

PERFORMANCE EVALUATION OF SOME SELECTED METAHEURISTIC TECHNIQUES FOR INTELLIGENT PATH PLANNING OF MOBILE AGENTS

QUADRI BOLAJI ABDULKAREEM,

Kwara State Polytechnic, Ilorin, Nigeria

STEPHEN OLATUNDE OLABIYISI AND FOLOWOSELE

ADEBOYE OLAIDE

Department of Computer Science,

Ladoke Akintola University of Technology, Ogbomosho, Nigeria

Email address:

quadribolajiabdulkareem@gmail.com soolabiyisi@lautech.edu.ng boye4christ@yahoo.com

DOI: 10.26821/IJSHRE.9.2.2021.9206

ABSTRACT

Mobile Agents are objects that migrate through the nodes of heterogeneous networks to perform intelligent path finding tasks, which leverages on metaheuristic techniques to obtain path patterns that are best suited for any given path planning problem. Therefore, this research evaluated the performance of some selected metaheuristic techniques Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA) and Genetic Algorithm (GA) for path planning. A prototype of Mobile Agents was introduced into a 2 Dimensional complex workspace of an ad-hoc mobile network consisting of 16 nodes, with a 4 by 4 coordinate dimension for path finding. The performance of the metaheuristic techniques was evaluated using Computational Time (CT), Cost Value (CV), Program Effort (PE) and Program Size (PS). The evaluation results obtained for the metaheuristic techniques indicated that CT for PSO, ACO, SA and GA are 32.40, 28.77, 40.20 and 38.23 seconds, respectively; corresponding values for CV are 4.10, 17.04, 39.10 and 5.4, respectively. Also, the corresponding value for PE are 24753.99, 58228.95, 13391.42 and 103016.47, respectively; while the corresponding value for PS are 9269, 7541, 7630 and 8639 kb, respectively. This work reveals that ACO performs better than other algorithms in terms of acceleration and time of convergence.

Keywords: Computational Time (CT), Cost Value (CV), Program Effort (PE) and Program Size (PS), Metaheuristic techniques.

1. INTRODUCTION

Rahul (2018) defines mobile agents as programs that can maneuver through a network by themselves. They are able to migrate from host to host and can interact with other resources and agents on each network. The mobile agent can suspend its execution at an arbitrary point, jump to another machine, and resume execution there. Each agent is typically composed of the agent code, the agent execution thread along with an execution stack and the agent data part, which corresponds to the values of the agent's global variables (1).

The main heuristic approaches employed in Mobile Agent Path Planning are Bug Algorithm (BA), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Gravitational Search Algorithm (GSA), Simulated Annealing (SA) and Tabu Search (TS) (2). In case of static environment, the location of the obstacle is fixed and does not transform with time (3). However, in case of dynamic environment, the position of the obstacle changes with time (4). In case of robot navigation, a mobile agent reaches the target destination from source station, avoiding collision with obstacles and upon iterations gives an optimal path without any human involvement.

There exists a number of metaheuristic path planning technique such as Particle Swarm Optimization (PSO), Ant-Colony Optimization (ACO), Simulated Annealing (SA) and Genetic Algorithm (GA) for mobile agent. However, the most suitable of these four (4) techniques relative to Shortest path and reduced search time of the mobile agent environment is still undetermined due to varying limitations of each of the techniques.

Enhanced execution of SA, GA, ACO and PSO regarding measurements like Shortest path and lessened pursuit time are looked for after capacities of every one of the systems. Thus, there is a need to recognize which of these procedures is most appropriate for path planning of mobile agent in unique condition with respect to these abilities. In this way, this examination is required to assess nearly the execution of every one of these procedures in mobile agent condition.

The aim of this research work is to comparatively evaluate the performance of some selected metaheuristic techniques for intelligent path planning of mobile agents. The Objectives are to:

- i. design Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA) and Ant Colony Optimization (ACO) path planning technique of mobile agents in a 2-D workspace,
- ii. implement the designed techniques in (i),
- iii. evaluate the performance of the techniques using computer execution time and Halstead complexity metrics.

