Constructive Initialization of a Genetic Algorithm for the Solution of a Highly Constrained Departmental Timetabling Problem

Peter. U. Eze, Dawn. C. Walker, Ifeyinwa E. Achumba

Abstract: ***The University or Departmental Timetabling Problem (UTP or DTP) is a scheduling problem ridden with numerous constraints. Each of the constraints has a complex effect on the ideal solution and their combined effect makes the problem harder to solve. As a solution to this problem, a genetic algorithm (GA) approach was augmented by a process of constructive initialisation and applied to an exemplar scheduling problem in the Department of Computer Science at the University of Sheffield. The problem entailed scheduling of timetabled slots for 33 modules across a range of taught programmes at various levels, delivered by 29 lecturers in 10 lecture theatres and 6 laboratories. A total of eight hard constraints and four soft constraints were considered, for problems of five levels of increasing complexity. It was found that the synergistic solution satisfied all the hard constraints, achieved up to 75% optimisation of the soft constraints, and converged within 500 iterations or an average of 2.74 minutes. These results indicate that the GA, when combined with constructive initialization, will give efficient solution to the DTP problem with constrained variables.***

Keywords: Departmental Timetabling Problem, Constructive Initialization, Genetic Algorithm, Scheduling, Constraints

I. INTRODUCTION

The University or Departmental Timetabling Problem (DTP) problem is a scheduling problem, which involves combinatorial allocation of resources under some predefined constraints [1]. The allocation is done within time slots (usually working hours) with the aim of optimizing the use of available space and time as well as avoiding the violation of the constraints.

The constraints may include the number of available lecture rooms, types of lecture rooms, sizes of lecture rooms, the number of lecturers and their availability, lecturer specialty, number of courses to be delivered, the number of students in a course, and distance between lecture rooms, among others. Some of the constraints are *hard* constraints that must be satisfied to make the solution valid, while others are *soft* constraints that are only desirable to be satisfied. Optimization involves satisfying the hard constraints and trying to satisfy as many of the soft constraints as possible.

The variations in constraints among institutions and departments and the complex interdependence of variables and constraints have increasingly made the problem hard to solve. Hence, it is hard to provide a generic model for all situations and for different universities.

Thus, the DTP scheduling problem is considered an NP-hard problem [1,2] or NP-Complete Problem [3], because there is no known deterministic polynomial time algorithm for its solution. Different researchers have used various methods in attempts to find a solution.

The work done by [4] suggested that the methods of neural networks, simulated annealing, Tabu Search constraint programming and greedy search could be used. They compared these methods to a Genetic Algorithms (GA) approach to solve the same problem and concluded that even though comparable results can be obtained from any of the methods, the GA approach achieves better performance in terms of efficiency and solution optimization.

A Genetic Algorithm (GA) is a particular class of evolutionary algorithm. Like many other artificial evolution-based algorithms, the GA takes its inspiration from the concepts of natural evolution.

These concepts include maintenance of a population of individuals, creation of diversity through random mutation or crossover, a selection mechanism and a process for transfer of genetic characteristics [5].

A general form of a GA is shown in figure 1. For highly constrained problems such as those used in this research, it is difficult to create a GA solution that can solve all scheduling problems of this nature. A GA to solve a specific problem must be manipulated in some way to direct it towards the desired solution. The level of this manipulation will determine if one is actually solving a problem or evolving a trivial solution without a target in mind.

Revised Version Manuscript Received on July 25, 2016.

Peter. U. Eze, Department of Computer Science, University of Sheffield, Sheffield S10 2TN, United Kingdom, Europe.

Dr. Dawn. C. Walker, Department of Computer Science, University of Sheffield, Sheffield S10 2TN, United Kingdom, Europe.

Dr. Ifeyinwa E. Achumba, Department of Electrical & Electronic Engineering, Federal University of Technology Owerri, Imo State, Nigeria.

