Scientific Research and Essays Vol. 7(17), pp. 1727-1733, 9 May, 2012
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE11.2103
ISSN 1992-2248 © 2012 Academic Journals

Full Length Research Paper

Performance evaluation of simulated annealing and genetic algorithm in solving examination timetabling problem

Oyeleye, C. Akinwale*, Olabiyisi, S. Olatunde, Omidiora, E. Olusayo and Oladosu, J. Babalola

Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria.

Accepted 23 April, 2012

This work details the performance evaluation of simulated annealing (SA) and genetic algorithm (GA) in terms of their software complexity measurement and simulation time in solving a typical University examination timetabling problem (ETP). Preparation of a timetable consists basically of allocating a number of events to a finite number of time periods (also called slots) in such a way that a certain set of constraints is satisfied. The developed software was used to schedule the first semester examination of Ladoke Akintola University of Technology, Ogbomoso Nigeria during the 2010/2011 session. A task involving 20,100 students, 652 courses, 52 examination venues for 17 days excluding Saturdays and Sundays.The use of the software resulted in significant time saving in the scheduling of the timetable, a shortening of the examination period and a well spread examination for the students. Also, none of the lecturers / examination invigilators was double booked or booked successively. It was clearly evident that simulated annealing performed better than genetic algorithm in most of the evaluated parameters.

Key words: Simulated annealing, genetic algorithm, examination timetabling, software complexity and simulation time.

INTRODUCTION

Preparation of a timetable consists basically of allocating a number of events to a finite number of time periods (also called slots) in such a way that a certain set of constraints is satisfied. Two types of constraints are usually considered, hard constrains, that have to be fulfilled under all circumstances, and soft constraints, that should be fulfilled if possible. In some cases, it is not possible to fully satisfy all the constraints, and the aim turns to be finding good solutions subject to certain quality criteria (for example, minimizing the number of violated constraints, or alternatively satisfaction of hard constraints, while the number of violated soft constraints is minimized (Keshav et al., 2007). Hard constraints are the most important constraints to completely resolve in order to prevent the double booking of lecturers, students or classrooms. A practical timetable that does not violate hard constraints is called a feasible timetable. The second international timetabling competition (ITC, 2007) described hard and soft constraints (Di Gaspero et al., 2007).

Timetabling are combinatorial optimization problems, which consist of scheduling a set of courses within a given number of rooms and time periods. Solving a real world timetabling problem manually often requires a significant amount of time, sometimes several days or even weeks (Abdennadher et al., 2000). The manual solution of the timetabling problem usually requires several days of work and the final solution may be unsatisfactory because it is a highly complex task to verify all constraints.

For the aforementioned reasons, considerable attention

*Corresponding author. E-mail: Letuskii@gmail.com or caoyeleye@lautech.edu.ng. Tel: +2348034295229.

has been devoted to automated timetabling. A large number of variants of the timetabling problem have been proposed in the literature, which differs from each other based on the type of institution involved and the type of constraints imposed by the examination policy of the institution. Preparation of an academic examination timetable is a typical scheduling problem that appears to be a tedious job in every academic institute once or twice a year. The problem involves the arrangement of courses, students, teachers and rooms at a fixed number of time-slots, respecting certain restrictions. Wren defines the general problem of timetabling as follows: “Timetabling is 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” (Wren, 1996).

The inability of the classical methods to handle the large number of real and integer variables involved in solving this class of problem and especially the number of constraints involved paved way for the adoption of non-classical techniques. Simulated annealing (SA), Tabu search (TS), Genetic algorithm (GA), Memetic algorithm (MA) and Ant colony system (ACS) are among the main algorithms for solving challenging problems of intelligent systems (Zahra, 2005). In this research, two of these techniques were carefully studied and compared in terms of their software complexity and simulation time.

