of the teachers.
4. There shouldn't be gaps in the activity of the students.

Classes timetable consist of six days; Saturday, Sunday, Monday, Tuesday, Wednesday, and Thursday. Each day consist of five slots; Slot 1, Slot 2, Slot 3, Slot 4, and Slot 5.

For the exam timetable, some of the previous constraints are modified such as:

1. A student can't have more than one exam per day.
2. A student can't have more than one exam at the same time.
3. There might be more than exam in the same room based on the size of the room.
4. A teacher cannot monitor more than one exam unless they are in the same room.

Based on these requirements, the solution has to fulfill the hard and soft constraints in a reasonable running time.

3. GRAPH COLORING AND GENETIC ALGORITHMS

Graph Coloring (GC) and Genetic Algorithms (GA) are two of the solutions that proved to be efficient in producing course timetable. In this section, we explore the modified version of the graph coloring and genetic algorithms approaches used for course timetable given in [4], [14] to fit our timetabling requirements.

3.1 Graph Coloring Approach

The graph coloring problem is a classical NP-complete problem in which you are given a graph G= (V, E), where V is the set of vertices of the graph, and E is the set of edges connecting vertices of the graph. The degree of a vertex is a value assigned to each vertex which represents the number of edges connected to it. While two vertices X and Y are said to be triples if and only if there exist a vertex Z such that;

- X is connected to Z.
- Y is connected to Z.
- X is not connected to Y.

The approach tends to divide V into the minimum number of groups, such that:

- Each group is colored with a distinct color.
- All the elements of the same group are colored with the same color.
- Any two vertices connected with an edge, should have different colors.

Course/Class timetable automation problem is reduced to a graph coloring problem. This reduction is done by representing each class with a vertex in the graph, and there's an edge between any two classes if and only if these two classes share a common student or teacher. As shown in Figure 1, the approach consists of three phases which are coloring vertices, assigning rooms and scheduling classes.

Figure 1: Graph coloring phases

In phase 1, as shown in Figure 2, best coloring to the graph vertices is attempt through 8 steps. In this phase we color the graph vertices so that no two adjacent vertices have the same color. In other words, the algorithm tries to figure out all courses that could be scheduled simultaneously.

The algorithm in Figure 2 consists of one main loop that continues to execute until no more triples exist in the graph G. This main loop is the set of instructions from 2 to 7. At step 2, the vertex with the maximum number of edges connected to it is selected and named as max, and then the set of triples of that vertex is fetched to get the maximum degree triple of max and name it maxTriple. These two selected vertices are merged by removing maxTriple and connecting max to all vertices connected to maxTriple. Finally we need to update the degree of max and check the main loop's ending condition

1. Let ***G*** = the set of all vertices in the graph.
2. Set ***max*** = the vertex with the maximum degree.
3. Set ***maxTriples*** = the set of all triples of max.
4. Set ***maxTriple*** = the vertex with the maximum degree in ***maxTriples***.
5. Merge ***max*** and ***maxTriple*** by deleting ***maxTriple*** from ***G*** and connecting ***max*** with all vertices connected to ***maxTriple***.
6. Update the degree of ***max***.
7. Go to step 2 until no more triples exist in ***G***.
8. Color each group of merged vertices with the same color.

Figure 2: Graph coloring phase 1

1. Let ***Rooms*** = the set of all rooms available.
2. Let ***Groups*** = the set of all colored groups.
3. For each group ***grp*** in ***Groups***:
    a. Copy ***Rooms*** to ***temp***.
    b. Sort ***grp*** in ascending order of capacity.
    c. For each class ***cls*** in ***grp***:
        i. Set the room of ***cls*** to the smallest room ***r*** in which ***cls*** fits.
        ii. Remove ***cls*** from ***grp***.
        iii. Remove ***r*** from ***temp***.
    d. Remove ***grp*** from ***Groups***.