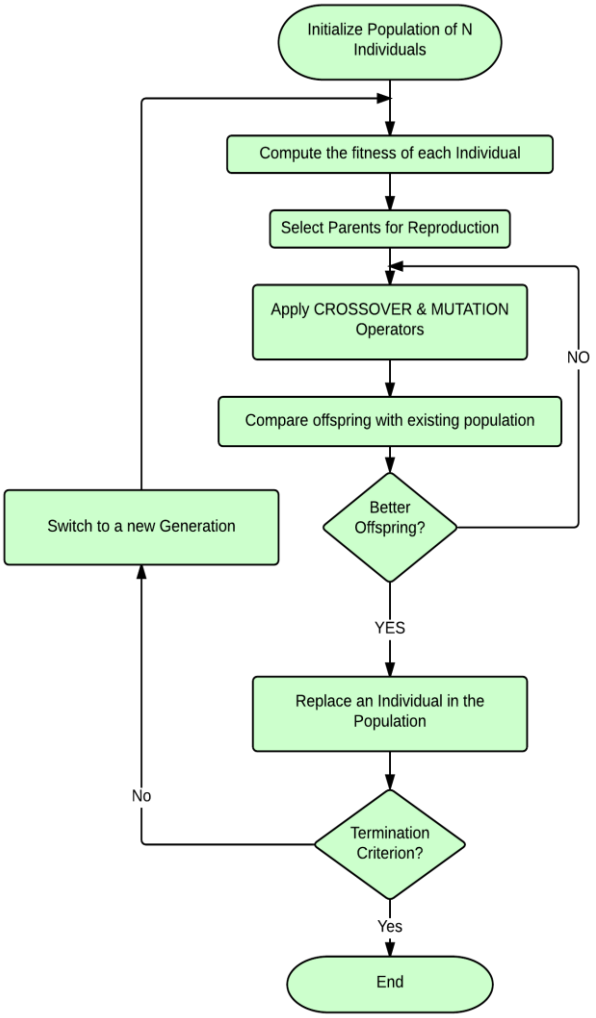

Figure 1 – A typical genetic algorithm

II. RELATED WORK ON APPLICATIONS OF GENETIC ALGORITHM TO SCHEDULING PROBLEMS

The work reported in [8, 9] shows that with a suitable representation method, a GA can be applied to various scheduling problems to obtain both minimally feasible (hard constraints satisfied) and optimized (most soft constraints also satisfied) solutions. The researchers in [8] took an approach that is typical of University systems. The problem size was moderately large in terms of number of events to be scheduled and number of constraints considered, precipitating the application of a grouping representation technique. However, the concept of different room types was not included.

The approach described in [13] is more generically applicable to a UTP. The problem space and size was large enough to represent a University faculty or school. A large number of hard and soft constraints, typical of a University environment, were also considered as were different room types The representation is similar to that described in [4], but a Sector-based Partially-mapped crossover (SB-PMX) operator was employed to ensure that the definition of the different types of rooms was maintained during the process.

Rushil et al [4] applied a GA to solving school weekly course timetabling problem, with some hard and soft constraints. In this case,, the notion of different room or class types was not taken into consideration, neither was the requirement of special facilities in a lecture room.

The work of Rupert et al [14] focused on an evolutionary algorithm for the solution of an Exam timetabling problem without violating the hardest constraints. They focused on the best form of heuristics that could be used to ensure the production of a highly optimized timetable. They concluded that the hybrid heuristic algorithm can perform well for very large problem sizes. However, 'how large' was not defined and they agreed that the problem instance used was small for experimentation purposes. They also did not take into consideration that some rooms may need to be specifically reserved for certain types of examination. Constraints are highly specific to each particular problem and there is no "one size fits all" approach. In this paper, we describe how a constructive initialization heuristic is applied to the population initialisation phase of a GA to solve a practical timetabling problem. The general design approach chosen for this work is to achieve feasibility of solution to this scheduling problem by applying a constructive initialisation heuristic and then optimize the obtained viable solutions using the iterative application of a GA. The major contribution of the work will be a design that avoids backtracking and infinite loops and the complexity of combination of multiple heuristics to achieve feasibility. We show that the use of simple but effective constructive initialisation heuristics increases speed as well as the probability of obtaining feasible solutions. The method we present is for a highly constrained problem. A pseudo-random GA initialization method will be explored to achieve both feasibility and optimization of the solution to the problem.

III. AN EXEMPLAR DEPARTMENTAL TIMETABLING PROBLEM

Generally, both the University and Departmental Timetabling Problems (UTPs and DTPs) are scheduling problems involving the allocation of resources at a particular time and location to ensure the success of a particular event. An event could be a lecture or laboratory class associated with a particular module, lecturer(s) and a cohort of students,. These resources (rooms, lecturers and time) are subject to certain constraints. A Departmental Timetabling Problem (DTP) is the subset of a UTP. A UTP has wider interdependence (e.g. competition between departments for resources such as teaching rooms) and greater complexity in logistics and will be considered in future works.

The problem presented here is a reduced version of the timetabling problem applicable to the Department of Computer Science at the University of Sheffield. The department runs various three and four year undergraduate degree programmes and a number of postgraduate taught programmes. They also contribute to dual honours degree and cross-faculty programmes. The undergraduate degree has up to four separate areas of specialization (Computer Science, Software engineering, Artificial Intelligence, and others) while the postgraduate taught masters degree has up to five separate programmes.

Each of the programmes has core modules as well as elective modules. Modules are commonly shared amongst two or more programmes, or across two or more levels of study. In some cases, multiple lecturers teach a module. This indicates complex and interdependent nature of the problem, even when restricted to the departmental level. In this context, cross-faculty and dual degree modules are not considered.

In the context of the work being presented, the timetabling problem considered is that of a weekly timetable that satisfies the following hard and soft constraints:

***Hard Constraint*:**

- **H1**: no two events can be fixed at the same venue at the same time.
- **H2**: multiple modules taught by a given lecturer cannot be fixed at the same time.
- **H3**: the core modules must not clash with one another for a given class group in the Department of Computer Science.
- **H4**: the time restrictions of all part-time lecturers must be accommodated.
- **H5**: all courses taught for the semester to all the cohorts must be accommodated.
- **H6**: lectures and labs that must be held at a specific time and venue within the week must be accommodated.
- **H7**: the room capacity must be sufficient for the number of students allocated to it for a particular lecture or lab.
- **H8**: each session must take place in an appropriate (lecture or laboratory) room type.