1.1 Justification of the Study

Wang and Zeng, (2014) (5) and Subramanian, Sudhagar and Rajeswari (2014) (6) have all worked on a number of metaheuristic path planning technique such as Particle Swarm Optimization (PSO), Ant-Colony Optimization (ACO), Simulated Annealing (SA) and Genetic Algorithm (GA) for mobile agent. However, of all the most recent literatures the most suitable of these four (4) techniques relative to shortest path and reduced search time of the mobile agent environment is still undetermined due to varying limitations of each of the techniques.

1.2 Related Works

Hachour (2012) proposed algorithm for path planning of autonomous mobile robot in an unknown environment. An autonomous robot reaches the target station avoiding collision with static obstacles. The robot travels within the environment sensing and avoiding obstacles that come across its way to the target station. Optimal path that would minimize cost, time, energy was planned when the mission was executed. The proposed algorithm was implemented in Borland C++ since it is suitable for graphic problems, afterwards tested with Visual Basic and DELPHI language. The simulation is an approach to the real expected result. This navigation approach has made the robot able to achieve to avoid obstacles, perception, deciding and to attend the target.

Milica and Miljković (2013) presented a methodology for the development of software application for integration of process planning, scheduling, and the mobile robot navigation in manufacturing environment. Proposed methodology is based on the Russian Theory of Inventive Problem Solving (TRIZ) and multiagent system (MAS). Contradiction matrix and inventive principles are proved as effective TRIZ tool to solve contradictions during conceptual phase of software development. The proposed MAS architecture consists of six intelligent agents: job agent, machine agent, optimization agent, path planning agent, machine learning agent and mobile robot agent. All agents work together to perform process plans optimization, schedule plans optimization, optimal path that mobile robot follows and classification of objects in a manufacturing environment. Experimental results show that developed software can be used for proposed integration in order to improve performance of intelligent manufacturing systems.

Subramanian, Sudhagar and Rajeswari (2014) identified an optimal path for a mobile robot agent which was considered to be one of the key challenges for researchers in the field of robotics. When the mobile robots agent decides its action, it is necessary to plan its movement more precisely and optimally. The work presents the use of search algorithm in identifying the optimal path along with its navigation controlling mechanism of a mobile robot agent navigational systems. The mobile robot agents are modelled to work independently through its intelligence without any human intervention. The movement details of a robot agent participating in dynamically changing environment using Breadth First Search (BFS) algorithm. The system evaluation was validated using Graphical User Interface (GUI) based test bed for robots called Robosim and the efficiency of the system was measured via simulation results through a defined complex arena. Simulation results proven that the applying the BFS algorithm in a unknown environment explores much faster than heuristic based other path planning algorithms.

2. METHODOLOGY

In this research, performance evaluation of four (4) metaheuristic techniques (Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO)) for intelligent path planning of mobile agents was conducted. Figure 1 shows the flowchart of the evaluation procedure. The flowchart shows the sequence involved in solving the path planning problems. The first step involves the initialization of mobile agent migration parameters. These initialized parameters become utilized by each of the metaheuristic techniques (ACO, PSO, SA, GA), after which optimization is performed based on the

selected optimization strategy for mobile agent path planning solution. Evaluation of the performance of the selected metaheuristic technique is done using Halstead measures, path length, speed of operation.

Problem Definition / Environment

A mobile path design is required between two indicated areas; a beginning and objective point. The path ought to have no crash and satisfies certain optimization criteria (briefest separation) utilizing any of previously mentioned Meta-Heuristic methods. The goal is to develop a briefest path from the source station S_t , to the objective station (objective point) D_t , which maintains a strategic distance from each obstruction in the guide in the ideal time; (a path which not touch any impediment). Figure 2 demonstrates the conduct of the mobile agent in the issue space. Path planning agent has an undertaking to choose paths with the allotted errand before development initiates. For the applications in the known static condition, path planning is generally explained in jumpers' means: (1) constructing a diagram to speak to the geometric structure of nature and (2) play out a chart hunt to discover path between the beginning and objective focuses in view of specific criteria.

Improved performance of PSO, ACO, SA and GA, in terms of metrics like Shortest path and reduced search time are sought-after capabilities of each of the techniques. Hence, there is a need to identify which of these techniques is most suitable for path planning of mobile agent in dynamic environment relative to these capabilities.

Problem Work Space

This examination centers on finding a snag free path for a mobile agent working in a 2-D complex workspace of a mobile system. A worldwide organize framework x-y is built up to speak to the workspace, where S_t and D_t signify the begin area and the goal area of the mobile agent individually. The x - pivot of the worldwide organize framework matches with line S_t - D_t and the y - hub of the worldwide arrange framework is opposite to line S_t - D_t .