Figure 3: Graph coloring phase 2

1. Let ***Groups*** = the set of colored groups.
2. For each group ***grp*** in ***Groups***:
    a. For each day ***d*** in the week:
        i. For each slot ***s*** in the day:
            1. Loop through all students and teachers to check all soft constraints if ***grp*** is assigned to slot ***s*** in day ***d***.
            2. If no constraints violated, assign ***grp*** to slot ***s*** in day ***d***.
    b. If all slots in all days violate constraints, assign ***grp*** to the first empty slot ***s*** in day ***d***.

Figure 4: Graph coloring phase 3

In phase 2, room assignment given in Figure 3, apply the room assignment algorithm to assign rooms to all courses, by assigning all available rooms to each group of courses colored with the same color once at a time, till all courses have their rooms assigned.

At this point, we have a set of groups, each group consists of a set of non-conflicting classes, and each class is as- signed to a room in which it best fits. In other words, we have satisfied all the hard constraints of the problem. Since the graph coloring technique is only useful in the satisfaction of the hard constraints, we have no other choice than the brute force algorithm to satisfy the soft constraints. This is done by a simple algorithm shown in Figure 4.

3.2 Genetic Algorithms Approach

Genetic Algorithm is a method for solving optimization problems based on local search. In GA, the solution is generated by combining parent solutions rather than by modifying a single solution. The GA is given a set of random solutions to be used for generating new solutions. Also, it produces the best solution available when the ending condition is met. Figure 5 shows the genetic algorithms flowchart.

The process starts with the generation of the initial population; this is done by generating random chromosomes to be used in later generations. The algorithm keeps track of some of the best chromosomes having the best fitness function in order not to be deleted. If the best fitness meets the GA stopping criteria, it terminates; otherwise, it goes through the crossover and mutation processes. Due to the randomness in the crossover and mutation , some of the chromosomes might violate the course rules; for instance, a course might exist more than one time or even doesn't exist at all in the chromosome (the number of times in which the class is scheduled is not equal to 1). Therefore, chromosome repair process is added to the GA to fix such problems. The pseudo code for the genetic algorithm is shown in Figure 6.

For the GA to work properly, there are three main important decisions have to be carefully taken. The first decision is the chromosome structure and the second is fitness/evaluation function while the third is the stopping criterion. In this paper, we selected the chromosome structure as the class timetable as depicted in Figure 7. In other word, a chromosome represents a valid and efficient class/exam schedule. At the same time, since the GA is used to solve the whole problem, therefore it should satisfy all of the hard constraints, and as much as possible of the soft constraints. Therefore, the fitness function, or the cost function, is selected to be a compound function that consists of two values, a hardFitness for the number of hard constraints violated, and a softFitness for the number of soft constraints violated. These two values are added together to give the fitness function. The third important factor i s the stopping condition, which is (hardFitness == 0 && softFitness < max) where max is the maximum allowable value for the number of soft constraints to be violated. This value is chosen experimentally. Other important parameters should be chosen such as; the number of iterations, type of crossover and probability, mutation percentage, population size, number of best chromosomes per iteration to keep, and number of new off-springs generated per iteration.