Soft Constraint Requirements

- **S1**: lecturers should not have more than 4 hours of teaching at a stretch per day.
- **S2**: students should not have more than 4 hours of teaching at a stretch per day.
- **S3**: there should be no lectures or labs fixed on Wednesday afternoons from 12noon.
- **S4**: lectures should not be fixed within lunchtime (1-2pm).

The challenges include formulation of appropriate chromosome representation for the problem, designing suitable fitness function, designing appropriate genetic operators and the use of constructive initialisation methods to solve the problem.

IV. THE DESIGN METHODOLOGY

The work presented in this paper is an adaptation and improvement of existing GA-based methods [4, 13] for solving timetabling problems. The methods of design relate to chromosome representation, GA population initialisation, genetic operators, fitness function and most importantly, the incorporation of a heuristic constructive initialisation technique, which we show to be critical in improving the efficiency/accuracy of our algorithm. The software system was designed and implemented using servlets and Java Server Pages (JSP) hosted on an Apache Tomcat web server. Web-based solution was important to ensure remote and simultaneous access to same data and application by various departments.

A. Chromosome representation

In order to apply GA operators, a suitable chromosome representation was adopted based on the work of other researchers in [13] in order to fit the DSP problem considered here.. The chromosome in [13] was represented as a 2-dimensional array in which the rooms are the rows while timeslots are the columns. The rooms are sub-divided internally into laboratory rooms and lecture-only rooms. A gene in the chromosome is defined as a locus of the intersection of a room and a timeslot. Teaching events (integer numbers) are then designated to a particular array cell. This 2-D approach was implemented in this context due to the tabular nature of the problem. The content of an intersection will be an integer, which represents a teaching event. This 2-D chromosome representation is shown in Table 1 below.

Table 1 – Sector-based representation technique [adapted from 13]

R/T	Mon1	Mon2	Mon3	…	Fri38	Fri39	Fri40
Room1	6			…		3	
Room2		1	3	…	7		6
Room3	2			…	9	8	
Room4		3		…	6		
Lab1	7			…		4	4
Lab2		6		…			
Lab3			5	…	3		10

Table 1 above shows that some genes are empty, representing a free period during which a room is not occupied. Also, it can also be seen that some modules (number in a cell) appear more than once. These represent modules with multiple periods or modules that have both tutorial and laboratory components.

Figure 2 below highlights the genotypic to phenotypic mapping used.

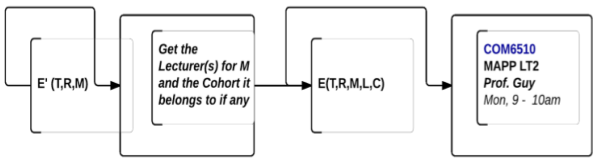

Figure 2 – Genotype – to Phenotype transformations for the UTP

It should be noted that T, R, M, L and C as used in Figure 2 represent integers that uniquely identify a time slot, lecture or lab room, module, lecturer and cohort respectively. E' is the event as directly represented in the chromosome while E is an intermediate transformation of the genetic content before it is finally mapped onto its equivalent phenotype that represents a part of a timetable.

The integer, which represents a module, has associated with it a lecturer and the student cohorts that take the module. With this internal representation, it will be possible to check clashes between scheduled modules, as well as display the final timetable in human readable form.

B. GA Population initialisation

The aim here is to use a quasi-random initialisation method that achieves a good balance between population diversity and the feasibility of the final solution.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Constraints H4 (Time preference of part-time lecturers) and H6 (lectures that must hold in specific room at a given time) are so rigid in nature that random initialisation will seldom lead to a feasible solution when the conventional genetic algorithm is applied to such a randomly initialised population. Hence, the process of building the initial population ensures that these constraints are embedded in the initial population of chromosomes.

The constructive GA population initialisation used in this research is shown as part of the overall adapted GA algorithm in figure 3. In constructive initialisation process, the most constrained events (H4 and H6) are scheduled first. After this, one should randomly schedule other events with fewer constraints. However, every event in this problem must lie within one of two domains: laboratory-based or tutorial-based events. The scheduling process from the CI and also the fitness function takes this concept of domain into consideration.

C. Design of Genetic Operators

The genetic operators considered and designed in this research represent selection, crossover and mutation.