In the global coordinate system, line S_t - D_t is equally divided into $n + 1$ subsections by n points, where n is a predefined parameter. After drawing n vertical lines l_1, l_2, \dots, l_n is indicated. Therefore, a point-to-point path for line S_t - D_t with respect to the lines in x and y is given as $Path = [S_t, d_1, d_2, \dots, d_n, D_t]$. All obstacles in the workspace are assumed to be static and are represented by various shape.

In the 2-D outline agent workspace, the begin point, the objective point, and the zones that contain impediments and free spaces are spoken to. In the free space, a gathering of associated waypoints are orchestrated in random areas. Each waypoint speaks to an area in the earth, and it is described by an identifier. A (X,Y) facilitates and an arrangement of neighboring waypoints as portrayed in Figure 4.2. The key thought of the model comprises in utilizing the waypoints as:

- ✓ Signposts to find the mobile agent
- ✓ Landmarks to direct the mobile agent towards the coveted goal.

The mobile agent must pick the best arrangement of waypoints to follow so as to locate the best path, which will be encoded as a succession of waypoints personality; for example, 0-3-9-12-17-16-22-25 is a path with begin position 0 and objective position 25. The mobile agent is relied upon to move between nearby way focuses that are associated. In the event that the two positions are not associated this implies there is a hindrance between them.

The Mobile Agent Path Planning Problem Formulation

Hunting down the best path is much of the time related with finding the most limited and safest path. Path length and path safety of the mobile agent are regarded as an important execution criteria of the path planning issue in this work.

Let d_0 and d_{n+1} denote the start location and the destination location of a mobile agent, respectively. The planning problem can be mathematically formulated as (Wang and Zeng, 2014):

$$\left\{ \begin{array}{l} \text{finding: } Pth = [St = d_0, d_1, d_2, \dots, d_n, Dt = d_{n+1}] \\ \\ \text{Minimize: } Q_{cost} = w_1 \cdot Q_L + w_2 \cdot \frac{1}{Q_S} \\ \\ St, d_k, d_{k+1} \in \text{semi-free workspace}, \quad 0 \leq k \leq n \end{array} \right.$$

3.1

Where Q_L and Q_S denote path length and path safety, respectively, and w_1 and w_2 are two weighting parameters indicating the relative importance of Q_L and Q_S ($w_1 = w_2 = 1$). To generate a point-to-point path, Q_L is calculated as: (Milika et al., 2013):

$$Q_L = \sum_{k=0}^n Eud(d_k, d_{k+1})$$

3.2

Where $Eud(d_k, d_{k+1})$ denotes the Euclidean distance between waypoint d_k and waypoint d_{k+1} .

Q_S is the summation of the minimum distance of each path segment from the nearest obstacle along the path, which is calculated as: (McGrath, 2000):

$$Q_S = \sum_{k=0}^n \min_{1 \leq l \leq N_{ob}} \{ \min Eud(d_k, d_{k+1}, Ob_l) \}$$

3.3

Where N_{ob} denotes the number of obstacles. $\min Eud(d_k, d_{k+1}, Ob_l)$ represents the minimum distance between path segment d_k, d_{k+1} and obstacle Ob_l .

Path Encoding Scheme

From figure 3.3, a path is constructed by a set of waypoints d_1, d_2, \dots, d_n while a set of lines l_1, l_2, \dots, l_n determines the x-axis values of these points. Since the set of lines l_1, l_2, \dots, l_n is given beforehand during the construction of the workspace. Values of d_1, d_2, \dots, d_n lie only with the y-axis values decided by l_1, l_2, \dots, l_n . This y-axis values, denoted as $(y_{d_1}, y_{d_2}, \dots, y_{d_n})$, are applied in order to encode the path. The following saturation strategy is used to modify y_{d_i} ($i = 1, 2, \dots, n$) when y_{d_i} is outside of the workspace as follows (Yong et al., 2013):

$$y_{d_i} = \begin{cases} \frac{width}{2}, & \text{if } y_{d_i} > \frac{width}{2} \\ -\frac{width}{2}, & \text{if } y_{d_i} < -\frac{width}{2} \\ y_{d_i}, & \text{otherwise} \end{cases}$$