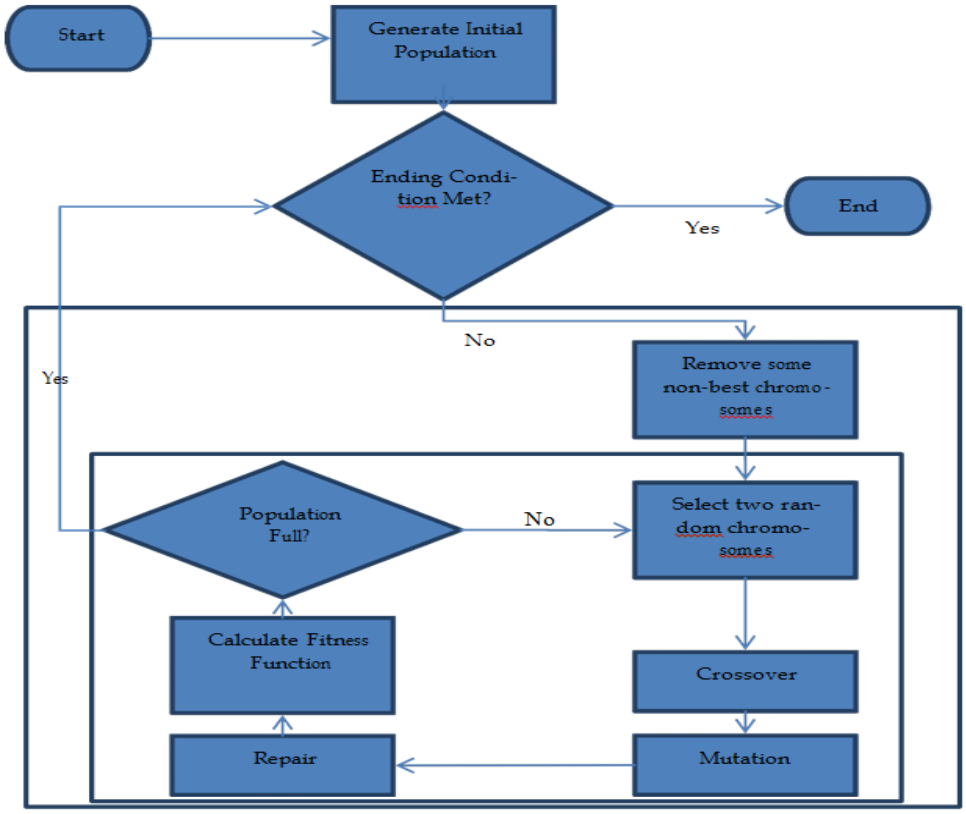

Figure 5: Genetic algorithm concept

```
1. Insert random chromosomes into the population until
   it's full.
2. While the ending condition is not met, do
      a. Remove some of the chromosomes available.
      b. Select two chromosomes by random.
      c. Cross them over to get a new offspring.
      d. Mutate the new offspring.
      e. Repair the new offspring.
      f. Evaluate the fitness function of the new
         offspring and add it to the population.
      g. Go to "b" until the population is full.
      h. Check the resetting condition, if met; replace
         all non-best chromosomes in the population with
         new randomly generated ones.
```

Figure 6: Genetic algorithm pseudo code

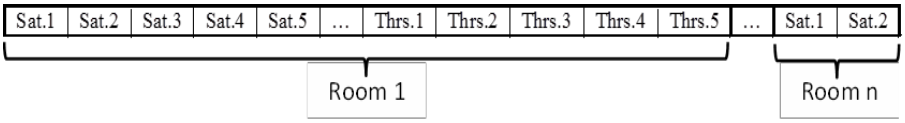

Figure 7: Genetic algorithms chromosome structure

3.3 Genetic Coloring

In this section, we propose to utilize both GC and GA for solving both class and exam timetabling problems. Based on our experience in using GA and GC in timetabling, we could summarize some of the drawbacks in both algorithms as follow:

1. GC uses a brute force algorithm to satisfy the soft constraints which is not good idea especially with large problem sizes.
2. GC assigns the class/course to the first empty slot if all slots violate one or more of the soft constraints.
3. In GA, the chromosome length increases with the increase of the problem size. For instance, in our case, the chromosome length could be 30*n (6 days * 5 slots), where n is the number of rooms. Assume n=100, therefore, the chromosome length will be 3000 genes which is very large. Certainly, this affects the chromosome processing and the overall GA running time.
4. GA should wait till the hardFitness value converges to zero which might take long time.

To solve these problems, we propose to use GC to satisfy the hard constraints and GA to satisfy the soft constraints. We believe that such combination will enhance the efficiency of the timetabling solution(s) as well as to reduce the running time in most of the cases. Therefore, no brute force is required for the soft constraints satisfaction. In addition genetic coloring assigns the class to the slot that minimizes the number of violated soft constraints. Moreover, the chromo- some length will be reduced to a constant value, (30 *6 days * 5 slots). Nevertheless, genetic coloring doesn't even check the hardFitness value as it is surely ZERO.