Genetic algorithm (GA) is one of the most popular optimization solutions. It has been implemented in various applications such as scheduling. The operators of GA such as selection, crossover and mutation are applied to populations of chromosomes. Simulated annealing (SA) is a random-search technique which exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) (Elmohamed et al., 1998; Omidiora et al., 2009). In addition, the search for a minimum in a more general system forms the basis of an optimization technique for solving combinatorial based problems. It is generally regarded as a modified version of hill climbing algorithm. It has been proved that by carefully controlling the rate of cooling of the temperature, SA can find the global optimum. However, this requires infinite time. Fast annealing and very fast simulated re-annealing (VFSR) or adaptive simulated annealing (ASA) are each in turn exponentially faster and overcome this problem. SA's major advantage over other methods is an ability to avoid becoming trapped in local minima. The algorithm employs a random search which not only accepts changes that decrease the objective function (assuming a minimization problem), but also some changes that increase it. An effective solution technique to the problem could be applied to other scheduling problems (Abramson, 1991). The problem drew the attention of the researchers in the early 60’s with the study of Gotlieb in 1962, who formulated a class-teacher timetabling problem by considering that each lecture contained one group of student, one teacher, and any number of time-slots which could be chosen freely.

Schaerf, surveyed that most of the early techniques for automated timetabling were based on successive augmentation (Schaerf, 1999), where a partial timetable was filled in lecture by lecture until either all lectures were scheduled or no further lecture could be scheduled without violating constraints. In another survey, Abramson (1991) reported the general techniques applied to the problem in the past, such as network flow analysis, random number generator, integer programming, and linear algorithm. In addition to these, worth mentioning methods are exact method-based heuristic algorithm (De Werra, 1985), and graph coloring theory (Neufeld and Tartar, 1974). However, the classical techniques are not fully capable to handle the large number of integer and/ or real variables and constraints, involved in the huge discrete search space of the timetabling problem. These inadequacies of classical techniques have drawn the attention of the researchers towards the non-classical techniques. Worth mentioning non-classical techniques, that are being used to solve the problem, are genetic algorithms (Colorni et al., 1994; Abramson and Abela, 1992), neural network, simulated annealing, and tabu search algorithm.

However, compared to other non-classical methods, the widely used are the genetic/ evolutionary algorithms (GAs/ EAs). The reason might be their successful implementation in a wider range of applications (Al-Attar, 1994). Piola (1994) applied three evolutive algorithms to school timetabling problem, and showed their capability to tackle highly constrained combinatorial problems, such as timetabling problem. A timetable is essentially a schedule which must suit a number of constraints. Constraints are almost universally employed by people dealing with timetabling problems (Burke et al., 1994).

METHODS

After representing the problem mathematically, the two algorithms employed were implemented using Matlab development kit on an Intel® Dual core CPU with 220 GHz speed, 2.91 GB Random Access Memory (Accessible) and 146 GB hard disk drive with windows 7 ultimate edition.

Problem representation

Examination timetabling is a specific case of the more general timetabling problem. In the case of examination timetabling, a set of exams $E = \{e1, . . . , en\}$ to be scheduled within a certain number of periods $P = \{p1, . . . , pm\}$ subject to a variety of hard and soft constraints (Piola, 1994; Burke and Ross, 1996). Table 1 contains the constraints considered in this work.

Simulated annealing pseudo code

The standard simulated annealing that was coded using the Matlab development kit is presented as follows:
Start with the system in a known configuration, at known energy E

Table 1. Summary of constraints considered.

Label	Definition
HC1	The number of exams a student will write at a time
HC2	Number of classes a teacher should be at a time
HC3	Number of examination in the schedule
HC4	The type and capacity of the room where a class is to be scheduled
HC5	Number of timeslot at which an examination of a course is to be scheduled
SC1	Total number of free time-slots between two examinations (or events) of students
SC2	Total number of consecutive classes of a teacher

HC: Hard constraints; SC: Soft constraints.

```
T = temperature = hot; frozen = false;
While (! frozen) {
        repeat {
          Perturb system slightly (e.g., moves a particle)
          Compute E, change in energy due to perturbation
          If (ΔE< 0)
 Then accept this perturbation, this is the new system config
          Else accept maybe, with probability = exp (-ΔE/KT)
          } until (the system is in thermal equilibrium at this T)
          If (ΔE still decreasing over the last few temperatures)
 Then T = 0.9T//cool the temperature; do more perturbations
 Else frozen = true
 }
return (final configuration as low-energy solution)
```