3.4

Where *width* represents the width of the workspace as shown in Figure 3.3

Evaluation of path and Handling requirement

Concerning condition 3.1, clearly the path planning relocation issue is a compelled optimization issue. Along these lines, handling imperative must be tended to for simple and productive arrangement. The imperative of the path planning mobile agent relocation issue is to produce a hindrance free path. It is sensible to assess the limitation infringement level of every candidate path by tallying its crash times with impediments. Given N_{ob} obstacles, the total constraint violation degree of candidate path Cp_1 is calculated as:

$$CV_{c_1} = \frac{1}{N_{ob}} \sum_{l=1}^{N_{ob}} V_{Cp_{1l}} \quad 3.5$$

$$V_{c_{1l}} = \begin{cases} 1 & \text{if } Cp_1 \text{ collides with Obstacle } i \\ 0 & \text{otherwise} \end{cases} \quad 3.6$$

Subsequent to figuring the imperative infringement degree and the wellness estimation of every candidate path, the achievability based administer is utilized to assess and select the world class path between any two candidate paths. The possibility based lead is portrayed as (Chao-Liet. al, 2011): for any two paths with the same imperative infringement degree, the path with better wellness is favored; for any two paths with various requirement infringement degrees, the path with a littler imperative infringement degree is favored.

Nature of Dataset Used

The Dataset used was generated by system when the parameter in Table 2 has been set in MATLAB environment as base measures.

4. IMPLEMENTATION AND RESULT

This examination work was done in Matrix Laboratory (MATLAB). An intelligent Graphic User Interface (GUI) was produced. The framework details are Windows 7 Ultimate 32-bit working framework, Intel® Pentium® CPU B960@2.20GHZ Central Processing Unit, 4GB Random Access Memory and 500GB hard plate drive.

4.1 Performance Evaluation of the Metaheuristic Techniques**Table 1: Formulae for Measuring the Halstead Complexity Metrics**

Complexity Metrics	Formulae
Program Length (N)	$N_1 + N_2$
Calculated Program Length (N_h)	$n_1 \log_2 n_1 + n_2 \log_2 n_2$
Program Vocabulary (n)	$n_1 + n_2$
Program Volume (V)	$N \log_2 n$
Potential Volume (v^*)	$(n_1^* + n_2^*) \log_2 (n_1^* + n_2^*)$
Program Level (L)	V^*/V or $(2n_2)/(n_1 N_2)$
Programming Difficulty	$(n_1 \times N_2)/2n_2$ or $1/L$
Programming Effort (E)	V/L or DV
Programming Time (T)	E/B or $E/18$
Language Level (LL)	LV^* or $L2V$
Intelligent content of the program (I)	LV or V/D

Source: Milica and Miljković (2013)

In order to evaluate the performance of the four meta-heuristics solution approach, software complexity evaluation of the four algorithms was considered. Specifically, the following quality measures or parameters were used:

- i. Halstead Software Complexity metrics
- ii. Line of Code
- iii. Program Size
- iv. Execution/Simulation Time (Speed of Operation), and

Halstead measure calculates Program Volume (V), Program Effort (E), Programming Difficulty (D) and Intelligent Content of the program (I). The formulae for these metrics have already been presented in Table 1.

The results of the implementation after the application of the four heuristic algorithms are shown in Table 2. The table shows the measured parameters and their various values used in calculating the software complexity of the four algorithms. As presented on the table, n_1 is the number of distinct operators found in the program, n_2 is the number of distinct operands, N_1 is the total number of operators, N_2 is the total number of operands, N is the addition of N_1 and N_2 and n is the addition of n_1 and n_2 . It was revealed from Table 4.1 that all the four meta-heuristic algorithms used the same distinct operator ($n_1 = 14$). GA used the highest number of distinct operands ($n_2 = 83$) while SA used the least amount of distinct operands ($n_2 = 44$). Therefore, GA among others have the high complexity while SA have the lowest complexity with respect to the number of operators and operands used. Figure 4.1 depict a bar chart showing the number of operands and operator with respect to the meta-heuristic algorithms.

Since the four considered algorithms produced feasible solutions. This is clearly evident in the summary of data obtained and presented in Table 4.2.

Results Obtained using the Halstead Measure

The Halstead measure includes Program Volume (PV), Program Effort (PE), Programming Difficulty (PD) and Intelligent Content of the program (ICP).