Figure 8: Genetic coloring phases

Similarly, genetic coloring algorithm consists of three phases as shown in figure 8. The problem is first reduced into a graph coloring problem and solve accordingly. This is similar to the first phase in graph coloring solution explained in section 3.1. The second phase is the room assignment which also similar to the second phase in graph coloring approach explained in section 3.1. The pseudo code for both phases is depicted in Figures 2 and 3 respectively. Here, it comes the role of the GA to work on the soft constraints. The output of phase 2 is a set of colored groups. The target of the genetic algorithm is to find the best way to assign these colored groups to the timetable's slots minimizing the number of soft constraints violated. The pseudo code for the phase 3, GA, is given in Figure 9. In this phase, the same GA process, explained before, takes place. The process includes the crossover, mutation, the repair procedure, and the fitness function evaluation. However, the GA process this time differs from the one stated before because of the change in some settings.

1. Insert random chromosomes into the population until it's full.
2. While the ending condition is not met, do
    a. Remove some of the chromosomes available.
    b. Select two chromosomes by random.
    c. Cross them over to get a new offspring.
    d. Mutate the new offspring.
    e. Repair the new offspring.
    f. Evaluate the fitness function of the new offspring and add it to the population.
    g. Go to "b" until the population is full.
    h. Check the resetting condition, if met; replace all non-best chromosomes in the population with new randomly generated ones.

Figure 9: Genetic coloring phase 3

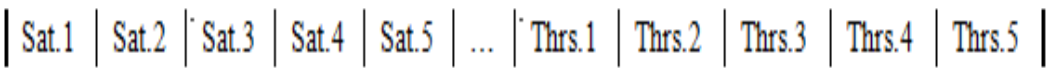

Figure 10: Genetic Coloring chromosome structure

The new chromosome structure is reduced to the number of days multiplied by the number of slots per day as shown in Figure 10. The fitness function, since the genetic algorithm will be used to satisfy the soft constraints only, there's no need to have a compound functions. Therefore the fitness function will be only the number of violated soft constraints. In addition, the stopping criterion is changed to be based only on the condition of softFitness < max, where max is the maximum allowable value for the number of soft constraints to be violated. This number we choose based on our simulation as we will show later in the simulation results. Other parameters stated previously in the GA section may have different value, depending on the used test cases.

4. EXPERIMENTS

In this section, we study the performance of hybrid of genetic and graph coloring algorithms compared to the GA and GC as standalone solutions to the timetabling problems. We implemented the three algorithms for this purpose and designed some of the test cases to measure their performances. All of the experiments done in this paper were implemented on Intel Quad Core 2.83GHz processor, with 12 MB cache, and Windows 7 32-bit having 4 GB of RAM. Our implementation utilized C# dot Net on dot Net framework 4 under Visual Studio 2010. In addition, to have fair comparisons among the proposed algorithms, we tested the Genetic algorithm in many test cases and found , on average, the best results produced when 1) mutation percentage equal to 1% , 2) maximum number of iterations equal to 1000, and 3) number of chromosomes in the population is equal to 30.

4.1 Test Case 1: Department Scope

This test case considers small size problem which we call it "department scope". This scope considers only 250 students, 20 professors, 70 courses, and 20 rooms. In fact, this is the actual size of our computer department. In terms of average running time, Figure 11 shows that graph coloring algorithm is the least running time while genetic algorithm takes almost 46 seconds to reach a suitable solution which is the worst among the three algorithms. At the same time, the genetic coloring time is not considered bad compared to the genetic algorithm but still worth than the graph coloring algorithm. However, Figure 12 shows another point of view in terms of the algorithms performance. For instance, as can be seen, it does not mean that because of the graph coloring algorithm is taking the least time, it produces the best timetable in terms of performance. In fact, Genetic coloring approach gives the best performance while the graph coloring algorithm violates most of the soft constraints. In addition, Genetic Coloring is something in between in terms of performance of other algorithms while its running time is very small compared to the Genetic Algorithm running time.