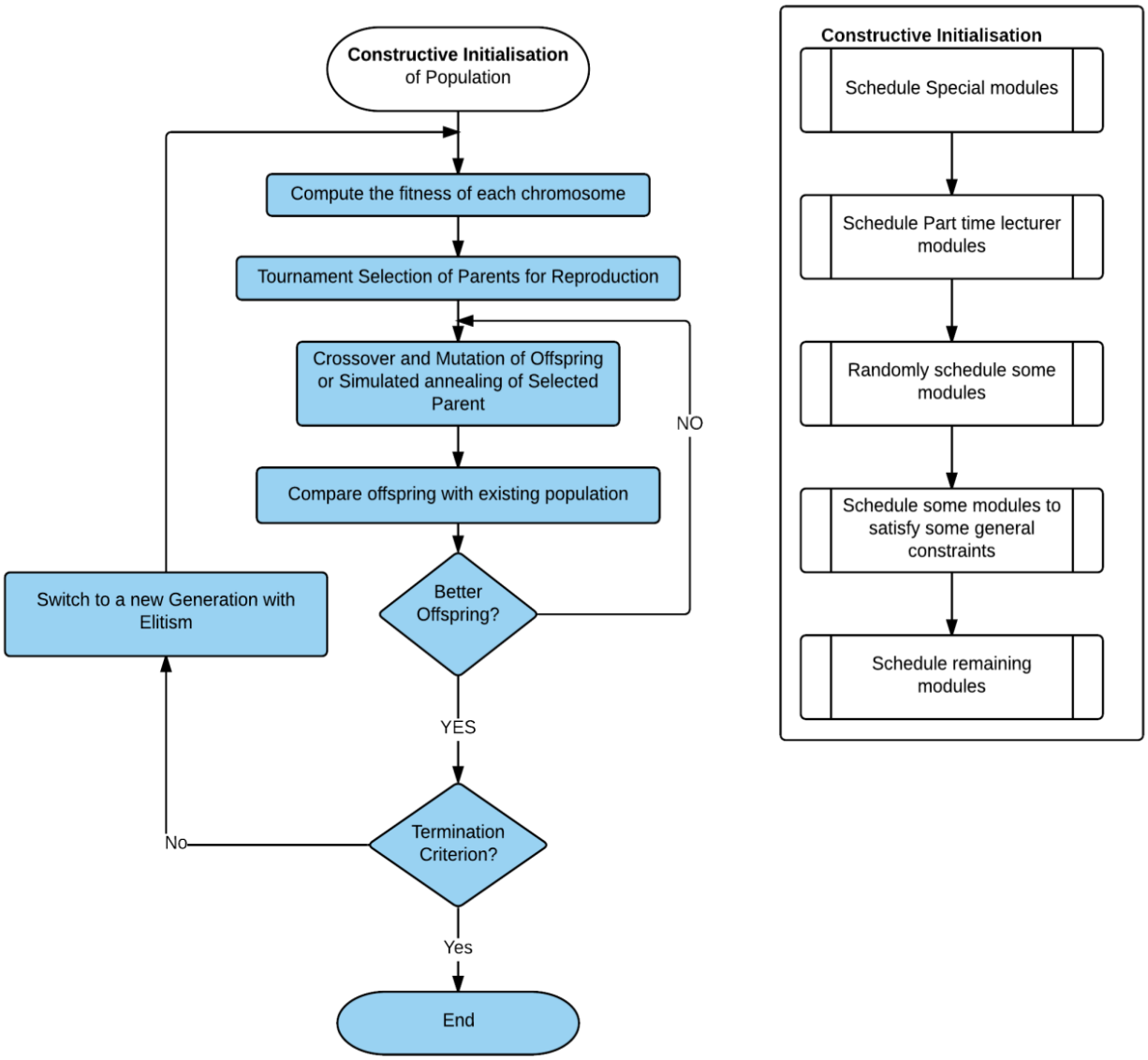

Figure 3 – Adapted GA through constructive Initialisation

The tournament selection operator was used in this research. The choice is based on the fact that selection probability can be tailored towards ensuring that better parents are more frequently chosen for reproduction. This method involves picking two individuals from a population and staging a contest to determine which one will be selected as a parent. As described in [16] and [7], this contest involves having a pre-determined selection probability and generating a random number between 0 and 1 and comparing it with this pre-determined value. The pseudo-code for the tournament operator designed for this work is given in Algorithm 1 below.

```
Randomly select any two chromosomes of different fitness from the entire
population
Set predetermined_probablity  Generate a random number between 0 and
1
  if random number <= pre-determined_probability
    select the fitter candidate
else
    select weaker candidate
```

Algorithm 1 – Design of Tournament Operator

A pre-set tournament selection probability of 0.7 will be used in this design. This is to favour the selection of better chromosome for reproduction. The crossover operator used in this research is 2-D in nature for compatibility with the 2-D chromosome on which it will operate. It is an adaptation of the Partially-mapped Crossover (PMX) operator used in [13]. The adaption of the normal 1-D PMX for a 2-D chromosome is shown in Algorithm 2 below

```
Randomly select a segment (a,b) of genes from parent 1 and copy them directly to
OffSpring2(O2).
For each element in same segment,i=a to b, in parent 2
        Check if P2i(r,t) has been copied to O2
        if not yet copied
                while P2i(r,t)is part of segment (a,b)
                        find the corresponding value,v, from parent1 i.e
                P1i(r,t)
                        get the location of v in Parent2 i.e P2v(r,t)
                        P2i(r,t) = P2v(r,t)
                repeat while
                O2i(r,t) = P2i(r,t)
        endif
repeat for
  Copy any remaining positions from Parent2 to Offspring2
```