Table 2: Parameters for Measuring the Computational Complexity

Base Measures	PSO	ACO	SA	GA
No. of Distinct Operators (n_1)	14.00	14.00	14.00	14.00
No. of Distinct Operands (n_2)	54.00	68.00	44.00	83.00
Total Number of Operators (N_1)	200.00	305.00	176.00	333.00
Total Number of Operands (N_2)	942.00	1243.00	778.00	1481.00
$N (N_1+N_2)$	1142.00	1548.00	954.00	1814.00
$n (n_1+n_2)$	68.00	82.00	58.00	97.00

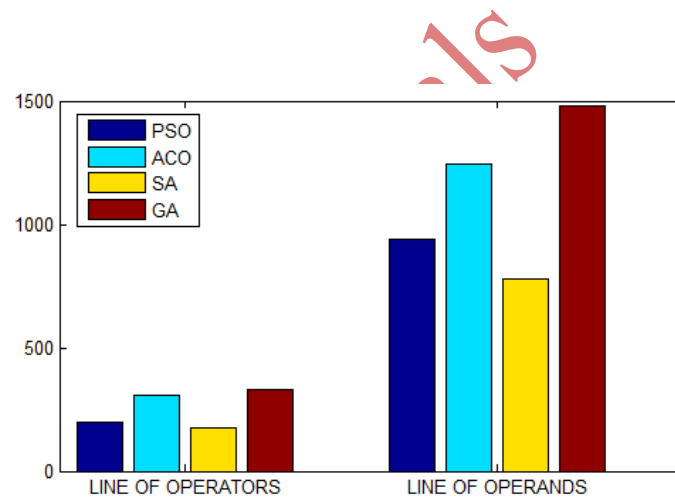


Fig. 1: Bar chart showing line of operators and line of operands for PSO, ACO, SA and GA

The result obtained from Table 3 in terms of Program Volume (V) revealed that PSO, ACO, SA and GA have 6951.88, 9841.49, 5588.51 and 11972.24 respectively. Similarly, in terms of Program Effort (PE), PSO, ACO, SA and GA have values of 24753.99, 58228.95, 13391.42 and 103016.47 respectively. The Programming Difficulty (PD) for PSO, ACO, SA and GA read 1.95, 1.66, 2.33 and 1.39 respectively. In terms of Intelligent Content of the program (I), PSO, ACO, SA and GA were found to have values of 35607.00, 591668.00, 239624.00 and 860461.00 respectively. With respect to the Halsted measure, results in Table 3 revealed that GA has the highest Program Volume (PV= 11972.24), Program Effort (PE=103016.47), Intelligent Content of the program (ICP= 860461.00) but the least programming difficulty (PD = 1.39); this further confirmed that GA high complexity of GA. On the other hand, SA has lowest Program Volume (PV= 5588.51), Program Effort (PE=13391.42), Intelligent Content of the program (ICP= 239624.00) but the highest

Programming Difficulty (PD = 2.33); this further confirmed the high complexity of GA. Similarly, this further confirms that SA among others has the least complexity. Figure 1,2,3,4 depicted bar charts using the Halsted measure against the techniques.

Results Obtained in term of Line of Code and Program Size

From table 3; PSO, ACO, SA and GA have 170, 216, 218 and 188 lines of code respectively. Here, SA has more lines of code compare to others and PSO has the least lines of code. Similarly, the program size in KB for PSO, ACO, SA and GA are 9269, 7541, 7630 and 8639 respectively. This revealed that ACO among others has the best space complexity followed by SA, GA while PSO has the worst space complexity.

Table 3: Summary of Result Obtained

Parameters	PSO	ACO	SA	GA
Lines of Code	170.00	216.00	218.00	188.00
Program Size (KB)	9269.00	7541.00	7630.00	8639.00
Program Volume (PV)	6951.88	9841.49	5588.51	11972.24
Program Effort (PE)	24753.99	58228.95	13391.42	103016.47
Intelligent Content of the Program (ICP)	356076.00	591668.00	239624.00	860461.00
Difficulty of Understanding the Program (PD)	1.95	1.66	2.33	1.39
Cost Value	22.62	5.80	4.60	0.52
Simulation Time (seconds)	295.83	8.69	22.73	9.22
Path Length	26.39	7.00	43.00	27.09

