SMART NAVIGATION SYSTEM FOR SELECTING THE SHORTEST ROUTES BASED ON HYBRID GENETIC ALGORITHM TECHNIQUE

Abadal-Salam. T. Hussain[1], Syed F. Ahmed[2], D. Hazry[3],

[1,2,3] Centre of Excellence for Unmanned Aerial Systems (COEUAS),
Universiti Malaysia Perlis (UniMAP)
Seriab, 01000, Kangar, Perlis, Malaysia.

ABSTRACT

In this research paper a new hybrid technique based on genetic algorithm (GAs) and logic systems is proposed which is used as an optimization tool for selecting the simplistic routes that reaches the desired target without calibration or initialization of the mobile robotic system. In this system trials of travelled sequences (routes) are used as inputs and each sequence comprises of many stages, called node represented by a binary number. These nodes are treated as populations and their reproduction is based on a random selection of the fitness functions. During experiment more than 30 travel trials are tested on proposed method, each of which contains a sequence of seven different nodes. These set of trials used as an input, are randomly selected for each run in the proposed GA and logic engine system to obtain one optimized route that reaches the desired output. Experimental results proved that proposed methodology is very accurate and fast, when it tested on simple, tricky, and complex routes.

Keywords: Simplistic Path Finder, Hybrid Genetic Algorithm, Optimization, Logic Representation and Simplification.

I INTRODUCTION

In the last three decades, Genetic Algorithms (GAs) have engaged the curiosity of many research communities. GAs are search procedures that are based on the mechanics of Natural Selection by N. Achour 8(2011). They were designed by John Holland (1975) at the University of Michigan. Since then, these algorithms have been applied successfully to a wide range of problems, including medical, financial, and other engineering problems, with a number of modifications to D. E. Goldberg (1989), S. Silva (2003 & 2004). However, the basic functionality of the GA and the philosophy behind its operators has remained very similar to the original concept.

In its simplest form, the GA begins with a (usually) randomly generated “population" of “individuals”, where each individual represents a possible solution. These individuals were evaluated to determine how close they are to the desired goal or target and assigned a value, or “goodness", based on that evaluation. Genetic operators are then applied according to the user defined probabilities; during this process, those individuals with higher values are

given more chances to reproduce and to produce higher valued offspring. This process continues until an optimal solution is achieved, which always covers the aimed target of reaching the path end within a shortest route, or until the user determines that an acceptable solution will not be reached.

II RELATED WORK

GA's are considered as optimization algorithms that search a space of potential solutions; thus, they are a useful tool to find a solution (Y) from a set of random selections (XN). The solution to determining the search route using Genetic Algorithms was proposed for the first time John Holland (1975), although there are also other methods proposed by D. E. Goldberg (1989). A task that is common to all the proposed methods involves choosing the initial population. Most of these methods then use a set of paths to determine the solution. The optimal path is calculated after several iterations.

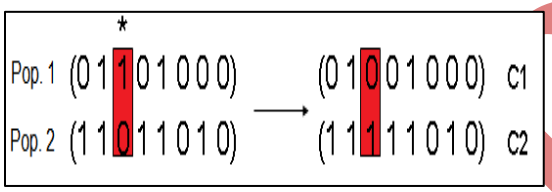


Figure 2: Single point mutation carried out by the C operator.

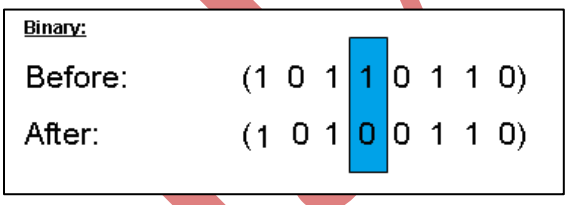


Figure 3: Mutation process carried out by the M operator.

