Performance Evaluation of Genetic Algorithm and Simulated Annealing in solving Kirkman Schoolgirl Problem

Christopher A. Oyelere 1, Victoria O. Dayo-Ajayi 2, Emmanuel Abiodun 3 and Alabi O. Bello 4

1Department of Information Systems, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
2Department of Computer Science, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
3Department of Computer Science, Kwara State Polytechnic, Ilorin, Nigeria
4Department of Mathematical and Physical Sciences, Afe Babalola University, Ado-Ekiti, Nigeria

Keywords: Genetic Algorithm, Simulated Annealing, Kirkman Schoolgirl, software complexity and simulation runtime

Abstract: This paper provides performance evaluation of Genetic Algorithm and Simulated Annealing in view of their software complexity and simulation runtime. Kirkman Schoolgirl is about arranging fifteen schoolgirls into five triplets in a week with a distinct constraint of no two schoolgirl must walk together in a week. The developed model was simulated using MATLAB version R2015a. The performance evaluation of both Genetic Algorithm (GA) and Simulated Annealing (SA) was carried out in terms of program size, program volume, program effort and the intelligent content of the program. The results obtained show that the runtime for GA and SA are 11.23sec and 6.20sec respectively. The program size for GA and SA are 2.01kb and 2.21kb, respectively. The lines of code for GA and SA are 324 and 404, respectively. The program volume for GA and SA are 1121.58 and 3127.92, respectively. The program effort for GA and SA are 135021.70 and 30633.26, respectively, while the intelligent content of the program for GA and SA are 72.461 and 41.06, respectively. Both algorithms are good solvers, however it can be concluded that Genetic Algorithm outperformed Simulated Annealing in most of the evaluated parameters.

1 INTRODUCTION

The Kirkman's schoolgirl problem (KSP) is a very hard combinatorial optimization problem. The problem is to arrange the fifteen schoolgirls into five triplets for seven days in a week, so that no girl will walk with any of her school-fellows in any triplet more than once. In most cases, this problem can be generalized for n girls (where n is divisible by three) into triplets to walk out for d days (where d = (n-1)/2). An instance of the KSP is then denoted by a Kirkman Steiner triple system with additional restrictive conditions which is part of combinatorial design (Bernabe, 2014). KSP belongs to a Constraint Satisfaction Problem. This kind of problem cannot be solved efficiently using traditional exact method (Yazdani, et al 2016).

In practice, due to the high complexity associated with such arrangement problems, several heuristic tools have evolved over the decade designed for solving such problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution (Desale, et al 2015). Therefore, efficient heuristics or meta-heuristics approaches can be considered as a shortcut to solve these problems in a reasonable amount of time, and to provide optimal solutions to such problem since it might require unacceptably huge time and space to discover an exact solution. GA and SA are meta-heuristics approach gaining increasing acceptance in diverse domain. The main advantage of GA and SA compared to others is their ability to clarify and find an answer to a complex problem with the aid of classifying these problems either into unconstrained (without constraints) or constrained (with constraints).

In this work, the performance evaluation of heuristics-based algorithm that is GA and SA in scheduling system was carried out. The analyses of performance were conducted in term simulation time, and Halstead software metrics. 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) (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.

2 REVIEW OF RELATED WORKS

The Kirkman schoolgirl problem (KSP) also known as the fifteen-schoolgirl problem is based on a very simple optimization problem proposed in 1850 by Rev Thomas Kirkman and has been studied for many decades by scientists. The problem is to arrange fifteen schoolgirls into five triplets for seven days in a week so that no girl will walk with any other schoolgirl in any triplet more than once. The goal is to solve this problem optimally, so as to be able to arrange these girls, in any group, that is subjective to one restrictive condition. Solving this problem with a small number of girls seems to be very easy. However, increasing the number of the girls, the problem becomes more complex and not easy to solve. This work will focus on solving the KSP using a three-dimensional array (DAY x GROUP NUMBER x GIRL NUMBER), just as a basis of the Steiner triple system.
To solve the Kirkman’s Schoolgirl Problem, a Constraint Satisfaction Problem (CSP) is required. With this in mind, as cited in Triska and Musliu (2012) normally the KSP is solved assuming the correct group sizes are within a class of isomorphic solutions as one having the following constraints:

1. Each girl walks exactly once each day.
2. Each group of girls can walk in the same group at most once.

Merta and Brandjesky (2018) worked on three-dimensional genetic algorithm for the Kirkman schoolgirl problem. The focus was on possibilities of multidimensional genetic algorithm and relevant of genetic operators which first result was not good due to non-convergence of the algorithm. Oyeleye et al (2012) worked on the performance evaluation of simulated annealing and genetic algorithm in solving examination timetabling problem. 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. The two considered algorithms produced a feasible examination timetable with simulated annealing performing better than genetic algorithm in most of the evaluated parameter.

3 RESEARCH METHODOLOGY

After representing the problem mathematically, the two algorithms were implemented using MATLAB development kit on an intel core i3 CPU with 2.00GHz and 4GB Random Access Memory with Window 8.1.