Genetic algorithm pseudo code

The standard genetic algorithm which was also coded using the Matlab development kit is also as follows:

```
Genetic algorithm: the Pseudo code
Input: $\mu, \lambda, \Theta_i, \Theta_r, \Theta_m, \Theta_s$
Output: $a^*$: The best individual found during the run,
        $P^*$: The best population found during the run.

1. t $\longleftarrow$ 0;
2. P (t) $\longleftarrow$ initialize ($\mu$);
3. F(t) $\longleftarrow$ evaluate (P (t), $\mu$);
4. While ($i$ ($P(t)$, $\Theta_i$) $\neq$ true) do
5. P′ (t) $\longleftarrow$ recombine (P (t), $\Theta_r$);
6. P′′ (t) $\longleftarrow$ mutate (P′(t), $\Theta_m$);
7. F(t) $\longleftarrow$ evaluate (P′′(t), $\lambda$);
8. P (t + 1) $\longleftarrow$ select (P′′(t), F(t), μ, $\Theta_s$);
9. t $\longleftarrow$ t + 1;
do
```

The input parameter sets Θ_i, Θ_r, Θ_m, and Θ_s of the basic operators. Notice that recombination was allowed to equal the identity mapping: that is, $P''(t) = P'(t)$ is possible.

Data used for the work

The following are the set of data used to automatically generate the examination timetable:

i. Available venues and their corresponding capacity
ii. Special examination venue (if any) and capacity
iii. List of subjects (exams) to be written
iv. The list of all registered students per exam or course
v. The list of available invigilators
vi. Maximum examination period (no of exam days or weeks)
vii. Duration of each examination (maximum number of hours)

Complexity of the two algorithms

Halstead software complexity and Lines of Code (LOC) were used to evaluate the two coded algorithms. Halstead measure calculates program volume (V), program effort (E), program level (L) and intelligence content of the program (I). Table 2 contains the formulae for measuring all the metrics. All these measures are valid under the assumption that the program is "pure," that is, free of the so-called "poor programming practices" (Olabiyisi et al., 2005, 2007).

RESULTS

After implementing the two algorithms, Table 3 shows the measured parameters and their various values used in calculating the software complexity of the two algorithms. It should be noted that n1 is the number of distinct operators found in the program, n2 is the number of distinct operands, N1 is the total number of operators, N2 is the total number of operands, N is the addition of N1 and N2 and n is the addition of n1 and n2.

DISCUSSION

As shown in Table 4, the two considered algorithms produced feasible solutions because none violated the constraints considered in this work.

Simulation time

The time utilized by an algorithm to run until the result is produced is usually called execution time or simulation time. Table 4 and Figure 1 show the measured values of the simulation time of the two considered algorithms. The simulation time of GA and SA are 19.73 and 56.16 seconds respectively to return a feasible examination timetable. This is clear evidence that SA utilized more time than GA.

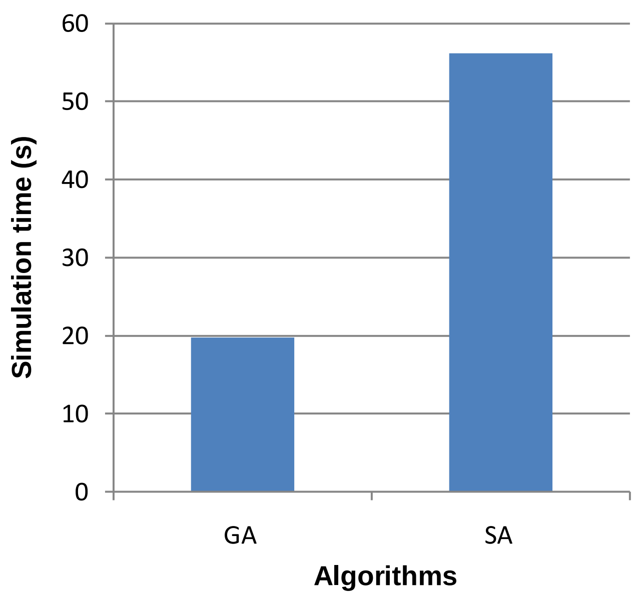

Figure 1. The simulation time of the two algorithms.