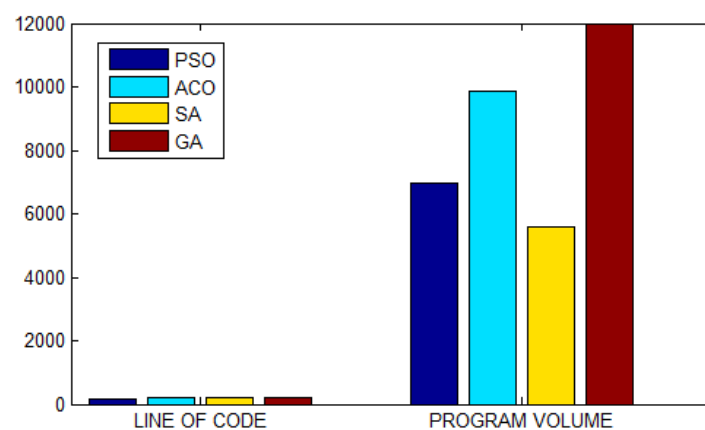


Fig. 2: Bar chart showing line of code and program volume for PSO, ACO, SA and GA

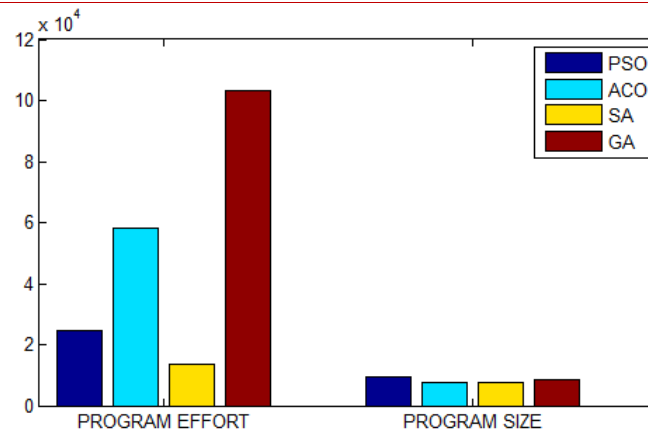


Fig. 3: Bar Chart showing program effort and program size for PSO, ACO, SA and GA

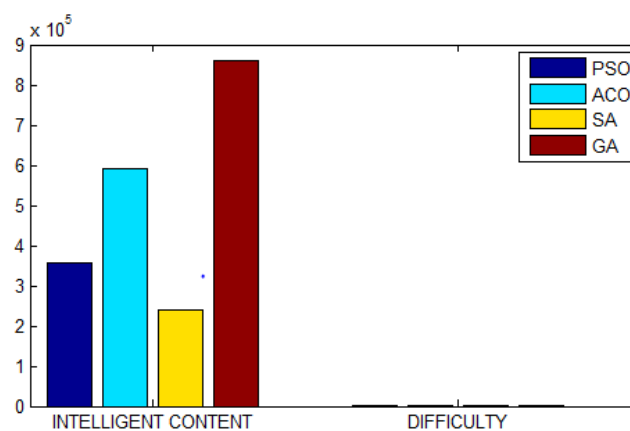


Fig. 4: Bar Chart showing intelligent content and difficulty for PSO, ACO, SA and GA

Results Obtained in term of Path length, Cost Value and Simulation time

A path planning problem majorly solves the problem of path length as well as the duration at which the optimization is done. A good path planning migration techniques will perform optimization at the shortest possible time and also have the shortest distance. PSO, ACO, SA and GA have path lengths of 26.39, 7.00, 43.00 and 27.09 respectively. Therefore, ACO achieved the best shortest distance between the source and the destination of the mobile agent. Similarly, PSO, ACO, SA and GA have simulation time of 295.83, 8.69, 22.73 and 9.22 seconds. ACO has the least simulation time. Therefore, ACO converged faster than other techniques. With respect to both path length and simulation time ACO produced a better path planning migration strategy for mobile agent in mobile network than PSO, SA and GA.

Furthermore, in terms of cost value PSO, ACO, SA and GA have 22.62, 5.80, 4.60 and 0.52. GA has the least cost value. Figure 5 shows Bar Chart which depicts simulation time, cost value and path length for PSO, ACO, SA and GA. Figure 6 showed Graph showing path length of PSO, ACO, SA and GA against iterations. The graph revealed that the path length increased with more iterations. Figure 7 showed a Graph showing cost fitness value of PSO, ACO, SA and GA against iterations while Figure 8 showed a Graph showing processing time of PSO, ACO, SA and GA against iterations.