3.1 MATHEMATICAL REPRESENTATION OF THE PROBLEM

The initial approach to solving a KSP is to formulate a mathematical model of the problem. The following assumptions and notations are used in developing the model:

Let there be G schoolgirls to be scheduled and s represents each schoolgirl

Let the fixed walking number of schoolgirls be S

Let the scheduling period be for D days, d = 1, ..., D

Let each day have five moves denoted by X = (x₁, x₂, x₃, x₄, x₅)

Zₛₐₜₜₛₖₜₜₛₖₗₜₜₛₖₘₜₜₛₖₙₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜₛₖₚₜₜسائرات 1 for each schoolgirl s, day d, move x – that is, a schoolgirl s is walking abreast a day d.

The model for the problem P is stated using the parameters stated above:

The number of schoolgirls walking together must be 3

\[
P: \text{MINIMIZE } \sum_{s=1}^{G} Z_s d_s x_s = S = 3 \tag{3.1}
\]

The schoolgirls must not walk together twice

\[
\sum_{s=1}^{G} \sum_{s=1}^{S-1} Z_s x_s x_{s+1} \quad \text{for all } i = 1, 2, \ldots, n \tag{3.2}
\]

In this problem, the two constraints are considered and are further categorized into hard and soft constraints as shown in Table I.

### 3.2 THE GENETIC ALGORITHM APPROACH

This approach incorporates some methods of population initialization, crossover and mutation working with the three-dimensional chromosome as shown in Fig 1.

| Table 1. Hard Constraints HC and Soft Constraint SC |
|---------------------------------------------------|
| **HC** | **Description** |
| HC1 | The total number of schoolgirls must not be more than 15 |
| HC2 | The number of schoolgirls walking together must be 3 |
| HC3 | No two schoolgirls should walk together more than once in a week |
| HC4 | All schoolgirls must feature each day in the schedule |
| HC5 | The schedule should span through seven days |
| HC6 | Each day only 5 groups of schoolgirls should feature |
| SCI | No schoolgirl should request for a particular girl to walk with |

| Hard Constraints HC | Description |
|--------------------|-------------|
| HC1 | The total number of schoolgirls must not be more than 15 |
| HC2 | The number of schoolgirls walking together must be 3 |
| HC3 | No two schoolgirls should walk together more than once in a week |
| HC4 | All schoolgirls must feature each day in the schedule |
| HC5 | The schedule should span through seven days |
| HC6 | Each day only 5 groups of schoolgirls should feature |
| SCI | No schoolgirl should request for a particular girl to walk with |

3.3 THE SIMULATED ANNEALING APPROACH

Several aspects of the SA algorithm are problem-oriented. Design of a good annealing algorithm is nontrivial, it generally comprises of neighbourhood structure, cost function and cooling schedule.
4 RESULTS

Halstead software complexity was used to evaluate the two algorithms. The parameter measures by Halstead are program volume (V), program effort(E), program level(L), and intelligent content of the program(I). The parameters used in measuring the Halsted software complexity metrics are: No of distinct operators (n₁), No of distinct operands (n₂), Total number of operators (N₁), Total number of operands (N₂), where N = (N₁+N₂) and n = (n₁+n₂). The results obtained for measuring the complexity of GA and SA are indicated in Table 2 with the formulae for measuring the complexity metrics shown in Table 3.

Table 2. Results obtained for measuring the complexity of both algorithms.

| Parameters for Halstead Complexity | GA  | SA  |
|-----------------------------------|-----|-----|
| No of distinct operators (n₁)     | 37  | 23  |
| No of distinct operands (n₂)      | 45  | 32  |
| Total number of operators (N₁)    | 387 | 118 |
| Total number of operands (N₂)     | 105 | 76  |
| N i.e. (N₁+N₂)                    | 492 | 194 |
| n i.e. (n₁+n₂)                    | 82  | 55  |

Table 3. Formulae for measuring the complexity metrics

| Complexity metrics | Formulae |
|--------------------|----------|
| Volume (V)         | N^* log2n |
| Effort (E)         | V/L      |
| Program level (L)  | (2*n2) / (n1*N2) |
| Intelligent content of the program (I) | L*V |

The two algorithms considered in this work as shown in Table 5 produced feasible solutions because none of the constraints was violated.

Simulation Runtime: This is the time utilized by an algorithm to run a program. Table 4 shows the values of the time taken by the two Algorithms. The simulation time of GA and SA are 6.44 and 11.78 seconds respectively. The result indicated that GA used more time to solve the problem than SA because GA evaluates more solutions and therefore takes more time.

Table 4. Performance of GA and SA

| Iterations | Method | Time (sec) |
|------------|--------|------------|
| 1000       | SA     | 6.44       |
| 1000       | GA     | 11.78      |

Program size: This is the amount of disk space occupied by the program and it is measured in bits or bytes. The program sizes of GA and SA are 2.01kb and 2.21kb, respectively as shown in Table 5. This implies that SA utilizes more disk space than GA.