Algorithm 2 – A 2-D PMX Algorithm

For the above algorithm, ***r = room index***, while ***t= time index***. A gene in our representation is defined by both indices. Based on results reported in [13] and [17] and also on our own trial results, the crossover probability was fixed at 0.75. A swap mutation operator was adopted for this research. This is due to the nature of the chromosome representation method. The algorithm designed for our mutation operator is shown in Algorithm 3 below.

```
Set a mutation_probability
formutation_probability x no_of_timeslots
    pick a room at random
    locate a module-occupied gene C1(r,t) in this room
    randomly pick another location C2(r,t)
    copy C1(r,t) to a temporary location temp
    copy C2(r,t) into C1(r,t)
    copy temp into C2(r,t)
repeat
```

Algorithm 3 – A 2-D Swap Mutation Operator

In the above algorithm C1(r,t) must not be empty while C2(r,t) may or may not be empty. This is because it would be meaningless to swap two empty genes.. A small swap mutation probability of 0.01 (or 1%) was used to limit possible disruption of very good chromosomes.

D. Design of Fitness function

The fitness function adopted for this problem was based on the concept of reward. Points are allocated for the non-violation of an instance of any of the 12 constraints in a particular chromosome in the entire population of possible solution to the problem. Hard and soft fitness functions are then defined based on the sum of points earned for each constraint. The Hard fitness function f(H), is simply the sum of all the percentage rewards associated with each hard (H) constraint. Thus, the algorithms evaluate all points of potential violation of a particular constraint and not just summarising if a constraint is violated or not. For instance, the algorithm can determine if only 8 out of 10 lectures can be scheduled in a room of adequate capacity. However, one might reason that it would prolong the run time for the algorithm, if there were many good genes in a chromosome. This is defined in equation 2 below.

F (H) = % of H2 + % of H3+% of H4+% of H5+ % of H6+ % of H7+ % of H8 ..... (2)

It should be noticed in equation (2) that H1 was omitted. This is intentional. The nature of the chromosome representation ensures that the constraint of fixing two events at the same venue and at the same time will never occur (see Table 1). Hence, all individuals in the population will always have same fitness value for this constraint. We simply assumed the fitness value to be zero so that it cancels out for all individuals in the population.

For a chromosome to be a feasible solution to the timetabling problem, equation 3 must be satisfied.

$F(H) = 100\% \text{ of } f(H)_{max}$ .....(3)

Where :

$F(H)_{max} = H2_{max} + H3_{max} + H4_{max} + H5_{max} + H6_{max} + H7_{max} + H8_{max}$ .....(4)

On the other hand, the Soft fitness function, f(S), is the sum of the rewards from each constraint considered as a soft(S) constraint. This is given by equation, 5:

$(S) = S9 + S10 + S11 + S12$ ..... (5)

The maximum value that f(S) can have is given as $f(S)_{max}$ and is given by equation 6.

$F(S)_{max} = S9_{max} + S10_{max} + S11_{max} + S12_{max}$ .....(6)

Thus, the overall fitness value, $FV_{overall}$, for a chromosome is therefore a combination of equations 2 and 5. Thus:

$FV_{overall} = F(H) + F$

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

(S)..... (7)

However, given that equation 3 must be true for an acceptable chromosome (due to the fact that all hard constraints must be satisfied), it becomes evident that equation 7 is the objective function that the genetic algorithm will tend to maximize.

The program file for the implementation of the algorithm in figure 3 and all these design computations is kept in the git repository:
https://github.com/KingPeter2014/Uosutpsolver.git.

V. EXPERIMENTS

The major aim of this research is to solve the University Timetabling problem as it occurs in the Department of Computer Science at the University of Sheffield by designing and implementing a suitable GA. Hence, experiments based on five levels of difficulty were carried out to determine the efficacy of the developed algorithms in solving the problem and also on the effectiveness of the representation method, initialisation method and the genetic operators used. The five levels are determined by the number of core and elective modules and on the number of constraints factored into the experiments. Table 2 states the data used for the five test cases.

Table 2: Test Cases and Test data for Experiments

		TEST CASES				
		1	**2A**	**2B**	**3A**	**3B**
1	No of Laboratories	6	6	6	6	6
2	No of lecture rooms	19	19	19	19	19
3	No of Lecturers	29	29	29	29	29
4	No of lecture-based modules	10	12	12	20	20
5	No of lab-based modules	5	7	7	13	13
6	No of Part-time lecturers	0	0	2	0	2
7	No of special rooms with time constraints	0	0	3	0	10

- CASE 1 - Only Level 1 (five core and one optional module) and 2 (six core) modules were considered. H4 (Part time lecturers) and H6 (Special room and time constraints) were not considered in calculating the number of hard constraints satisfied.
- CASE 2A – Core and optional Levels 1 to 3 modules were considered but H4 and H6 constraints were not considered.
- CASE 2B - It is the same as 2A but there are two part time lecturers and three modules with special room and time requirements.
- CASE 3A - It includes all core and optional modules from levels 1-3, level 4 (M.Eng) and level 6 (postgraduate M.Sc). However, no part-time or special room and time requirements were considered.
- CASE 3B - It is the same as 3A but it has up to ten modules with special room and time requirements plus two part time lecturer requirements.