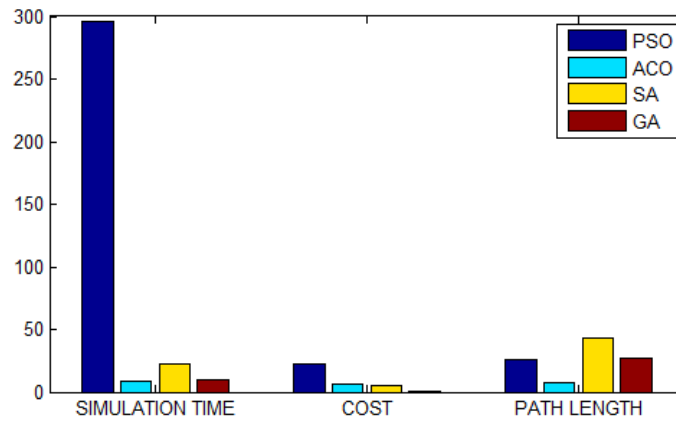


Fig. 5: Bar Chart showing simulation time, cost value and path length for PSO, ACO, SA and GA

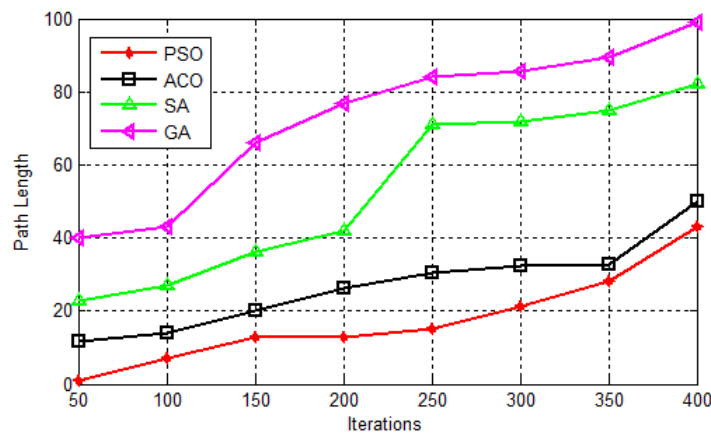


Fig. 6: Graph showing path length of PSO, ACO, SA and GA against iterations

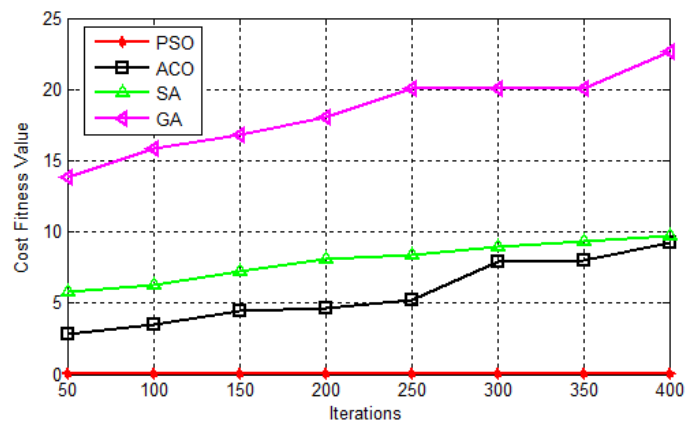


Fig. 7: Graph showing cost fitness value of PSO, ACO, SA and GA against iterations

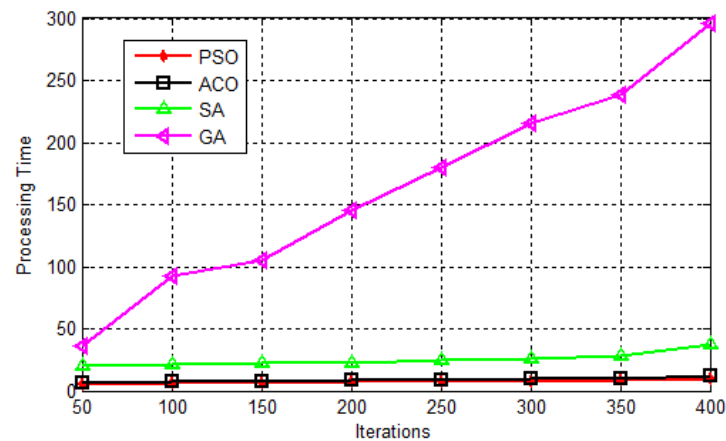


Fig. 8: Graph showing processing time of PSO, ACO, SA and GA against iterations

In this research work, four different metaheuristic approach were adopted and used to solve an intelligent path planning problem for mobile agent. The results were analyzed and compared using Halsted Software Complexity metrics. This research work has been able to develop and simulate a suitable technique for generating the shortest path for intelligent path planning for mobile agents.