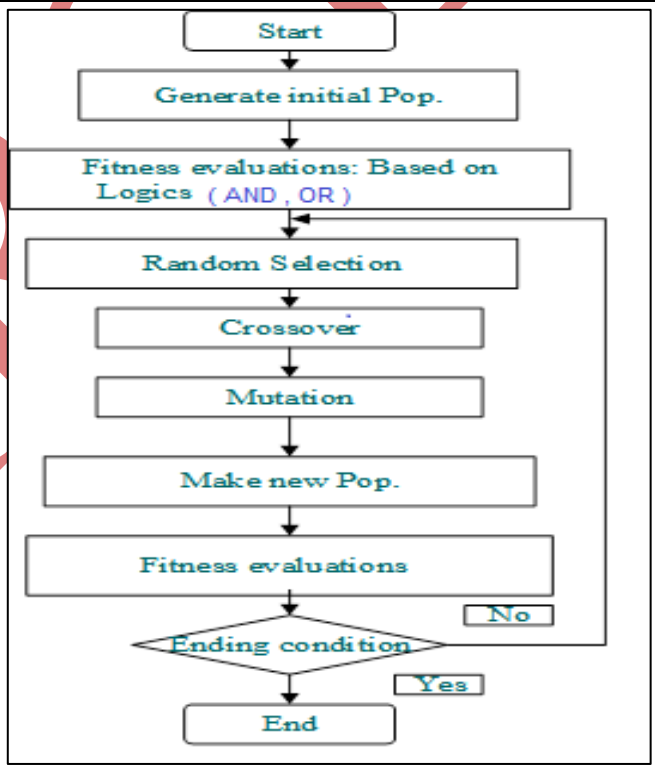


Figure 1: Flow chart of GA Process.

During each of the iteration, the fitness function (optimization criterion) was evaluated. 14 S. Silva (2003) proposed the use of a performance function determined by the linear distance between the robots position from the target. Some papers have focused on using GAs in dynamic environments C. E. Thomas (1999), whereas others have attempted to investigate the synergism of fuzzy logic – GAs by R. Guernane (2009) and neural networks - GAs by

C. Fayad (2006) and N. Noguch (1997). In this article, we developed a route optimization algorithm to identify the shortest route for mobile robots or tracks. We mainly focused on using genetic algorithms to calculate the optimal route and we demonstrated that the use of this application using the logic principle reduces the iteration process and thus the simulation time by R. Bohlin 10(2000), and author A. T. Hussain (2005).

'Figure 1' illustrates the sequence of the GA process steps. The three basic operators used in GAs are selection, crossover, and mutation. The first step involves encoding the problem as an artificial chromosome or chromosomes. These chromosomes can be strings of 1s and 0s, parameter lists, integers or even complex codes. The Reproduction (R) operator simply reproduces the individual. The Crossover (C) operator exchanges the chromosome strings of two individuals starting from a random index. In essence, the C operator randomly chooses a locus and exchanges a sequence of random length before and after the locus between two chromosomes to create two off springs. The most common types of C operations used with binary representations are single-point, which is illustrated in 'Fig. 2', double-point and multi-points crossovers. The Mutation (M) operator inverts one bit of the individual, i.e., the 'M' operator creates a mutation, as shown in 'Fig. 3'. In addition, the n-point M operator creates n mutations by applying the 'M' operator n times. The parameter n can be fixed for the entire simulation or be randomly chosen each time the operator is applied by S. Silva (2003), J. H. Holland (1975).

Throughout the GA process, a set of steps, or loop, which are illustrated in 'Fig. 4', are continuously iterated until the optimization is finished.

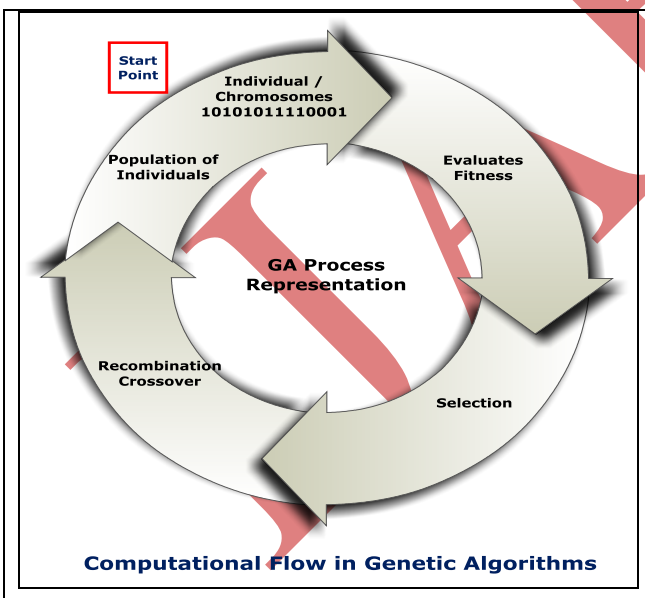


Figure 4: Computational flow in genetic algorithms.