All other constraints, whether soft or hard, were always included for all calculations. For each of the test cases, following the constructive initialization step, the GA was run ten times with a population size of 100 for 500 generations. Each experiment was repeated 10 times and average value of the results taken. The experiments were run on MAC OSX 10.9.5 with Intel Core i5 with processor speed of 2.6GHz and 8GB 1600MHz DDR3 RAM memory.

VI. RESULTS AND DISCUSSION

The success of the designed algorithm is measured in terms of convergence rate, average number of feasible solutions obtained, level of soft constraint satisfaction and the time taken (speed) in terms of CPU time to run the GA in each case. The results will be presented and discussed using appropriate graphs. Case 3B above is the case that is most typical of the degree of complexity of real timetabling problem and will thus be the focus of the evaluation.

- ***Convergence rate***

The goal of evolutionary algorithms is to improve a candidate solution until it is maximally optimised and cannot be improved further (convergence). Here, we measure the rate of increase of fitness value of the best chromosome over the generations for each run. The average convergence rate for case 3B is shown in figure 4. The fitness value increased progressively and consistently from fitness value of about 93.8 per cent to 95.8 per cent from zero to 500 generations. Note that the constructive initialisation has fixed the fitness value at about 93.8 per cent in most cases before evolution of the chromosomes.

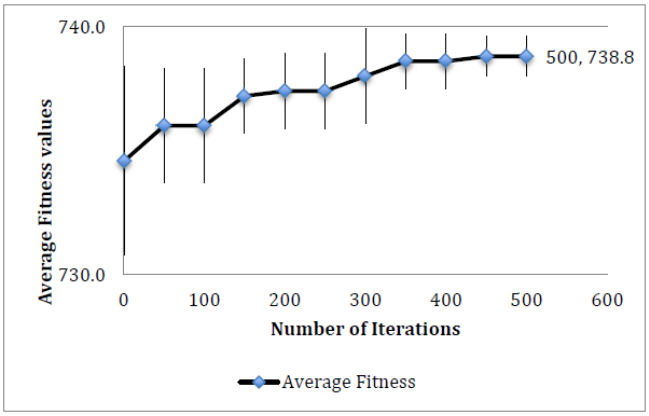

Figure 4 – Convergence and Solution optimisation of the GA

The error bars represent standard deviations for the 10 runs of the GA

- ***Number of feasible solutions***

The distribution of the number of runs that produced at least one feasible solution per run per test case is shown in figure 5 below.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

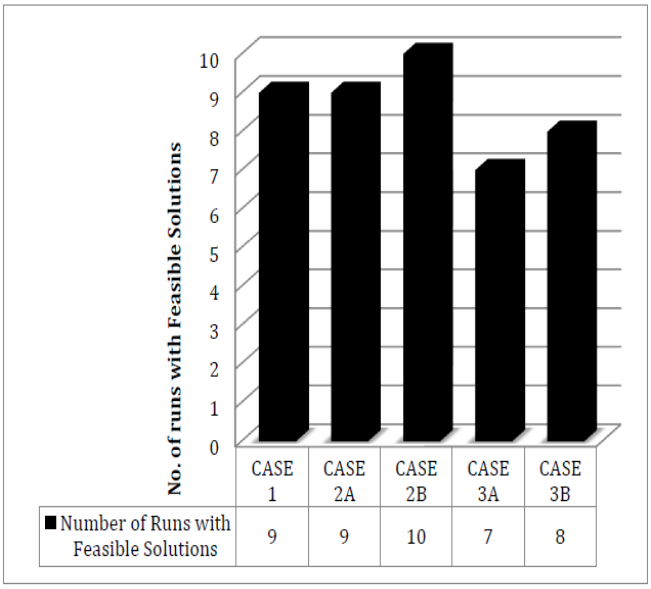

Figure 5 – Number of runs from a total of 10 per case with at least one feasible solution out of 100 candidates

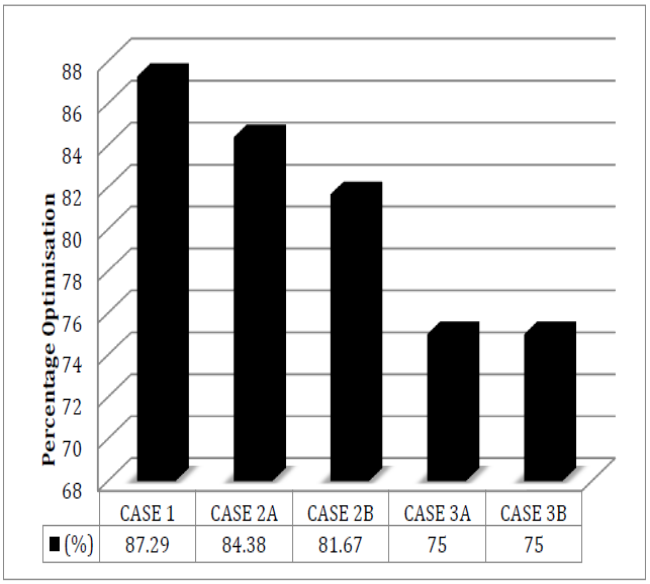

Figure 6 – Percentage optimisation of soft constraints obtained per test case

Hence, at least 7 out of every 10 runs for each of the test cases produced at least one feasible solution. As shown in figure 5 above, 8 out of 10 runs of the algorithm with Case 3B produced feasible solutions. It should be noted that this case contains all the taught modules offered by the Department of Computer Science in the autumn semester for all levels of study. Thus, with constructive initialisation, feasible solutions can be obtained to solve the timetabling problem. Note that in the absence of constructive initialization, no feasible solutions were found for any case even after running the GA for 1000 iterations.