Table 2. Formulae for measuring the complexity metrics of both algorithms.

Complexity metrics	Formulae
Volume (V)	N* $\log_2 n$
Effort (E)	V/L
Program level (L)	(2*n2) / (n1*N2)
Intelligent content of the program (I)	L*V

Table 3. Data obtained for measuring the complexity of both algorithms.

Parameters for measuring complexity	GA	SA
No. of distinct operators (n1)	14	10
No. of distinct operands (n2)	42	54
Total number of operators (N1)	267	260
Total number of operands (N2)	96	88
N i.e. (N1+N2)	363	348
n i.e. (n1+n2)	56	64

Program size

The program size is the amount of disk space occupied and it is usually measured in bits, bytes, kilobytes, Megabytes, Gigabyte, Terabytes, etc depending on the actual size under consideration. Table 4 and Figure 2 show that the program sizes of GA and SA are 20 and 16.5 Kb respectively. This is an indication that GA code utilized more disk space than SA.

Lines of code

The lines of code (LOC) are the number of lines of the executable codes in a program. Table 4 shows the LOC of GA and SA as 500 and 256 respectively. These values show that GA code has more number of executable lines of code than SA.This is an indication that the implementation time and effort required by GA was more than that of SA.

Program volume

The program volume is the value that signifies the volume of the computer memory being utilized during the execution of the implemented algorithms. The program volume for Genetic Algorithm, Simulated annealing are 2108.07 and 2088.00 respectively. Table 4 and Figure 3 show that SA occupies lesser memory space in terms of volume than GA.

Program effort

This is widely known as the number of discriminations made in the preparation of a program, it specifies the extent to which personnel involved in software production are effectively engaged. It could also be referred to as the quantitative measure of the effort involved in the implementation of an algorithm.

The measured values are presented in Table 4 and Figure 4. The program effort of GA and SA are 33729.12 and 17013.33 respectively. This is an indication that the program effort of GA is higher than that of SA.

Table 4. Data obtained during and after the execution of both algorithms.

Parameter	GA	SA
Simulation time (seconds)	19.73	56.16
Number of Courses clashed	0	0
Number of lecturers double booked	0	0
Program size (KB)	20	16.5
Lines of code	500	256
Program volume (V)	2108.07	2088.00
Program effort (E)	33729.12	17013.33
Difficulty of understanding the program (L)	0.06	0.12
Intelligent content of the program (I)	131.75	256.25

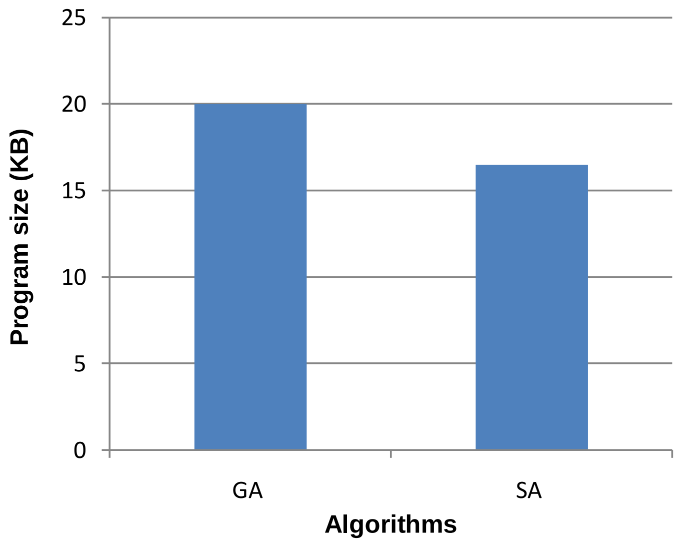

Figure 2. The program size of the two algorithms.