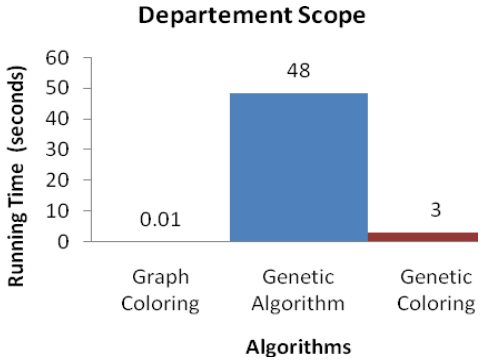

Figure 11: Running time of the three algorithms in terms of department scope

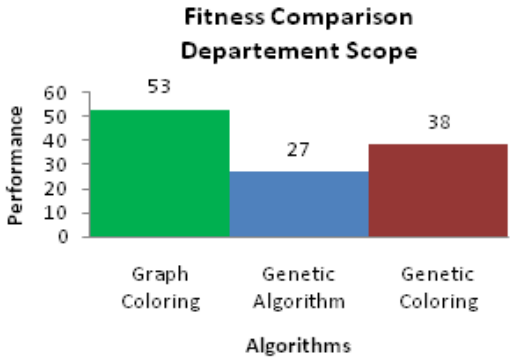

Figure 12: Fitness comparison of the three algorithms in terms of department scope

4.2 Test Case 2: Faculty Scope

Here, again, we test the performance of the three algorithms with medium size problems in which we call it "faculty scope". The faculty scope used in this section are about 500 students, 40 professors, 40 teaching assistance, 140 courses, and 40 rooms. In terms of running time, the results as shown in Figure 13 is similar to the department results where genetic algorithm is the most time consuming algorithm. Graph coloring still better than the Genetic coloring by almost half of the time. However, both graph coloring and genetic coloring are much better than the genetic algorithm by almost 14 times. On the other hand, in terms of the algorithms fitness, genetic coloring, as shown in Figure 14 is somehow between the genetic and graph coloring. Therefore, it is a tradeoff between the time and fitness.

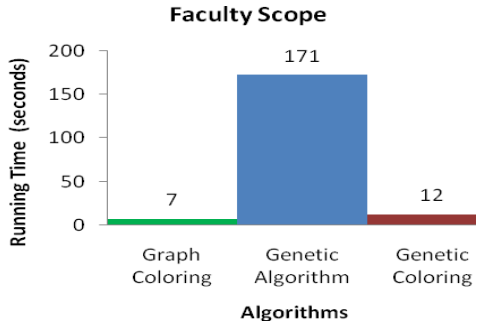

Figure 13: Running time of the three algorithms in terms of Faculty scope

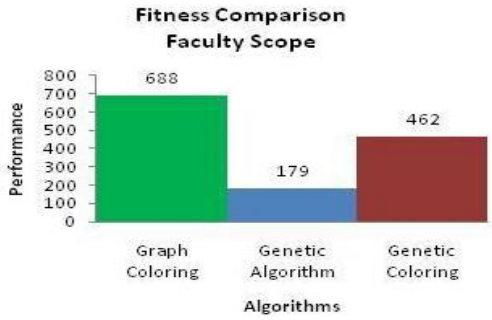

Figure 14: Fitness comparison of the three algorithms in terms of Faculty scope

4.3 Test Case 3: University Scope

Another test case is examined in this section for a university scope in which problems with 750 students, 60 professors, 60 teaching assistance, 210 courses, and 60 rooms are inspected. In terms of running time, Figure 15, it seems that the running time is doubled compared to the faculty scope problems. For instance, the genetic algorithm takes almost 7 hours to produce a solution. However, genetic algorithm still over fits both graph coloring and genetic coloring algorithms in terms of performance as shown in Figure 16. At the same time, genetic coloring is much better than graph coloring algorithm.