Lines of Code: The line of code measures are the most traditional measures used to quantify software complexity. Table 5 shows that the LOC of GA and SA as 324 and 404 respectively. This show SA code has a greater number of executable lines of code than GA.

Program Volume: The program volume V is the information contents of the program measured in bits needed to encode the program. The results as shown in Table 5 revealed that GA has V of 3127.92 which occupies more memory space in terms of volume than SA with V of 1121.58.

Program Effort (E): The program effort is the number of mental discriminations required to implement the program and also the effort required to read and understand the program. The program effort (E) of GA and SA are 135021.70 and 30633.26, respectively as shown in Table 5. This indicates that the program effort (E) of GA is higher than that of SA.

Program Level (L): The program level is the measure of a person's ability to understand a program. Table 5 revealed that GA and SA have 0.02 and 0.04, respectively. The result indicates that SA is more difficult to understand than GA and takes less effort to develop.

Intelligent Content of the Program (I): The Intelligent Content of the Program is used to determine the amount of intelligence presented or stated in a program. Table 5 shows the intelligent content of the program for GA and SA to be 72.461 and 41.06, respectively.
Table 5. Results obtained during and after the execution of both algorithms.

| Parameters                        | GA  | SA  |
|-----------------------------------|-----|-----|
| Number of Moves Violated          | 0   | 0   |
| Lines of Code                     | 342 | 404 |
| Program Size (KB)                 | 2.01| 2.21|
| Program Volume (V)                | 3127.92| 1121.58|
| Program Effort (E)                | 135021.70| 30633.26|
| Intelligient Content of the Program (I) | 72.46| 41.06|

5 CONCLUSION
The two algorithms considered were able to return a feasible solution satisfying the hard constraints with GA using more execution time, higher program volume, more program effort and higher intelligent content than SA. Likewise, SA codes occupy more disk space, higher difficulty of understanding the program and has more lines of code than GA. In conclusion, both algorithms are very good solvers and were able to provide optimal solutions with the appropriate set of parameters.

However, the outcome of the experiment shows that Simulated Annealing finds a solution for the problem in less time within a limited number of iterations with a 2.13GHz processor. Alternatively, Genetic Algorithm can provide quality solutions when a large population is set, thereby greatly increases its run time. However, one of the approaches that can be used to run these algorithms for better solutions to be generated is parallelizing both algorithms. Also, other metrics can be integrated with Halstead complexity so as to provide an aggregated measure of complexity from both metrics.

REFERENCES
Abed, M. H., and Alicia, Y. (2013). Hybridizing Genetic Algorithm and Record-to-Record Travel Algorithm for Solving Uncapacitated Examination Timetabling Problem. *Electronic Journal of Computer Science and Information Technology (EJCIST)*, 4(1).

Adewole, A.P., Otubamowo, K., Egunjobi, T. O., and Ng, K. M. (2012). A comparative study of simulated annealing and genetic algorithm for solving the travelling salesman problem. *International Journal of Applied Information Systems (IIAIS)*, 4(4), 6-12.

Akinwale, O.C., Olutunde, O.S., Oluusayo, O. E., and Temitayo, F. (2014). Hybrid Metaheuristic Simulated Annealing and Genetic Algorithm for Solving Examination Timetabling Problem. *International Journal of Computer Science and Engineering (IJCSE)*, India, 3(5), 7-22.

Bhatia, A., Singh, A., and Goyal, R. (2015). A Hybrid Autonomic Computing-Based Approach to Distributed Constraint Satisfaction Problems. *Computers*, 4(1), 2-23.

Blum, C., Puchinger, J., Raidl, G. R., and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6), 4135-4151.

Cao, P., Fan, Z., Gao, R. X., and Tang, J. (2017). Solving Configuration Optimization Problem with Multiple Hard Constraints: An Enhanced Multi-Objective Simulated Annealing Approach. *arXiv preprint arXiv:1706.03141*.

Froncek, D., and Kubesa, M. (2011). Rectangular table negotiation problem revisited. *Open Mathematics*, 9(5), 1114-1120.

Hanafy, T. O., Zaini, H., Shoush, K. A., and Aly, A. A. (2014). Recent trends in soft computing techniques for solving real time engineering problems. *International Journal of Control, Automation and Systems*, 3(1), 27-33.

Nopiah, Z. M., Khairil, M. L., Abdullah, S., Baharin, M. N., and Ariffin, A. (2010, February). Time complexity analysis of the genetic algorithm clustering method. In *Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation, ISPRA* (pp. 171-176).

Repe, L. R., Fanany, M. I., & Aymurhythy, A. M. (2015). Simulated annealing algorithm for deep learning. *Procedia Computer Science*, 72, 137-144.

Yazdani, M., Khalili, S. M., and Jolai, F. (2016). A parallel machine scheduling problem with two-agent and tool change activities: an efficient hybrid metaheuristic algorithm. *International Journal of Computer Integrated Manufacturing*, 29(10), 1075-1088.