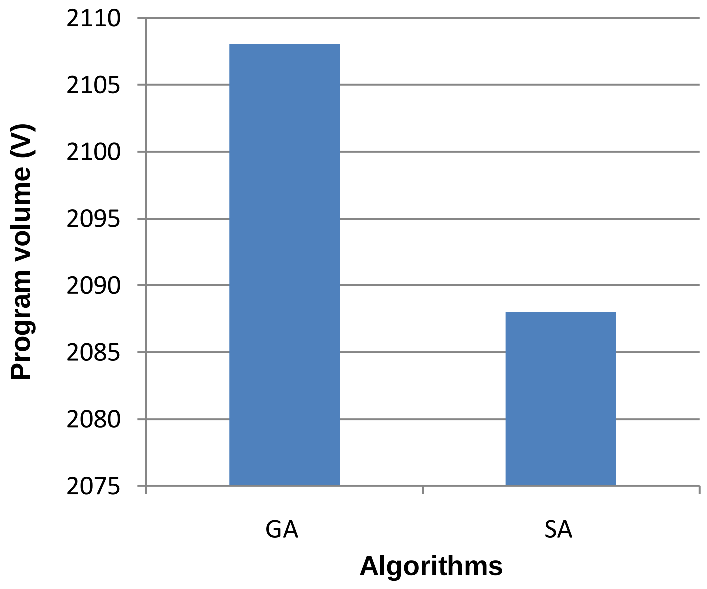

Figure 3. The program volume of both algorithms.

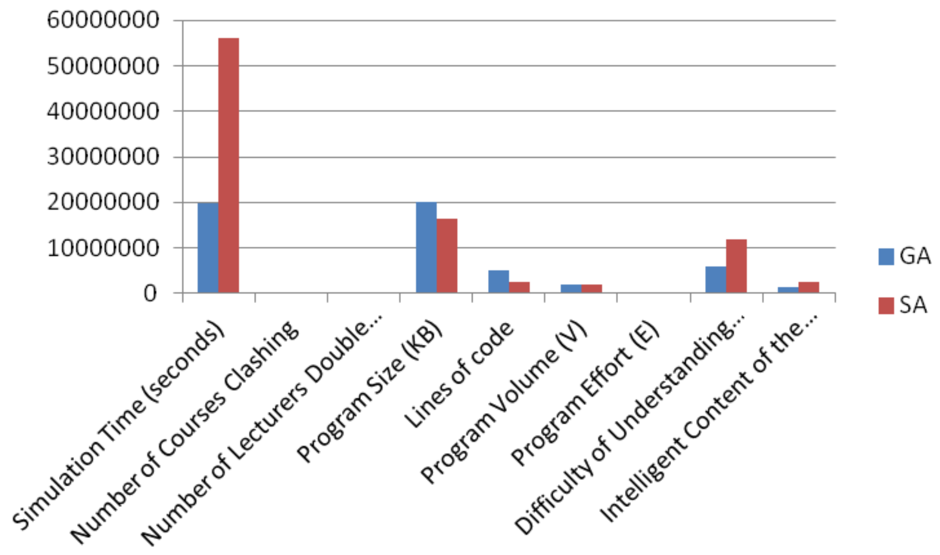

Figure 4. Comparative view of the evaluated parameters of SA and GA.

Program level / difficulty of understanding the program

This program level otherwise called difficulty of understanding a program. As presented in Table 4 and Figure 4, that GA and SA has 0.06 and 0.12 respectively as their values for the difficulty of understanding the program. The result revealed that SA is more difficult to understand than GA.

Intelligent content of the program

The Intelligent Content of the Program is the quantitative representation of how logically reasonable and experienced the program writer is. Table 4 and Figure 4 show the intelligent content of the program for GA and SA to be 131.75 and 256.25 respectively.

In view of the aforementioned, the two considered algorithms produced feasible university examination timetable with SA using more simulation time, higher difficulty of understanding the program and higher intelligent content than GA. On the other hand, GA code occupied more disk space, has more lines of code, higher program volume with more program effort than SA.