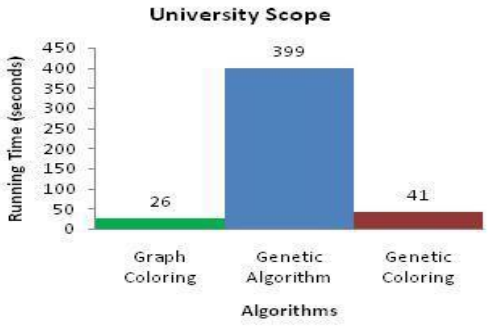

Figure 15: Running time of the three algorithms in terms of University scope

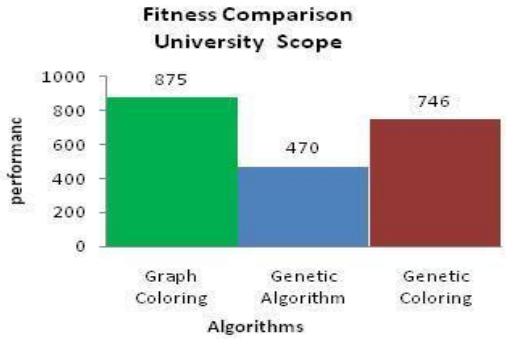

Figure 16: Fitness comparison of the three algorithms in terms of University scope

5. CONCLUSION

In this paper, we compared between different techniques for timetabling problem including the exam timetabling. We introduced three algorithms which are graph coloring, genetic algorithms, and genetic coloring as a hybrid algorithm. Graph Coloring and Genetic Algorithms are modified from previous algorithms while the Genetic Coloring algorithm is the one we proposed in this paper. After the implementation to the algorithms, we experimented with different test cases through different problem scopes, department, faculty, and university scopes. Our results show that although our Genetic Coloring was not the best algorithm in terms in its fitness in most of the cases, however, it is better than the GA in terms of running time and better than Graph coloring in terms of fitness performance.

6. REFERENCES

[1] A. Chaudhuri and D. Kajal, "Fuzzy Genetic Heuristic for University Course Timetable Problem," Int.J.Advance. Soft Computing. Applications, Vol. 2, No. 1, March 2010 ISSN 2074-8523; 2010.

[2] B. Paechter, A. Cumming,M. G. Norman and H. Luchian, "Extensionsto a Memetic Timetabling System", Proceedings of the 1st In ternational Confe- rence on the Practice and Theory of Automated Timetabling, (1995)

[3] B.Sigl,M.Golub,andV.Mornar,"SolvingTimetableSchedulingProblemUsingGeneticAlgorithms",25thInternationalConference Information Tech- nology Interfaces, Cavtat, Croatia, (2003).

[4] Burke E.K., Elliman D.G. and Weare R.F. (1993) "A University Timetabling System Based on Graph Colouring and Constraint Manip ulation", in the Journal of Research on Computing in Education. Vol. 26.issue 4.

[5] C.Marco,B.Mauro and S. Krzysztof, "An Effective Hybrid Algorithm forUniversity Course Timetabling", Journal of Scheduling, Vol.9, No.5, (2006), pp.403-432.

[6] C.Reeves, "Genetic Algorithms", Modern Heuristic Techniques for Combinatorial Problems, V. J. Rayward-Smith (Editors), McGraw-Hill International, UK, (1995), pp.151-196.

[7] D. Corne, H. S. Fang and C. Mellish, "Solving the Modular Exam Scheduling Problem with Genetic Algorithms", Proceedings of the 6th International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, (1993).

[8] D. De Werra, "An introduction to Time tabling", European Journal of Operations Research, Vol.19, (1985),pp.151-162.

[9] D. Dubois and H. Prade, Possibility Theory: an Approach to Computerized Processing of Uncertainty, New York, (1988).

[10] E.K.BrukeandJ. P.Newall, "Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings", Annals of Operations Research,Vol.129, (2004), pp.107-134.

[11] E.K.BrukeandS.Petrovic,"RecentResearchdirectionsinAutomatedTimetabling",EuropeanJournalofOperationalResearch, Vol.140, No.2, (2002), pp.266-280.

[12] E.K.BurkeandJ.P.Newall,"A New Adaptive Heuristic Framework for Examination Timetabling Problems", University of N ottingham, Working Group on Automated Timetabling, TR–2002-1, (2002).

[13] E. K. Burke, B. McCollum and A. Meisels, "A Graph based Hyper Heuristic for Educational Timetabling Problems", European Journal of O perational Research, Vol.176, No.1, (2007), pp.177-192.

[14] E.K. Burke, D. Elliman, R. Weare, A genetic algorithm based university timetabling system, in: Proceedings of the 2nd East-West International Confe- rence on Computers in Education, Crimea, Ukraine, 19th–23rd September 1994, vol. 1, (1994), pp. 35–40.