Software complexity metrics were used to evaluate the performance of the four metaheuristic based on data sets used. ACO algorithm in this study was discovered to be the most efficient algorithm among the other three algorithms for path planning migration problem because it gets out of local minima and converged in less time. The analysis of the implementation complexity of the four algorithms show that PSO and GA metaheuristics yield better solutions in terms of their cost value when compared with ACO and SA. In general, both algorithms were able to generate good solutions to the problem instances.

5. CONCLUSION AND RECOMMENDATION

These research has shown that ACO algorithm, performs better than other metaheuristic methods, namely the GA, PSO and SA algorithms in terms of time and acceleration of convergence. This is because it gives better solution in both simulation mode using MATLAB and also requires less time to execute by locating the shortest path. The results of these simulations and experiments are very encouraging. ACO has good contribution to the path planning of mobile agent in unknown environment having static obstacles in different positions. Based on the result obtained in this work, improvement is sought on Ant Colony Optimization (ACO) in terms of cost value and Intelligent Content of the program, on PSO in terms of Intelligent Content of the program and computation time. The results of this research justified by showing the best performance out of the four selected heuristics algorithm (PSO, SA, GA and ACO) for solving path planning problem of mobile agents. Also it ascertains the heuristics that perform efficiently for a formulated intelligent path planning model in mobile agent by using Halstead Software Complexity metrics, path length, speed of operation. The techniques are evaluated on a vehicle routing problem, travelling salesman and on a high dimensional problem space with 500 nodes.

ACKNOWLEDGEMENTS

All forms of praises and commendations and attributed to Almighty God for His protection and guidance throughout the duration of carrying out this research. I acknowledge and appreciate the hard work of my supervisor for his support throughout my research.

REFERENCES

1. Felner (2014) Introduction to Evolutionary Computing. Springer.
2. Dorigo, M., and Stutzle, (2014). Ant colony optimization: a new meta-heuristic. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 2). IEEE.
3. Geem, Z. W., Joong H. K., & Loganathan, G. V. (2011). A new heuristic optimization algorithm: harmony search. *Simulation* 76.2 (2011): 60-68.
4. Hachour (2012). Genetic Algorithms in optimization, search and machine learning. Addison Wesley, New York.
5. Milica and Miljković (2013) Dynamic vehicle path planning using an enhanced simulated annealing approach for supply chains. *International Journal of Enterprise Network Management*, 5(2), 197-218.
6. Rahul S C. (2018) "Mobile Agent Programming Paradigm and its Application Scenarios", *International Journal of Current Microbiology and Applied Sciences* ISSN: 2319-7706 Vol.7 No 5.
7. Sivanandam and Deepa, (2015) "Optimal Path Planning for Intelligent Mobile Robot Navigation using Modified Particle Swarm Optimization", *International Journal of Engineering and Advanced Technology*, vol. 2, Issue - 4, pp. 260-267.
8. Soh and Coelho L. D., (2006) "Path Planning Optimization for Mobile Robots Based on Bacteria Colony Approach", *Springer Book Series of Applied Soft Computing Technologies: The challenge of complexity*, vol. 34, pp.187-198.
9. Subramanian, Sudhagar and Rajeswari (2014) "Collective Robotic Search Using Hybrid Techniques: Fuzzy Logic and Swarm Intelligence inspired by Nature", *Journal of Engineering Applications of Artificial Intelligence*, vol. 22, pp. 431-441.
10. Wang, X. and Zeng, G. (2014). A Dynamic Planning Method for Mobile Agent Implementation based on Service Recommendation. *An International Journal of Applied Mathematics & Information Sciences China: Shandong University, Jinan 250101, P. R. 3*, pp. 1247-1255.