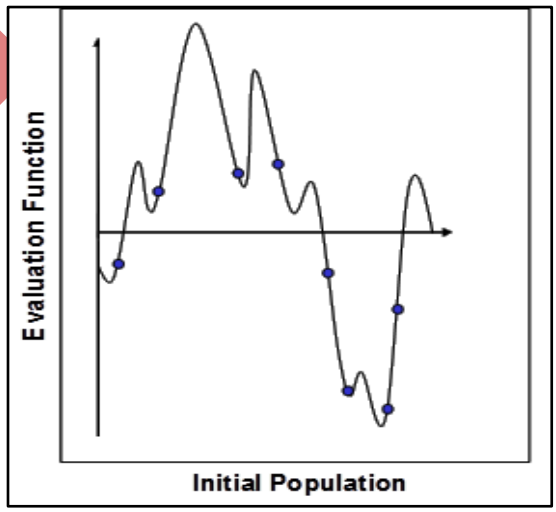


Figure 5: Evaluation function applied to the initial population.

'Fig.5' shows the initial state of the population that is estimated by the evaluation function and 'Fig. 6' describes the evaluation of the final state of the population. As shown in 'Fig. 6', the final population recognizes the peak points; in addition, the symbol "*" represents the best individual solution.

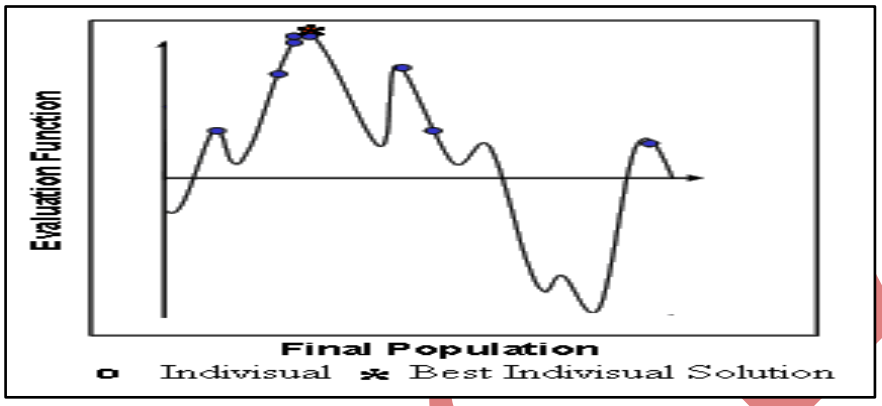


Figure 6: Evaluation Function Applied To The Final Population.

The overall procedure of the genetic algorithm can be listed as follows:

1. *Set k=0;*
2. *Create an initial population based on travel trials;*
3. *Initial logic representation*
4. *Evaluate the population using the fitness function;*
5. ***While** < the termination conditions are not met >*
6. *Set k=k+1;*
7. *Reproduce a new population based on the evaluation;*
8. *Apply crossover;*
9. *Apply mutation;*
10. *Evaluate the new population using the fitness function;*
11. ***End While***
12. *Output the solution;*
13. *Optimized logic representation of shortest path selection.*

III FITNESS FUNCTION

The most important part of the GA process is the fitness function, which guides the optimization process. In addition, it is necessary to determine a procedure for discriminating the good solutions from the bad ones. The simplest method involves having a human intuitively choose the better solutions over the worse ones. More elaborate computational simulations can also be developed to identify the good solutions. In essence, a function determines the relative fitness of the solution, which in turn determines whether the evaluated solution will be used to guide the evolution of future generations. One common application of GAs is function optimization, in which the

goal is to determine a set of parameter values that maximizes a complex parameter function. In many cases, the fitness of a string is the function value at that point.