[15] Even, S. Itai, A., & Shamir, A., On the Complexity of timetabling and multicommodity flow problems, SIAM Journal of Computation, Vol.5, No.4, 1976, pp.691-703.

[16] G.Kendall and N.M.Hussain, "A Tabu Search Hyper Heuristic Approach to the Examination Time tabling Problem at the MARA University of Technology", Lecture Notes in Computer Science, Springer Verlag, Vol.3616, (2005), pp.270- 293.

[17] G.White,B.XieandS.Zonjic,"UsingTabuSearchwithLongerTermMemoryandRelaxationtocreateExaminationTimetables",EuropeanJournalofOperational Research, Vol.153, No.16, (2004), pp.80-91.

[18] H. Asmuni, E. K. Burke and J.Garibaldi, "Fuzzy Multiple Heuristic Ordering for Course Timetabling", Proceedings of the 5th United Kingdom Workshop on Computational Intelligence, London, (2005).

[19] J.F.GonçalvesandJ.R.DeAlmeida,"A Hybrid Genetic Algorithm for Assembly Line Balancing", Journal of Heuristics,Vol.8,(2002),pp.629-642.

[20] K. Socha, J. Knowles and M. Samples, "A Max-Min Ant System for the University Course Timetabling Problem", Proceedings of the 3rd International Workshop on Ant Algorithms, Lecture Notes in Computer Science, Springer Verlag, Vol.2463, (2002), pp.1-13.

[21] L. M. Mooney, Tabu Search Heuristics for Resource Scheduling with Course Scheduling Applications, PhD Dissertation, Purdue Un iversity, (1991).

[22] Lewis, R. (2008) 'A Survey of Metaheuristic-based Techniques for University Timetabling Problems'. OR Spectrum, vol. 30(1), pp 167-190.

[23] M.A.SalehandP.Coddington,"AComparisonofAnnealingtechniquesforAcademicCourseScheduling",LectureNotesinComputerScience,SpringerVerlag, Vol. 1408, (1998), pp.92-114.

[24] M. Battarra, B. Golden and D. Vigo, "Tuning a Parametric Clarke-Wright Heuristic via a Genetic Algorithm", Università di Bologna, Dipartimento di

ElettronicaInformatica e Sistemistica, TR-DEIS OR.INGCE 2006/1R, (2006).

[25] M.GendreauandJ.Potvin,"TabuSearch",IntroductoryTutorialsinOptimization,DecisionSupportandSearchMethodology, E. K. Burke and G. Ken- dall (Editors), Springer Verlag, Chapter 6, (2005), pp.165-186.

[26] M.Omar,R.N.Ainon,andR.Zainuddin,"UsingaGeneticAlgorithmOptimizerTooltogenerategoodqualityTimetables",Proceedingsofthe10thIEEE International Conference, Electronics, Circuits and Systems, Vol.3, (2003), pp.1300-1303.

[27] M.R.Malim,A.T.KhaderandAMustafa,"ArtificialImmuneAlgorithmsforUniversityTimetabling",Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Czech Republic, (2006).

[28] M.Tuga,R.BerrettaandA.Mendes,"AHybridSimulatedAnnealingwithKempeChainNeighborhoodfortheUniversityTimetablingProblem",Com- puter and Information Science, (2007).

[29] N.D.Thanh,"SolvingTimetablingProblemusingGeneticand HeuristicAlgorithms",Proceedings of 8th ACIS International Conference, (2007).

[30] N.D.Thanh,"SolvingTimetablingProblemusingGeneticand HeuristicAlgorithms",Proceedingsof8thACISInternationalConference,(2007).

[31] P.AdamidisandP.Arapakis,"EvolutionaryAlgorithmsinLectureTimetabling",Proceedingsofthe1999IEEECongressonEvolutionary Computation, (1999), pp.1145-1151.

[32] P. De Causmaecker, P. Demeester, G. VandenBerghe: Evaluation of the University Course Timetabling Problem with the Linear Numberings Method, UK Plan SIG 2006, Nottingham, 14-15 December 2006, pp. 154-155.