Conclusively, the results generated indicates a very high consumption of computing resources by genetic algorithm but with high optimality while simulated annealing results showed that though the consumption of computing resources is reduced yet the two algorithms still consume considerable computing resources. This paper therefore proposes the development of a hybrid algorithm of both GA and SA in solving examination timetabling problem. Such hybridized algorithm should attempt to reduce the weaknesses of genetic algorithm and simulated annealing and combine their strength to solve the problem.

REFERENCES

Abdennadher S, Marte M (2000).University course timetabling using constraint handling rules. J. Appl. Artif. Intell., 14(4): 311-326.

Abramson D (1991). Constructing school timetables using simulated annealing.sequentialand parallel algorithms. Manage. Sci., 37(1): 98-113.

Abramson D, Abela J (1992). A parallel genetic algorithm for solving the school timetabling problem.In Proceedings of 15 Australian Comput. Sci. Conf. Hobart, pp. 1-11.

Al-Attar A (1994). A hybrid GA-heuristic search strategy. A white paper, AI Expert. USA.

Andrea Schaerf (1999). A survey of automated timetabling, Artif. Intell. Rev. Kluwer Acad. Publ., 13(2): 87-127.

Burke EK, Elliman DG, Weare RF (1994). A University Timetabling System based on Graph Colouring and Constraint Manipulation. J. Res. Comput. Educ., 27: 1-18.

Burke EK, Ross P (1996). Practice and Theory of Automated Timetabling. Volume 1153 of Lecture Notes in Computer Science. Springer-Verlag. Berlin. Heidelberg.

Colorni A, Dorigo M, Maniezzo V, Trubian M (1994). Ant system for job-shop scheduling.Belgium J. Oper. Res. Stat. Comput. Sci., 34(1): 39-53.

De Werra D (1985). An introduction to Timetabling. Eur. J. Oper. Res., 19: 151-162.

Di Gaspero L, McCollum B, Schaerf A (2007). Curriculum-based course timetabling track. The second international timetabling competition (ITC-2007). Proceedings of the 14th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion, Rome, Italy.

Elmohamed S, Coddington P, Fox G (1998). A Comparison of Annealing Techniques for Academic Course Scheduling. Practice and Theory of Automated Timetabling II. Springer-Verlag, 1408: 92-112.

Keshav P, Dahal KCT, Peter IC (2007). Evolutionary Scheduling. Springer. Berlin Heidelberg Germany. p. 49.

Neufeld GA, Tartar J (1974). Graph Coloring Conditions for the Existence of Solutions to the Timetable Problem. Commun. ACM, 17(8): 450-453.

Olabiyisi SO, Akanmu TA, Oyeleye CA, Sobayo OD, Adelana JB (2007). Complexity Analysis of A New Edge-Adaptive Zooming Algorithm For Digital Images. J. Res. Phys. Sci., 3(2): 67-71.

Olabiyisi SO, Omidiora EO, Oyeleye CA (2005). Asymptotic Time Complexity Analysis For Two-Dimensional Object Inspection Using String matching. Sci. Focus, Nigeria, 10(3): 83-87.

Omidiora EO, Olabiyisi SO, Arulogun OT, Oyeleye CA, Adegbola AA (2009). A Prototype of An Access Control System For a Computer Laboratory Scheduling. AICTTRA 2009 Proceedings, Obafemi Awolowo University. Ile-Ife. Nigeria, pp. 114-120.

Piola R (1994). Evolutionary solutions to a highly constrained combinatorial problem. Proceedings of IEEE Conference on Evolutionary Computation, First World Congress Comput. Intell. Orlando. Florida, 1: 446-445.

Wren A (1996). Scheduling, timetabling and rostering - A special relationship?. In Practice and Theory of Automated Timetabling. 1153 of Lecture Notes in Computer Science. Springer-Verlag. Berlin. Heidelberg. pp. 46-75.

Zahra NajiAzimi (2005). Hybrid heuristics for Examination Timetabling problem. Appl. Math. Comput., 163(15): 705-733.