The rather unexpected pattern of feasible solutions needs some explanation. One would expect the number of feasible solutions to decrease progressively as the difficulty level of the problem increases. The reason that this is not observed here is that a degree of randomisation was introduced into the constructive initialisation process to ensure diversity of the population. Hence, some random genes were introduced before constructively created genes were inserted. Feasibility was not guaranteed for any of the test cases during initialisation but was possible. Hence, the above result was stochastic in nature. Most of the solutions that were not feasible were very close to feasibility (each up to 98% feasible).

- ***Level of Soft Constraint satisfaction***

It can be seen from figure 6 that, as expected, Case 1 has the best level of solution optimisation. A trend apparent in figure 7 is that optimisation reduces as problem size and constraint stringency increases. One can see the progressive decrease in optimisation from 87% in case 1 to 75% in case 3B. However, the optimisation level of 75% for case 3B, which is of greatest interest in this research, represents a promising level of soft constraint satisfaction.

- ***Solution time***

Figure 7 shows the average total time taken to run the GA for each test case. According to the profiling that was performed, the GA took longer to evaluate a chromosome with better genes than a chromosome that has bad genes. The reason for this is that the GA was designed to minimize the amount of time spent on iterations that will not produce a better offspring. A reward-based fitness function was used, where a condition is initially evaluated to determine whether a gene is fit to satisfy an instance of a constraint. Only in the event that the gene is found to be fit, will the body of the conditional statement be evaluated in order to give a reward to the chromosome containing the gene. Hence, if the gene is "bad", the body of the conditional is not executed. This implies that the more "bad " genes that exist in a population, the less time it will take to evaluate the entire population for that particular model iteration.

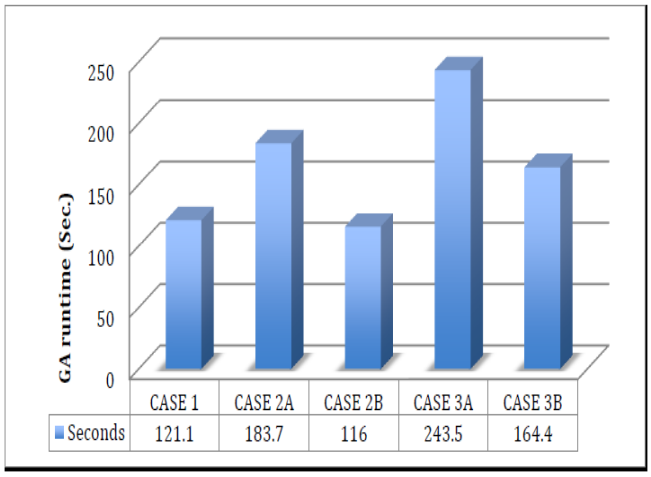

Figure 7 – Average time to run the GA

Without constructive initialisation (CI), the solution never converged to a feasible solution within the 500 generations or the upper limit of runtime allotted for the GA to run. For case 3B, constraints H4 (time preference of part-time lecturers) and H6 (lectures that

must be held in a specific room at a given time) may never be satisfied by a totally random initialisation process. Though eventually a feasible solution may be possible, a higher capacity computer may be needed to ascertain the possibility. However, the time (computational and clock) will definitely be longer than necessary.

VII. CONCLUSION AND RECOMMENDATIONS

Here we have demonstrated the effectiveness of a genetic algorithm incorporating a constructive initialization pre-processing stage, for solving a subset of the UTP problem, based on a real problem from the Department of Computer Science at the University of Sheffield. The complexity of the problem was large – incorporating 40 time slots, 25 type-specific teaching rooms and up to 33 modules, giving rise to chromosome containing1000 genes. With 100 candidate solutions and iterating up to 500 generations, one should appreciate the problem size as compared to the test system described in [13]. Thus, the average runtime of **2.74 minutes per run of the GA** is acceptable based on the fact that it takes the timetabling officer hours of work to resolve equivalent problems. Comparing this with the time currently spent by the timetabling officer (hours of work for each manual change and resolution of clashes), it is obvious that there is a significant improvement in the time taken to generate a feasible and flexible timetable based on changes in input parameters and constraints. Future extensions to this work include adapting the algorithm to take into account the fact that other departments also compete for teaching space, allowing for the fact some modules may have irregular teaching periods for some weeks of a semester,. More generic investigations leading to potential improvements could include a comparative investigation into the speed of reward-based and penalty-based fitness functions and the design of appropriate genetic operators that could allow the GA to obtain a solution to a highly constrained problems without constructive initialization, or the possibility of developing a distributed form of the algorithm that could be run on a multicore cluster or a GPU-based environment.