The selection and other genetic operators can process the population interactivity to create a sequence of populations that hopefully contain increasingly better solutions. These better organisms, or solutions, are assigned higher fitness function values.

IV APPLICATION AND RESULTS

We used our algorithm over a hundred times to obtain the shortest route reaches the desired target while the starting points are not the same. The results that were obtained exhibited extremely high accuracy. The GA and logic engine, were shown in 'Fig. 7', uses a large number of inputs (travel trials), from one to as many as hundreds. Each travel trial contains many sequences (nodes). The output of the algorithm is the shortest route(s) to reach the target.

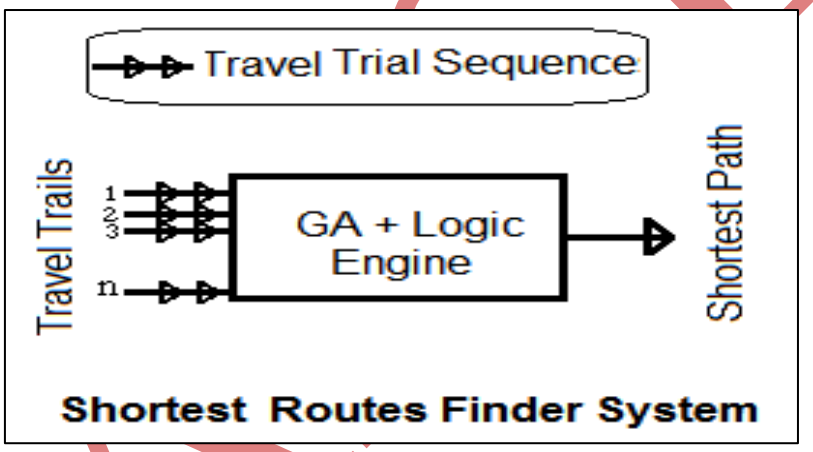


Figure 7: Illustration of the system that identifies the shortest path reaches the target.

TABLE 1: Per-Diacritic Statistics.

Possible Robot paths	TRAVEL TRIALS (INPUTS SEQUENCES) X1 X2 X3 X4 X5 X6 X7 … X_N	REACHING DESIRED TARGET (OUTPUT) Y
P1	1 0 1 0 0 1 1	1
P2	0 1 1 0 1 1 1	0
P3	0 1 0 1 0 1 1	0
P4	0 0 0 1 1 1 1	1
P5	1 0 1 0 0 1 0	0
P6	1 0 1 0 1 1 1	1
P7	0 0 1 1 0 1 1	1
P8	0 0 1 1 1 1 1	0
P9	0 1 0 1 0 1 1	0
P10	0 1 0 0 1 1 0	1
P11	1 1 1 1 1 1 0	1
P12	0 1 0 0 1 1 0	0
P13	0 1 1 0 0 1 0	1
P14	0 1 1 0 1 1 0	0
P14	0 1 0 1 0 1 0	0
		1

P15	**0 1 1 0 1 0 1**	**0**
P16	**1 0 0 0 0 1 1**	**1**
P17	**1 0 1 0 1 1 0**	**1**
P18	**1 0 0 1 0 1 1**	**0**
P19	**1 0 0 1 0 1 1**	**1**
P20	**1 0 1 0 0 1 1**	**0**
P21	**1 0 1 0 1 0 0**	**0**
P22	**1 0 1 0 0 1 0**	**1**
P23	**1 0 1 0 1 0 1**	**1**
P24	**1 1 0 0 0 1 0**	**0**
P25	**0 1 0 0 1 1 1**	**0**
P26	**0 1 0 1 0 1 0**	**1**
P27	**0 1 0 1 0 1 0**	**0**
P28	**1 0 1 0 0 1 0**	**1**
P29	**1 0 1 0 1 0 1**	**1**
P30	**0 1 0 1 0 1 1**	**0**
P31	**0 0 0 0 1 0 0**	
P_M	**.. .. ..**	
X_1 … X_7 are the stages of each sequence (nodes) on travels; "1" means pass through the node and "0" doesn't pass through the node. X_N could be N nodes and P_M could be M paths.		**"1" Means Reach Desired Target, "0" Means Doesn't Reach**

The traveling sequences that the reaches the desired target (Y) were marked by "1" on Table 1. 'Fig. 8' shows the logic representation of the traveling paths. So tree on the 'Fig. 8' can be represented by these logic 'Statements 1' and 'Statement 2'.

The logic representation of Circle A is A= AND (X7, X2) (1)

The logic representation of Circle B is B= NAND (A, X4) (2)

And so on for all other branches of the tree.

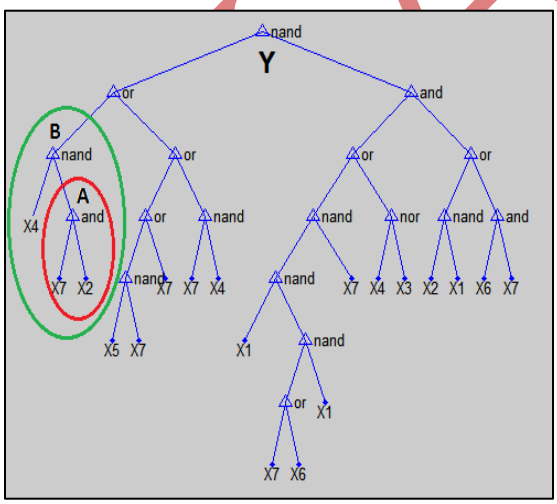


Fig 8

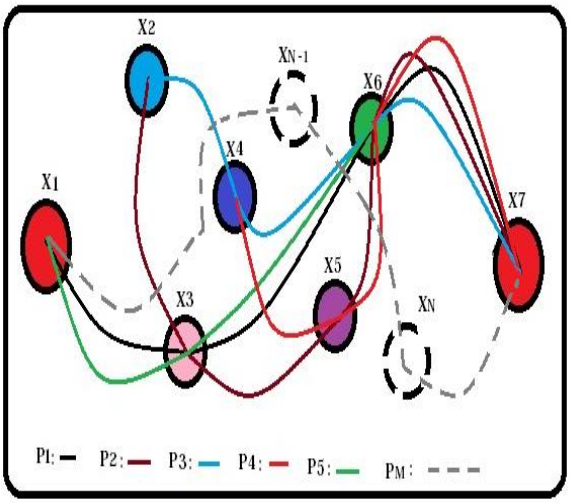
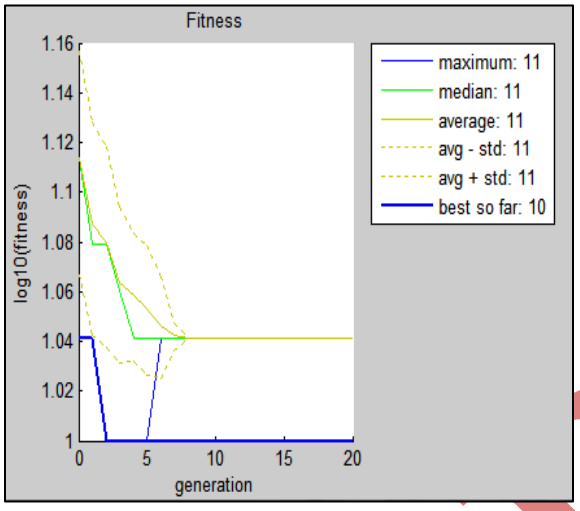


Fig 9

Figure 8: Logical representation of the traveled paths.

Figure 9: Graphical representation of some robot paths illustrated on table 1.

The use of GAs is an optimization tool to minimize (simplify) the logic representation of the path sequences. In our application, the fitness function is dynamic and depends on how far the solution is from the desired output. 'Fig 10', 'Fig 11', 'Fig 12' and 'Fig 13' shows the behaviour of the GA and the logic engine. 'Fig. 10' shows the description of the dynamic fitness function. 'Figure 11' illustrates the relationship between the population complexity and the tree depth throughout the optimization process.



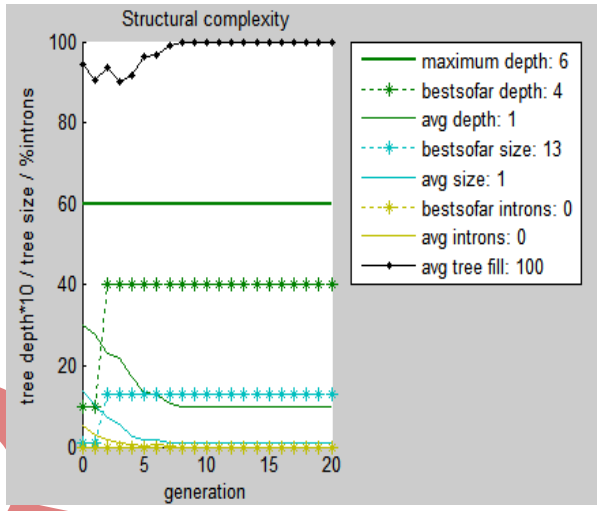
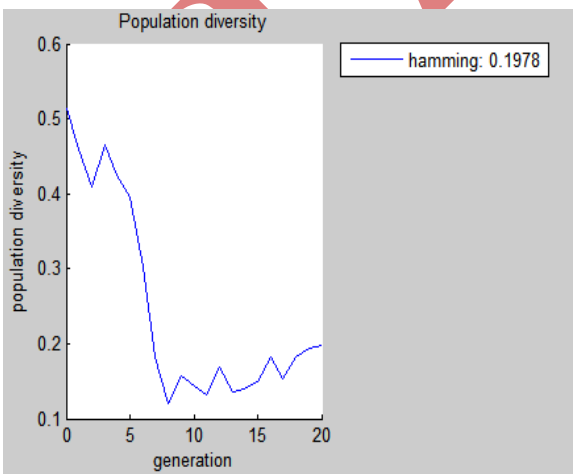


Figure 10: Cascade of dynamic fitness functions. **Figure 11: Structure complexity.**

'Figure 12' shows the relationship between the syndrome generation and the population diversity that is observed throughout the optimization process. 'Figure 13' shows the results of the optimization of the shortest route that reaches the desired output. As shown on the logic tree, the shortest path that reached the desired target can be read from the tree in 'Fig. 13' which can be written in logic statements as follows:



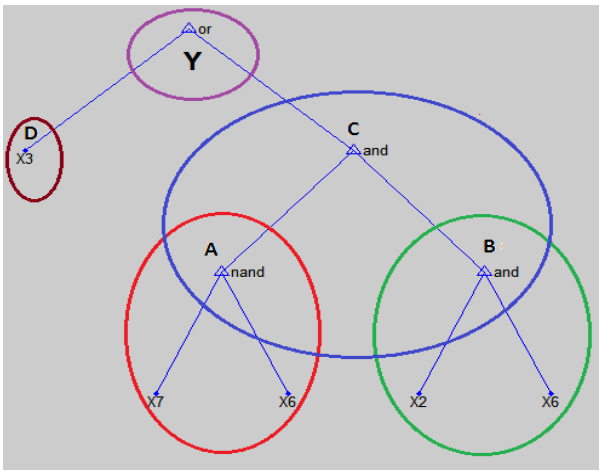


Fig 12 **Fig 13**

Figure 12: Population Diversity of the GA and logic process.
Figure 13: Graphical tree representing output of the logic representation of the shortest path that reach to the desired target.

The logic representation of Circle A is A= NAND (X7,X6) (3)

The logic representation of Circle B is B= AND (X2 , X6) (4)

The logic representation of Circle C is C= AND (A , B) (5)

X3 has no effect to the logic statement of C, so it can be neglected.

From 'Equation 3', A must be logically "1" means X7 is "0" and X6 is "0" as well or any of them must be "0" while the other one is "1". From 'Equation 4', B must be logically "1" means X2 is "1" and X6 is "1" as well.

And (Y) is the desired target to be reached, so from 'Equation 5', C must be logically "1" means A is "1" and B is "1" as well. This was satisfied by 'Equations 3' and 'Equation 4'.

With every run of the system, the obtainable results will not be the same. This GA and logic optimization system is based on randomization; therefore, each time it is used, it will return with different optimized results. However, the results exhibit some variances; will still represent the shortest path that reaches the desired target.

The results described above were obtained using the GPLAB software with modifications to achieve an optimum results (A Genetic Programming Toolbox for MATLAB)

V CONCLUSION

In this paper, we have proposed a robust technique for finding the Paths Simplicity Selector that reaches a desired target among many others, which were not the same distance, number of nodes or starting points. The system is using the hybrid GAs with logic engine system to identify the logic function that represents the input paths as well as to represent the simplicity selected path. We used GAs with its dynamic fitness function in our research. The fitness function parameters used the basic logic functions (NOT, AND, NAND, OR and/or NOR).

The system works by using a GAs as an optimizer to the logic engine, so when the optimization took place means the shortest simplicity path to reach the desired target been obtained and selected.

The main advantages of this system were its ability to identify the shortest path for a desired target without needs for any calibrations or initialization as well as no need for a specific started point. This system has a high accuracy of obtaining the optimized path because of the logic simplification took place by the optimization of the hybrid GAs.

REFERENCES

1. T. Hussain, "*Genetic Algorithms is an Optimization Tool for Digital Design*", Proceeding of the First International Conference on Modeling, Simulation and Applied Optimization, Sharjah, U.A.E. February 1-3, 2005.
2. E. Thomas, M. A. C. Pacheco, M.M., and B.R.Vellasco, "*Mobile Robot Path Planning Using Genetic Algorithms*,". In foundations and tools for neural modeling, vol. 1606/1999, Springer Berlin/ Heidelberg, 1990, pp. 671–679.

3. Fayad and P. Webb, "*Development of a hybrid crisp-fuzzy logic algorithm optimised by genetic algorithms for path-planning of an autonomous mobile robot*" in Journal of Intelligent & Fuzzy Systems, 2006, pp. 15-26.
4. N. Noguchi and H. Terao, "*Path planning of an agricultural mobile robot by neural network and genetic algorithm"* in Journal Computers and electronics in agriculture, vol. 18, no2-3, 1997, pp. 187–204.
5. E. Goldberg, "*Genetic algorithms in search, optimization, and machine learning*", Reading, MA. Addison-Wesley, 1989.
6. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1st ed. MA: Addison-Wesley, 1989.
7. J. H. Holland, Adaptation in natural and artificial systems, Ann Arbor: University of Michigan Press, 1975.
8. N. Achour, "*Mobile Robots Path Planning using Genetic Algorithms*", the Seventh International Conference on Autonomic and Autonomous SystemsBaker", ICAS, 2011.
9. R. Guernane, and N. Achour, "*An Algorithm for Generating Safe and Execution-Optimized Paths*" in Proceedings of the Int. Conference on Autonomic and Autonomous Systems, 2009, pp. 16–21, Spain.
10. R. Bohlin, and L. E Kavraki, 2000. "*Path Planning Using Lazy PRM" in Proceedings of the IEEE Int. Conference on Robot & Autom*, pp. 521–528, IEEE Press, San Francisco.
11. S. Yang, H. Cheng, and Fang Wang, "*Genetic Algorithms With Immigrants and Memory Schemes for Dynamic Shortest Path Routing*," IEEE Trans on Syst man, and cyb, vol. 40, n°1, 2010, pp. 52–63.
12. Sara Silva, GPLAB (A Genetic Programming Toolbox for MATLAB) - ECOS - Evolutionary and Complex Systems Group, University of Coimbra, Portugal, Version 3, 2007.
13. S. Silva, Costa, E. "*Dynamic limits for bloat control - variations on size and depth*" to appear in Proceedings of GECCO-2004, Berlin. Springer Verlag (2004).
14. S. Silva, Almeida, J. "*Dynamic maximum tree depth - a simple tech¬nique for avoiding bloat in tree-based GP*", in E. Cant´u-Paz, Foster, J.A., Deb, K., Davis, L.D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Pot-ter, M.A., Schultz, A.C., Dowsland, K.A., Jonoska, N., Miller, J., editors, Proceedings of GECCO-2003, Berlin. Springer Verlag (2003) 1776–1787.