[33] P.Kostuch,"TheUniversityCourseTimetablingProblemwitha 3-phaseApproach",Proceedingsofthe5thInternationalConference on the Practice and Theory of Automated Timetabling, E. K. Burke, M. Trick, (Editors), Lecture Notes in Computer Science, Springer Verlag, Vol.3616, (2005), pp.109–125.

[34] P.Kostuch,"TheUniversityCourseTimetablingProblemwitha 3-stage Approach",Proceedingsofthe5thInternationalConferenceonthePractice and Theory of Automated Timetabling, (2004), pp.251-266.

[35] R. Lewis andB. Paechter, "Application of the Grouping Genetic Algorithm to University Course Timetabling", Evolutionary Comp utation in Combina- torial Optimization, V. G. Raidl and J. Gottlieb, (Editors), Lecture Notes in Computer Science, Springer Verlag, Vol.3448, (2005), pp.144-153.

[36] R.LewisandB.Paechter,"NewCrossoverOperatorsforTimetablingwithEvolutionary Algorithms",5thInternationalConferenceonRecentAdvances in Soft Computing, Nottingham, England, (2004).

[37] R.Lewis,"ASurveyofMetaheuristicbasedtechniquesforUniversityTimetablingproblems",ORSpectrum,Vol.30,(2008),pp .1 67-190.

[38] R.Lewis,"MetaheuristicsforUniversityCourseTimetabling" PhDThesis,SchoolofComputing,NapierUniversity,Edinburgh, (2006).

[39] R. Lewis, B. Paechter and B. McCollum, "Post Enrolment based Course Timetabling: A description of the Problem Model used for Track Two of the SecondInternationalTimetabling",CardiffUniversity,Cardiff Business School, Accounting and Finance Section, (2007).

[40] R. Marti, H. Lourenco and M. Laguna, "Assigning Proctors to Exams with Scatter Search", Economics Working Paper Seri es No.534, Department of Economics and Business, Universitat Pompeu Fabr, (2001).

[41] R.QuandE.K.Burke,"AdaptiveDecompositionandConstructionforExaminationTimetablingProblems",MultidisciplinaryInternational Scheduling: Theory and Applications, P. Baptiste, G. Kendall, A. Munier-Kordon, F. Sourd, (Editors.), (2007), pp.418-425.

[42] R. Qu and E. K. Burke, "Adaptive Decomposition and Construction for Examination Timetabling Problems", Multidisciplinary International Scheduling: Theory and Applications, P. Baptiste, G. Kendall, A. Munier-Kordon, F. Sourd, (Editors.), (2007), pp.418-425.

[43] S.Abdullah,E.K.BurkeandB.McCollum,"A Hybrid Evolutionary Approach to the University Course Timetabling Problem", Proc eedings of the IEEE Congress on Evolutionary Computation, Singapore, (2007).

[44] S.O.TasanandS.Tunali,"A Review of the Current Applications of Genetic Algorithms in Assembly Line Balancing", Journalof Intelligent Manufactur- ing, Vol.19, (2008), pp.49-69.

[45] Schaerf A., A Survey of Automated Timetabling, Tech. rep. CS-R9567, CWI, Amsterdam,(1995).

[46] Scholl and C. Becker, "State-of-the-art Exact and Heuristic solution procedures for Simple Assembly Line Balancing", European Journal of Operation Research, Vol.168, (2006), pp.666-693.

[47] T. Duong and K. Lam, "CombiningConstraint Programming and Simulated Annealing on University Exam Timetabling", Proceedings o f RIVF Confe- rence, Hanoi, Vietnam, (2004).

[48] Wren, Scheduling, "Timetabling and Rostering– A Special Relationship!", The Practiceand Theory of Automated Timetabling:Selected Papers from the 1st Int'l Conf. on the Practice and Theory of Automated Timetabling, E. K. Burke, P. Ross (Editors), Lecture Notes in Computer Science, Springer Verlag, Vol.1153, (1996), pp.46-75.

[49] Z. W. Geem, "School Bus Routing using Harmony Search", Proceedings of Genetic and Evolutionary Computation Conference, Washington, D.C., (2005).