REFERENCES

1. Sadaf N.S and Shengxiang Y., (2009). "A guided search Genetic Algorithm for the University Course timetabling problem" In Multidisciplinary International Conference on Scheduling: Theory and applications 10 -12 August 2009, Dublin Ireland.
2. Even S., Itai A., and Shamir A., (1976) ."On the complexity of timetable and multi commodity flow problems" In SIAM Journal on Computing, 5(4) pp 691 - 703.
3. Jeffrey H. Kingston (2006). "Hierarchical Timetable Construction" In Edmund K. Burke & Hana Rudova (Eds.). Practiceand Theory of Automated Timetabling. Proceedings of the 6th International Conference on the practice and Theory of Automated Timetabling, 30th August - 1st September 2006. pp 196 - 208.
4. Rushil Raghavjee and NetishaPilley (2008) "An Application of Genetic Algorithms to the School Timetabling Problem" In SAICSIT Conference Proceeding, 6-8 October, 2008. pp.193 - 199. url: www.titan.cs.unp.ac.za/~nelishiap/uploads/45.pdf.
5. Dario Floreano and Claudio Mattiussi (2008).Bio-Inspired Artificial Intelligence: theories, methods and technologies. MIT Press Cambridge, USA.
6. Bashir S.A (2014). Developing a Java-based Genetic Algorithm to solve the Travelling Salesmans Problem. MSc Dissertation, Department of Computer Science, University of Sheffield.
7. Mehdi et al (2012). "Solving University Course Timetabling Problem using Genetic Algorithm" In 2nd World Conferenceon Information Technology. AWER Procedia Information Technology and Computer Science. Vol 1 (2012).pp 1033 - 1040.
8. Abubakar M.S et al (2006). "Maintaining diversity for Genetic Algorithm: A case study of timetabling problem"In JurnalTeknologi 44 (D) June 2006, pp.123 - 130.
9. Tormos P. et al.(2008). "A Genetic Algorithm for Railway Scheduling Problems" In Studies in Computational Intelligence (SCI) 128, pp. 255–276.
10. Fraser G. and Acuri A. "ALarge Scale Evaluation of Automated Unit Test Generation Using Evo Suite". [online]: http://www.evosuite.org/wpcontent/papercite-data/pdf/tosem_evaluation.pdf , retrieved on 19th March, 2015.
11. Els R. and Pillay N. (2010)."An Evolutionary Algorithm Hyper-Heuristic for Producing Feasible Timetables for the Curriculum Based University Course Timetabling Problem" In 2010 Second World on Nature and Biologically Inspired Computing Dec. 15-17, 2010 in Japan.Pp 460 – 466.
12. Yu and K. Sung (2002)."A genetic Algorithm for weekly courses timetabling problem" In International transactions in Operational Research, 9 (2001), pp 703 - 717.
13. W. Rupert, B. Edmund and E. Dave (1995). A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems. Computer Science Technical Report No. NOTTCS-TR-1995-8.
14. S.A. Oyebanjo (2013). Development of a University Timetabling Automation System.B.Sc Project, Department of Computer and Information Science, Covenant University, Nigeria.
15. D.W. Dayer (2010). Evolutionary Computation in Java: A practical guide to the watchmaker Framework. [Online]: http://watchmaker.uncommons.org/manual/index.html retrieved 3rd March 2015.
16. W. Chinnasri, S. Krootjohn, and N. Sureerattanan (2012) "Performance comparison of Genetic Algorithm's crossover operators on University Course Timetabling Problem" In Proceedings of 8th International Conference on Computing and Information Management (ICCM), 24th -26h April, 2012 in South Korea.

AUTHORS PROFILE

Eze Peter Uchenna obtained Masters of Science in Engineering (M.Sc (Eng.) in Advanced Software Engineering from University of Sheffield in 2015. He obtained Bachelors of Engineering (First Class Hons.) in Electrical/Electronic Engineering at Federal University of Technology Owerri, Nigeria in 2010. He worked as a Softtware Engineer at Internet Experts Nigeria Ltd in 2011 - 2012 and currently works as an Assistant Lecturer in the Department of Elecltrical/Electronic Engineering, Federal University of Technology, Owerri, Nigeria. His areas of research interest include Agile Software design, biometric and steganographic information security systems and software-controlled electronic systems.

Dr. Dawn C. Walker Profile obtained BSc Hons Physics from the University of Durham UK in 1996 and a Ph.D. in Medical Physics from the University of Sheffield, UK in 2001. She was appointed as a lecturer in Sheffield in 2010 and as a Senior Lecturer in 2014. Her research interests relate to the use of computational simulation for the understanding of biological systems and the use of bio-inspired algorithms for the solution of real world problems.

Dr. Ifeyinwa E. Achumba is a Senior Lecturer in the department of Electrical and Electronic Engineering at Federal University of Technology Owerri, Imo State Nigeria. Her major area of research interest is in the application of Artificial Intelligence in solving human problems.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication